HANDS-ON LAB GUIDE

Snowflake Cortex AI with Streamlit: capture, analyse and secure maintenance requests

A two-part lab. Part 1: Cortex AI and row-level security in SQL. Part 2: a Streamlit app with input, AI insights and the same row-level security.

Prepared for	BHP training participants
Platform	Snowflake trial account (Enterprise edition), Snowsight, Streamlit in Snowflake
Data	Real BHP asset and site names. All maintenance requests, equipment numbers, people, phone numbers and emails are dummy data.

How to use this guide

Conventions

Item	Meaning
Numbered steps	Do the steps in the given order. Do one step at a time.
Projects » Workspaces	A path in the Snowsight menu. Select each item from left to right.
Why: box (purple)	Why you do the step, and which part of the architecture it builds. Read it before you do the step.
❶ ❷ ❸	The numbered parts of the architecture diagram (Figure 1).
Do: box (blue)	The actions for one step: where to go, what to select, what to type.
Dark code box	The SQL of the step. The same SQL is in the lab file.
Green code box	Python code for the Streamlit app.
Result box	The result that you must see. If your result is different, stop and speak to the trainer.
Ctrl + Enter	Push the two keys at the same time.
NOTE / **CAUTION**	Information that helps you / a risk of an error or data loss. Obey a CAUTION.

The Snowsight screen

Part of the screen	What it does
Navigation menu (left)	Opens the pages. **Projects** is in the group **Work with data**. **Admin** is in the group **Manage**.
User menu (bottom left)	Your user name. Select it to switch the role or to see the account details.
Projects » Workspaces	The SQL editor. Your private workspace is **My Workspace**: a file list, an editor and a **Results** pane.
Projects » Streamlit	The list of Streamlit apps that your role can use.

> **NOTE:** Snowflake updates Snowsight often. If a menu item is in a different place, use the search box at the top of the navigation menu.

Lab files

All code is in this guide. The same code is in `BHP_Asset_Care_Lab_Files.zip`. In the Checklist you upload the files into Snowsight, so you do not copy SQL from this PDF.

File	Used in
`00_preflight_check.sql`	Checklist, before Part 1

`01_setup.sql`	Part 1, Lab 1
`02_data.sql`	Part 1, Lab 2
`03_cortex_ai.sql`	Part 1, Lab 3
`04_security.sql`	Part 1, Lab 4
`05_app_owner.sql`	Part 2, Lab 5
`streamlit_app_v1.py`, `_v2.py`, `_v3.py`	Part 2, Labs 5, 6 and 7
`06_share_app.sql`	Part 2, Lab 8
`07_change_group.sql`	Part 2, Lab 9
`99_cleanup.sql`	After the training (optional)

How to run one step in Workspaces

1. Select **Projects » Workspaces**.
2. In the file list, select the lab file, for example `01_setup.sql`.
3. Find the comment line of the step, for example `-- Step 1.1`.
4. Highlight all lines of the step, to the last semicolon `;` of the step.
5. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
6. Look at the **Results** pane below the editor. If it is closed, push Ctrl + Alt + ↑ (Mac: Control + Option + ↑).

> **CAUTION:** Run each file from **Step x.0** (or Step 1.1). Step x.0 sets the role and the warehouse for the file. If you do not run it, you can get *No active warehouse* or *Insufficient privileges*.

> **NOTE:** For a block that starts with `EXECUTE IMMEDIATE $$` or `CREATE ... PROCEDURE`, highlight the full block to the last line `$$;`.

The use case

The business story

BHP operates large mines, plants and ports. The equipment includes haul trucks, shovels, draglines, crushers, grinding mills, conveyors and ship loaders. When equipment breaks, production stops.

Maintenance planners and operators write maintenance requests. The text often contains names, phone numbers and emails. The requests are in many places, so nobody sees the repeat problems early.

In this lab you make one app: **Asset Care Requests**. Users type a request. The app saves it in Snowflake. Cortex AI hides the personal data, writes a short summary, and finds the repeat problems. Each asset team sees only its own asset. Head office sees all assets.

Asset (real name)	Commodity	Dummy sites in this lab	Group role
Western Australia Iron Ore (WAIO)	Iron ore	Mt Newman, Mining Area C, Jimblebar, Port Hedland	`MAINT_WAIO`
Escondida (Chile)	Copper	Escondida Pit, Laguna Seca Concentrator, Los Colorados Concentrator, Coloso Port	`MAINT_ESCONDIDA`
BHP Mitsubishi Alliance (BMA)	Steelmaking coal	Goonyella Riverside, Peak Downs, Saraji, Hay Point	`MAINT_BMA`
Head office	All	All 12 sites	`MAINT_HQ`

The requirement and the lab feature

Requirement	How the lab does it
End users give input in a Streamlit app.	Tab **New request** (Lab 6).
The input is saved in Snowflake.	The app inserts a row into `MAINT_LAB.OPS.MAINT_REQUESTS` (Lab 6).
Cortex AI analyses the data in Snowflake.	AI functions in SQL (Lab 3). AI on each new row (Lab 6). AI analysis in the app (Lab 7).
Streamlit shows the insights.	Tab **Insights** and tab **AI analysis** (Lab 7).
App users are managed with Snowflake users and groups.	Users log in with their Snowflake user. Snowflake roles are the groups (Lab 4). App access is a grant to a role (Lab 8).
The same row-level security (RLS) as in Snowflake.	One row access policy on the tables. It filters SQL and the app in the same way, also inside AI functions (Labs 4, 8, 9).
No live company account. Dummy data only.	A Snowflake trial account and dummy data.

Architecture

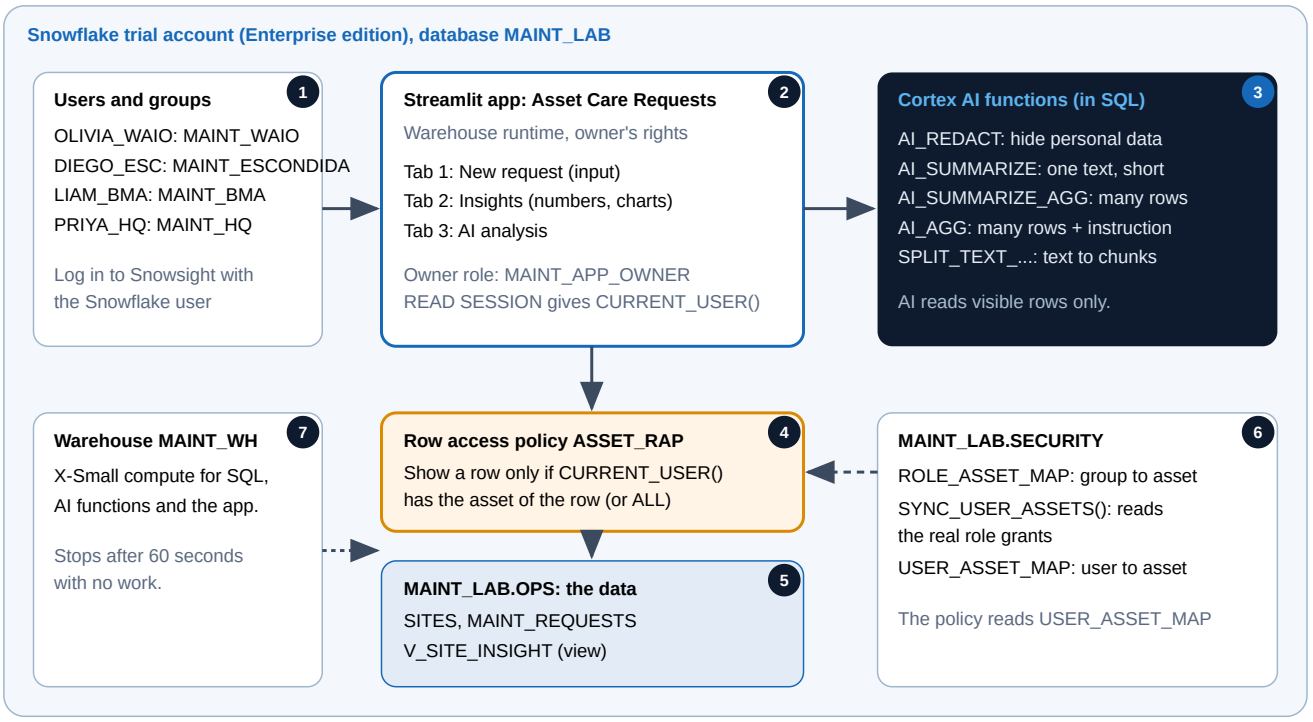


Figure 1. Architecture of the lab. The numbers show the parts that each lab builds.

Part	Name	What it does	Built in
1	Users and groups	Snowflake users. Snowflake roles are the groups (one for each asset, one for HQ).	Lab 4
2	Streamlit app	The user interface: input, insights, AI analysis. Runs with the owner role.	Labs 5 to 8
3	Cortex AI functions	AI that runs inside SQL. The data does not leave Snowflake.	Labs 3, 6, 7
4	Row access policy	The RLS rule. Snowflake adds it to every query on the tables.	Lab 4
5	Data (OPS schema)	Sites, maintenance requests and the insight view.	Labs 1, 2, 3
6	Security tables	Group to asset rules, and the user to asset list that the policy reads.	Lab 4, Lab 9
7	Warehouse	The compute that runs SQL, AI functions and the app.	Lab 1

Data privacy flow

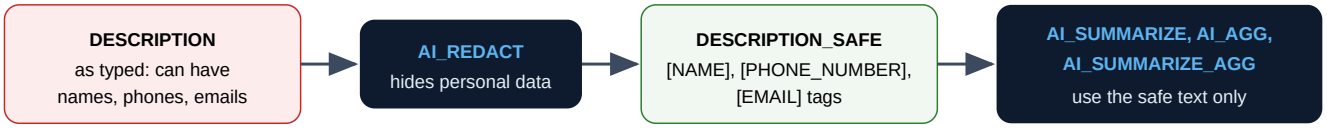


Figure 2. AI_REDACT makes a safe copy of the text. All other AI functions and the app use only the safe copy.

Flow: save a request and see insights

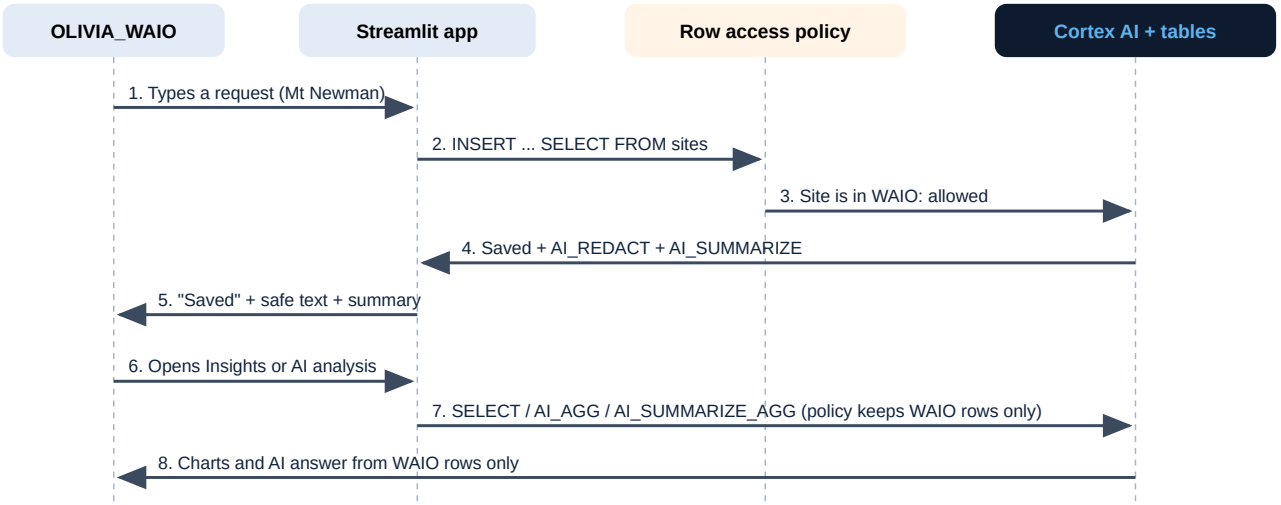


Figure 3. What happens when OLIVIA_WAIO (WAIO group) uses the app.

Cortex AI in a trial account

A self-service trial account does not allow all Cortex AI functions. We tested each function in a new Enterprise trial account (October 2026, no credit card). This lab uses only the functions that work.

Function	Trial	Use in this lab
`AI_REDACT`	**Works**	Hide names, phone numbers and emails (Labs 3, 6)
`AI_SUMMARIZE` (Preview)	**Works**	Short summary of one request (Labs 3, 6)
`AI_SUMMARIZE_AGG`	**Works**	One summary of many requests (Labs 3, 7)
`AI_AGG`	**Works**	Your own instruction over many requests (Labs 3, 7)
`SPLIT_TEXT_RECURSIVE_CHARACTER`	**Works**	Cut long text into chunks (Lab 3)
AI_COMPLETE, AI_CLASSIFY, AI_FILTER, AI_SENTIMENT, AI_EXTRACT, AI_TRANSLATE, AI_EMBED, AI_SIMILARITY, AI_PARSE_DOCUMENT, AI_TRANSCRIBE, AI_COUNT_TOKENS	**Blocked**	Not used. Error: *AI function ... is not available for trial accounts.*
Old `SNOWFLAKE.CORTEX.*` functions (COMPLETE, SENTIMENT, ...)	**Blocked**	Not used. New accounts must use the AI_* functions.

> **NOTE:** This lab does not need a credit card. In a paid account, all functions in the table are available. The same app can then add, for example, AI_CLASSIFY for the failure mode and AI_COMPLETE for free questions.

The two parts

Part	Labs	What you make
Part 1 (day 1)	Lab 1 Setup Lab 2 Dummy data Lab 3 Cortex AI in SQL Lab 4 Users, groups and RLS	A database with 60 maintenance requests. Safe text and AI summaries in the table. Demo users in groups. A row access policy that you test.
Part 2 (day 2)	Streamlit primer Lab 5 App owner and first app Lab 6 Save input with AI Lab 7 AI insights Lab 8 Share and test as other users Lab 9 Change a group	The Asset Care Requests app. It saves input, runs Cortex AI and shows insights. Each user sees only the assets of his or her groups.

Before the training: checklist

Each person (trainer and participants) has his or her own trial account. Do steps C1 to C8 before Part 1.

C1. Sign up for a trial account

> **Why:** Each person needs an own account, because the labs create users, roles and account settings. Only an account administrator can do this.

1. In Google Chrome or Microsoft Edge, go to **signup.snowflake.com**.
2. Type your first name, last name and your own email address. Select the reason for the sign-up. Select **Continue**.
3. Type your company name and your job title.
4. For **Snowflake edition**, select **Enterprise**.
5. For the cloud provider, select **Amazon Web Services (AWS)**. For the region, select **US West (Oregon)**.
6. Read the terms. Select the check box to accept them.
7. Select **Get started**. If Snowflake shows a short survey, answer it or skip it.

> **CAUTION:** Select **Enterprise**. Row access policies (part 4) are not available in the Standard edition.

C2. Activate the account

1. Open the email from Snowflake (subject similar to *Activate your Snowflake account*).
2. Select the activation button. A new browser tab opens.
3. Type a user name and a password. Keep them in a safe place.
4. Select **Get started**. Snowsight opens.

> **NOTE:** If Snowflake asks you to set up multi-factor authentication (MFA) for your own user, follow the instructions on the screen.

C3. Copy your account URL

> **Why:** In Part 2 you log in as demo users in a private window. You need the account URL for this.

1. Select your user name at the bottom of the navigation menu.
2. Open the account selector (your account name). Select **View account details**.
3. Copy the **Account URL**. Paste it into a text file. Close the dialog.

C4. Use the role ACCOUNTADMIN

1. Select your user name at the bottom of the navigation menu.
2. Select **Switch role**. Select **ACCOUNTADMIN**.
3. Make sure that **ACCOUNTADMIN** shows below your user name.

C5. Get the lab files

1. Download `BHP_Asset_Care_Lab_Files.zip` from the trainer.
2. Extract the zip file. You get 9 SQL files and 3 Python files.

C6. Upload the lab files to Workspaces

> **Why:** When the files are in Snowsight, you run the SQL from the file. You do not copy code from the PDF, so you do not make copy errors.

1. Select **Projects » Workspaces**. **My Workspace** opens.
2. At the top of the file list, select **+ Add new** (or the **+** next to **My Workspace**).
3. Select **Upload Files**. Select all 12 files. Select **Open**.
4. Make sure that the 12 files show in the file list.

> **NOTE:** If you cannot upload, select **+ Add new** » **SQL File**. Hover over the new file, select ⋮ » **Rename**, and type the file name. Copy the code from Appendix B.

C7. Run the preflight check

> **Why:** The check makes sure that the 5 AI functions of this lab work in your account, before the class starts.

1. In the file list, select `00_preflight_check.sql`. Click one time in the editor.
2. Push Ctrl + Shift + Enter (Windows) or Cmd + Shift + Return (Mac). This runs all statements, one after the other.
3. Wait until the run stops (less than 1 minute).
4. Look at the **Results** pane. The last result must be *PREFLIGHT_WH successfully dropped.*

Result: No error message shows. The last result is *PREFLIGHT_WH successfully dropped.*

> **CAUTION:** If an error shows, the run stops at that statement. Do not start Part 1. Speak to the trainer.

```
-- ============================================================================
-- PREFLIGHT CHECK (all participants, before Part 1)
-- File: 00_preflight_check.sql
-- Run as: ACCOUNTADMIN, on a new Enterprise trial account.
-- Purpose: Make sure that the 5 Cortex AI functions of this lab work in
--          your account. Each statement must give a result with no error.
-- ============================================================================

-- Check 1: Use the administrator role.
USE ROLE ACCOUNTADMIN;

-- Check 2: Region and account. Write the values in your notes.
SELECT CURRENT_REGION() AS region, CURRENT_ACCOUNT_NAME() AS account, CURRENT_VERSION() AS
version;

-- Check 3: Turn on cross-region inference (backup for AI models).
ALTER ACCOUNT SET CORTEX_ENABLED_CROSS_REGION = 'ANY_REGION';
```

```
-- Check 4: A temporary warehouse for the checks.
CREATE WAREHOUSE IF NOT EXISTS preflight_wh WAREHOUSE_SIZE = 'XSMALL' AUTO_SUSPEND = 60;
USE WAREHOUSE preflight_wh;

-- Check 5: The 5 AI functions of this lab.
SELECT AI_REDACT('Operator Jack Wilson, mobile 0412 555 019, reported a leak.') AS check_redact;

SELECT AI_SUMMARIZE('The haul truck stopped on the ramp because the engine overheated. A coolant
hose burst. This is the second engine problem this month on this truck, so the planner wants a
root cause check.') AS check_summarize;

SELECT AI_SUMMARIZE_AGG(t) AS check_summarize_agg
FROM (SELECT column1 AS t FROM VALUES ('The conveyor belt has a rip.'), ('The crusher bearing is
hot.'));

SELECT AI_AGG(t, 'List the equipment problems as bullet points.') AS check_agg
FROM (SELECT column1 AS t FROM VALUES ('The conveyor belt has a rip.'), ('The crusher bearing is
hot.'));

SELECT SNOWFLAKE.CORTEX.SPLIT_TEXT_RECURSIVE_CHARACTER('hello world are you here', 'none', 15,
10) AS check_split;

-- Check 6: Remove the temporary warehouse.
DROP WAREHOUSE IF EXISTS preflight_wh;
```

C8. Make sure that you can open a private window

1. In Chrome or Edge, push Ctrl + Shift + N (Mac: Cmd + Shift + N).
2. Close the private window. You use it in Lab 4 and Part 2.

> **NOTE: Trainer only:** 1 day before the training, do the full lab one time in a new trial account. Then run `99_cleanup.sql`.

Names in this lab

Object	Name
Warehouse 7	`MAINT_WH`
Database and schemas	`MAINT_LAB.OPS` 5, `MAINT_LAB.SECURITY` 6, `MAINT_LAB.APP` 2
Group roles 1	`MAINT_HQ`, `MAINT_WAIO`, `MAINT_ESCONDIDA`, `MAINT_BMA`, parent `MAINT_APP_USER`
App owner role	`MAINT_APP_OWNER`
Streamlit app 2	`MAINT_LAB.APP.ASSET_CARE`, title **Asset Care Requests**
Demo users 1	`OLIVIA_WAIO`, `DIEGO_ESC`, `LIAM_BMA`, `PRIYA_HQ`
Demo password	`AssetCare#Lab2026` (training only)

Part 1: Cortex AI and row-level security in SQL

Part 1 (day 1)

Labs 1 to 4. You build parts 7 5 3 6 4 1 of the architecture with SQL in Snowsight Workspaces.

> **NOTE:** Before you start: log in to Snowsight. Make sure that your role is ACCOUNTADMIN (C4).

Lab 1: Set up the account

> **Goal:** Make the warehouse, the database and the schemas.

> **Why:** Every query and every AI function needs compute (the warehouse). The data, the security rules and the app need a place (database and schemas). We make 3 schemas, so that the data, the security rules and the app are separate. **Architecture:** 7 5

Open the file

Do:

1. Select **Projects » Workspaces**.
2. In the file list of **My Workspace**, select `01_setup.sql`. The file opens in the editor.
3. Read the comments in the file. Each step starts with a comment line `-- Step ...`.

Step 1.1 to 1.2: Use ACCOUNTADMIN and turn on cross-region inference

> **Why:** ACCOUNTADMIN can change account settings. Cross-region inference lets Cortex use a model in a different region if your region does not have it. The data does not leave Snowflake. **Architecture:** 3

Do:

1. In the editor of `01_setup.sql`, find **Step 1.1 and Step 1.2**.
2. Highlight all lines of **Step 1.1 and Step 1.2**, from its comment line to the last semicolon.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

```
-- Step 1.1: Use the administrator role.
USE ROLE ACCOUNTADMIN;

-- Step 1.2: Let Cortex AI use a model in a different region
-- if the model is not available in your region.
ALTER ACCOUNT SET CORTEX_ENABLED_CROSS_REGION = 'ANY_REGION';
```

Result: *Statement executed successfully.*

Step 1.3 to 1.4: Make the warehouse, database and schemas

> **Why:** X-Small is the smallest and cheapest size. AUTO_SUSPEND = 60 stops the warehouse after 60 seconds with no work, so it does not use credits when nobody works. **Architecture:** 7 5

Do:

1. In the editor of `01_setup.sql`, find **Step 1.3 and Step 1.4**.
2. Highlight all lines of **Step 1.3 and Step 1.4**, from its comment line to the last semicolon.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

```
-- Step 1.3: Create a small warehouse. It stops after 60 seconds of no work.
CREATE WAREHOUSE IF NOT EXISTS maint_wh
  WAREHOUSE_SIZE = 'XSMALL'
  AUTO_SUSPEND = 60
  AUTO_RESUME = TRUE
  INITIALLY_SUSPENDED = TRUE
  COMMENT = 'Warehouse for the BHP maintenance Cortex AI lab';

USE WAREHOUSE maint_wh;

-- Step 1.4: Create the database and three schemas.
--   OPS      = business data (sites and maintenance requests)
--   SECURITY = access rules (who can see which asset)
--   APP      = the Streamlit app (Part 2)
CREATE DATABASE IF NOT EXISTS maint_lab COMMENT = 'BHP maintenance Cortex AI lab';
CREATE SCHEMA IF NOT EXISTS maint_lab.ops;
CREATE SCHEMA IF NOT EXISTS maint_lab.security;
CREATE SCHEMA IF NOT EXISTS maint_lab.app;

USE SCHEMA maint_lab.ops;
```

Result: *Statement executed successfully.* for each statement.

Step 1.5 to 1.6: Do a check

> **Why:** A check after each lab finds a problem early, when it is easy to correct.

Do:

1. Highlight **Step 1.5** only. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
2. Highlight **Step 1.6** only. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).

```
-- Step 1.5: Do a check. You must see 3 schemas: APP, OPS, SECURITY
-- (plus INFORMATION_SCHEMA and PUBLIC).
SHOW SCHEMAS IN DATABASE maint_lab;

-- Step 1.6: Do a check of the AI setting. The value must be ANY_REGION.
SHOW PARAMETERS LIKE 'CORTEX_ENABLED_CROSS_REGION' IN ACCOUNT;
```

Result: Step 1.5 shows APP, OPS and SECURITY. Step 1.6 shows `ANY_REGION` in the column **value**.

Lab 2: Load the dummy data

Goal: Make the SITES and MAINT_REQUESTS tables. Load 12 sites and 60 requests.

Why: The app and the AI need data. SITES tells which asset each site belongs to. The column ASSET is the column that the security rule uses in Lab 4. MAINT_REQUESTS is the table that users write to in Part 2. **Architecture:** 5

Open the file

Do:

1. Select **Projects » Workspaces**.
2. In the file list of **My Workspace**, select `02_data.sql`. The file opens in the editor.
3. Read the comments in the file. Each step starts with a comment line `-- Step ...`.

Step 2.0 to 2.2: Set the context and make the tables

Why: DESCRIPTION keeps the text as the user typed it. DESCRIPTION_SAFE and AI_SUMMARY are empty now. Cortex AI fills them in Lab 3. We keep the raw text and the safe text in different columns, so that the app can show only the safe text. **Architecture:** 5

Do:

1. In the editor of `02_data.sql`, find **Step 2.0 to Step 2.2**.
2. Highlight all lines of **Step 2.0 to Step 2.2**, from its comment line to the last semicolon.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

```
-- Step 2.0: Set the role and the warehouse for this file.
USE ROLE ACCOUNTADMIN;
USE WAREHOUSE maint_wh;
USE SCHEMA maint_lab.ops;

-- Step 2.1: Create the SITES table. One row for each site.
CREATE OR REPLACE TABLE maint_lab.ops.sites (
  site_id    VARCHAR(10)  NOT NULL,   -- for example WA01
  site_name  VARCHAR(100) NOT NULL,
  asset      VARCHAR(20)  NOT NULL,   -- WAIO, ESCONDIDA or BMA
  commodity  VARCHAR(50)  NOT NULL,
  country    VARCHAR(50)  NOT NULL
);

-- Step 2.2: Create the MAINT_REQUESTS table. One row for each request.
-- DESCRIPTION is the text that the person typed. It can contain names,
-- phone numbers and emails. The last two columns are empty now.
-- Cortex AI fills them in Lab 3.
CREATE OR REPLACE TABLE maint_lab.ops.maint_requests (
  request_id       NUMBER AUTOINCREMENT START 1 INCREMENT 1,
  site_id          VARCHAR(10)   NOT NULL,
  asset            VARCHAR(20)   NOT NULL,
```

```
  equipment_id     VARCHAR(20)   NOT NULL,   -- for example HT-0412
  equipment_type   VARCHAR(50)   NOT NULL,   -- Haul truck, Conveyor, ...
  failure_mode     VARCHAR(50)   NOT NULL,   -- selected from a list
  priority         VARCHAR(2)    NOT NULL,   -- P1 (highest) to P4
  downtime_hours   NUMBER(6,1)   NOT NULL,
  submitted_by     VARCHAR(100)  NOT NULL,   -- Snowflake user name
  submitted_at     TIMESTAMP_NTZ NOT NULL,
  description      VARCHAR(4000) NOT NULL,   -- text as typed (can contain personal data)
  description_safe VARCHAR(4000),            -- AI result: personal data hidden (Lab 3)
  ai_summary       VARCHAR(1000)             -- AI result: short summary (Lab 3)
);
```

Result: *Table SITES successfully created.* and *Table MAINT_REQUESTS successfully created.*

Step 2.3 to 2.4: Load the data

> **Why:** The dummy data has 20 requests for each asset. Some texts contain names, phone numbers and emails, so that you can see AI_REDACT work. Some equipment has repeat failures, so that AI_AGG can find them. **Architecture:** 5

Do:

1. In the editor of `02_data.sql`, find **Step 2.3 and Step 2.4**.
2. Highlight all lines of **Step 2.3 and Step 2.4**, from its comment line to the last semicolon.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

This is the full dummy data: 12 sites and 60 maintenance requests.

```
-- Step 2.3: Load 12 sites.
INSERT INTO maint_lab.ops.sites (site_id, site_name, asset, commodity, country) VALUES
  ('WA01', 'Mt Newman', 'WAIO', 'Iron ore', 'Australia'),
  ('WA02', 'Mining Area C', 'WAIO', 'Iron ore', 'Australia'),
  ('WA03', 'Jimblebar', 'WAIO', 'Iron ore', 'Australia'),
  ('WA04', 'Port Hedland', 'WAIO', 'Iron ore', 'Australia'),
  ('ES01', 'Escondida Pit', 'ESCONDIDA', 'Copper', 'Chile'),
  ('ES02', 'Laguna Seca Concentrator', 'ESCONDIDA', 'Copper', 'Chile'),
  ('ES03', 'Los Colorados Concentrator', 'ESCONDIDA', 'Copper', 'Chile'),
  ('ES04', 'Coloso Port', 'ESCONDIDA', 'Copper', 'Chile'),
  ('BM01', 'Goonyella Riverside', 'BMA', 'Steelmaking coal', 'Australia'),
  ('BM02', 'Peak Downs', 'BMA', 'Steelmaking coal', 'Australia'),
  ('BM03', 'Saraji', 'BMA', 'Steelmaking coal', 'Australia'),
  ('BM04', 'Hay Point', 'BMA', 'Steelmaking coal', 'Australia');

-- Step 2.4: Load 60 maintenance requests.
INSERT INTO maint_lab.ops.maint_requests
  (site_id, asset, equipment_id, equipment_type, failure_mode, priority,
   downtime_hours, submitted_by, submitted_at, description)
VALUES
  ('WA01', 'WAIO', 'HT-0412', 'Haul truck', 'Hydraulics', 'P2', 6.5, 'SEED_DATA', '2026-10-01
06:00:00',
   'Operator Jack Wilson (mobile 0412 555 019) reported a hydraulic oil leak at the left hoist
cylinder of HT-0412. The truck was parked at the go-line. Oil on the ground was cleaned with a
spill kit. Please replace the cylinder seal kit before the next shift.'),
  ('WA01', 'WAIO', 'HT-0418', 'Haul truck', 'Tyres', 'P1', 12.0, 'SEED_DATA', '2026-10-01
16:00:00',
   'Front right tyre on HT-0418 has a deep cut in the sidewall after contact with rock spillage
```

```
on the ramp. Truck stopped for safety. Tyre change crew needed. Contact the shift supervisor Emma
Clarke on emma.clarke@example.com.'),
  ('WA01', 'WAIO', 'EX-0203', 'Excavator', 'Engine', 'P2', 8.0, 'SEED_DATA', '2026-10-02
02:00:00',
   'EX-0203 shows high coolant temperature alarms after 3 hours of work. Radiator fins are
blocked with dust. Clean the radiator and check the fan belt tension.'),
  ('WA01', 'WAIO', 'CV-0110', 'Conveyor', 'Belt or idler', 'P3', 2.0, 'SEED_DATA', '2026-10-02
12:00:00',
   'Noisy idlers on CV-0110 near transfer point 3. Three idlers turn slowly. Replace at the next
planned shutdown.'),
  ('WA01', 'WAIO', 'HT-0412', 'Haul truck', 'Hydraulics', 'P1', 9.0, 'SEED_DATA', '2026-10-02
22:00:00',
   'Second hydraulic leak on HT-0412 this month, again at the left hoist cylinder. Maintenance
planner Ravi Kumar (0400 123 777) asks for a root cause check of the hose routing.'),
  ('WA02', 'WAIO', 'SH-0301', 'Shovel', 'Electrical', 'P1', 14.0, 'SEED_DATA', '2026-10-02
17:00:00',
   'Electric shovel SH-0301 tripped three times on the night shift. Main breaker trips when the
swing motor starts. Electrician Sam Taylor checked the cables and found heat damage at the slip
ring.'),
  ('WA02', 'WAIO', 'HT-0522', 'Haul truck', 'Brakes', 'P1', 5.0, 'SEED_DATA', '2026-10-03
03:00:00',
   'Brake temperature warning on HT-0522 on the long downhill haul road. The operator parked the
truck. Brake pads look worn. Do a full brake inspection before the truck goes back into
service.'),
  ('WA02', 'WAIO', 'DR-0105', 'Drill rig', 'Hydraulics', 'P3', 3.0, 'SEED_DATA', '2026-10-03
13:00:00',
   'DR-0105 rotation head is slow. Hydraulic pressure is lower than normal. Check the pump and
the filters.'),
  ('WA02', 'WAIO', 'CV-0215', 'Conveyor', 'Belt or idler', 'P2', 7.5, 'SEED_DATA', '2026-10-03
23:00:00',
   'Belt on CV-0215 runs off-centre at the tail pulley. Spillage builds up under the belt. Adjust
the training idlers and clean the spillage.'),
  ('WA02', 'WAIO', 'SH-0301', 'Shovel', 'Electrical', 'P2', 4.0, 'SEED_DATA', '2026-10-04
09:00:00',
   'SH-0301 swing motor fault again after the slip ring repair. Fault code SW-112. Please send
the OEM technician.'),
  ('WA03', 'WAIO', 'HT-0611', 'Haul truck', 'Engine', 'P2', 10.0, 'SEED_DATA', '2026-10-04
04:00:00',
   'HT-0611 has low engine power on the ramp and black smoke. Turbocharger noise. Operator Mia
Brown, phone 0455 220 981, says the problem started after the fuel change.'),
  ('WA03', 'WAIO', 'CR-0301', 'Crusher', 'Bearing', 'P1', 16.0, 'SEED_DATA', '2026-10-04
14:00:00',
   'Primary crusher CR-0301 eccentric bearing temperature is high, 92 degrees. Crusher stopped.
Check lubrication flow and the oil cooler.'),
  ('WA03', 'WAIO', 'CR-0301', 'Crusher', 'Lubrication', 'P2', 6.0, 'SEED_DATA', '2026-10-05
00:00:00',
   'Lube oil pump on CR-0301 makes noise and the pressure goes up and down. Replace the pump
coupling.'),
  ('WA03', 'WAIO', 'DZ-0204', 'Dozer', 'Tyres', 'P4', 1.0, 'SEED_DATA', '2026-10-05 10:00:00',
   'Dozer DZ-0204 track tension low on the right side. Adjust at next service.'),
  ('WA03', 'WAIO', 'CV-0330', 'Conveyor', 'Sensor or control', 'P3', 2.5, 'SEED_DATA', '2026-10-
05 20:00:00',
   'Belt rip detector on CV-0330 gives false alarms. The conveyor stops each time. Calibrate the
sensor.'),
  ('WA04', 'WAIO', 'SL-0101', 'Ship loader', 'Structural crack', 'P1', 20.0, 'SEED_DATA', '2026-
10-05 15:00:00',
   'Crack found on the boom of ship loader SL-0101 during the weekly inspection. Ship loading
stopped. Structural engineer must inspect. Report by inspector Noah Martin,
noah.martin@example.com.'),
  ('WA04', 'WAIO', 'SR-0202', 'Stacker reclaimer', 'Belt or idler', 'P2', 5.5, 'SEED_DATA',
'2026-10-06 01:00:00',
```

```
   'Stacker reclaimer SR-0202 boom conveyor belt has a damaged splice. Replace the splice before
the belt tears.'),
  ('WA04', 'WAIO', 'CV-0401', 'Conveyor', 'Electrical', 'P2', 4.0, 'SEED_DATA', '2026-10-06
11:00:00',
   'Drive motor of CV-0401 trips on overload when the belt starts with load. Check the soft
starter settings.'),
  ('WA04', 'WAIO', 'SL-0101', 'Ship loader', 'Hydraulics', 'P3', 3.0, 'SEED_DATA', '2026-10-06
21:00:00',
   'Luffing cylinder of SL-0101 has a small oil leak. Monitor and repair during the next planned
stop.'),
  ('WA04', 'WAIO', 'SR-0202', 'Stacker reclaimer', 'Sensor or control', 'P3', 1.5, 'SEED_DATA',
'2026-10-07 07:00:00',
   'Anti-collision sensor on SR-0202 is dirty with dust and gives warnings. Clean the sensor
lens.'),
  ('ES01', 'ESCONDIDA', 'HT-1102', 'Haul truck', 'Tyres', 'P1', 11.0, 'SEED_DATA', '2026-10-07
02:00:00',
   'Haul truck HT-1102 rear tyre pressure dropped fast on the main ramp. Truck stopped. Operator
Diego Morales (+56 9 8765 4321) reported a possible rock cut.'),
  ('ES01', 'ESCONDIDA', 'SH-1201', 'Shovel', 'Electrical', 'P1', 13.0, 'SEED_DATA', '2026-10-07
12:00:00',
   'Electric shovel SH-1201 lost power. The trailing cable has damage from a dozer track. Replace
the cable section and test the insulation.'),
  ('ES01', 'ESCONDIDA', 'EX-1301', 'Excavator', 'Hydraulics', 'P2', 6.0, 'SEED_DATA', '2026-10-07
22:00:00',
   'EX-1301 boom drifts down when the lever is in neutral. Check the boom holding valve.'),
  ('ES01', 'ESCONDIDA', 'DR-1105', 'Drill rig', 'Engine', 'P3', 4.0, 'SEED_DATA', '2026-10-08
08:00:00',
   'Drill DR-1105 engine starts with difficulty in the cold morning at high altitude. Check the
glow plugs and the battery.'),
  ('ES01', 'ESCONDIDA', 'HT-1102', 'Haul truck', 'Tyres', 'P2', 7.0, 'SEED_DATA', '2026-10-08
18:00:00',
   'HT-1102 has a second tyre problem this week, front left. Tyre wear is not even. Check the
suspension and the wheel alignment.'),
  ('ES02', 'ESCONDIDA', 'ML-2101', 'Grinding mill', 'Bearing', 'P1', 18.0, 'SEED_DATA', '2026-10-
08 13:00:00',
   'SAG mill ML-2101 trunnion bearing temperature is high. Mill throughput reduced. Lubrication
engineer Valentina Rojas, valentina.rojas@example.com, asks for an oil analysis.'),
  ('ES02', 'ESCONDIDA', 'CV-2203', 'Conveyor', 'Belt or idler', 'P2', 5.0, 'SEED_DATA', '2026-10-
08 23:00:00',
   'Ore feed conveyor CV-2203 belt is worn near the skirt boards. Replace the skirt rubber and
inspect the belt thickness.'),
  ('ES02', 'ESCONDIDA', 'PU-2301', 'Slurry pump', 'Bearing', 'P2', 6.0, 'SEED_DATA', '2026-10-09
09:00:00',
   'Slurry pump PU-2301 vibration is high. The bearing makes noise. Change over to the standby
pump and replace the bearing.'),
  ('ES02', 'ESCONDIDA', 'ML-2101', 'Grinding mill', 'Lubrication', 'P2', 4.5, 'SEED_DATA', '2026-
10-09 19:00:00',
   'Lube system filter on ML-2101 has high differential pressure. Change the filter element.'),
  ('ES02', 'ESCONDIDA', 'FL-2401', 'Flotation cell', 'Sensor or control', 'P3', 2.0, 'SEED_DATA',
'2026-10-10 05:00:00',
   'Level sensor on flotation cell FL-2401 reads wrong values. Recalibrate the sensor.'),
  ('ES03', 'ESCONDIDA', 'ML-3101', 'Grinding mill', 'Bearing', 'P2', 8.0, 'SEED_DATA', '2026-10-
10 00:00:00',
   'Ball mill ML-3101 pinion bearing temperature goes up slowly. Check alignment and
lubrication.'),
  ('ES03', 'ESCONDIDA', 'PU-3302', 'Slurry pump', 'Bearing', 'P1', 9.0, 'SEED_DATA', '2026-10-10
10:00:00',
   'Slurry pump PU-3302 failed. Bearing seized. Standby pump also has high vibration. Mechanic
Felipe Soto (+56 9 1234 5678) asks for spare bearings from the warehouse.'),
  ('ES03', 'ESCONDIDA', 'CV-3104', 'Conveyor', 'Electrical', 'P3', 3.0, 'SEED_DATA', '2026-10-10
20:00:00',
```

```
    'Pull-wire switch on CV-3104 is damaged and stops the belt. Replace the switch.'),
  ('ES03', 'ESCONDIDA', 'CR-3201', 'Crusher', 'Structural crack', 'P2', 10.0, 'SEED_DATA', '2026-
10-11 06:00:00',
    'Pebble crusher CR-3201 has a crack in the mainframe near the feed opening. Do a non-
destructive test.'),
  ('ES03', 'ESCONDIDA', 'PU-3302', 'Slurry pump', 'Lubrication', 'P3', 2.0, 'SEED_DATA', '2026-
10-11 16:00:00',
    'Grease lines to the PU-3302 bearings are blocked. Clean the lines and check the auto-lube
unit.'),
  ('ES04', 'ESCONDIDA', 'SL-4101', 'Ship loader', 'Electrical', 'P2', 6.0, 'SEED_DATA', '2026-10-
11 11:00:00',
    'Concentrate ship loader SL-4101 travel motor stops in high humidity. Check the cable
connectors for corrosion.'),
  ('ES04', 'ESCONDIDA', 'CV-4202', 'Conveyor', 'Belt or idler', 'P3', 3.0, 'SEED_DATA', '2026-10-
11 21:00:00',
    'Covered conveyor CV-4202 has seized return idlers. Replace four idlers.'),
  ('ES04', 'ESCONDIDA', 'PU-4301', 'Slurry pump', 'Bearing', 'P2', 5.0, 'SEED_DATA', '2026-10-12
07:00:00',
    'Pipeline booster pump PU-4301 bearing temperature is high. Shift lead Camila Fuentes,
camila.fuentes@example.com, reports a smell of hot oil.'),
  ('ES04', 'ESCONDIDA', 'SL-4101', 'Ship loader', 'Structural crack', 'P3', 2.0, 'SEED_DATA',
'2026-10-12 17:00:00',
    'Small corrosion crack on the SL-4101 walkway support. Repair at the next shutdown.'),
  ('ES04', 'ESCONDIDA', 'CV-4202', 'Conveyor', 'Sensor or control', 'P4', 1.0, 'SEED_DATA',
'2026-10-13 03:00:00',
    'Belt speed sensor on CV-4202 needs cleaning. No stop.'),
  ('BM01', 'BMA', 'DL-0101', 'Dragline', 'Structural crack', 'P1', 22.0, 'SEED_DATA', '2026-10-12
22:00:00',
    'Dragline DL-0101 boom chord has a crack near the lower lug. Dragline stopped. Structural
engineer Liam O''Brien (0423 678 901) will inspect today.'),
  ('BM01', 'BMA', 'HT-5104', 'Haul truck', 'Brakes', 'P2', 5.0, 'SEED_DATA', '2026-10-13
08:00:00',
    'HT-5104 retarder grid overheats on the long ramp. Check the cooling fan of the retarder.'),
  ('BM01', 'BMA', 'DZ-5201', 'Dozer', 'Hydraulics', 'P3', 3.0, 'SEED_DATA', '2026-10-13
18:00:00',
    'Dozer DZ-5201 ripper cylinder leaks oil. Replace the rod seal.'),
  ('BM01', 'BMA', 'CV-5301', 'Conveyor', 'Belt or idler', 'P2', 6.0, 'SEED_DATA', '2026-10-14
04:00:00',
    'Overland conveyor CV-5301 belt has a rip of 30 cm. Belt splicing crew needed.'),
  ('BM01', 'BMA', 'DL-0101', 'Dragline', 'Electrical', 'P2', 7.0, 'SEED_DATA', '2026-10-14
14:00:00',
    'Dragline DL-0101 hoist motor brush wear is high. Replace the brushes and clean the
commutator.'),
  ('BM02', 'BMA', 'HT-5211', 'Haul truck', 'Engine', 'P2', 8.0, 'SEED_DATA', '2026-10-14
09:00:00',
    'HT-5211 engine oil pressure is low at idle. Operator Chloe Nguyen, chloe.nguyen@example.com,
says the warning light came on twice.'),
  ('BM02', 'BMA', 'EX-5302', 'Excavator', 'Hydraulics', 'P1', 10.0, 'SEED_DATA', '2026-10-14
19:00:00',
    'Excavator EX-5302 main hydraulic pump makes a loud noise and the bucket is slow. Stop and
check the pump before more damage.'),
  ('BM02', 'BMA', 'DR-5401', 'Drill rig', 'Sensor or control', 'P3', 2.0, 'SEED_DATA', '2026-10-
15 05:00:00',
    'GPS guidance on drill DR-5401 loses signal near the high wall. Check the antenna.'),
  ('BM02', 'BMA', 'HT-5211', 'Haul truck', 'Engine', 'P1', 12.0, 'SEED_DATA', '2026-10-15
15:00:00',
    'HT-5211 stopped on the haul road with engine overheating. Coolant hose burst. Second engine
problem this month on this truck.'),
  ('BM02', 'BMA', 'CV-5501', 'Conveyor', 'Electrical', 'P3', 3.0, 'SEED_DATA', '2026-10-16
01:00:00',
    'Conveyor CV-5501 motor cooling fan is noisy. Replace the fan.'),
```

```
  ('BM03', 'BMA', 'DL-0301', 'Dragline', 'Electrical', 'P1', 15.0, 'SEED_DATA', '2026-10-15
20:00:00',
   'Dragline DL-0301 swing motor fault, fault code DS-410. Electrician Jack Thompson (0499 111
222) found a damaged cable at the house slip ring.'),
  ('BM03', 'BMA', 'HT-5306', 'Haul truck', 'Tyres', 'P2', 6.0, 'SEED_DATA', '2026-10-16
06:00:00',
   'HT-5306 tyre has a cut after driving over spillage at the dump. Clean the dump area and
change the tyre.'),
  ('BM03', 'BMA', 'CR-5601', 'Crusher', 'Bearing', 'P2', 9.0, 'SEED_DATA', '2026-10-16 16:00:00',
   'Sizer CR-5601 bearing is noisy. Vibration is above the alarm limit. Plan a bearing change.'),
  ('BM03', 'BMA', 'DZ-5303', 'Dozer', 'Engine', 'P3', 4.0, 'SEED_DATA', '2026-10-17 02:00:00',
   'Dozer DZ-5303 air filter blocks fast in dusty conditions. Change to the heavy duty filter.'),
  ('BM03', 'BMA', 'DL-0301', 'Dragline', 'Electrical', 'P2', 5.0, 'SEED_DATA', '2026-10-17
12:00:00',
   'DL-0301 swing motor fault again after the cable repair. Check the motor insulation.'),
  ('BM04', 'BMA', 'SL-6101', 'Ship loader', 'Hydraulics', 'P2', 6.0, 'SEED_DATA', '2026-10-17
07:00:00',
   'Ship loader SL-6101 slew hydraulic motor leaks oil onto the wharf. Contain the spill and
replace the motor seal. Contact wharf supervisor Olivia Smith, olivia.smith@example.com.'),
  ('BM04', 'BMA', 'SR-6201', 'Stacker reclaimer', 'Structural crack', 'P1', 19.0, 'SEED_DATA',
'2026-10-17 17:00:00',
   'Crack in the bucket wheel boom of stacker reclaimer SR-6201. Machine stopped. Engineer
inspection needed today.'),
  ('BM04', 'BMA', 'CV-6301', 'Conveyor', 'Belt or idler', 'P2', 5.0, 'SEED_DATA', '2026-10-18
03:00:00',
   'Yard conveyor CV-6301 has carryback coal on the return belt. Belt scraper is worn. Replace
the scraper blade.'),
  ('BM04', 'BMA', 'SR-6201', 'Stacker reclaimer', 'Electrical', 'P3', 3.0, 'SEED_DATA', '2026-10-
18 13:00:00',
   'SR-6201 long travel limit switch is damaged. Replace the switch.'),
  ('BM04', 'BMA', 'CV-6301', 'Conveyor', 'Sensor or control', 'P4', 1.0, 'SEED_DATA', '2026-10-18
23:00:00',
   'Weightometer on CV-6301 needs calibration. No stop.');
```

Result: Two results: *number of rows inserted* = 12, then 60.

Step 2.5 to 2.6: Do a check

Do:

1. Highlight **Step 2.5**. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
2. Highlight **Step 2.6**. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).

```
-- Step 2.5: Do a check. You must see 12 sites and 60 requests.
SELECT 'sites' AS table_name, COUNT(*) AS row_count FROM maint_lab.ops.sites
UNION ALL
SELECT 'maint_requests', COUNT(*) FROM maint_lab.ops.maint_requests;

-- Step 2.6: Look at the requests and the downtime for each asset.
SELECT asset,
       COUNT(*)                AS requests,
       SUM(downtime_hours)     AS downtime_hours,
       COUNT_IF(priority = 'P1') AS p1_requests
FROM maint_lab.ops.maint_requests
GROUP BY asset
ORDER BY asset;
```

Result: Step 2.5: sites = 12, maint_requests = 60. Step 2.6: 3 assets with 20 requests each. Total downtime is 417 hours.

Lab 3: Use Cortex AI on the requests

Goal: Use the 5 Cortex AI functions in SQL. Save the AI results in the table. Make an insight view.

Why: Cortex AI functions run inside SQL, next to the data. You do not move data to another AI tool. In Part 2 the app calls the same functions. Here you learn what each function does. **Architecture:** 3 5

Function	What it does	Business use here
`AI_REDACT`	Finds personal data and puts a tag in its place.	Share requests with no names, phones or emails.
`AI_SUMMARIZE`	Makes one long text short.	A one-line view of each request.
`AI_SUMMARIZE_AGG`	Makes one summary from many rows (like SUM, for text).	One summary for each asset.
`AI_AGG`	Reads many rows and follows your instruction.	Top repeat problems and actions.
`SPLIT_TEXT_RECURSIVE_CHARACTER`	Cuts a long text into small chunks.	Prepare long notes and manuals for AI search.

Open the file and set the context

Do:

1. Select **Projects » Workspaces**.
2. In the file list of **My Workspace**, select `03_cortex_ai.sql`. The file opens in the editor.
3. Read the comments in the file. Each step starts with a comment line `-- Step ...`.

Do:

1. In the editor of `03_cortex_ai.sql`, find **Step 3.0**.
2. Highlight all lines of **Step 3.0**, from its comment line to the last semicolon.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

```
-- Step 3.0: Set the role and the warehouse for this file.
USE ROLE ACCOUNTADMIN;
USE WAREHOUSE maint_wh;
USE SCHEMA maint_lab.ops;
```

Step 3.1: AI_REDACT

Why: Maintenance text often contains names and phone numbers. Before we share the text or give it to other AI functions, we hide the personal data. This protects privacy. **Architecture:** 3

Do:

1. In the editor of `03_cortex_ai.sql`, find **Step 3.1**.
2. Highlight all lines of **Step 3.1**, from its comment line to the last semicolon.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

```
-- ------------------------------------------------------------------
-- Step 3.1: AI_REDACT. Hide personal data (names, phone numbers, emails).
-- Result: the same text, with [NAME], [PHONE_NUMBER] and [EMAIL] tags.
-- ------------------------------------------------------------------
SELECT request_id,
       description,
       AI_REDACT(description) AS description_safe
FROM maint_lab.ops.maint_requests
WHERE request_id IN (1, 2, 5, 16, 21)
ORDER BY request_id;
```

Result: 5 rows. In DESCRIPTION_SAFE, names show as `[NAME]`, phone numbers as `[PHONE_NUMBER]` and emails as an email tag. The other words do not change.

Step 3.2: AI_SUMMARIZE

> **Why:** A planner reads many requests each day. A short summary saves time. In the app, the summary shows next to each request. **Architecture:** 3

Do:

1. In the editor of `03_cortex_ai.sql`, find **Step 3.2**.
2. Highlight all lines of **Step 3.2**, from its comment line to the last semicolon.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

```
-- ------------------------------------------------------------------
-- Step 3.2: AI_SUMMARIZE. Make one short summary of one long text.
-- Result: a short summary for each request.
-- NOTE: AI_SUMMARIZE is a Preview function. A text of 110 characters
--       or less comes back with no change.
-- ------------------------------------------------------------------
SELECT request_id,
       LENGTH(description)       AS characters,
       AI_SUMMARIZE(description) AS ai_summary
FROM maint_lab.ops.maint_requests
WHERE request_id IN (1, 6, 16, 26, 41)
ORDER BY request_id;
```

Result: 5 rows. AI_SUMMARY is shorter than the description and keeps the main facts.

> **NOTE:** AI_SUMMARIZE is a Preview function. An AI answer is not always the same. Your text can be different from the text of other participants.

Step 3.3 to 3.5: Save the AI results in the table

> **Why:** We run the AI one time and save the result. Then the app reads the result fast, and we do not pay for the same AI work again. Step 3.4 summarises the **safe** text, so the summary also has no personal data (Figure 2). **Architecture:** 3 5

Do:

1. In the editor of `03_cortex_ai.sql`, find **Step 3.3**.
2. Highlight all lines of **Step 3.3**, from its comment line to the last semicolon.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

Do:

1. Wait until Step 3.3 stops (about 1 minute).
2. Highlight **Step 3.4**. Push Ctrl + Enter (Windows) or Cmd + Return (Mac). Wait until it stops.
3. Highlight **Step 3.5**. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).

```
-- ------------------------------------------------------------------
-- Step 3.3: Save the safe text in the table (all 60 rows).
-- Wait about 1 minute.
-- ------------------------------------------------------------------
UPDATE maint_lab.ops.maint_requests
SET description_safe = AI_REDACT(description)
WHERE description_safe IS NULL;

-- ------------------------------------------------------------------
-- Step 3.4: Save a summary of the SAFE text (all 60 rows).
-- We summarise the safe text, so the summary has no personal data.
-- ------------------------------------------------------------------
UPDATE maint_lab.ops.maint_requests
SET ai_summary = AI_SUMMARIZE(description_safe)
WHERE ai_summary IS NULL;

-- Step 3.5: Do a check. All 60 rows must have a safe text and a summary.
SELECT COUNT(*)                     AS total_rows,
       COUNT(description_safe)      AS rows_with_safe_text,
       COUNT(ai_summary)            AS rows_with_summary,
       COUNT_IF(description_safe LIKE '%[NAME]%') AS rows_with_hidden_names
FROM maint_lab.ops.maint_requests;
```

Result: Step 3.3 and Step 3.4: *number of rows updated* = 60. Step 3.5: total_rows = 60, rows_with_safe_text = 60, rows_with_summary = 60, rows_with_hidden_names is 10 or more.

Step 3.6: AI_SUMMARIZE_AGG

> **Why:** A manager wants one view of an asset, not 20 requests. AI_SUMMARIZE_AGG reads all rows of a group and writes one summary. It works like SUM, but for text. **Architecture:** 3

Do:

1. In the editor of `03_cortex_ai.sql`, find **Step 3.6**.
2. Highlight all lines of **Step 3.6**, from its comment line to the last semicolon.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

```
-- ------------------------------------------------------------------
-- Step 3.6: AI_SUMMARIZE_AGG. One summary from many rows.
-- Result: one summary for each asset.
-- ------------------------------------------------------------------
SELECT asset,
       COUNT(*)                                AS requests,
       AI_SUMMARIZE_AGG(description_safe)      AS asset_summary
FROM maint_lab.ops.maint_requests
GROUP BY asset
ORDER BY asset;
```

Result: 3 rows (BMA, ESCONDIDA, WAIO). Each row has one summary of 20 requests.

Step 3.7: AI_AGG

Why: AI_AGG reads many rows and follows your instruction. Here it acts as a reliability engineer and finds repeat failures, for example HT-0412 with two hydraulic leaks. The app uses the same idea in Lab 7. **Architecture:** 3

Do:

1. In the editor of `03_cortex_ai.sql`, find **Step 3.7**.
2. Highlight all lines of **Step 3.7**, from its comment line to the last semicolon.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

```
-- ------------------------------------------------------------------
-- Step 3.7: AI_AGG. Give your own instruction for many rows.
-- Result: the top 3 repeat problems and one action for each asset.
-- ------------------------------------------------------------------
SELECT asset,
       AI_AGG(
         CONCAT(equipment_id, ' (', equipment_type, '): ', description_safe),
         'You are a maintenance reliability engineer at a mining company. Find the top 3 repeat
equipment problems in these requests. For each problem, give the equipment ID and one action. Use
short bullet points.'
       ) AS top_repeat_problems
FROM maint_lab.ops.maint_requests
GROUP BY asset
ORDER BY asset;
```

Result: 3 rows. Each row has 3 bullet points with an equipment ID and an action.

Step 3.8: SPLIT_TEXT_RECURSIVE_CHARACTER

Why: Long texts (shift handover notes, manuals) are too big for one AI call or one search result. We cut them into small chunks with a small overlap. This is the first step of AI search on documents. **Architecture:** 3

Do:

1. In the editor of `03_cortex_ai.sql`, find **Step 3.8**.
2. Highlight all lines of **Step 3.8**, from its comment line to the last semicolon.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

```
-- ------------------------------------------------------------------
-- Step 3.8: SPLIT_TEXT_RECURSIVE_CHARACTER. Cut a long text into chunks.
-- Long documents (shift handover notes, manuals) are cut into small
-- parts before AI search. Result: one row for each chunk.
-- ------------------------------------------------------------------
SELECT c.index       AS chunk_number,
       c.value::STRING AS chunk_text
FROM TABLE(FLATTEN(input => SNOWFLAKE.CORTEX.SPLIT_TEXT_RECURSIVE_CHARACTER(
  'Shift handover, night shift, Mt Newman. HT-0412 is parked at the go-line with a hydraulic leak
at the left hoist cylinder. '
  || 'The seal kit is in the warehouse, bin 14. CV-0110 idlers at transfer point 3 are noisy;
replace them at the planned shutdown. '
  || 'EX-0203 had coolant temperature alarms; the radiator was cleaned at 02:00 and the alarms
stopped. '
  || 'Safety: one tyre cut on HT-0418 from rock spillage on the main ramp; the ramp was cleaned.
No injuries.',
  'none', 200, 30))) c;
```

Result: 3 or more rows. Each chunk has 200 characters or less. The end of one chunk and the start of the next chunk can be the same (overlap).

Step 3.9 to 3.10: Make the insight view

Why: The view calculates the numbers for each site: requests, downtime, P1 requests and the top failure mode. The app reads the view, so the calculation is in one place. **Architecture:** 5

Do:

1. In the editor of `03_cortex_ai.sql`, find **Step 3.9**.
2. Highlight all lines of **Step 3.9**, from its comment line to the last semicolon.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

Do:

1. Highlight **Step 3.10**. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).

```
-- ------------------------------------------------------------------
-- Step 3.9: Make an insight view. The Streamlit app uses it in Part 2.
-- ------------------------------------------------------------------
CREATE OR REPLACE VIEW maint_lab.ops.v_site_insight AS
```

```
SELECT s.asset,
       s.site_id,
       s.site_name,
       COUNT(r.request_id)               AS requests,
       COALESCE(SUM(r.downtime_hours), 0) AS downtime_hours,
       COUNT_IF(r.priority = 'P1')       AS p1_requests,
       MODE(r.failure_mode)              AS top_failure_mode
FROM maint_lab.ops.sites s
LEFT JOIN maint_lab.ops.maint_requests r
  ON r.site_id = s.site_id
GROUP BY s.asset, s.site_id, s.site_name;

-- Step 3.10: Look at the sites with the most downtime.
SELECT *
FROM maint_lab.ops.v_site_insight
ORDER BY downtime_hours DESC;
```

Result: Step 3.9: *View V_SITE_INSIGHT successfully created.* Step 3.10: 12 rows, the site with the most downtime first.

Lab 4: Users, groups and row-level security

Goal: Make the groups (roles) and 4 demo users. Make a row access policy. Each user sees only the rows of his or her groups.

Why: This is the security requirement. The WAIO team must not see Escondida or BMA requests. Head office sees all. One policy on the table does this for all tools: SQL, views, AI functions and the app. **Architecture:** 1 4 6

How it works:

1. A Snowflake role is a group. Example: role `MAINT_WAIO` is the WAIO group.
2. Table `ROLE_ASSET_MAP` tells which group sees which asset.
3. Procedure `SYNC_USER_ASSETS` reads the real role grants and writes "user to asset" rows into `USER_ASSET_MAP`.
4. The row access policy reads `USER_ASSET_MAP` with `CURRENT_USER()`.

NOTE: Why does the policy not use the role directly? The Streamlit app runs with the role of the app owner. In the app, the current role is always the owner role, but the current user is the real person. Thus the policy uses the user, and the groups give the user his or her assets.

Open the file and set the context

Do:

1. Select **Projects » Workspaces**.
2. In the file list of **My Workspace**, select `04_security.sql`. The file opens in the editor.
3. Read the comments in the file. Each step starts with a comment line `-- Step ...`.

Do:

1. In the editor of `04_security.sql`, find **Step 4.0**.
2. Highlight all lines of **Step 4.0**, from its comment line to the last semicolon.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

```
-- Step 4.0: Set the role and the warehouse for this file.
USE ROLE ACCOUNTADMIN;
USE WAREHOUSE maint_wh;
```

Step 4.1 to 4.2: Make the group roles

> **Why:** Each asset team gets one role. All group roles get the parent role MAINT_APP_USER. Then we give the common access (warehouse, database, app) one time, to one role. **Architecture:** 1

Do:

1. In the editor of `04_security.sql`, find **Step 4.1 and Step 4.2**.
2. Highlight all lines of **Step 4.1 and Step 4.2**, from its comment line to the last semicolon.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

```
-- ----------------------------------------------------------------------
-- Step 4.1: Create the group roles.
--   MAINT_APP_USER  = common access for all app users
--   MAINT_HQ        = group head office, sees all assets
--   MAINT_WAIO, MAINT_ESCONDIDA, MAINT_BMA = one asset each
-- ----------------------------------------------------------------------
CREATE ROLE IF NOT EXISTS maint_app_user  COMMENT = 'Common access for Asset Care app users';
CREATE ROLE IF NOT EXISTS maint_hq        COMMENT = 'Group: head office, all assets';
CREATE ROLE IF NOT EXISTS maint_waio      COMMENT = 'Group: Western Australia Iron Ore';
CREATE ROLE IF NOT EXISTS maint_escondida COMMENT = 'Group: Escondida';
CREATE ROLE IF NOT EXISTS maint_bma       COMMENT = 'Group: BHP Mitsubishi Alliance';

-- Each group role gets the common access.
GRANT ROLE maint_app_user TO ROLE maint_hq;
GRANT ROLE maint_app_user TO ROLE maint_waio;
GRANT ROLE maint_app_user TO ROLE maint_escondida;
GRANT ROLE maint_app_user TO ROLE maint_bma;

-- Step 4.2: Give the common role access to the warehouse and the data.
GRANT USAGE  ON WAREHOUSE maint_wh                   TO ROLE maint_app_user;
GRANT USAGE  ON DATABASE  maint_lab                  TO ROLE maint_app_user;
GRANT USAGE  ON SCHEMA    maint_lab.ops              TO ROLE maint_app_user;
GRANT USAGE  ON SCHEMA    maint_lab.app              TO ROLE maint_app_user;
GRANT SELECT ON ALL TABLES IN SCHEMA maint_lab.ops    TO ROLE maint_app_user;
GRANT SELECT ON ALL VIEWS  IN SCHEMA maint_lab.ops    TO ROLE maint_app_user;
```

Result: *Statement executed successfully.* for each statement.

Step 4.3 to 4.4: Make the demo users and put them in groups

> **Why:** The demo users act as the asset planners and the head office analyst. In Part 2 you log in as each user and see the effect of the policy. **Architecture:** 1

Do:

1. In the editor of `04_security.sql`, find **Step 4.3 and Step 4.4**.
2. Highlight all lines of **Step 4.3 and Step 4.4**, from its comment line to the last semicolon.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

```
-- ------------------------------------------------------------------
-- Step 4.3: Create 4 demo users. Each user is in one group.
-- CAUTION: This is a training password. Do not use it in production.
-- ------------------------------------------------------------------
CREATE USER IF NOT EXISTS olivia_waio
  PASSWORD = 'AssetCare#Lab2026' MUST_CHANGE_PASSWORD = FALSE TYPE = PERSON
  DEFAULT_ROLE = maint_waio DEFAULT_WAREHOUSE = maint_wh
  COMMENT = 'Demo user: WAIO maintenance planner';
CREATE USER IF NOT EXISTS diego_esc
  PASSWORD = 'AssetCare#Lab2026' MUST_CHANGE_PASSWORD = FALSE TYPE = PERSON
  DEFAULT_ROLE = maint_escondida DEFAULT_WAREHOUSE = maint_wh
  COMMENT = 'Demo user: Escondida maintenance planner';
CREATE USER IF NOT EXISTS liam_bma
  PASSWORD = 'AssetCare#Lab2026' MUST_CHANGE_PASSWORD = FALSE TYPE = PERSON
  DEFAULT_ROLE = maint_bma DEFAULT_WAREHOUSE = maint_wh
  COMMENT = 'Demo user: BMA maintenance planner';
CREATE USER IF NOT EXISTS priya_hq
  PASSWORD = 'AssetCare#Lab2026' MUST_CHANGE_PASSWORD = FALSE TYPE = PERSON
  DEFAULT_ROLE = maint_hq DEFAULT_WAREHOUSE = maint_wh
  COMMENT = 'Demo user: head office reliability analyst';

-- Step 4.4: Put each user in a group (grant the role to the user).
GRANT ROLE maint_waio      TO USER olivia_waio;
GRANT ROLE maint_escondida TO USER diego_esc;
GRANT ROLE maint_bma       TO USER liam_bma;
GRANT ROLE maint_hq        TO USER priya_hq;
```

Result: *User OLIVIA_WAIO successfully created.* (and the same for the other 3 users). Then *Statement executed successfully.*

Step 4.5: Put your user in the HQ group

> **Why:** After Step 4.10 the policy filters all queries, also your queries. If your user is not in a group, you see 0 rows. **Architecture:** 1

Do:

1. In the editor of `04_security.sql`, find **Step 4.5**.
2. Highlight the full block, from the comment line **Step 4.5** to the last line `$$;`.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

```
-- Step 4.5: Put YOUR user in the HQ group. Then you can see all rows.
-- CAUTION: Do this step. If you do not, you see 0 rows after Step 4.10.
EXECUTE IMMEDIATE $$
BEGIN
  EXECUTE IMMEDIATE 'GRANT ROLE maint_hq TO USER "' || CURRENT_USER() || '"';
  RETURN 'MAINT_HQ is granted to ' || CURRENT_USER();
END;
$$;
```

Result: *MAINT_HQ is granted to* followed by your user name.

Step 4.6 to 4.7: Make the access tables

Why: The rule "which group sees which asset" is data, not code. To change the rule, you change a row. You do not change the policy. **Architecture:** 6

Do:

1. In the editor of `04_security.sql`, find **Step 4.6 and Step 4.7**.
2. Highlight all lines of **Step 4.6 and Step 4.7**, from its comment line to the last semicolon.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

```
-- ------------------------------------------------------------------
-- Step 4.6: Create the access rule table: which group sees which asset.
-- 'ALL' means all assets.
-- ------------------------------------------------------------------
CREATE OR REPLACE TABLE maint_lab.security.role_asset_map (
  role_name VARCHAR(100) NOT NULL,
  asset     VARCHAR(20)  NOT NULL
);

INSERT INTO maint_lab.security.role_asset_map (role_name, asset) VALUES
  ('MAINT_HQ',        'ALL'),
  ('MAINT_WAIO',      'WAIO'),
  ('MAINT_ESCONDIDA', 'ESCONDIDA'),
  ('MAINT_BMA',       'BMA');

-- Step 4.7: Create the table that the policy reads: user to asset.
-- A procedure fills it from the real role grants (next step).
CREATE OR REPLACE TABLE maint_lab.security.user_asset_map (
  user_name VARCHAR(100) NOT NULL,
  asset     VARCHAR(20)  NOT NULL,
  via_role  VARCHAR(100) NOT NULL,
  synced_at TIMESTAMP_NTZ DEFAULT CURRENT_TIMESTAMP()
);
```

Result: Two tables are created. The INSERT gives *number of rows inserted* = 4.

Step 4.8: Make the sync procedure

Why: Administrators manage access with normal Snowflake role grants. The procedure copies the grants into USER_ASSET_MAP. Thus Snowflake user management stays the one source of truth. **Architecture:** 6

Do:

1. In the editor of `04_security.sql`, find **Step 4.8**.
2. Highlight the full block, from the comment line **Step 4.8** to the last line `$$;`.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

```
-- ------------------------------------------------------------------
-- Step 4.8: Create the sync procedure. It reads "who has which group
-- role" from Snowflake and writes the result to USER_ASSET_MAP.
-- Run it again each time you add or remove a user from a group.
-- ------------------------------------------------------------------
CREATE OR REPLACE PROCEDURE maint_lab.security.sync_user_assets()
RETURNS VARCHAR
LANGUAGE SQL
EXECUTE AS CALLER
AS
$$
DECLARE
  c1 CURSOR FOR SELECT role_name, asset FROM maint_lab.security.role_asset_map;
  total_rows INTEGER DEFAULT 0;
BEGIN
  TRUNCATE TABLE maint_lab.security.user_asset_map;
  FOR rec IN c1 DO
    LET v_role  VARCHAR := rec.role_name;
    LET v_asset VARCHAR := rec.asset;
    -- List all users and roles that have this group role.
    EXECUTE IMMEDIATE 'SHOW GRANTS OF ROLE ' || v_role;
    -- Keep only the users.
    INSERT INTO maint_lab.security.user_asset_map (user_name, asset, via_role)
      SELECT "grantee_name", :v_asset, :v_role
      FROM TABLE(RESULT_SCAN(LAST_QUERY_ID()))
      WHERE "granted_to" = 'USER';
    total_rows := total_rows + SQLROWCOUNT;
  END FOR;
  RETURN 'User-asset rows written: ' || total_rows;
END;
$$;
```

Result: *Function SYNC_USER_ASSETS successfully created.* (Snowsight can show the word Function for a procedure.)

Step 4.9: Run the sync procedure

Do:

1. In the editor of `04_security.sql`, find **Step 4.9**.
2. Highlight all lines of **Step 4.9**, from its comment line to the last semicolon.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

```
-- Step 4.9: Run the sync. You must see "User-asset rows written: 5".
CALL maint_lab.security.sync_user_assets();

SELECT * FROM maint_lab.security.user_asset_map ORDER BY asset, user_name;
```

Result: `User-asset rows written: 5`. The SELECT shows OLIVIA_WAIO (WAIO), DIEGO_ESC (ESCONDIDA), LIAM_BMA (BMA), PRIYA_HQ (ALL) and your user (ALL).

Step 4.10: Make the row access policy and attach it

Why: The policy is the RLS rule. When it is attached to a table, Snowflake adds the rule to each query on that table. Nobody can go around it: not a view, not an AI function, not the app. **Architecture:** 4

Do:

1. In the editor of `04_security.sql`, find **Step 4.10**.
2. Highlight all lines of **Step 4.10**, from its comment line to the last semicolon.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

```
-- ------------------------------------------------------------------
-- Step 4.10: Create the row access policy and attach it to both tables.
-- The rule: show a row only if the current user has the asset of the
-- row, or has 'ALL'.
-- ------------------------------------------------------------------
CREATE OR REPLACE ROW ACCESS POLICY maint_lab.security.asset_rap
AS (row_asset VARCHAR) RETURNS BOOLEAN ->
  EXISTS (
    SELECT 1
    FROM maint_lab.security.user_asset_map m
    WHERE m.user_name = CURRENT_USER()
      AND (m.asset = row_asset OR m.asset = 'ALL')
  );

ALTER TABLE maint_lab.ops.sites          ADD ROW ACCESS POLICY maint_lab.security.asset_rap ON
(asset);
ALTER TABLE maint_lab.ops.maint_requests ADD ROW ACCESS POLICY maint_lab.security.asset_rap ON
(asset);
```

Result: *Row access policy ASSET_RAP successfully created.* Then 2 times *Statement executed successfully.*

Step 4.11: Test the policy

Why: POLICY_CONTEXT runs a query "as if" a different user runs it. You can test each user without a new login. **Architecture:** 4

Do:

1. Highlight only the first SELECT under **Step 4.11** (your user). Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
2. Highlight the next `EXECUTE USING POLICY_CONTEXT ...` statement (2 lines). Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
3. Do the same for each of the next 3 statements, one at a time.
4. Compare each result with the table below.

```
-- ------------------------------------------------------------------
-- Step 4.11: Test the policy. POLICY_CONTEXT shows the result
```

```
-- "as if" another user runs the query. You do not have to log in again.
-- ------------------------------------------------------------------
-- You (HQ): 3 assets, 60 rows.
SELECT asset, COUNT(*) AS rows_seen
FROM maint_lab.ops.maint_requests GROUP BY asset ORDER BY asset;

-- OLIVIA_WAIO: only WAIO, 20 rows.
EXECUTE USING POLICY_CONTEXT(CURRENT_USER => 'OLIVIA_WAIO')
AS SELECT asset, COUNT(*) AS rows_seen FROM maint_lab.ops.maint_requests GROUP BY asset;

-- DIEGO_ESC: only ESCONDIDA, 20 rows.
EXECUTE USING POLICY_CONTEXT(CURRENT_USER => 'DIEGO_ESC')
AS SELECT asset, COUNT(*) AS rows_seen FROM maint_lab.ops.maint_requests GROUP BY asset;

-- PRIYA_HQ: 3 assets, 60 rows.
EXECUTE USING POLICY_CONTEXT(CURRENT_USER => 'PRIYA_HQ')
AS SELECT asset, COUNT(*) AS rows_seen FROM maint_lab.ops.maint_requests GROUP BY asset;

-- The view also obeys the policy. LIAM_BMA sees only the 4 BMA sites.
EXECUTE USING POLICY_CONTEXT(CURRENT_USER => 'LIAM_BMA')
AS SELECT asset, site_name, requests, downtime_hours FROM maint_lab.ops.v_site_insight;
```

Query	Result that you must see
Your user	BMA 20, ESCONDIDA 20, WAIO 20
OLIVIA_WAIO	WAIO 20 (only)
DIEGO_ESC	ESCONDIDA 20 (only)
PRIYA_HQ	BMA 20, ESCONDIDA 20, WAIO 20
LIAM_BMA on the view	4 BMA sites

Step 4.12 (optional): Test with a real login

1. Open a private window (C8). Paste your account URL (C3). Push Enter.
2. Type the user name `OLIVIA_WAIO` and the password `AssetCare#Lab2026`. Select **Sign in**.
3. Select **Projects » Workspaces**. Select **+ Add new** » **SQL File**.
4. Type: `SELECT asset, COUNT(*) FROM maint_lab.ops.maint_requests GROUP BY asset;`
5. Highlight the line. Push Ctrl + Enter (Windows) or Cmd + Return (Mac). Then close the private window.

Result: OLIVIA_WAIO sees only WAIO, 20 rows. The app uses the same rule in Part 2.

End of Part 1: checkpoint

Item	Done?
MAINT_REQUESTS has 60 rows with DESCRIPTION_SAFE and AI_SUMMARY	☐
View V_SITE_INSIGHT gives 12 rows	☐
4 demo users and 4 group roles exist	☐

USER_ASSET_MAP has 5 rows	☐
POLICY_CONTEXT tests give the results in Step 4.11	☐

> **CAUTION:** Do not delete the objects at the end of day 1. Part 2 uses them.

Part 2: The Streamlit app with AI and row-level security

Part 2 (day 2)

Labs 5 to 9. You build part **2** of the architecture and connect it to parts **1** **3** **4** **5** **6** .

Streamlit primer

Idea	Meaning
Script from top to bottom	Streamlit runs the full Python script each time a user changes a widget or pushes a button.
Widget	An input, for example `st.selectbox`, `st.text_input`, `st.number_input`, `st.text_area`, `st.button`.
Form	`st.form` keeps many inputs together. The script runs again only when the user pushes the submit button.
Output	`st.write`, `st.metric`, `st.dataframe`, `st.bar_chart`, `st.markdown`.
Layout	`st.sidebar`, `st.tabs`, `st.columns`.
Snowflake session	`get_active_session()` gives the session of the app. No password is necessary. `session.sql(...)` runs SQL.
Bind variables	`session.sql("... WHERE id = ?", params=[value])`. Always use `?` for user input. Do not put user text directly into SQL.
Owner's rights	The app runs SQL with the privileges of the owner role. With READ SESSION, `CURRENT_USER()` gives the real viewer.
Warehouse runtime	The app runs on the warehouse MAINT_WH. It does not need a compute pool.

```
import streamlit as st
from snowflake.snowpark.context import get_active_session

session = get_active_session()                # the session of the app
name = st.text_input("Your name")             # a widget
if st.button("Say hello"):                    # True when the user pushes the button
    df = session.sql("SELECT CURRENT_USER() AS u").to_pandas()
    st.write(f"Hello {name}. Snowflake user: {df.iloc[0]['U']}")
```

Lab 5: Make the app owner role and the app

Goal: Make the owner role. Make the app on the warehouse runtime. Show who the user is and which rows the user can see.

> **Why:** The app needs an owner role with exactly the privileges that it needs: read and insert requests, read sites, use Cortex, and read the viewer session. This is the principle of least privilege.
> **Architecture:** 2

> **NOTE:** Before you start: make sure that your role is ACCOUNTADMIN (C4).

Open the file and set the context

Do:

1. Select **Projects » Workspaces**.
2. In the file list of **My Workspace**, select `05_app_owner.sql`. The file opens in the editor.
3. Read the comments in the file. Each step starts with a comment line `-- Step ...`.

Do:

1. In the editor of `05_app_owner.sql`, find **Step 5.0**.
2. Highlight all lines of **Step 5.0**, from its comment line to the last semicolon.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

```
-- Step 5.0: Set the role and the warehouse for this file.
USE ROLE ACCOUNTADMIN;
USE WAREHOUSE maint_wh;
```

Step 5.1 to 5.6: Make the owner role and give the privileges

> **Why:** Step 5.3 gives INSERT only on MAINT_REQUESTS, because the app writes only there. Step 5.6 (READ SESSION) lets `CURRENT_USER()` in the app give the real viewer. Without it, the policy cannot filter for each user. **Architecture:** 2 4

Do:

1. In the editor of `05_app_owner.sql`, find **Step 5.1 to Step 5.6**.
2. Highlight all lines of **Step 5.1 to Step 5.6**, from its comment line to the last semicolon.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

```
-- Step 5.1: Create the owner role.
CREATE ROLE IF NOT EXISTS maint_app_owner COMMENT = 'Owns the Asset Care Streamlit app';
GRANT ROLE maint_app_owner TO ROLE SYSADMIN;

-- Step 5.2: Give access to the warehouse, the database and the schemas.
GRANT USAGE ON WAREHOUSE maint_wh           TO ROLE maint_app_owner;
GRANT USAGE ON DATABASE  maint_lab          TO ROLE maint_app_owner;
GRANT USAGE ON SCHEMA    maint_lab.ops      TO ROLE maint_app_owner;
GRANT USAGE ON SCHEMA    maint_lab.security TO ROLE maint_app_owner;
GRANT USAGE ON SCHEMA    maint_lab.app      TO ROLE maint_app_owner;

-- Step 5.3: Give access to the data. The app reads and writes MAINT_REQUESTS.
GRANT SELECT, INSERT ON TABLE maint_lab.ops.maint_requests       TO ROLE maint_app_owner;
GRANT SELECT         ON TABLE maint_lab.ops.sites                TO ROLE maint_app_owner;
GRANT SELECT         ON VIEW  maint_lab.ops.v_site_insight       TO ROLE maint_app_owner;
```

```
GRANT SELECT         ON TABLE maint_lab.security.user_asset_map  TO ROLE maint_app_owner;

-- Step 5.4: Let the role create a Streamlit app in the APP schema.
GRANT CREATE STREAMLIT ON SCHEMA maint_lab.app TO ROLE maint_app_owner;

-- Step 5.5: Let the role use Cortex AI functions.
-- NOTE: By default, the PUBLIC role already has the USE AI FUNCTIONS
-- privilege. This grant makes sure that the owner role can use Cortex.
GRANT DATABASE ROLE SNOWFLAKE.CORTEX_USER TO ROLE maint_app_owner;

-- Step 5.6: IMPORTANT. Let the app read the session of the viewer.
-- Without this grant, CURRENT_USER() in the app does not give the
-- viewer, and the row access policy cannot filter for each user.
GRANT READ SESSION ON ACCOUNT TO ROLE maint_app_owner;
```

Result: *Statement executed successfully.* for each statement.

Step 5.7 to 5.8: Give the owner role to your user and do a check

Do:

1. In the editor of `05_app_owner.sql`, find **Step 5.7**.
2. Highlight the full block, from the comment line **Step 5.7** to the last line `$$;`.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

Do:

1. Highlight **Step 5.8**. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).

```
-- Step 5.7: Give the owner role to YOUR user.
EXECUTE IMMEDIATE $$
BEGIN
  EXECUTE IMMEDIATE 'GRANT ROLE maint_app_owner TO USER "' || CURRENT_USER() || '"';
  RETURN 'MAINT_APP_OWNER is granted to ' || CURRENT_USER();
END;
$$;

-- Step 5.8: Do a check. The list must show USAGE, INSERT, SELECT,
-- CREATE STREAMLIT, READ SESSION and a row for SNOWFLAKE.CORTEX_USER.
SHOW GRANTS TO ROLE maint_app_owner;
```

Result: Step 5.7: *MAINT_APP_OWNER is granted to* your user name. Step 5.8: the list shows CREATE STREAMLIT, INSERT, SELECT, USAGE, READ SESSION and a row for SNOWFLAKE.CORTEX_USER.

Step 5.9 to 5.12: Make the Streamlit app

> **Why:** We make the app with SQL. Then the name is always ASSET_CARE, and the app always uses the warehouse runtime. The container runtime needs a compute pool, and a compute pool can fail on a trial account. **Architecture:** 2 7

Do:

1. In the editor of `05_app_owner.sql`, find **Step 5.9 to Step 5.12**.
2. Highlight all lines of **Step 5.9 to Step 5.12**, from its comment line to the last semicolon.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

```
-- Step 5.9: Change to the owner role. The owner role creates the app.
USE ROLE maint_app_owner;
USE WAREHOUSE maint_wh;

-- Step 5.10: Create the Streamlit app on the WAREHOUSE runtime.
-- RUNTIME_NAME makes sure that the app does not use a compute pool.
-- CAUTION: Run this step one time only. If you run it again later,
--          it replaces the app and removes your code.
CREATE OR REPLACE STREAMLIT maint_lab.app.asset_care
  RUNTIME_NAME = 'SYSTEM$WAREHOUSE_RUNTIME'
  QUERY_WAREHOUSE = maint_wh
  TITLE = 'Asset Care Requests'
  COMMENT = 'Maintenance request app with Cortex AI and row-level security';

-- Step 5.11: Make the app active.
ALTER STREAMLIT maint_lab.app.asset_care ADD LIVE VERSION FROM LAST;

-- Step 5.12: Do a check. You must see one row: ASSET_CARE,
-- title "Asset Care Requests", query_warehouse MAINT_WH.
SHOW STREAMLITS IN SCHEMA maint_lab.app;
```

Result: *Streamlit ASSET_CARE successfully created.* Then *Statement executed successfully.* The SHOW result has one row: name ASSET_CARE, title Asset Care Requests, query_warehouse MAINT_WH.

Step 5.13: Change your Snowsight role to MAINT_APP_OWNER

> **Why:** Snowsight shows the apps of your current role. To edit the app, use the role that owns it.

1. Select your user name at the bottom of the navigation menu.
2. Select **Switch role**. Select **MAINT_APP_OWNER**.
3. Make sure that **MAINT_APP_OWNER** shows below your user name. If it is not in the list, refresh the page (F5).

Step 5.14: Copy the code of version 1

1. Select **Projects » Workspaces**. In the file list, select `streamlit_app_v1.py`.
2. Click in the editor. Push Ctrl + A (Mac: Cmd + A). Push Ctrl + C (Mac: Cmd + C).

Step 5.15: Open the app and paste the code

> **Why:** Version 1 shows only who you are and which rows you can see. It is the simplest test of the full chain: login 1, app 2, policy 4, data 5.

1. Select **Projects » Streamlit**. Select **Asset Care Requests**. The app opens (it can open in a new tab).
2. Select **Edit**. The code editor opens on the left side.
3. Click in the code editor. Push Ctrl + A (Mac: Cmd + A). Push Delete.

4. Push Ctrl + V (Mac: Cmd + V). Select **Run**. Wait until the app shows on the right side.

Result: The sidebar shows your user and the asset ALL (group MAINT_HQ). The first table shows 3 assets with 20 requests each.

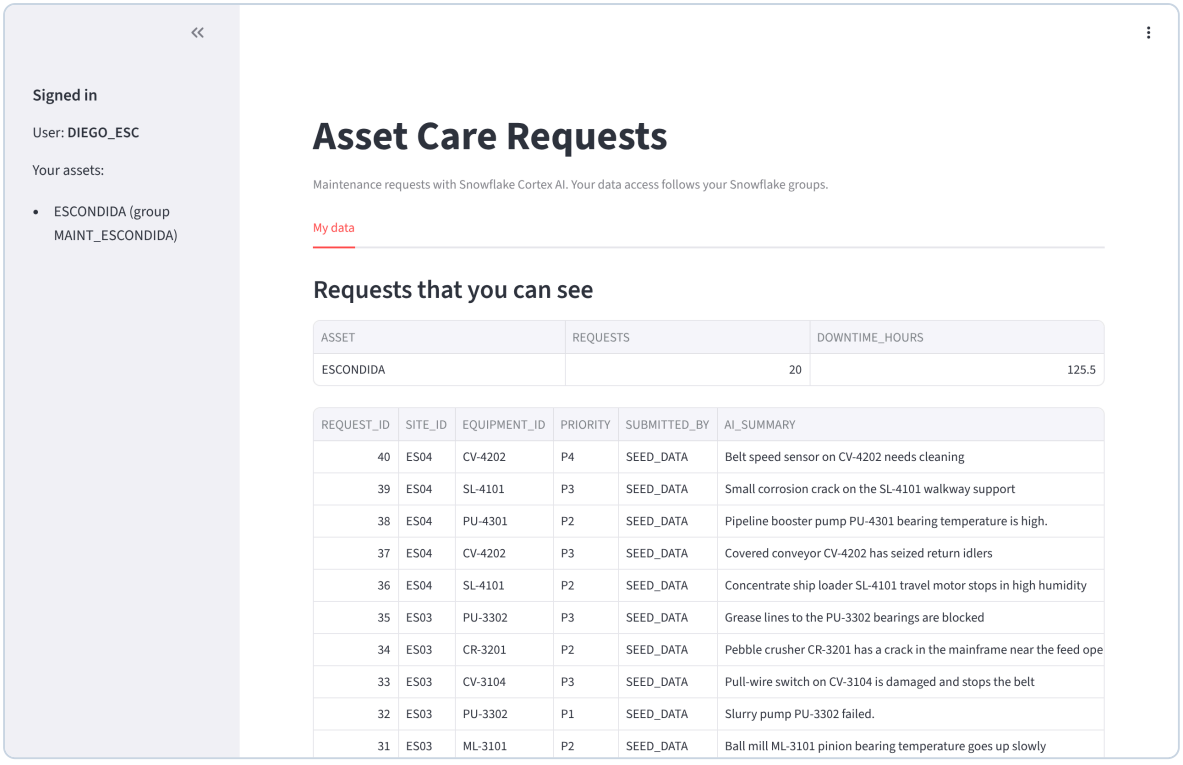


Figure 4. Version 1 for DIEGO_ESC (local preview with test data). DIEGO_ESC sees only ESCONDIDA rows.

NOTE: The app screenshots come from a local preview of the same code with test data. In Snowflake the app looks almost the same, inside the Snowsight frame.

CAUTION: Copy Python code from the `.py` file in Workspaces, not from the PDF. A PDF can change the spaces at the start of a line.

The full version 1 code:

```
# ========================================================================
# Asset Care Requests · PART 2 · APP VERSION 1: who am I and my data
# Streamlit in Snowflake, warehouse runtime, owner's rights.
#
# What the app does:
#   1. Shows who you are and which assets your Snowflake groups give you.
#   2. Shows the requests that you can see. The row access policy
#      filters every query.
# ========================================================================
import streamlit as st
from snowflake.snowpark.context import get_active_session

st.set_page_config(layout="wide")

# The Snowflake session of the app. No password is necessary.
session = get_active_session()

def run_query(sql, params=None):
    """Run a query and give the result as a pandas DataFrame."""
```

```
    return session.sql(sql, params=params).to_pandas()


# ----------------------------------------------------------------------
# Header and sidebar: who is the viewer?
# CURRENT_USER() gives the viewer, because the owner role has the
# READ SESSION privilege (Lab 5).
# ----------------------------------------------------------------------
me = run_query("SELECT CURRENT_USER() AS user_name").iloc[0]["USER_NAME"]
access = run_query(
    """SELECT asset, via_role
       FROM maint_lab.security.user_asset_map
       WHERE user_name = CURRENT_USER()
       ORDER BY asset"""
)

st.title("Asset Care Requests")
st.caption("Maintenance requests with Snowflake Cortex AI. Your data access follows your
Snowflake groups.")

with st.sidebar:
    st.subheader("Signed in")
    st.write(f"User: **{me}**")
    if access.empty:
        st.error("You are not in a maintenance group. Speak to the administrator.")
    else:
        st.write("Your assets:")
        for row in access.itertuples():
            st.write(f"- {row.ASSET} (group {row.VIA_ROLE})")

tab_data, = st.tabs(["My data"])

# ----------------------------------------------------------------------
# Tab: My data. The row access policy filters these queries.
# ----------------------------------------------------------------------
with tab_data:
    st.subheader("Requests that you can see")
    st.dataframe(
        run_query(
            """SELECT asset, COUNT(*) AS requests, SUM(downtime_hours) AS downtime_hours
               FROM maint_lab.ops.maint_requests
               GROUP BY asset
               ORDER BY asset"""
        ),
        hide_index=True,
    )
    st.dataframe(
        run_query(
            """SELECT request_id, site_id, equipment_id, priority, submitted_by,
                      ai_summary, description_safe
               FROM maint_lab.ops.maint_requests
               ORDER BY request_id DESC
               LIMIT 20"""
        ),
        hide_index=True,
    )
```

Lab 6: Save input in Snowflake with Cortex AI

Goal: Add a form. When the user saves a request, the app inserts a row. Cortex AI hides the personal data and writes a summary.

Why: This is the input requirement. The app writes the request to the same table as the dummy data, so the insights and the policy work on the new rows at once. The AI runs one time, when the row is saved. **Architecture:** 2 3 5

Read this SQL from the app before you change the code:

```
INSERT_SQL = """
INSERT INTO maint_lab.ops.maint_requests
  (site_id, asset, equipment_id, equipment_type, failure_mode, priority,
   downtime_hours, submitted_by, submitted_at, description,
   description_safe, ai_summary)
SELECT s.site_id,
       s.asset,
       y.eq_id,
       y.eq_type,
       y.fmode,
       y.prio,
       y.hours,
       CURRENT_USER(),
       CURRENT_TIMESTAMP()::TIMESTAMP_NTZ,
       y.txt,
       y.safe,
       AI_SUMMARIZE(y.safe)
FROM (SELECT x.*, AI_REDACT(x.txt) AS safe
      FROM (SELECT ? AS sid, ? AS eq_id, ? AS eq_type, ? AS fmode,
                   ? AS prio, ? AS hours, ? AS txt) x) y
JOIN maint_lab.ops.sites s
  ON s.site_id = y.sid
"""
```

1. The `?` marks are bind variables. The app sends the values separately. This stops SQL injection.
2. `AI_REDACT(x.txt)` makes the safe text. `AI_SUMMARIZE(y.safe)` summarises the safe text, not the raw text.
3. The JOIN to SITES uses the row access policy. A user can save a request only for a site that he or she can see.

Step 6.1: Change the app to version 2

1. Select **Projects » Workspaces**. Select `streamlit_app_v2.py`. Click in the editor. Push Ctrl + A (Mac: Cmd + A). Push Ctrl + C (Mac: Cmd + C).
2. Go back to the app (**Projects » Streamlit** » **Asset Care Requests**). If the code editor is closed, select **Edit**.
3. Click in the code editor. Push Ctrl + A (Mac: Cmd + A). Push Delete. Push Ctrl + V (Mac: Cmd + V). Select **Run**.

Result: The app has 2 tabs: **New request** and **My data**.

Step 6.2: Save a request

> **Why:** You type a name and a phone number on purpose. Then you see that AI_REDACT hides them before the text is shown or analysed.

1. Select the tab **New request**.
2. In **Site**, select **WA01 · Mt Newman (WAIO)**.
3. In **Equipment ID**, type `CV-0215`. In **Equipment type**, select **Conveyor**.
4. In **Failure mode**, select **Belt or idler**. In **Priority**, select **P2**. In **Downtime (hours)**, type `3`.
5. In **Description**, type: `Conveyor CV-0215 belt splice is damaged and the belt slips. Fitter Tom Baker, mobile 0412 987 654, asks for a splicing crew on day shift.`
6. Select **Save request**. Wait for the green message (about 5 to 15 seconds).
7. Select the tab **My data**. Find the new row at the top.

Result: The message **The request is saved in Snowflake.** shows. DESCRIPTION_SAFE shows `[NAME]` and `[PHONE_NUMBER]`. AI_SUMMARY is a short summary.

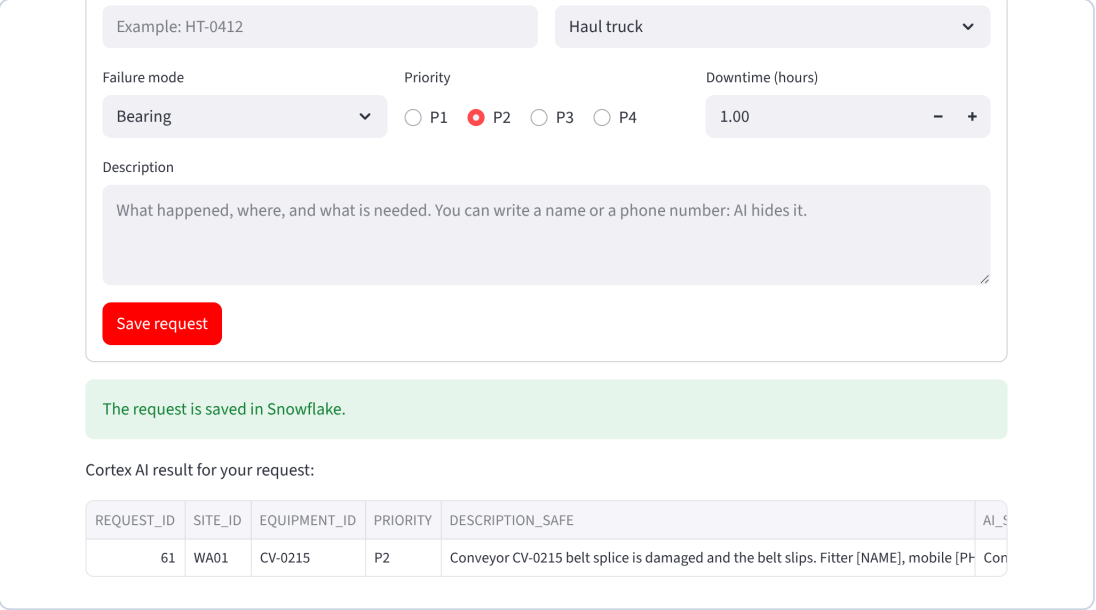


Figure 5. The request is saved. The name and the phone number are hidden (local preview).

Step 6.3: Do a check in SQL

1. Select **Projects » Workspaces**. Select **+ Add new** » **SQL File**. Rename it to `lab6_check.sql` (⋮ » **Rename**).
2. Copy the SQL below into the file. Highlight all lines. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).

```
-- Lab 6 check: your new request must be at the top.
USE ROLE ACCOUNTADMIN;
USE WAREHOUSE maint_wh;
SELECT request_id, site_id, equipment_id, submitted_by, description, description_safe, ai_summary
FROM maint_lab.ops.maint_requests
ORDER BY request_id DESC
LIMIT 3;
```

Result: The first row is your request (request_id 61). DESCRIPTION has the name and phone. DESCRIPTION_SAFE does not.

Lab 7: Show insights and AI analysis

Goal: Add the Insights tab and the AI analysis tab. All queries go through the row access policy, also the AI queries.

Why: This is the insight requirement. The numbers and charts use normal SQL. The AI analysis uses AI_AGG and AI_SUMMARIZE_AGG on the safe text. The policy filters the rows before the AI reads them, so the AI cannot see other assets. **Architecture:** 2 3 4

Step 7.1: Change the app to version 3 (final)

1. Select **Projects » Workspaces**. Select `streamlit_app_v3.py`. Click in the editor. Push Ctrl + A (Mac: Cmd + A). Push Ctrl + C (Mac: Cmd + C).
2. Go back to the app. If the code editor is closed, select **Edit**.
3. Click in the code editor. Push Ctrl + A (Mac: Cmd + A). Push Delete. Push Ctrl + V (Mac: Cmd + V). Select **Run**.

Result: The app has 3 tabs: **New request**, **Insights** and **AI analysis**.

Part of the app	What it uses
4 numbers	`COUNT(*)`, `SUM(downtime_hours)`, `COUNT_IF(priority = 'P1')`, number of sites
Downtime by site	SQL GROUP BY site
Requests by failure mode and priority	SQL GROUP BY failure mode and priority
P1 requests	AI_SUMMARY and DESCRIPTION_SAFE columns
Site table	View `V_SITE_INSIGHT`
AI analysis: 3 analyses	`AI_AGG` with a fixed instruction for each analysis
AI analysis: Summarise all requests	`AI_SUMMARIZE_AGG`

NOTE: The AI instructions are fixed text in the app. The user selects an analysis but does not type the instruction. This keeps the AI output predictable and safe.

Step 7.2: Look at the insights

1. Select the tab **Insights**. Look at the 4 numbers and the 2 charts.
2. Scroll down. Look at **P1 requests** and **Site table**.

Result: For your user (HQ): Requests = 61, Downtime = 420 hours, 12 sites.

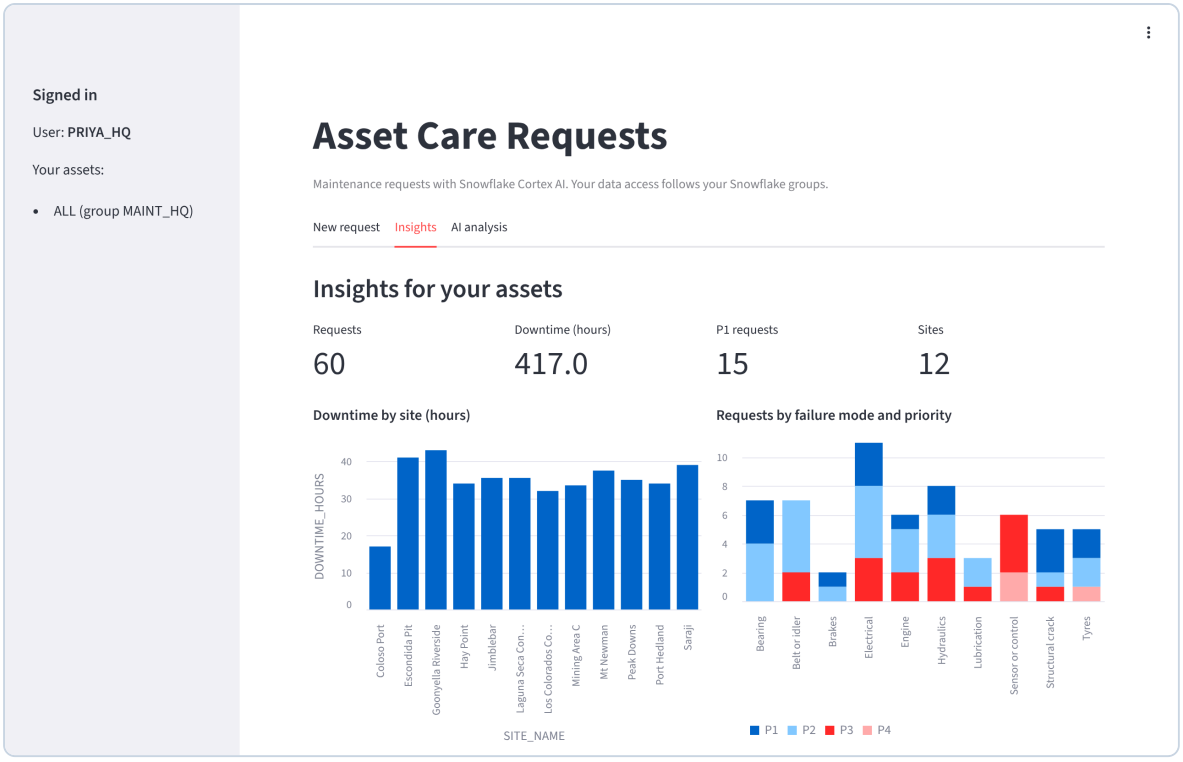
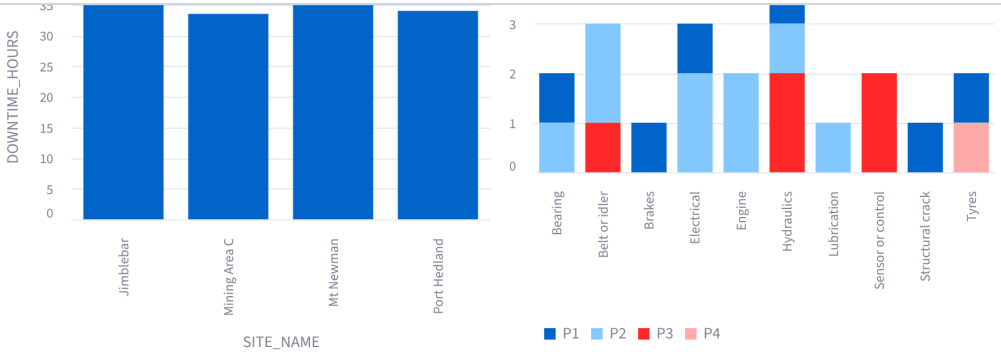


Figure 6. Tab Insights for PRIYA_HQ: all 3 assets (local preview).



P1 requests (newest first). Personal data is hidden by AI_REDACT.

REQUEST_ID	SITE_ID	EQUIPMENT_ID	EQUIPMENT_TYPE	DOWNTIME_HOURS	AI_SUMMARY
16	WA04	SL-0101	Ship loader	20	Crack found on the boom of ship loader SL-0101 during th
12	WA03	CR-0301	Crusher	16	Primary crusher CR-0301 eccentric bearing temperature is
7	WA02	HT-0522	Haul truck	5	Brake temperature warning on HT-0522 on the long downl
5	WA01	HT-0412	Haul truck	9	Second hydraulic leak on HT-0412 this month, again at the
6	WA02	SH-0301	Shovel	14	Electric shovel SH-0301 tripped three times on the night sl
2	WA01	HT-0418	Haul truck	12	Front right tyre on HT-0418 has a deep cut in the sidewall

Site table

ASSET	SITE_ID	SITE_NAME	REQUESTS	DOWNTIME_HOURS	P1_REQUESTS	TOP_FAILURE_MODE
WAIO	WA01	Mt Newman	6	38.5	2	Hydraulics
WAIO	WA02	Mining Area C	5	33.5	2	Electrical
WAIO	WA03	Jimblebar	5	35.5	1	Bearing
WAIO	WA04	Port Hedland	5	34	1	Belt or idler

Figure 7. P1 requests and site table for OLIVIA_WAIO: only WAIO rows (local preview).

Step 7.3: Run the AI analysis

1. Select the tab **AI analysis**.
2. In **Scope**, keep **All my sites**. In **Analysis**, select **Top 3 repeat problems and actions**.
3. Select **Run AI analysis**. Wait for the result (about 10 to 30 seconds).
4. Select a different analysis and a single site. Select **Run AI analysis** again.
5. Select **Summarise all requests**.

Result: 3 short bullet points with equipment IDs and actions. The caption tells how many requests the AI used.

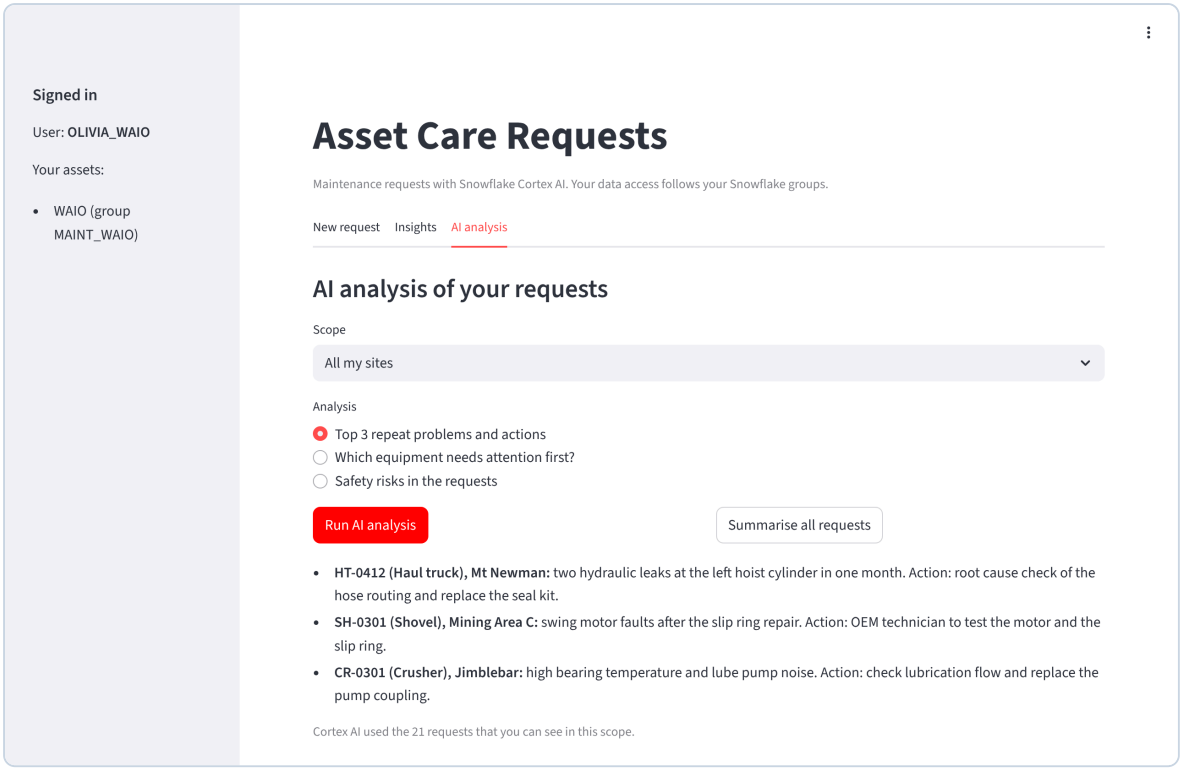


Figure 8. AI analysis for OLIVIA_WAIO. The AI reads only WAIO requests (local preview; in Snowflake the AI writes its own text).

Lab 8: Share the app and test as other users

Goal: Give the app to the groups. Log in as each demo user. Make sure that each user sees only his or her assets.

Why: This is the user management requirement. Users log in with their Snowflake user. Access to the app is a grant to a Snowflake role. The data inside the app follows the same policy as SQL.
Architecture: 1 2 4

Step 8.1 to 8.3: Share the app

Do:

1. Select **Projects » Workspaces**.
2. In the file list of **My Workspace**, select `06_share_app.sql`. The file opens in the editor.
3. Read the comments in the file. Each step starts with a comment line `-- Step ...`.

Do:

1. In the editor of `06_share_app.sql`, find **Step 8.1 to Step 8.3**.
2. Highlight all lines of **Step 8.1 to Step 8.3**, from its comment line to the last semicolon.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

```
-- Step 8.1: Use the owner role.
USE ROLE maint_app_owner;
USE WAREHOUSE maint_wh;

-- Step 8.2: Share the app with MAINT_APP_USER.
-- All group roles (MAINT_HQ, MAINT_WAIO, MAINT_ESCONDIDA, MAINT_BMA)
-- have MAINT_APP_USER, so all demo users get the app.
GRANT USAGE ON STREAMLIT maint_lab.app.asset_care TO ROLE maint_app_user;

-- Step 8.3: Do a check. The list must show USAGE on STREAMLIT
-- MAINT_LAB.APP.ASSET_CARE.
SHOW GRANTS TO ROLE maint_app_user;
```

Result: *Statement executed successfully.* The grant list of MAINT_APP_USER has a row: privilege USAGE, granted_on STREAMLIT, name MAINT_LAB.APP.ASSET_CARE.

> **NOTE:** You can also share in the app: **Projects » Streamlit** » **Asset Care Requests** » **Share**. In **Grant to a role or person**, type `MAINT_APP_USER`. Select **View only**. Select **Done**.

Step 8.4: Test as OLIVIA_WAIO

1. Open a private window (C8). Paste your account URL (C3). Push Enter.
2. Log in as `OLIVIA_WAIO`, password `AssetCare#Lab2026`.
3. Make sure that **MAINT_WAIO** shows below the user name (bottom left).
4. Select **Projects » Streamlit** » **Asset Care Requests**.
5. Look at the sidebar, the tab **Insights**, and the **Site** list in the tab **New request**.

Result: The sidebar shows WAIO (group MAINT_WAIO). The numbers show only WAIO (21 requests, with your Lab 6 request). The Site list shows only WA01 to WA04.

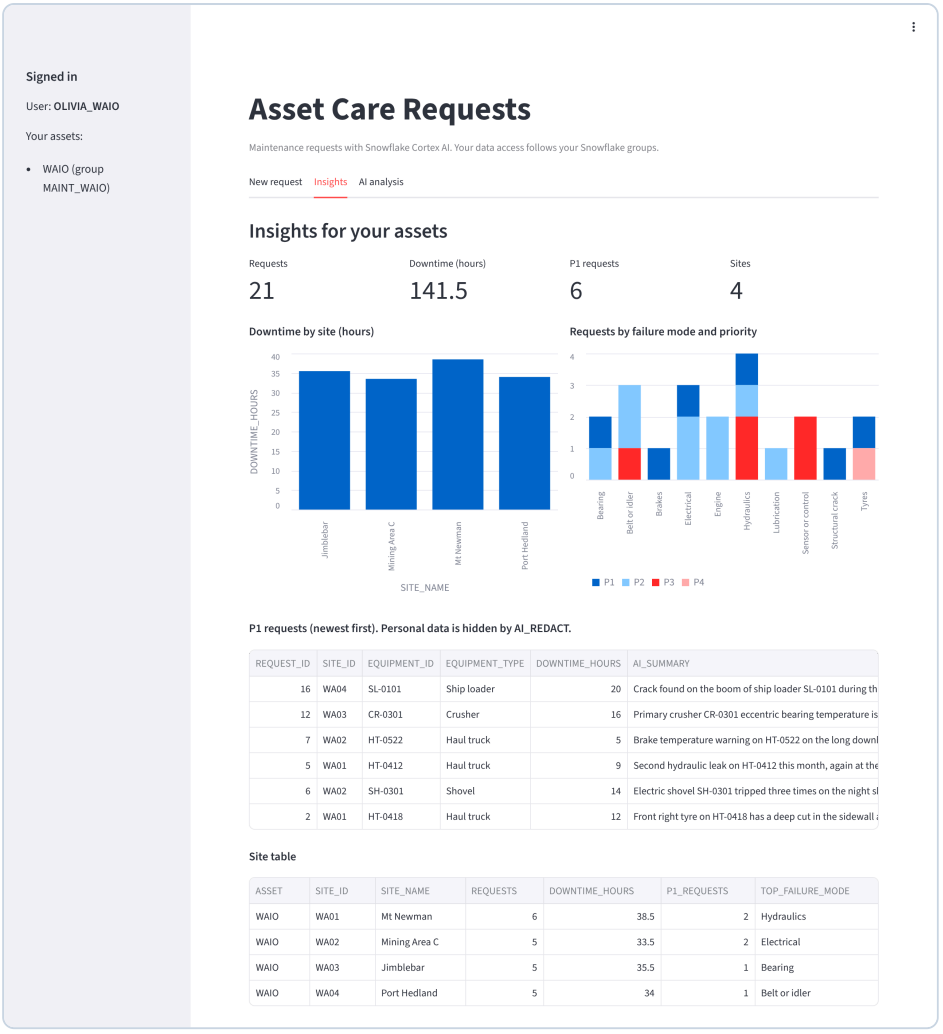


Figure 9. OLIVIA_WAIO sees only the 4 WAIO sites (local preview).

Step 8.5: Test as the other users

1. Close the private window. Open a new private window.
2. Log in as `DIEGO_ESC`. Open the app. Make sure that only ESCONDIDA shows. Close the window.
3. Do the same for `LIAM_BMA` (BMA only) and `PRIYA_HQ` (all assets).

User	Sidebar	Requests	Sites in the list
OLIVIA_WAIO	WAIO	21	WA01 to WA04
DIEGO_ESC	ESCONDIDA	20	ES01 to ES04
LIAM_BMA	BMA	20	BM01 to BM04
PRIYA_HQ	ALL	61	All 12

Lab 9: Change a group, see the app change

Goal: Add OLIVIA_WAIO to the BMA group. The app shows 2 assets for her. Then remove her again.

> **Why:** This shows that access is managed only in Snowflake. You do not change the app or the policy. You change a role grant and run the sync. **Architecture:** 1 6 4

Step 9.1: Open a window for OLIVIA_WAIO

1. Open a private window. Log in as `OLIVIA_WAIO`. Open the app. Keep the window open.

Step 9.2: Add OLIVIA_WAIO to the BMA group (your own window)

1. Go back to your normal browser window. Make sure that your role is ACCOUNTADMIN (C4).

Do:

1. Select **Projects » Workspaces**.
2. In the file list of **My Workspace**, select `07_change_group.sql`. The file opens in the editor.
3. Read the comments in the file. Each step starts with a comment line `-- Step ...`.

Do:

1. In the editor of `07_change_group.sql`, find **Step 9.0 to Step 9.3**.
2. Highlight all lines of **Step 9.0 to Step 9.3**, from its comment line to the last semicolon.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

```
-- Step 9.0: Set the role and the warehouse for this file.
USE ROLE ACCOUNTADMIN;
USE WAREHOUSE maint_wh;

-- Step 9.1: Put OLIVIA_WAIO also in the BMA group.
GRANT ROLE maint_bma TO USER olivia_waio;

-- Step 9.2: Sync the groups to the access table.
CALL maint_lab.security.sync_user_assets();

-- Step 9.3: Do a check. OLIVIA_WAIO now has WAIO and BMA.
SELECT * FROM maint_lab.security.user_asset_map WHERE user_name = 'OLIVIA_WAIO';

EXECUTE USING POLICY_CONTEXT(CURRENT_USER => 'OLIVIA_WAIO')
AS SELECT asset, COUNT(*) AS rows_seen FROM maint_lab.ops.maint_requests GROUP BY asset;
```

Result: The CALL gives `User-asset rows written: 6`. OLIVIA_WAIO has WAIO and BMA. POLICY_CONTEXT shows BMA and WAIO.

Step 9.4: Look at the app of OLIVIA_WAIO

1. Go to the private window of OLIVIA_WAIO. Refresh the page (F5, Mac: Cmd + R).
2. Look at the sidebar and the tab **Insights**.

```
-- Step 9.4: In the app window of OLIVIA_WAIO, refresh the page.
-- The sidebar shows 2 assets. The Insights tab shows 2 assets.
```

Result: The sidebar shows BMA and WAIO. The tab Insights shows 41 requests and 8 sites.

Step 9.5: Remove OLIVIA_WAIO from the BMA group

Do:

1. In the editor of `07_change_group.sql`, find **Step 9.5**.
2. Highlight all lines of **Step 9.5**, from its comment line to the last semicolon.
3. Push Ctrl + Enter (Windows) or Cmd + Return (Mac).
4. Look at the **Results** pane below the editor. Compare it with the **Result** box.

```
-- Step 9.5: Remove OLIVIA_WAIO from the BMA group again.
REVOKE ROLE maint_bma FROM USER olivia_waio;
CALL maint_lab.security.sync_user_assets();

EXECUTE USING POLICY_CONTEXT(CURRENT_USER => 'OLIVIA_WAIO')
AS SELECT asset, COUNT(*) AS rows_seen FROM maint_lab.ops.maint_requests GROUP BY asset;
```

Result: The CALL gives `User-asset rows written: 5`. Refresh the window of OLIVIA_WAIO: only WAIO shows.

NOTE: In production, a scheduled task can run the sync. A newer option is the container runtime with restricted caller's rights: the app then uses the roles of the viewer directly. It needs a compute pool, so this lab does not use it.

Troubleshooting

Problem	Cause and fix
AI function ... is not available for trial accounts.	The function is blocked on a trial account. Use only the 5 functions of this lab. Make sure that you use the lab files with no change.
Error: *No active warehouse*, or the object does not exist.	Run Step x.0 (or Step 1.1) of the file. Then run the step again.
Error on `CREATE ROW ACCESS POLICY`: feature not supported.	The account is Standard edition. Make a new trial with the Enterprise edition (C1).
Your user sees 0 rows after Step 4.10.	Your user is not in a group. Do Step 4.5 again. Then run Step 4.9.
Step 4.9 writes fewer than 5 rows.	Do Steps 4.3 to 4.5 again. Then run Step 4.9.
Step 4.10: the policy is already attached, or it cannot be replaced.	You ran Step 4.10 two times. The first run worked. Go to Step 4.11.
You ran `02_data.sql` again after Part 1. Now all users see all rows.	New tables have no policy. Run Steps 3.3 to 3.9. Then run only the two `ALTER TABLE ... ADD ROW ACCESS POLICY` statements of Step 4.10. Then run Step 4.2 and Step 5.3.
MAINT_APP_OWNER is not in the **Switch role** list.	Refresh the page. If it is still not there, run Step 5.7 again.
The app is not in **Projects » Streamlit**.	Your Snowsight role is not MAINT_APP_OWNER (Step 5.13). Demo users: do Lab 8 first.
In the app, every user sees the same rows, or 0 rows.	MAINT_APP_OWNER does not have READ SESSION. Run Step 5.6 as ACCOUNTADMIN. Then select **Run** in the app.
The sidebar says "You are not in a maintenance group".	Grant a group role to the user. Run the sync (Step 4.9).
Python error: `IndentationError` or `SyntaxError`.	Copy all the code again from the `.py` file in Workspaces.
Demo user cannot log in.	Check the user name and the password `AssetCare#Lab2026`. If Snowsight asks for MFA, enrol with an authenticator app or a passkey.
"The request is not saved" message.	The user cannot see the site. This is correct behaviour of the policy.

Clean up (optional, after the training)

> **CAUTION:** This removes all lab objects, users and roles. Do this only after Part 2.

Do:

1. Select **Projects » Workspaces**.
2. In the file list of **My Workspace**, select `99_cleanup.sql`. The file opens in the editor.
3. Read the comments in the file. Each step starts with a comment line `-- Step ...`.

Do:

1. Click in the editor. Push Ctrl + Shift + Enter (Windows) or Cmd + Shift + Return (Mac).

```
-- ============================================================================
-- CLEAN UP (optional, after the training)
-- File: 99_cleanup.sql
-- Run as: ACCOUNTADMIN
-- CAUTION: This removes all lab objects, users and roles.
-- ============================================================================

-- Step C.0: Set the role.
USE ROLE ACCOUNTADMIN;

DROP DATABASE IF EXISTS maint_lab;      -- also removes the app, tables, policy and procedure
DROP WAREHOUSE IF EXISTS maint_wh;

DROP USER IF EXISTS olivia_waio;
DROP USER IF EXISTS diego_esc;
DROP USER IF EXISTS liam_bma;
DROP USER IF EXISTS priya_hq;

DROP ROLE IF EXISTS maint_app_owner;
DROP ROLE IF EXISTS maint_hq;
DROP ROLE IF EXISTS maint_waio;
DROP ROLE IF EXISTS maint_escondida;
DROP ROLE IF EXISTS maint_bma;
DROP ROLE IF EXISTS maint_app_user;
```

Appendix A: Verification and sources

Check	Status
AI functions on a trial account	Tested by the trainer in a new Enterprise trial account (October 2026). The 5 functions of this lab passed. Other functions are blocked.
SQL syntax and Snowsight paths	Checked against the Snowflake documentation (October 2026).
Python code	Compiled with no error. The app ran end to end in a local preview with test data (Streamlit 1.50).
Full run in Snowflake	**Trainer must do it** one time before the training: C1 to C8 and Labs 1 to 9.

- BHP operated assets (WAIO, Escondida, BMA): BHP Group Form 20-F FY2026, sec.gov
- Trial accounts: docs.snowflake.com/en/user-guide/admin-trial-account
- Migrate from legacy SNOWFLAKE.CORTEX functions: docs.snowflake.com/en/user-guide/snowflake-cortex/aisql-migrate-legacy-functions
- AI_REDACT, AI_SUMMARIZE, AI_SUMMARIZE_AGG, AI_AGG, SPLIT_TEXT_RECURSIVE_CHARACTER: docs.snowflake.com/en/sql-reference/functions/
- Row access policies in Streamlit in Snowflake: docs.snowflake.com/en/developer-guide/streamlit/features/row-access
- CREATE STREAMLIT: docs.snowflake.com/en/sql-reference/sql/create-streamlit
- Sharing Streamlit apps: docs.snowflake.com/en/developer-guide/streamlit/features/sharing-streamlit-apps
- Workspaces: docs.snowflake.com/en/user-guide/ui-snowsight/workspaces
- POLICY_CONTEXT: docs.snowflake.com/en/sql-reference/functions/policy_context
- Snowflake editions: docs.snowflake.com/user-guide/intro-editions

Appendix B: Full code

> **CAUTION:** Use the lab files in Workspaces when possible. A PDF can change the spaces at the start of a line.

01_setup.sql

```
-- ============================================================================
-- PART 1 · LAB 1 · SETUP
-- File: 01_setup.sql
-- Run as: ACCOUNTADMIN
-- Purpose: Create the warehouse, the database and the schemas.
-- ============================================================================

-- Step 1.1: Use the administrator role.
USE ROLE ACCOUNTADMIN;

-- Step 1.2: Let Cortex AI use a model in a different region
-- if the model is not available in your region.
ALTER ACCOUNT SET CORTEX_ENABLED_CROSS_REGION = 'ANY_REGION';

-- Step 1.3: Create a small warehouse. It stops after 60 seconds of no work.
CREATE WAREHOUSE IF NOT EXISTS maint_wh
  WAREHOUSE_SIZE = 'XSMALL'
  AUTO_SUSPEND = 60
  AUTO_RESUME = TRUE
  INITIALLY_SUSPENDED = TRUE
  COMMENT = 'Warehouse for the BHP maintenance Cortex AI lab';

USE WAREHOUSE maint_wh;

-- Step 1.4: Create the database and three schemas.
--   OPS      = business data (sites and maintenance requests)
--   SECURITY = access rules (who can see which asset)
--   APP      = the Streamlit app (Part 2)
CREATE DATABASE IF NOT EXISTS maint_lab COMMENT = 'BHP maintenance Cortex AI lab';
CREATE SCHEMA IF NOT EXISTS maint_lab.ops;
CREATE SCHEMA IF NOT EXISTS maint_lab.security;
CREATE SCHEMA IF NOT EXISTS maint_lab.app;

USE SCHEMA maint_lab.ops;

-- Step 1.5: Do a check. You must see 3 schemas: APP, OPS, SECURITY
-- (plus INFORMATION_SCHEMA and PUBLIC).
SHOW SCHEMAS IN DATABASE maint_lab;

-- Step 1.6: Do a check of the AI setting. The value must be ANY_REGION.
SHOW PARAMETERS LIKE 'CORTEX_ENABLED_CROSS_REGION' IN ACCOUNT;
```

02_data.sql

```
-- ============================================================================
-- PART 1 · LAB 2 · DUMMY DATA
-- File: 02_data.sql
```

```
-- Run as: ACCOUNTADMIN
-- Purpose: Create two tables and load dummy maintenance data:
--          12 sites in 3 BHP assets and 60 maintenance requests.
-- NOTE: The asset and site names are real BHP names. All requests,
--       equipment numbers, people, phones and emails are DUMMY data.
-- ================================================================

-- Step 2.0: Set the role and the warehouse for this file.
USE ROLE ACCOUNTADMIN;
USE WAREHOUSE maint_wh;
USE SCHEMA maint_lab.ops;

-- Step 2.1: Create the SITES table. One row for each site.
CREATE OR REPLACE TABLE maint_lab.ops.sites (
  site_id    VARCHAR(10)  NOT NULL,   -- for example WA01
  site_name  VARCHAR(100) NOT NULL,
  asset      VARCHAR(20)  NOT NULL,   -- WAIO, ESCONDIDA or BMA
  commodity  VARCHAR(50)  NOT NULL,
  country    VARCHAR(50)  NOT NULL
);

-- Step 2.2: Create the MAINT_REQUESTS table. One row for each request.
-- DESCRIPTION is the text that the person typed. It can contain names,
-- phone numbers and emails. The last two columns are empty now.
-- Cortex AI fills them in Lab 3.
CREATE OR REPLACE TABLE maint_lab.ops.maint_requests (
  request_id       NUMBER AUTOINCREMENT START 1 INCREMENT 1,
  site_id          VARCHAR(10)   NOT NULL,
  asset            VARCHAR(20)   NOT NULL,
  equipment_id     VARCHAR(20)   NOT NULL,   -- for example HT-0412
  equipment_type   VARCHAR(50)   NOT NULL,   -- Haul truck, Conveyor, ...
  failure_mode     VARCHAR(50)   NOT NULL,   -- selected from a list
  priority         VARCHAR(2)    NOT NULL,   -- P1 (highest) to P4
  downtime_hours   NUMBER(6,1)   NOT NULL,
  submitted_by     VARCHAR(100)  NOT NULL,   -- Snowflake user name
  submitted_at     TIMESTAMP_NTZ NOT NULL,
  description      VARCHAR(4000) NOT NULL,   -- text as typed (can contain personal data)
  description_safe VARCHAR(4000),            -- AI result: personal data hidden (Lab 3)
  ai_summary       VARCHAR(1000)             -- AI result: short summary (Lab 3)
);

-- Step 2.3: Load 12 sites.
INSERT INTO maint_lab.ops.sites (site_id, site_name, asset, commodity, country) VALUES
  ('WA01', 'Mt Newman', 'WAIO', 'Iron ore', 'Australia'),
  ('WA02', 'Mining Area C', 'WAIO', 'Iron ore', 'Australia'),
  ('WA03', 'Jimblebar', 'WAIO', 'Iron ore', 'Australia'),
  ('WA04', 'Port Hedland', 'WAIO', 'Iron ore', 'Australia'),
  ('ES01', 'Escondida Pit', 'ESCONDIDA', 'Copper', 'Chile'),
  ('ES02', 'Laguna Seca Concentrator', 'ESCONDIDA', 'Copper', 'Chile'),
  ('ES03', 'Los Colorados Concentrator', 'ESCONDIDA', 'Copper', 'Chile'),
  ('ES04', 'Coloso Port', 'ESCONDIDA', 'Copper', 'Chile'),
  ('BM01', 'Goonyella Riverside', 'BMA', 'Steelmaking coal', 'Australia'),
  ('BM02', 'Peak Downs', 'BMA', 'Steelmaking coal', 'Australia'),
  ('BM03', 'Saraji', 'BMA', 'Steelmaking coal', 'Australia'),
  ('BM04', 'Hay Point', 'BMA', 'Steelmaking coal', 'Australia');

-- Step 2.4: Load 60 maintenance requests.
INSERT INTO maint_lab.ops.maint_requests
  (site_id, asset, equipment_id, equipment_type, failure_mode, priority,
   downtime_hours, submitted_by, submitted_at, description)
VALUES
```

```
  ('WA01', 'WAIO', 'HT-0412', 'Haul truck', 'Hydraulics', 'P2', 6.5, 'SEED_DATA', '2026-10-01
06:00:00',
   'Operator Jack Wilson (mobile 0412 555 019) reported a hydraulic oil leak at the left hoist
cylinder of HT-0412. The truck was parked at the go-line. Oil on the ground was cleaned with a
spill kit. Please replace the cylinder seal kit before the next shift.'),
  ('WA01', 'WAIO', 'HT-0418', 'Haul truck', 'Tyres', 'P1', 12.0, 'SEED_DATA', '2026-10-01
16:00:00',
   'Front right tyre on HT-0418 has a deep cut in the sidewall after contact with rock spillage
on the ramp. Truck stopped for safety. Tyre change crew needed. Contact the shift supervisor Emma
Clarke on emma.clarke@example.com.'),
  ('WA01', 'WAIO', 'EX-0203', 'Excavator', 'Engine', 'P2', 8.0, 'SEED_DATA', '2026-10-02
02:00:00',
   'EX-0203 shows high coolant temperature alarms after 3 hours of work. Radiator fins are
blocked with dust. Clean the radiator and check the fan belt tension.'),
  ('WA01', 'WAIO', 'CV-0110', 'Conveyor', 'Belt or idler', 'P3', 2.0, 'SEED_DATA', '2026-10-02
12:00:00',
   'Noisy idlers on CV-0110 near transfer point 3. Three idlers turn slowly. Replace at the next
planned shutdown.'),
  ('WA01', 'WAIO', 'HT-0412', 'Haul truck', 'Hydraulics', 'P1', 9.0, 'SEED_DATA', '2026-10-02
22:00:00',
   'Second hydraulic leak on HT-0412 this month, again at the left hoist cylinder. Maintenance
planner Ravi Kumar (0400 123 777) asks for a root cause check of the hose routing.'),
  ('WA02', 'WAIO', 'SH-0301', 'Shovel', 'Electrical', 'P1', 14.0, 'SEED_DATA', '2026-10-02
17:00:00',
   'Electric shovel SH-0301 tripped three times on the night shift. Main breaker trips when the
swing motor starts. Electrician Sam Taylor checked the cables and found heat damage at the slip
ring.'),
  ('WA02', 'WAIO', 'HT-0522', 'Haul truck', 'Brakes', 'P1', 5.0, 'SEED_DATA', '2026-10-03
03:00:00',
   'Brake temperature warning on HT-0522 on the long downhill haul road. The operator parked the
truck. Brake pads look worn. Do a full brake inspection before the truck goes back into
service.'),
  ('WA02', 'WAIO', 'DR-0105', 'Drill rig', 'Hydraulics', 'P3', 3.0, 'SEED_DATA', '2026-10-03
13:00:00',
   'DR-0105 rotation head is slow. Hydraulic pressure is lower than normal. Check the pump and
the filters.'),
  ('WA02', 'WAIO', 'CV-0215', 'Conveyor', 'Belt or idler', 'P2', 7.5, 'SEED_DATA', '2026-10-03
23:00:00',
   'Belt on CV-0215 runs off-centre at the tail pulley. Spillage builds up under the belt. Adjust
the training idlers and clean the spillage.'),
  ('WA02', 'WAIO', 'SH-0301', 'Shovel', 'Electrical', 'P2', 4.0, 'SEED_DATA', '2026-10-04
09:00:00',
   'SH-0301 swing motor fault again after the slip ring repair. Fault code SW-112. Please send
the OEM technician.'),
  ('WA03', 'WAIO', 'HT-0611', 'Haul truck', 'Engine', 'P2', 10.0, 'SEED_DATA', '2026-10-04
04:00:00',
   'HT-0611 has low engine power on the ramp and black smoke. Turbocharger noise. Operator Mia
Brown, phone 0455 220 981, says the problem started after the fuel change.'),
  ('WA03', 'WAIO', 'CR-0301', 'Crusher', 'Bearing', 'P1', 16.0, 'SEED_DATA', '2026-10-04
14:00:00',
   'Primary crusher CR-0301 eccentric bearing temperature is high, 92 degrees. Crusher stopped.
Check lubrication flow and the oil cooler.'),
  ('WA03', 'WAIO', 'CR-0301', 'Crusher', 'Lubrication', 'P2', 6.0, 'SEED_DATA', '2026-10-05
00:00:00',
   'Lube oil pump on CR-0301 makes noise and the pressure goes up and down. Replace the pump
coupling.'),
  ('WA03', 'WAIO', 'DZ-0204', 'Dozer', 'Tyres', 'P4', 1.0, 'SEED_DATA', '2026-10-05 10:00:00',
   'Dozer DZ-0204 track tension low on the right side. Adjust at next service.'),
  ('WA03', 'WAIO', 'CV-0330', 'Conveyor', 'Sensor or control', 'P3', 2.5, 'SEED_DATA', '2026-10-
05 20:00:00',
   'Belt rip detector on CV-0330 gives false alarms. The conveyor stops each time. Calibrate the
```

```
sensor.'),
  ('WA04', 'WAIO', 'SL-0101', 'Ship loader', 'Structural crack', 'P1', 20.0, 'SEED_DATA', '2026-
10-05 15:00:00',
   'Crack found on the boom of ship loader SL-0101 during the weekly inspection. Ship loading
stopped. Structural engineer must inspect. Report by inspector Noah Martin,
noah.martin@example.com.'),
  ('WA04', 'WAIO', 'SR-0202', 'Stacker reclaimer', 'Belt or idler', 'P2', 5.5, 'SEED_DATA',
'2026-10-06 01:00:00',
   'Stacker reclaimer SR-0202 boom conveyor belt has a damaged splice. Replace the splice before
the belt tears.'),
  ('WA04', 'WAIO', 'CV-0401', 'Conveyor', 'Electrical', 'P2', 4.0, 'SEED_DATA', '2026-10-06
11:00:00',
   'Drive motor of CV-0401 trips on overload when the belt starts with load. Check the soft
starter settings.'),
  ('WA04', 'WAIO', 'SL-0101', 'Ship loader', 'Hydraulics', 'P3', 3.0, 'SEED_DATA', '2026-10-06
21:00:00',
   'Luffing cylinder of SL-0101 has a small oil leak. Monitor and repair during the next planned
stop.'),
  ('WA04', 'WAIO', 'SR-0202', 'Stacker reclaimer', 'Sensor or control', 'P3', 1.5, 'SEED_DATA',
'2026-10-07 07:00:00',
   'Anti-collision sensor on SR-0202 is dirty with dust and gives warnings. Clean the sensor
lens.'),
  ('ES01', 'ESCONDIDA', 'HT-1102', 'Haul truck', 'Tyres', 'P1', 11.0, 'SEED_DATA', '2026-10-07
02:00:00',
   'Haul truck HT-1102 rear tyre pressure dropped fast on the main ramp. Truck stopped. Operator
Diego Morales (+56 9 8765 4321) reported a possible rock cut.'),
  ('ES01', 'ESCONDIDA', 'SH-1201', 'Shovel', 'Electrical', 'P1', 13.0, 'SEED_DATA', '2026-10-07
12:00:00',
   'Electric shovel SH-1201 lost power. The trailing cable has damage from a dozer track. Replace
the cable section and test the insulation.'),
  ('ES01', 'ESCONDIDA', 'EX-1301', 'Excavator', 'Hydraulics', 'P2', 6.0, 'SEED_DATA', '2026-10-07
22:00:00',
   'EX-1301 boom drifts down when the lever is in neutral. Check the boom holding valve.'),
  ('ES01', 'ESCONDIDA', 'DR-1105', 'Drill rig', 'Engine', 'P3', 4.0, 'SEED_DATA', '2026-10-08
08:00:00',
   'Drill DR-1105 engine starts with difficulty in the cold morning at high altitude. Check the
glow plugs and the battery.'),
  ('ES01', 'ESCONDIDA', 'HT-1102', 'Haul truck', 'Tyres', 'P2', 7.0, 'SEED_DATA', '2026-10-08
18:00:00',
   'HT-1102 has a second tyre problem this week, front left. Tyre wear is not even. Check the
suspension and the wheel alignment.'),
  ('ES02', 'ESCONDIDA', 'ML-2101', 'Grinding mill', 'Bearing', 'P1', 18.0, 'SEED_DATA', '2026-10-
08 13:00:00',
   'SAG mill ML-2101 trunnion bearing temperature is high. Mill throughput reduced. Lubrication
engineer Valentina Rojas, valentina.rojas@example.com, asks for an oil analysis.'),
  ('ES02', 'ESCONDIDA', 'CV-2203', 'Conveyor', 'Belt or idler', 'P2', 5.0, 'SEED_DATA', '2026-10-
08 23:00:00',
   'Ore feed conveyor CV-2203 belt is worn near the skirt boards. Replace the skirt rubber and
inspect the belt thickness.'),
  ('ES02', 'ESCONDIDA', 'PU-2301', 'Slurry pump', 'Bearing', 'P2', 6.0, 'SEED_DATA', '2026-10-09
09:00:00',
   'Slurry pump PU-2301 vibration is high. The bearing makes noise. Change over to the standby
pump and replace the bearing.'),
  ('ES02', 'ESCONDIDA', 'ML-2101', 'Grinding mill', 'Lubrication', 'P2', 4.5, 'SEED_DATA', '2026-
10-09 19:00:00',
   'Lube system filter on ML-2101 has high differential pressure. Change the filter element.'),
  ('ES02', 'ESCONDIDA', 'FL-2401', 'Flotation cell', 'Sensor or control', 'P3', 2.0, 'SEED_DATA',
'2026-10-10 05:00:00',
   'Level sensor on flotation cell FL-2401 reads wrong values. Recalibrate the sensor.'),
  ('ES03', 'ESCONDIDA', 'ML-3101', 'Grinding mill', 'Bearing', 'P2', 8.0, 'SEED_DATA', '2026-10-
10 00:00:00',
```

```
   'Ball mill ML-3101 pinion bearing temperature goes up slowly. Check alignment and
lubrication.'),
  ('ES03', 'ESCONDIDA', 'PU-3302', 'Slurry pump', 'Bearing', 'P1', 9.0, 'SEED_DATA', '2026-10-10
10:00:00',
   'Slurry pump PU-3302 failed. Bearing seized. Standby pump also has high vibration. Mechanic
Felipe Soto (+56 9 1234 5678) asks for spare bearings from the warehouse.'),
  ('ES03', 'ESCONDIDA', 'CV-3104', 'Conveyor', 'Electrical', 'P3', 3.0, 'SEED_DATA', '2026-10-10
20:00:00',
   'Pull-wire switch on CV-3104 is damaged and stops the belt. Replace the switch.'),
  ('ES03', 'ESCONDIDA', 'CR-3201', 'Crusher', 'Structural crack', 'P2', 10.0, 'SEED_DATA', '2026-
10-11 06:00:00',
   'Pebble crusher CR-3201 has a crack in the mainframe near the feed opening. Do a non-
destructive test.'),
  ('ES03', 'ESCONDIDA', 'PU-3302', 'Slurry pump', 'Lubrication', 'P3', 2.0, 'SEED_DATA', '2026-
10-11 16:00:00',
   'Grease lines to the PU-3302 bearings are blocked. Clean the lines and check the auto-lube
unit.'),
  ('ES04', 'ESCONDIDA', 'SL-4101', 'Ship loader', 'Electrical', 'P2', 6.0, 'SEED_DATA', '2026-10-
11 11:00:00',
   'Concentrate ship loader SL-4101 travel motor stops in high humidity. Check the cable
connectors for corrosion.'),
  ('ES04', 'ESCONDIDA', 'CV-4202', 'Conveyor', 'Belt or idler', 'P3', 3.0, 'SEED_DATA', '2026-10-
11 21:00:00',
   'Covered conveyor CV-4202 has seized return idlers. Replace four idlers.'),
  ('ES04', 'ESCONDIDA', 'PU-4301', 'Slurry pump', 'Bearing', 'P2', 5.0, 'SEED_DATA', '2026-10-12
07:00:00',
   'Pipeline booster pump PU-4301 bearing temperature is high. Shift lead Camila Fuentes,
camila.fuentes@example.com, reports a smell of hot oil.'),
  ('ES04', 'ESCONDIDA', 'SL-4101', 'Ship loader', 'Structural crack', 'P3', 2.0, 'SEED_DATA',
'2026-10-12 17:00:00',
   'Small corrosion crack on the SL-4101 walkway support. Repair at the next shutdown.'),
  ('ES04', 'ESCONDIDA', 'CV-4202', 'Conveyor', 'Sensor or control', 'P4', 1.0, 'SEED_DATA',
'2026-10-13 03:00:00',
   'Belt speed sensor on CV-4202 needs cleaning. No stop.'),
  ('BM01', 'BMA', 'DL-0101', 'Dragline', 'Structural crack', 'P1', 22.0, 'SEED_DATA', '2026-10-12
22:00:00',
   'Dragline DL-0101 boom chord has a crack near the lower lug. Dragline stopped. Structural
engineer Liam O''Brien (0423 678 901) will inspect today.'),
  ('BM01', 'BMA', 'HT-5104', 'Haul truck', 'Brakes', 'P2', 5.0, 'SEED_DATA', '2026-10-13
08:00:00',
   'HT-5104 retarder grid overheats on the long ramp. Check the cooling fan of the retarder.'),
  ('BM01', 'BMA', 'DZ-5201', 'Dozer', 'Hydraulics', 'P3', 3.0, 'SEED_DATA', '2026-10-13
18:00:00',
   'Dozer DZ-5201 ripper cylinder leaks oil. Replace the rod seal.'),
  ('BM01', 'BMA', 'CV-5301', 'Conveyor', 'Belt or idler', 'P2', 6.0, 'SEED_DATA', '2026-10-14
04:00:00',
   'Overland conveyor CV-5301 belt has a rip of 30 cm. Belt splicing crew needed.'),
  ('BM01', 'BMA', 'DL-0101', 'Dragline', 'Electrical', 'P2', 7.0, 'SEED_DATA', '2026-10-14
14:00:00',
   'Dragline DL-0101 hoist motor brush wear is high. Replace the brushes and clean the
commutator.'),
  ('BM02', 'BMA', 'HT-5211', 'Haul truck', 'Engine', 'P2', 8.0, 'SEED_DATA', '2026-10-14
09:00:00',
   'HT-5211 engine oil pressure is low at idle. Operator Chloe Nguyen, chloe.nguyen@example.com,
says the warning light came on twice.'),
  ('BM02', 'BMA', 'EX-5302', 'Excavator', 'Hydraulics', 'P1', 10.0, 'SEED_DATA', '2026-10-14
19:00:00',
   'Excavator EX-5302 main hydraulic pump makes a loud noise and the bucket is slow. Stop and
check the pump before more damage.'),
  ('BM02', 'BMA', 'DR-5401', 'Drill rig', 'Sensor or control', 'P3', 2.0, 'SEED_DATA', '2026-10-
15 05:00:00',
```

```
   'GPS guidance on drill DR-5401 loses signal near the high wall. Check the antenna.'),
  ('BM02', 'BMA', 'HT-5211', 'Haul truck', 'Engine', 'P1', 12.0, 'SEED_DATA', '2026-10-15
15:00:00',
   'HT-5211 stopped on the haul road with engine overheating. Coolant hose burst. Second engine
problem this month on this truck.'),
  ('BM02', 'BMA', 'CV-5501', 'Conveyor', 'Electrical', 'P3', 3.0, 'SEED_DATA', '2026-10-16
01:00:00',
   'Conveyor CV-5501 motor cooling fan is noisy. Replace the fan.'),
  ('BM03', 'BMA', 'DL-0301', 'Dragline', 'Electrical', 'P1', 15.0, 'SEED_DATA', '2026-10-15
20:00:00',
   'Dragline DL-0301 swing motor fault, fault code DS-410. Electrician Jack Thompson (0499 111
222) found a damaged cable at the house slip ring.'),
  ('BM03', 'BMA', 'HT-5306', 'Haul truck', 'Tyres', 'P2', 6.0, 'SEED_DATA', '2026-10-16
06:00:00',
   'HT-5306 tyre has a cut after driving over spillage at the dump. Clean the dump area and
change the tyre.'),
  ('BM03', 'BMA', 'CR-5601', 'Crusher', 'Bearing', 'P2', 9.0, 'SEED_DATA', '2026-10-16 16:00:00',
   'Sizer CR-5601 bearing is noisy. Vibration is above the alarm limit. Plan a bearing change.'),
  ('BM03', 'BMA', 'DZ-5303', 'Dozer', 'Engine', 'P3', 4.0, 'SEED_DATA', '2026-10-17 02:00:00',
   'Dozer DZ-5303 air filter blocks fast in dusty conditions. Change to the heavy duty filter.'),
  ('BM03', 'BMA', 'DL-0301', 'Dragline', 'Electrical', 'P2', 5.0, 'SEED_DATA', '2026-10-17
12:00:00',
   'DL-0301 swing motor fault again after the cable repair. Check the motor insulation.'),
  ('BM04', 'BMA', 'SL-6101', 'Ship loader', 'Hydraulics', 'P2', 6.0, 'SEED_DATA', '2026-10-17
07:00:00',
   'Ship loader SL-6101 slew hydraulic motor leaks oil onto the wharf. Contain the spill and
replace the motor seal. Contact wharf supervisor Olivia Smith, olivia.smith@example.com.'),
  ('BM04', 'BMA', 'SR-6201', 'Stacker reclaimer', 'Structural crack', 'P1', 19.0, 'SEED_DATA',
'2026-10-17 17:00:00',
   'Crack in the bucket wheel boom of stacker reclaimer SR-6201. Machine stopped. Engineer
inspection needed today.'),
  ('BM04', 'BMA', 'CV-6301', 'Conveyor', 'Belt or idler', 'P2', 5.0, 'SEED_DATA', '2026-10-18
03:00:00',
   'Yard conveyor CV-6301 has carryback coal on the return belt. Belt scraper is worn. Replace
the scraper blade.'),
  ('BM04', 'BMA', 'SR-6201', 'Stacker reclaimer', 'Electrical', 'P3', 3.0, 'SEED_DATA', '2026-10-
18 13:00:00',
   'SR-6201 long travel limit switch is damaged. Replace the switch.'),
  ('BM04', 'BMA', 'CV-6301', 'Conveyor', 'Sensor or control', 'P4', 1.0, 'SEED_DATA', '2026-10-18
23:00:00',
   'Weightometer on CV-6301 needs calibration. No stop.');

-- Step 2.5: Do a check. You must see 12 sites and 60 requests.
SELECT 'sites' AS table_name, COUNT(*) AS row_count FROM maint_lab.ops.sites
UNION ALL
SELECT 'maint_requests', COUNT(*) FROM maint_lab.ops.maint_requests;

-- Step 2.6: Look at the requests and the downtime for each asset.
SELECT asset,
       COUNT(*)                 AS requests,
       SUM(downtime_hours)      AS downtime_hours,
       COUNT_IF(priority = 'P1') AS p1_requests
FROM maint_lab.ops.maint_requests
GROUP BY asset
ORDER BY asset;
```

03_cortex_ai.sql

```
-- ============================================================================
-- PART 1 · LAB 3 · CORTEX AI ON THE MAINTENANCE REQUESTS
-- File: 03_cortex_ai.sql
-- Run as: ACCOUNTADMIN
-- Purpose: Use 5 Cortex AI functions in SQL. Save the AI results in the
--          MAINT_REQUESTS table. Make an insight view.
-- NOTE: Run one step at a time. Highlight the statement, then push
--       Ctrl+Enter (Windows) or Cmd+Return (Mac).
-- ============================================================================

-- Step 3.0: Set the role and the warehouse for this file.
USE ROLE ACCOUNTADMIN;
USE WAREHOUSE maint_wh;
USE SCHEMA maint_lab.ops;

-- ----------------------------------------------------------------------------
-- Step 3.1: AI_REDACT. Hide personal data (names, phone numbers, emails).
-- Result: the same text, with [NAME], [PHONE_NUMBER] and [EMAIL] tags.
-- ----------------------------------------------------------------------------
SELECT request_id,
       description,
       AI_REDACT(description) AS description_safe
FROM maint_lab.ops.maint_requests
WHERE request_id IN (1, 2, 5, 16, 21)
ORDER BY request_id;

-- ----------------------------------------------------------------------------
-- Step 3.2: AI_SUMMARIZE. Make one short summary of one long text.
-- Result: a short summary for each request.
-- NOTE: AI_SUMMARIZE is a Preview function. A text of 110 characters
--       or less comes back with no change.
-- ----------------------------------------------------------------------------
SELECT request_id,
       LENGTH(description)       AS characters,
       AI_SUMMARIZE(description) AS ai_summary
FROM maint_lab.ops.maint_requests
WHERE request_id IN (1, 6, 16, 26, 41)
ORDER BY request_id;

-- ----------------------------------------------------------------------------
-- Step 3.3: Save the safe text in the table (all 60 rows).
-- Wait about 1 minute.
-- ----------------------------------------------------------------------------
UPDATE maint_lab.ops.maint_requests
SET description_safe = AI_REDACT(description)
WHERE description_safe IS NULL;

-- ----------------------------------------------------------------------------
-- Step 3.4: Save a summary of the SAFE text (all 60 rows).
-- We summarise the safe text, so the summary has no personal data.
-- ----------------------------------------------------------------------------
UPDATE maint_lab.ops.maint_requests
SET ai_summary = AI_SUMMARIZE(description_safe)
WHERE ai_summary IS NULL;

-- Step 3.5: Do a check. All 60 rows must have a safe text and a summary.
SELECT COUNT(*)                    AS total_rows,
       COUNT(description_safe)     AS rows_with_safe_text,
```

```
       COUNT(ai_summary)           AS rows_with_summary,
       COUNT_IF(description_safe LIKE '%[NAME]%') AS rows_with_hidden_names
FROM maint_lab.ops.maint_requests;

-- ---------------------------------------------------------------------
-- Step 3.6: AI_SUMMARIZE_AGG. One summary from many rows.
-- Result: one summary for each asset.
-- ---------------------------------------------------------------------
SELECT asset,
       COUNT(*)                                AS requests,
       AI_SUMMARIZE_AGG(description_safe)      AS asset_summary
FROM maint_lab.ops.maint_requests
GROUP BY asset
ORDER BY asset;

-- ---------------------------------------------------------------------
-- Step 3.7: AI_AGG. Give your own instruction for many rows.
-- Result: the top 3 repeat problems and one action for each asset.
-- ---------------------------------------------------------------------
SELECT asset,
       AI_AGG(
         CONCAT(equipment_id, ' (', equipment_type, '): ', description_safe),
         'You are a maintenance reliability engineer at a mining company. Find the top 3 repeat
equipment problems in these requests. For each problem, give the equipment ID and one action. Use
short bullet points.'
       ) AS top_repeat_problems
FROM maint_lab.ops.maint_requests
GROUP BY asset
ORDER BY asset;

-- ---------------------------------------------------------------------
-- Step 3.8: SPLIT_TEXT_RECURSIVE_CHARACTER. Cut a long text into chunks.
-- Long documents (shift handover notes, manuals) are cut into small
-- parts before AI search. Result: one row for each chunk.
-- ---------------------------------------------------------------------
SELECT c.index       AS chunk_number,
       c.value::STRING AS chunk_text
FROM TABLE(FLATTEN(input => SNOWFLAKE.CORTEX.SPLIT_TEXT_RECURSIVE_CHARACTER(
  'Shift handover, night shift, Mt Newman. HT-0412 is parked at the go-line with a hydraulic leak
at the left hoist cylinder. '
  || 'The seal kit is in the warehouse, bin 14. CV-0110 idlers at transfer point 3 are noisy;
replace them at the planned shutdown. '
  || 'EX-0203 had coolant temperature alarms; the radiator was cleaned at 02:00 and the alarms
stopped. '
  || 'Safety: one tyre cut on HT-0418 from rock spillage on the main ramp; the ramp was cleaned.
No injuries.',
  'none', 200, 30))) c;

-- ---------------------------------------------------------------------
-- Step 3.9: Make an insight view. The Streamlit app uses it in Part 2.
-- ---------------------------------------------------------------------
CREATE OR REPLACE VIEW maint_lab.ops.v_site_insight AS
SELECT s.asset,
       s.site_id,
       s.site_name,
       COUNT(r.request_id)               AS requests,
       COALESCE(SUM(r.downtime_hours), 0) AS downtime_hours,
       COUNT_IF(r.priority = 'P1')       AS p1_requests,
       MODE(r.failure_mode)              AS top_failure_mode
FROM maint_lab.ops.sites s
LEFT JOIN maint_lab.ops.maint_requests r
```

```
  ON r.site_id = s.site_id
GROUP BY s.asset, s.site_id, s.site_name;

-- Step 3.10: Look at the sites with the most downtime.
SELECT *
FROM maint_lab.ops.v_site_insight
ORDER BY downtime_hours DESC;
```

04_security.sql

```
-- ======================================================================
-- PART 1 · LAB 4 · USERS, GROUPS (ROLES) AND ROW-LEVEL SECURITY
-- File: 04_security.sql
-- Run as: ACCOUNTADMIN
-- Purpose: Snowflake roles are the "groups". An asset role gives a user
--          access to the rows of that asset. A row access policy
--          applies the rule to every query: in worksheets, in views,
--          and (in Part 2) in the Streamlit app.
-- ======================================================================

-- Step 4.0: Set the role and the warehouse for this file.
USE ROLE ACCOUNTADMIN;
USE WAREHOUSE maint_wh;

-- ----------------------------------------------------------------------
-- Step 4.1: Create the group roles.
--   MAINT_APP_USER  = common access for all app users
--   MAINT_HQ        = group head office, sees all assets
--   MAINT_WAIO, MAINT_ESCONDIDA, MAINT_BMA = one asset each
-- ----------------------------------------------------------------------
CREATE ROLE IF NOT EXISTS maint_app_user  COMMENT = 'Common access for Asset Care app users';
CREATE ROLE IF NOT EXISTS maint_hq        COMMENT = 'Group: head office, all assets';
CREATE ROLE IF NOT EXISTS maint_waio      COMMENT = 'Group: Western Australia Iron Ore';
CREATE ROLE IF NOT EXISTS maint_escondida COMMENT = 'Group: Escondida';
CREATE ROLE IF NOT EXISTS maint_bma       COMMENT = 'Group: BHP Mitsubishi Alliance';

-- Each group role gets the common access.
GRANT ROLE maint_app_user TO ROLE maint_hq;
GRANT ROLE maint_app_user TO ROLE maint_waio;
GRANT ROLE maint_app_user TO ROLE maint_escondida;
GRANT ROLE maint_app_user TO ROLE maint_bma;

-- Step 4.2: Give the common role access to the warehouse and the data.
GRANT USAGE  ON WAREHOUSE maint_wh                    TO ROLE maint_app_user;
GRANT USAGE  ON DATABASE  maint_lab                   TO ROLE maint_app_user;
GRANT USAGE  ON SCHEMA    maint_lab.ops               TO ROLE maint_app_user;
GRANT USAGE  ON SCHEMA    maint_lab.app               TO ROLE maint_app_user;
GRANT SELECT ON ALL TABLES IN SCHEMA maint_lab.ops    TO ROLE maint_app_user;
GRANT SELECT ON ALL VIEWS  IN SCHEMA maint_lab.ops    TO ROLE maint_app_user;

-- ----------------------------------------------------------------------
-- Step 4.3: Create 4 demo users. Each user is in one group.
-- CAUTION: This is a training password. Do not use it in production.
-- ----------------------------------------------------------------------
CREATE USER IF NOT EXISTS olivia_waio
  PASSWORD = 'AssetCare#Lab2026' MUST_CHANGE_PASSWORD = FALSE TYPE = PERSON
  DEFAULT_ROLE = maint_waio DEFAULT_WAREHOUSE = maint_wh
  COMMENT = 'Demo user: WAIO maintenance planner';
CREATE USER IF NOT EXISTS diego_esc
```

```
  PASSWORD = 'AssetCare#Lab2026' MUST_CHANGE_PASSWORD = FALSE TYPE = PERSON
  DEFAULT_ROLE = maint_escondida DEFAULT_WAREHOUSE = maint_wh
  COMMENT = 'Demo user: Escondida maintenance planner';
CREATE USER IF NOT EXISTS liam_bma
  PASSWORD = 'AssetCare#Lab2026' MUST_CHANGE_PASSWORD = FALSE TYPE = PERSON
  DEFAULT_ROLE = maint_bma DEFAULT_WAREHOUSE = maint_wh
  COMMENT = 'Demo user: BMA maintenance planner';
CREATE USER IF NOT EXISTS priya_hq
  PASSWORD = 'AssetCare#Lab2026' MUST_CHANGE_PASSWORD = FALSE TYPE = PERSON
  DEFAULT_ROLE = maint_hq DEFAULT_WAREHOUSE = maint_wh
  COMMENT = 'Demo user: head office reliability analyst';

-- Step 4.4: Put each user in a group (grant the role to the user).
GRANT ROLE maint_waio      TO USER olivia_waio;
GRANT ROLE maint_escondida TO USER diego_esc;
GRANT ROLE maint_bma       TO USER liam_bma;
GRANT ROLE maint_hq        TO USER priya_hq;

-- Step 4.5: Put YOUR user in the HQ group. Then you can see all rows.
-- CAUTION: Do this step. If you do not, you see 0 rows after Step 4.10.
EXECUTE IMMEDIATE $$
BEGIN
  EXECUTE IMMEDIATE 'GRANT ROLE maint_hq TO USER "' || CURRENT_USER() || '"';
  RETURN 'MAINT_HQ is granted to ' || CURRENT_USER();
END;
$$;

-- ------------------------------------------------------------------
-- Step 4.6: Create the access rule table: which group sees which asset.
-- 'ALL' means all assets.
-- ------------------------------------------------------------------
CREATE OR REPLACE TABLE maint_lab.security.role_asset_map (
  role_name VARCHAR(100) NOT NULL,
  asset     VARCHAR(20)  NOT NULL
);

INSERT INTO maint_lab.security.role_asset_map (role_name, asset) VALUES
  ('MAINT_HQ',        'ALL'),
  ('MAINT_WAIO',      'WAIO'),
  ('MAINT_ESCONDIDA', 'ESCONDIDA'),
  ('MAINT_BMA',       'BMA');

-- Step 4.7: Create the table that the policy reads: user to asset.
-- A procedure fills it from the real role grants (next step).
CREATE OR REPLACE TABLE maint_lab.security.user_asset_map (
  user_name VARCHAR(100) NOT NULL,
  asset     VARCHAR(20)  NOT NULL,
  via_role  VARCHAR(100) NOT NULL,
  synced_at TIMESTAMP_NTZ DEFAULT CURRENT_TIMESTAMP()
);

-- ------------------------------------------------------------------
-- Step 4.8: Create the sync procedure. It reads "who has which group
-- role" from Snowflake and writes the result to USER_ASSET_MAP.
-- Run it again each time you add or remove a user from a group.
-- ------------------------------------------------------------------
CREATE OR REPLACE PROCEDURE maint_lab.security.sync_user_assets()
RETURNS VARCHAR
LANGUAGE SQL
EXECUTE AS CALLER
AS
```

```
$$
DECLARE
  c1 CURSOR FOR SELECT role_name, asset FROM maint_lab.security.role_asset_map;
  total_rows INTEGER DEFAULT 0;
BEGIN
  TRUNCATE TABLE maint_lab.security.user_asset_map;
  FOR rec IN c1 DO
    LET v_role  VARCHAR := rec.role_name;
    LET v_asset VARCHAR := rec.asset;
    -- List all users and roles that have this group role.
    EXECUTE IMMEDIATE 'SHOW GRANTS OF ROLE ' || v_role;
    -- Keep only the users.
    INSERT INTO maint_lab.security.user_asset_map (user_name, asset, via_role)
      SELECT "grantee_name", :v_asset, :v_role
      FROM TABLE(RESULT_SCAN(LAST_QUERY_ID()))
      WHERE "granted_to" = 'USER';
    total_rows := total_rows + SQLROWCOUNT;
  END FOR;
  RETURN 'User-asset rows written: ' || total_rows;
END;
$$;

-- Step 4.9: Run the sync. You must see "User-asset rows written: 5".
CALL maint_lab.security.sync_user_assets();

SELECT * FROM maint_lab.security.user_asset_map ORDER BY asset, user_name;

-- ------------------------------------------------------------------
-- Step 4.10: Create the row access policy and attach it to both tables.
-- The rule: show a row only if the current user has the asset of the
-- row, or has 'ALL'.
-- ------------------------------------------------------------------
CREATE OR REPLACE ROW ACCESS POLICY maint_lab.security.asset_rap
AS (row_asset VARCHAR) RETURNS BOOLEAN ->
  EXISTS (
    SELECT 1
    FROM maint_lab.security.user_asset_map m
    WHERE m.user_name = CURRENT_USER()
      AND (m.asset = row_asset OR m.asset = 'ALL')
  );

ALTER TABLE maint_lab.ops.sites          ADD ROW ACCESS POLICY maint_lab.security.asset_rap ON
(asset);
ALTER TABLE maint_lab.ops.maint_requests ADD ROW ACCESS POLICY maint_lab.security.asset_rap ON
(asset);

-- ------------------------------------------------------------------
-- Step 4.11: Test the policy. POLICY_CONTEXT shows the result
-- "as if" another user runs the query. You do not have to log in again.
-- ------------------------------------------------------------------
-- You (HQ): 3 assets, 60 rows.
SELECT asset, COUNT(*) AS rows_seen
FROM maint_lab.ops.maint_requests GROUP BY asset ORDER BY asset;

-- OLIVIA_WAIO: only WAIO, 20 rows.
EXECUTE USING POLICY_CONTEXT(CURRENT_USER => 'OLIVIA_WAIO')
AS SELECT asset, COUNT(*) AS rows_seen FROM maint_lab.ops.maint_requests GROUP BY asset;

-- DIEGO_ESC: only ESCONDIDA, 20 rows.
EXECUTE USING POLICY_CONTEXT(CURRENT_USER => 'DIEGO_ESC')
AS SELECT asset, COUNT(*) AS rows_seen FROM maint_lab.ops.maint_requests GROUP BY asset;
```

```
-- PRIYA_HQ: 3 assets, 60 rows.
EXECUTE USING POLICY_CONTEXT(CURRENT_USER => 'PRIYA_HQ')
AS SELECT asset, COUNT(*) AS rows_seen FROM maint_lab.ops.maint_requests GROUP BY asset;

-- The view also obeys the policy. LIAM_BMA sees only the 4 BMA sites.
EXECUTE USING POLICY_CONTEXT(CURRENT_USER => 'LIAM_BMA')
AS SELECT asset, site_name, requests, downtime_hours FROM maint_lab.ops.v_site_insight;
```

05_app_owner.sql

```
-- ============================================================================
-- PART 2 · LAB 5 · THE APP OWNER ROLE AND THE APP
-- File: 05_app_owner.sql
-- Run as: ACCOUNTADMIN
-- Purpose: Create the role that owns the Streamlit app. The app runs
--          with the privileges of this role ("owner's rights").
--          Then create the app on the warehouse runtime.
-- ============================================================================

-- Step 5.0: Set the role and the warehouse for this file.
USE ROLE ACCOUNTADMIN;
USE WAREHOUSE maint_wh;

-- Step 5.1: Create the owner role.
CREATE ROLE IF NOT EXISTS maint_app_owner COMMENT = 'Owns the Asset Care Streamlit app';
GRANT ROLE maint_app_owner TO ROLE SYSADMIN;

-- Step 5.2: Give access to the warehouse, the database and the schemas.
GRANT USAGE ON WAREHOUSE maint_wh           TO ROLE maint_app_owner;
GRANT USAGE ON DATABASE  maint_lab          TO ROLE maint_app_owner;
GRANT USAGE ON SCHEMA    maint_lab.ops      TO ROLE maint_app_owner;
GRANT USAGE ON SCHEMA    maint_lab.security TO ROLE maint_app_owner;
GRANT USAGE ON SCHEMA    maint_lab.app      TO ROLE maint_app_owner;

-- Step 5.3: Give access to the data. The app reads and writes MAINT_REQUESTS.
GRANT SELECT, INSERT ON TABLE maint_lab.ops.maint_requests       TO ROLE maint_app_owner;
GRANT SELECT         ON TABLE maint_lab.ops.sites                TO ROLE maint_app_owner;
GRANT SELECT         ON VIEW  maint_lab.ops.v_site_insight       TO ROLE maint_app_owner;
GRANT SELECT         ON TABLE maint_lab.security.user_asset_map  TO ROLE maint_app_owner;

-- Step 5.4: Let the role create a Streamlit app in the APP schema.
GRANT CREATE STREAMLIT ON SCHEMA maint_lab.app TO ROLE maint_app_owner;

-- Step 5.5: Let the role use Cortex AI functions.
-- NOTE: By default, the PUBLIC role already has the USE AI FUNCTIONS
-- privilege. This grant makes sure that the owner role can use Cortex.
GRANT DATABASE ROLE SNOWFLAKE.CORTEX_USER TO ROLE maint_app_owner;

-- Step 5.6: IMPORTANT. Let the app read the session of the viewer.
-- Without this grant, CURRENT_USER() in the app does not give the
-- viewer, and the row access policy cannot filter for each user.
GRANT READ SESSION ON ACCOUNT TO ROLE maint_app_owner;

-- Step 5.7: Give the owner role to YOUR user.
EXECUTE IMMEDIATE $$
BEGIN
  EXECUTE IMMEDIATE 'GRANT ROLE maint_app_owner TO USER "' || CURRENT_USER() || '"';
  RETURN 'MAINT_APP_OWNER is granted to ' || CURRENT_USER();
```

```
END;
$$;

-- Step 5.8: Do a check. The list must show USAGE, INSERT, SELECT,
-- CREATE STREAMLIT, READ SESSION and a row for SNOWFLAKE.CORTEX_USER.
SHOW GRANTS TO ROLE maint_app_owner;

-- Step 5.9: Change to the owner role. The owner role creates the app.
USE ROLE maint_app_owner;
USE WAREHOUSE maint_wh;

-- Step 5.10: Create the Streamlit app on the WAREHOUSE runtime.
-- RUNTIME_NAME makes sure that the app does not use a compute pool.
-- CAUTION: Run this step one time only. If you run it again later,
--          it replaces the app and removes your code.
CREATE OR REPLACE STREAMLIT maint_lab.app.asset_care
  RUNTIME_NAME = 'SYSTEM$WAREHOUSE_RUNTIME'
  QUERY_WAREHOUSE = maint_wh
  TITLE = 'Asset Care Requests'
  COMMENT = 'Maintenance request app with Cortex AI and row-level security';

-- Step 5.11: Make the app active.
ALTER STREAMLIT maint_lab.app.asset_care ADD LIVE VERSION FROM LAST;

-- Step 5.12: Do a check. You must see one row: ASSET_CARE,
-- title "Asset Care Requests", query_warehouse MAINT_WH.
SHOW STREAMLITS IN SCHEMA maint_lab.app;
```

06_share_app.sql

```
-- ========================================================================
-- PART 2 · LAB 8 · SHARE THE APP WITH THE GROUPS
-- File: 06_share_app.sql
-- Run as: MAINT_APP_OWNER (the role that owns the app)
-- Purpose: Give the app to all users in the maintenance groups.
-- ========================================================================

-- Step 8.1: Use the owner role.
USE ROLE maint_app_owner;
USE WAREHOUSE maint_wh;

-- Step 8.2: Share the app with MAINT_APP_USER.
-- All group roles (MAINT_HQ, MAINT_WAIO, MAINT_ESCONDIDA, MAINT_BMA)
-- have MAINT_APP_USER, so all demo users get the app.
GRANT USAGE ON STREAMLIT maint_lab.app.asset_care TO ROLE maint_app_user;

-- Step 8.3: Do a check. The list must show USAGE on STREAMLIT
-- MAINT_LAB.APP.ASSET_CARE.
SHOW GRANTS TO ROLE maint_app_user;
```

07_change_group.sql

```
-- ========================================================================
-- PART 2 · LAB 9 · CHANGE A GROUP, SEE THE APP CHANGE
-- File: 07_change_group.sql
-- Run as: ACCOUNTADMIN
```

```
-- Purpose: Show that the app follows Snowflake user management.
--          You change a role grant in Snowflake. The app changes too.
-- ========================================================================

-- Step 9.0: Set the role and the warehouse for this file.
USE ROLE ACCOUNTADMIN;
USE WAREHOUSE maint_wh;

-- Step 9.1: Put OLIVIA_WAIO also in the BMA group.
GRANT ROLE maint_bma TO USER olivia_waio;

-- Step 9.2: Sync the groups to the access table.
CALL maint_lab.security.sync_user_assets();

-- Step 9.3: Do a check. OLIVIA_WAIO now has WAIO and BMA.
SELECT * FROM maint_lab.security.user_asset_map WHERE user_name = 'OLIVIA_WAIO';

EXECUTE USING POLICY_CONTEXT(CURRENT_USER => 'OLIVIA_WAIO')
AS SELECT asset, COUNT(*) AS rows_seen FROM maint_lab.ops.maint_requests GROUP BY asset;

-- Step 9.4: In the app window of OLIVIA_WAIO, refresh the page.
-- The sidebar shows 2 assets. The Insights tab shows 2 assets.

-- Step 9.5: Remove OLIVIA_WAIO from the BMA group again.
REVOKE ROLE maint_bma FROM USER olivia_waio;
CALL maint_lab.security.sync_user_assets();

EXECUTE USING POLICY_CONTEXT(CURRENT_USER => 'OLIVIA_WAIO')
AS SELECT asset, COUNT(*) AS rows_seen FROM maint_lab.ops.maint_requests GROUP BY asset;
```

streamlit_app_v2.py

```
# ========================================================================
# Asset Care Requests · PART 2 · APP VERSION 2: save requests with Cortex AI
# Streamlit in Snowflake, warehouse runtime, owner's rights.
#
# What the app does:
#   1. Shows who you are and which assets your Snowflake groups give you.
#   2. Saves a new maintenance request in Snowflake. Cortex AI hides the
#      personal data (AI_REDACT) and writes a short summary (AI_SUMMARIZE).
#   3. Shows the requests that you can see.
# ========================================================================
import streamlit as st
from snowflake.snowpark.context import get_active_session

st.set_page_config(layout="wide")

# The Snowflake session of the app. No password is necessary.
session = get_active_session()

# ------------------------------------------------------------------------
# Settings
# ------------------------------------------------------------------------
EQUIPMENT_TYPES = [
    "Haul truck", "Excavator", "Shovel", "Dragline", "Dozer", "Drill rig", "Conveyor",
    "Crusher", "Grinding mill", "Slurry pump", "Flotation cell", "Ship loader", "Stacker
reclaimer",
]
FAILURE_MODES = [
```

```
    "Bearing", "Belt or idler", "Brakes", "Electrical", "Engine", "Hydraulics",
    "Lubrication", "Sensor or control", "Structural crack", "Tyres",
]

# SQL to save one request. The ? marks are bind variables.
# AI_REDACT hides personal data. AI_SUMMARIZE writes a summary of the safe text.
# The JOIN to SITES makes sure that the user can only save a request for a
# site that the row access policy lets him or her see.
INSERT_SQL = """
INSERT INTO maint_lab.ops.maint_requests
  (site_id, asset, equipment_id, equipment_type, failure_mode, priority,
   downtime_hours, submitted_by, submitted_at, description,
   description_safe, ai_summary)
SELECT s.site_id,
       s.asset,
       y.eq_id,
       y.eq_type,
       y.fmode,
       y.prio,
       y.hours,
       CURRENT_USER(),
       CURRENT_TIMESTAMP()::TIMESTAMP_NTZ,
       y.txt,
       y.safe,
       AI_SUMMARIZE(y.safe)
FROM (SELECT x.*, AI_REDACT(x.txt) AS safe
      FROM (SELECT ? AS sid, ? AS eq_id, ? AS eq_type, ? AS fmode,
                   ? AS prio, ? AS hours, ? AS txt) x) y
JOIN maint_lab.ops.sites s
  ON s.site_id = y.sid
"""

def run_query(sql, params=None):
    """Run a query and give the result as a pandas DataFrame."""
    return session.sql(sql, params=params).to_pandas()


# ------------------------------------------------------------------------
# Header and sidebar: who is the viewer?
# CURRENT_USER() gives the viewer, because the owner role has the
# READ SESSION privilege (Lab 5).
# ------------------------------------------------------------------------
me = run_query("SELECT CURRENT_USER() AS user_name").iloc[0]["USER_NAME"]
access = run_query(
    """SELECT asset, via_role
       FROM maint_lab.security.user_asset_map
       WHERE user_name = CURRENT_USER()
       ORDER BY asset"""
)

st.title("Asset Care Requests")
st.caption("Maintenance requests with Snowflake Cortex AI. Your data access follows your
Snowflake groups.")

with st.sidebar:
    st.subheader("Signed in")
    st.write(f"User: **{me}**")
    if access.empty:
        st.error("You are not in a maintenance group. Speak to the administrator.")
    else:
        st.write("Your assets:")
```

```
        for row in access.itertuples():
            st.write(f"- {row.ASSET} (group {row.VIA_ROLE})")

tab_new, tab_data = st.tabs(["New request", "My data"])

# ---------------------------------------------------------------------
# Tab 1: New request (input saved in Snowflake)
# ---------------------------------------------------------------------
with tab_new:
    st.subheader("New maintenance request")
    sites = run_query(
        """SELECT site_id, site_name, asset
           FROM maint_lab.ops.sites
           ORDER BY site_id"""
    )
    if sites.empty:
        st.warning("You cannot see any site. Speak to the administrator.")
    else:
        labels = {
            row.SITE_ID: f"{row.SITE_ID} · {row.SITE_NAME} ({row.ASSET})"
            for row in sites.itertuples()
        }
        with st.form("request_form", clear_on_submit=True):
            site_id = st.selectbox("Site", options=list(labels.keys()),
                                   format_func=lambda k: labels[k])
            c1, c2 = st.columns(2)
            equipment_id = c1.text_input("Equipment ID", max_chars=20, placeholder="Example: HT-
0412")
            equipment_type = c2.selectbox("Equipment type", EQUIPMENT_TYPES)
            c3, c4, c5 = st.columns(3)
            failure_mode = c3.selectbox("Failure mode", FAILURE_MODES)
            priority = c4.radio("Priority", ["P1", "P2", "P3", "P4"], index=1, horizontal=True)
            downtime = c5.number_input("Downtime (hours)", min_value=0.0, max_value=999.0,
                                       value=1.0, step=0.5)
            description = st.text_area(
                "Description", max_chars=4000,
                placeholder="What happened, where, and what is needed. You can write a name or a
phone number: AI hides it.")
            submitted = st.form_submit_button("Save request", type="primary")

        if submitted:
            if not equipment_id.strip() or not description.strip():
                st.error("Type the equipment ID and the description. Then push Save request
again.")
            else:
                with st.spinner("Saving the request and running Cortex AI..."):
                    result = session.sql(
                        INSERT_SQL,
                        params=[site_id, equipment_id.strip().upper(), equipment_type,
failure_mode,
                                priority, float(downtime), description.strip()],
                    ).collect()
                inserted = result[0][0] if result else 0
                if inserted == 1:
                    st.success("The request is saved in Snowflake.")
                    latest = run_query(
                        """SELECT request_id, site_id, equipment_id, priority,
                                  description_safe, ai_summary
                           FROM maint_lab.ops.maint_requests
                           WHERE submitted_by = CURRENT_USER()
                           ORDER BY request_id DESC
```

```
                            LIMIT 1"""
                    )
                    st.write("Cortex AI result for your request:")
                    st.dataframe(latest, hide_index=True)
                else:
                    st.error("The request is not saved. You do not have access to this site.")

# ---------------------------------------------------------------------
# Tab: My data. The row access policy filters these queries.
# ---------------------------------------------------------------------
with tab_data:
    st.subheader("Requests that you can see")
    st.dataframe(
        run_query(
            """SELECT asset, COUNT(*) AS requests, SUM(downtime_hours) AS downtime_hours
               FROM maint_lab.ops.maint_requests
               GROUP BY asset
               ORDER BY asset"""
        ),
        hide_index=True,
    )
    st.dataframe(
        run_query(
            """SELECT request_id, site_id, equipment_id, priority, submitted_by,
                      ai_summary, description_safe
               FROM maint_lab.ops.maint_requests
               ORDER BY request_id DESC
               LIMIT 20"""
        ),
        hide_index=True,
    )
```

streamlit_app_v3.py

```
# ======================================================================
# Asset Care Requests · PART 2 · FINAL APP (version 3)
# Streamlit in Snowflake, warehouse runtime, owner's rights.
#
# What the app does:
#   1. Shows who you are and which assets your Snowflake groups give you.
#   2. Saves a new maintenance request in Snowflake. Cortex AI hides the
#      personal data (AI_REDACT) and writes a short summary (AI_SUMMARIZE).
#   3. Shows insights. The row access policy filters every query, so each
#      user sees only the assets of his or her groups.
#   4. Runs AI analysis on the requests that you can see (AI_AGG and
#      AI_SUMMARIZE_AGG).
# ======================================================================
import streamlit as st
from snowflake.snowpark.context import get_active_session

st.set_page_config(layout="wide")

# The Snowflake session of the app. No password is necessary.
session = get_active_session()

# ---------------------------------------------------------------------
# Settings
# ---------------------------------------------------------------------
EQUIPMENT_TYPES = [
```

```
    "Haul truck", "Excavator", "Shovel", "Dragline", "Dozer", "Drill rig", "Conveyor",
    "Crusher", "Grinding mill", "Slurry pump", "Flotation cell", "Ship loader", "Stacker
reclaimer",
]
FAILURE_MODES = [
    "Bearing", "Belt or idler", "Brakes", "Electrical", "Engine", "Hydraulics",
    "Lubrication", "Sensor or control", "Structural crack", "Tyres",
]

# SQL to save one request. The ? marks are bind variables.
# AI_REDACT hides personal data. AI_SUMMARIZE writes a summary of the safe text.
# The JOIN to SITES makes sure that the user can only save a request for a
# site that the row access policy lets him or her see.
INSERT_SQL = """
INSERT INTO maint_lab.ops.maint_requests
  (site_id, asset, equipment_id, equipment_type, failure_mode, priority,
   downtime_hours, submitted_by, submitted_at, description,
   description_safe, ai_summary)
SELECT s.site_id,
       s.asset,
       y.eq_id,
       y.eq_type,
       y.fmode,
       y.prio,
       y.hours,
       CURRENT_USER(),
       CURRENT_TIMESTAMP()::TIMESTAMP_NTZ,
       y.txt,
       y.safe,
       AI_SUMMARIZE(y.safe)
FROM (SELECT x.*, AI_REDACT(x.txt) AS safe
      FROM (SELECT ? AS sid, ? AS eq_id, ? AS eq_type, ? AS fmode,
                   ? AS prio, ? AS hours, ? AS txt) x) y
JOIN maint_lab.ops.sites s
  ON s.site_id = y.sid
"""

# Fixed AI instructions for the AI analysis tab. The user selects one.
# The instructions are fixed text, so user input never goes into an AI prompt.
ANALYSES = {
    "Top 3 repeat problems and actions":
        "You are a maintenance reliability engineer at a mining company. Find the top 3 repeat
equipment problems in these requests. For each problem, give the equipment ID and one action. Use
short bullet points.",
    "Which equipment needs attention first?":
        "You are a maintenance planner at a mining company. Name the 3 pieces of equipment that
need attention first and say why, based on priority, downtime and repeat failures. Use short
bullet points.",
    "Safety risks in the requests":
        "You are a safety advisor at a mining company. List the safety risks that these
maintenance requests show, for example structural cracks, tyre damage, brake problems or oil
spills. Use short bullet points.",
}


def run_query(sql, params=None):
    """Run a query and give the result as a pandas DataFrame."""
    return session.sql(sql, params=params).to_pandas()


# ------------------------------------------------------------------
```

```
# Header and sidebar: who is the viewer?
# CURRENT_USER() gives the viewer, because the owner role has the
# READ SESSION privilege (Lab 5).
# ------------------------------------------------------------------
me = run_query("SELECT CURRENT_USER() AS user_name").iloc[0]["USER_NAME"]
access = run_query(
    """SELECT asset, via_role
       FROM maint_lab.security.user_asset_map
       WHERE user_name = CURRENT_USER()
       ORDER BY asset"""
)

st.title("Asset Care Requests")
st.caption("Maintenance requests with Snowflake Cortex AI. Your data access follows your
Snowflake groups.")

with st.sidebar:
    st.subheader("Signed in")
    st.write(f"User: **{me}**")
    if access.empty:
        st.error("You are not in a maintenance group. Speak to the administrator.")
    else:
        st.write("Your assets:")
        for row in access.itertuples():
            st.write(f"- {row.ASSET} (group {row.VIA_ROLE})")

tab_new, tab_insight, tab_ai = st.tabs(["New request", "Insights", "AI analysis"])

# ------------------------------------------------------------------
# Tab 1: New request (input saved in Snowflake)
# ------------------------------------------------------------------
with tab_new:
    st.subheader("New maintenance request")
    sites = run_query(
        """SELECT site_id, site_name, asset
           FROM maint_lab.ops.sites
           ORDER BY site_id"""
    )
    if sites.empty:
        st.warning("You cannot see any site. Speak to the administrator.")
    else:
        labels = {
            row.SITE_ID: f"{row.SITE_ID} · {row.SITE_NAME} ({row.ASSET})"
            for row in sites.itertuples()
        }
        with st.form("request_form", clear_on_submit=True):
            site_id = st.selectbox("Site", options=list(labels.keys()),
                                   format_func=lambda k: labels[k])
            c1, c2 = st.columns(2)
            equipment_id = c1.text_input("Equipment ID", max_chars=20, placeholder="Example: HT-
0412")
            equipment_type = c2.selectbox("Equipment type", EQUIPMENT_TYPES)
            c3, c4, c5 = st.columns(3)
            failure_mode = c3.selectbox("Failure mode", FAILURE_MODES)
            priority = c4.radio("Priority", ["P1", "P2", "P3", "P4"], index=1, horizontal=True)
            downtime = c5.number_input("Downtime (hours)", min_value=0.0, max_value=999.0,
                                       value=1.0, step=0.5)
            description = st.text_area(
                "Description", max_chars=4000,
                placeholder="What happened, where, and what is needed. You can write a name or a
phone number: AI hides it.")
```

```
            submitted = st.form_submit_button("Save request", type="primary")

        if submitted:
            if not equipment_id.strip() or not description.strip():
                st.error("Type the equipment ID and the description. Then push Save request
again.")
            else:
                with st.spinner("Saving the request and running Cortex AI..."):
                    result = session.sql(
                        INSERT_SQL,
                        params=[site_id, equipment_id.strip().upper(), equipment_type,
failure_mode,
                                priority, float(downtime), description.strip()],
                    ).collect()
                inserted = result[0][0] if result else 0
                if inserted == 1:
                    st.success("The request is saved in Snowflake.")
                    latest = run_query(
                        """SELECT request_id, site_id, equipment_id, priority,
                                  description_safe, ai_summary
                           FROM maint_lab.ops.maint_requests
                           WHERE submitted_by = CURRENT_USER()
                           ORDER BY request_id DESC
                           LIMIT 1"""
                    )
                    st.write("Cortex AI result for your request:")
                    st.dataframe(latest, hide_index=True)
                else:
                    st.error("The request is not saved. You do not have access to this site.")

# ---------------------------------------------------------------------
# Tab 2: Insights (all queries are filtered by the row access policy)
# ---------------------------------------------------------------------
with tab_insight:
    st.subheader("Insights for your assets")
    kpi = run_query(
        """SELECT COUNT(*)                     AS requests,
                  COALESCE(SUM(downtime_hours), 0) AS downtime,
                  COUNT_IF(priority = 'P1')    AS p1,
                  COUNT(DISTINCT site_id)      AS sites
           FROM maint_lab.ops.maint_requests"""
    ).iloc[0]
    k1, k2, k3, k4 = st.columns(4)
    k1.metric("Requests", int(kpi["REQUESTS"]))
    k2.metric("Downtime (hours)", float(kpi["DOWNTIME"]))
    k3.metric("P1 requests", int(kpi["P1"]))
    k4.metric("Sites", int(kpi["SITES"]))

    left, right = st.columns(2)
    with left:
        st.markdown("**Downtime by site (hours)**")
        by_site = run_query(
            """SELECT s.site_name, SUM(r.downtime_hours) AS downtime_hours
               FROM maint_lab.ops.maint_requests r
               JOIN maint_lab.ops.sites s ON s.site_id = r.site_id
               GROUP BY s.site_name
               ORDER BY 2 DESC"""
        )
        if not by_site.empty:
            st.bar_chart(by_site, x="SITE_NAME", y="DOWNTIME_HOURS")
    with right:
```

```
        st.markdown("**Requests by failure mode and priority**")
        by_mode = run_query(
            """SELECT failure_mode, priority, COUNT(*) AS requests
               FROM maint_lab.ops.maint_requests
               GROUP BY 1, 2"""
        )
        if not by_mode.empty:
            wide = by_mode.pivot_table(index="FAILURE_MODE", columns="PRIORITY",
                                       values="REQUESTS", aggfunc="sum", fill_value=0)
            st.bar_chart(wide)

    st.markdown("**P1 requests (newest first). Personal data is hidden by AI_REDACT.**")
    p1 = run_query(
        """SELECT request_id, site_id, equipment_id, equipment_type, downtime_hours,
                  ai_summary, description_safe
           FROM maint_lab.ops.maint_requests
           WHERE priority = 'P1'
           ORDER BY submitted_at DESC
           LIMIT 10"""
    )
    st.dataframe(p1, hide_index=True)

    st.markdown("**Site table**")
    st.dataframe(
        run_query("SELECT * FROM maint_lab.ops.v_site_insight ORDER BY asset, site_id"),
        hide_index=True,
    )

# ---------------------------------------------------------------------
# Tab 3: AI analysis (AI_AGG and AI_SUMMARIZE_AGG on visible rows only)
# ---------------------------------------------------------------------
with tab_ai:
    st.subheader("AI analysis of your requests")
    site_choice = st.selectbox(
        "Scope", options=["ALL"] + (list(labels.keys()) if not sites.empty else []),
        format_func=lambda k: "All my sites" if k == "ALL" else labels[k],
    )
    analysis = st.radio("Analysis", list(ANALYSES.keys()))
    col_a, col_b = st.columns(2)
    run_agg = col_a.button("Run AI analysis", type="primary")
    run_sum = col_b.button("Summarise all requests")

    if run_agg or run_sum:
        n = int(run_query(
            """SELECT COUNT(*) AS n FROM maint_lab.ops.maint_requests
               WHERE (? = 'ALL' OR site_id = ?)""",
            params=[site_choice, site_choice],
        ).iloc[0]["N"])
        if n == 0:
            st.warning("You cannot see any request in this scope, so Cortex AI has no data.")
        elif run_agg:
            # The instruction is fixed text from ANALYSES, not user input.
            instruction = ANALYSES[analysis].replace("'", "''")
            with st.spinner("Cortex AI reads the requests..."):
                answer = run_query(
                    f"""SELECT AI_AGG(
                                CONCAT(equipment_id, ' (', equipment_type, ', ', priority, ', ',
                                       downtime_hours::VARCHAR, ' h): ', description_safe),
                                '{instruction}') AS answer
                        FROM maint_lab.ops.maint_requests
                        WHERE (? = 'ALL' OR site_id = ?)""",
```

```
                params=[site_choice, site_choice],
            ).iloc[0]["ANSWER"]
        st.markdown(answer)
        st.caption(f"Cortex AI used the {n} requests that you can see in this scope.")
    else:
        with st.spinner("Cortex AI reads the requests..."):
            answer = run_query(
                """SELECT AI_SUMMARIZE_AGG(description_safe) AS answer
                   FROM maint_lab.ops.maint_requests
                   WHERE (? = 'ALL' OR site_id = ?)""",
                params=[site_choice, site_choice],
            ).iloc[0]["ANSWER"]
        st.markdown(answer)
        st.caption(f"Cortex AI used the {n} requests that you can see in this scope.")
```