Microsoft Azure Data Fundamentals

Master Azure data services and analytics

Learn core data concepts, relational and non-relational data, data analytics, and Azure data services with real-world examples and hands-on practice.

Core Data Concepts

Structured, semi-structured, and unstructured data

Relational & NoSQL

SQL databases and non-relational stores

Analytics at Scale

Data warehousing and lakehouse patterns

Data Visualization

Power BI and real-time analytics

YOUR PRESENTER

Mohammed Arif, PhD

GenAI Architect & Data Scientist

DP-900: Microsoft Azure Data Fundamentals

Table of Contents

5 Modules

1 Course Overview

01. Core Data Concepts

Data types & storage

- Structured, semi-structured, unstructured
- Data storage formats (CSV, JSON, Parquet)
- OLTP vs OLAP workloads

3 topics

02. Data Roles & Cloud Services

DBA, Engineer, Analyst

- Database Administrator
- Data Engineer
- Data Analyst

3 roles

03. Relational Data Essentials

SQL & Azure SQL

- Tables, normalization, keys
- SQL basics (DDL, DML, DCL)
- Azure SQL services

4 topics

04. Non-Relational & Storage

Cosmos DB, Blob, Files

- Azure Cosmos DB
- Blob & File storage
- Data Lake Gen2

05. Analytics at Scale

Databricks, Fabric, Power BI

- Azure Databricks
- Microsoft Fabric
- Power BI visualization

Course Resources

Access all course materials and supplementary resources

DP-900

Course URL

https://arif.works/mbb/

Microsoft Learn Resources

DP-900 Course
Official training course

Azure Documentation
Complete documentation

Video Tutorials
Step-by-step guides

Access all resources through Microsoft Learn links

Structured Data

Tables with rows & columns

Example: Bank Customer Database

ID	Name	Email
1	Joe Jones	joe@bank.com
2	Samir Nadoy	samir@bank.com

- **Schema-defined:** Fixed structure, data types enforced
- **Relational:** Tables with relationships
- **SQL queries:** Easy to query with SQL

Why it matters:

Structured data is the foundation of transactional systems like banking, where data integrity is critical.

Semi-structured

JSON/CSV with flexible schema

Example: Social Media Post JSON

```
{ "post_id": "12345", "user": "john_doe", "content": "Hello world!", "likes": 42, "tags": ["social", "tech"] }
```

- **Flexible:** Schema can change
- **Nested data:** Supports complex structures
- **API-friendly:** Easy to parse

Why it matters:

Semi-structured data is ideal for APIs and social media, where flexibility is key.

Unstructured

Media files & documents

Example: YouTube Video

- Video file (MP4, AVI)
- Thumbnail image
- Audio track
- Metadata (title, description)

- **Media-rich:** Images, videos, audio
- **Large files:** No fixed structure
- **Storage:** Requires blob storage

Why it matters:

Unstructured data powers media platforms like YouTube, where content is diverse.

1 Data Storage Formats Overview

CSV

Simple, human-readable format with comma-separated values. Great for data interchange and small datasets.

- ✓ Best for: Data import/export, spreadsheets

Human-readable

JSON

Flexible, lightweight format for APIs and configuration. Supports nested structures and arrays.

- ✓ Best for: APIs, configuration files

API-friendly

XML

Metadata-rich format with strong schema validation. Used for complex data structures.

- ✓ Best for: Configuration, SOAP APIs

Schema-based

Parquet

Columnar format optimized for analytics. Compressed, efficient for large-scale data processing.

- ✓ Best for: Analytics, data lakes

Columnar

Avro

Row-based format with schema evolution support. Good for streaming data and serialization.

- ✓ Best for: Streaming, serialization

Schema evolution

ORC

Optimized columnar format for Hadoop workloads. High compression and query performance.

- ✓ Best for: Hadoop, big data

Optimized

Real-world example:
Data engineers use **Parquet** for cost-effective analytics because it's **50-70% smaller** than CSV and **10x faster** for queries. Perfect for storing clickstream data in Azure Data Lake.

Relational vs Non-relational Databases

Feature	Relational (SQL)	Non-relational (NoSQL)
Data Structure	**Structured Tables** Rows and columns with defined schema Banking accounts · ERP systems	**Flexible Documents** JSON, key-value, graph, column-family Social media feeds · IoT sensor data
ACID Compliance	Full ACID Atomicity, Consistency, Isolation, Durability	Eventual Consistency Sacrifices strict consistency for performance
Scalability	Vertical Scaling Scale up with bigger servers	Horizontal Scaling Scale out across multiple servers
Relationships	Joins Supported Complex relationships between tables	No Joins Embedded documents or references
Query Language	**SQL** Standard query language	**Varies (JSON, CQL, etc.)** API-based queries
Best For	✓ Transactional data ✓ Complex queries ✓ Data integrity	✓ High volume writes ✓ Flexible schema ✓ Global distribution

Choose based on data structure, consistency needs, and scale requirements

Microsoft Learn: Data Store Overview

OLTP vs OLAP Workloads

OLTP (Online Transaction Processing)

Operational data for day-to-day transactions

- **Many small transactions** - High volume of read/write operations
- **ACID properties** - Atomicity, Consistency, Isolation, Durability
- **Real-time processing** - Immediate response for applications
- **Row-level locking** - Concurrent access without conflicts

Example: ATM withdrawal system

Each withdrawal is a small transaction with immediate confirmation

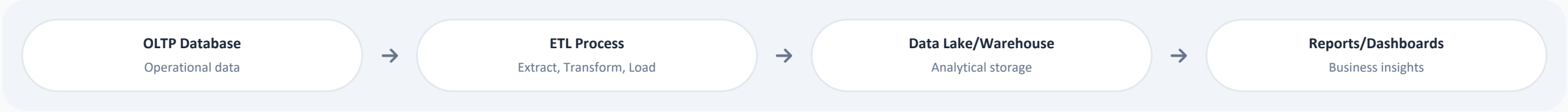
OLAP (Online Analytical Processing)

Analytics for business intelligence

- **Aggregations & analytics** - Complex queries on large datasets
- **Historical data** - Time-series analysis and trends
- **Read-heavy workloads** - Optimized for reporting
- **Data warehousing** - Centralized data storage

Example: Monthly sales dashboard

Analyzing trends across millions of transactions



OLTP for operational data, OLAP for analytics - Choose based on workload patterns

Microsoft Learn: Data Warehousing

1 Data Professional Roles

Database Administrator

Manages database infrastructure, security, and performance. Like the caretaker of your data warehouse.

- Database provisioning & configuration
- Security & user access management
- Backups & disaster recovery
- Performance monitoring & optimization

Real Example:
In an e-commerce company, the DBA secures the orders database and ensures 99.99% uptime.

Data Engineer

Builds data pipelines and ETL processes. Like the plumber who connects data sources to destinations.

- Data integration pipelines & ETL
- Data cleansing & transformation
- Analytical data store schemas
- Data loads & processing

Real Example:
The DE builds pipelines to process customer data from multiple sources into a data lake.

Data Analyst

Analyzes data and creates visualizations. Like the storyteller who turns data into insights.

- Analytical modeling & reporting
- Data visualization (Power BI)
- Business intelligence insights
- Dashboard creation

Real Example:
The DA creates revenue dashboards showing sales trends and customer behavior.

2 Azure Data Services Overview

Operational Data
Azure SQL, MySQL, PostgreSQL, Cosmos DB

Managed database services for transactional workloads with high availability, security, and scalability.

Real-world example:
Banking systems use Azure SQL for secure transaction processing with 99.99% uptime

- ✓ Azure SQL
- ✓ MySQL
- ✓ PostgreSQL
- ✓ Cosmos DB

Storage Services
Blob, Files, ADLS Gen2, Table

Scalable storage solutions for structured, semi-structured, and unstructured data.

Real-world example:
Netflix stores millions of video files in Azure Blob Storage for global streaming

- ✓ Blob Storage
- ✓ ADLS Gen2
- ✓ File Storage

Analytics
Databricks PaaS, Microsoft Fabric SaaS

Big data processing and analytics platforms for insights and visualization.

Real-world example:
Retail companies use Databricks to analyze customer behavior patterns

- ✓ Databricks
- ✓ Fabric
- ✓ Synapse

Governance
Microsoft Purview

Data governance, cataloging, and compliance management across the organization.

Real-world example:
Financial institutions use Purview to manage sensitive data compliance

- ✓ Data Catalog
- ✓ Compliance
- ✓ Lineage

Relational Tables

Understanding table structure, keys, and relationships

DP-900

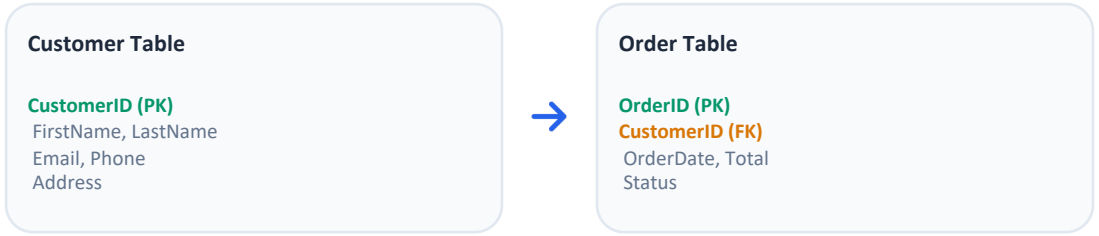
Table Structure Basics

Tables are the fundamental building blocks of relational databases. They consist of:

- **Rows:** Individual records (e.g., each customer)
- **Columns:** Attributes/fields (e.g., name, email)
- **Data Types:** Define what data each column can hold
- **Keys:** Primary keys (unique IDs) and foreign keys (relationships)

Column	Data Type	Description
CustomerID (PK)	INT	Unique identifier
FirstName	VARCHAR(50)	Customer first name
LastName	VARCHAR(50)	Customer last name
Email	VARCHAR(100)	Contact email

Key Relationships



- **Primary Key (PK)**: Unique identifier for each row
- **Foreign Key (FK)**: Links to primary key in another table

Real-World Example

E-commerce Database:

- **Customers:** Store customer info (name, email, address)
- **Orders:** Track purchases with customer reference
- **Products:** Catalog with prices and inventory
- **OrderItems:** Link orders to products

Learn more: SQL Tables Documentation

DP-900: Microsoft Azure Data Fundamentals

Normalization Basics

Data Design | Relational

Goal

Reduce redundancy, improve integrity

Why Normalize?

Normalization organizes data to minimize redundancy and improve data integrity by eliminating duplicate data and ensuring consistent data storage.

- **Eliminate redundancy:** Store each piece of data only once
- **Improve integrity:** Ensure data consistency
- **Save storage:** Reduce data duplication
- **Better queries:** Faster data retrieval

Key Benefits

- Prevents update anomalies
- Simplifies data maintenance
- Improves data quality

Process

Separate entities, define keys

Steps to Normalize

1. **Separate entities:** Split into distinct tables
2. **Define primary keys:** Unique identifier for each row
3. **Use foreign keys:** Link related tables
4. **Remove transitive dependencies:** Avoid indirect relationships

Key Concepts

- **Primary Key:** Unique identifier
- **Foreign Key:** Links tables
- **1NF:** Atomic values
- **2NF:** No partial dependencies
- **3NF:** No transitive dependencies

Example

Sales table normalization

Before: Denormalized

OrderNo	Customer	Product	Qty
1000	Joe Jones	Hammer	1
1000	Joe Jones	Screwdriver	2

After: Normalized

- **Customer** table (ID, Name)
- **Order** table (OrderNo, CustomerID)
- **LineItem** table (OrderNo, ProductID, Qty)
- **Product** table (ID, Name, Price)

DDL - Data Definition Language
Define database structure

```sql
-- Create a new table
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Email VARCHAR(100)
);

-- Modify table structure
ALTER TABLE Customers
ADD Phone VARCHAR(20);

-- Remove table
DROP TABLE Customers;
```

Key Operations

CREATE ALTER DROP RENAME

- **Schema changes:** Define tables, columns, constraints
- **Structure:** Create database objects
- **Modifications:** Change existing objects

DML - Data Manipulation Language
Manipulate data in tables

```sql
-- Query data
SELECT FirstName, LastName
FROM Customers
WHERE City = 'Seattle';

-- Insert new record
INSERT INTO Customers (CustomerID, FirstName, Las
VALUES (1, 'John', 'Doe');

-- Update existing data
UPDATE Customers
SET Email = 'new@email.com'
WHERE CustomerID = 1;

-- Delete records
DELETE FROM Customers
WHERE CustomerID = 1;
```

Key Operations

SELECT INSERT UPDATE DELETE

- **Data access:** Query and retrieve data
- **CRUD operations:** Create, Read, Update, Delete
- **Filtering:** Use WHERE clause

DCL - Data Control Language
Control access to data

```sql
-- Grant permissions
GRANT SELECT, INSERT, UPDATE
ON Customers
TO user1;

-- Revoke permissions
REVOKE DELETE
ON Customers
FROM user1;

-- Deny access
DENY SELECT
ON Customers
TO user2;
```

Key Operations

GRANT REVOKE DENY

- **Security:** Control data access
- **Permissions:** Grant/revoke privileges
- **Access control:** User management

Views
Virtual tables from queries

Example: Delivery View

```
CREATE VIEW Deliveries
AS
SELECT o.OrderNo, o.OrderDate,
c.Address, c.City
FROM Order AS o
JOIN Customer AS c
ON o.Customer = c.ID;
```

- **Virtual table:** Pre-defined SQL queries
- **Join data:** Combine multiple tables
- **Security:** Restrict access to specific columns

Why it matters:

Views simplify complex queries and provide a security layer by exposing only necessary data.

Stored Procedures
Parameterized routines

Example: Rename Product

```
CREATE PROCEDURE RenameProduct
@ProductID INT,
@NewName VARCHAR (20)
AS
UPDATE Product
SET Name = @NewName
WHERE ID = @ProductID;
```

- **Parameterized:** Accept input parameters
- **Reusable:** Execute multiple times
- **Performance:** Pre-compiled SQL

Why it matters:

Stored procedures improve performance and encapsulate business logic close to the data.

Indexes
Tree structures for fast queries

Example: Product Name Index

```
CREATE INDEX idx_ProductName
ON Product(Name);

-- B-tree structure for O(log n) lookup
```

- **Fast lookup:** O(log n) search time
- **Sort optimization:** Faster ORDER BY
- **Trade-off:** Slower writes, faster reads

Why it matters:

Indexes dramatically improve query performance for large datasets.

Azure SQL Family - Choose the Right Option

Feature	SQL Server on Azure VM	Azure SQL Managed Instance	Azure SQL Database
Service Type	IaaS Infrastructure as a Service	PaaS Platform as a Service	PaaS Platform as a Service
Control Level	✓ Full OS control ✓ SQL Server config ✓ Windows/Linux	✓ Instance-level ✓ Near 100% compat ✗ No OS access	✓ Database-level ✓ Elastic pools ✗ Limited control
Maintenance	Customer manages everything OS patches, SQL updates	Microsoft manages infrastructure Auto backups, Patching	Fully managed service Auto maintenance, Monitoring
Best For	✓ Legacy apps ✓ Complex migrations ✓ Full SQL Server	✓ Most on-prem DBs ✓ Minimal maintenance ✓ Instance consolidation	✓ New cloud apps ✓ Elastic scaling ✓ Cost optimization
Example Use Case	Existing SQL Server migration ERP systems, Legacy apps	Consolidate multiple databases SaaS apps, Multi-tenant	Modern cloud applications Web apps, Microservices

Choose based on migration needs, control requirements, and maintenance preferences

Microsoft Learn: Choose the right SQL option

3 Open-Source Databases on Azure

Azure Database for MySQL

Flexible Server, LAMP stacks, web apps

PaaS implementation of MySQL Community Edition in Azure cloud. Optimized for LAMP application architectures and web applications.

Real-world example:

E-commerce platforms like Shopify use MySQL for product catalogs and user management

- ✓ Flexible Server
- ✓ LAMP Stack
- ✓ Web Apps

Azure Database for PostgreSQL

Flexible Server/Citus, geospatial, JSONB

Database service based on PostgreSQL Community Edition. Supports geospatial data, JSONB, and advanced analytics.

Real-world example:

Location-based apps like Uber use PostgreSQL for geospatial data and routing

- ✓ Flexible Server
- ✓ Citus
- ✓ JSONB

Azure Database for MariaDB

MariaDB workloads, Oracle compatibility

Implementation of MariaDB Community Edition adapted to run in Azure. Compatible with Oracle Database features.

Real-world example:

Financial institutions use MariaDB for legacy system migration

- ✓ MariaDB
- ✓ Oracle Compat
- ✓ Migration

4 Azure Storage Services Overview

Blob Storage

Object storage for unstructured data with Hot, Cool, and Archive tiers

Block Blobs | Page Blobs | Append Blobs

Example: Store videos like YouTube (Hot tier), old backups (Archive tier)

Data Lake Gen2

Hierarchical namespace for big data analytics with high performance

Analytics | Hierarchical | Big Data

Example: Store and analyze petabytes of IoT data

Azure Files

SMB/NFS shares for file-based workloads and legacy applications

SMB | NFS | File Shares

Example: Shared folders for team collaboration

Table Storage

NoSQL key-value storage for structured data

Key-Value | NoSQL | Scalable

Example: Store IoT telemetry data

Learn more: Azure Storage Documentation

Azure Cosmos DB Overview

Multi-model, globally distributed NoSQL database

DP-900

What is Azure Cosmos DB?

A **multi-model, globally distributed NoSQL database** management system designed for modern app development. It provides:

- **Multi-model support:** Document, key-value, graph, and column-family data models
- **Global distribution:** Data replicated across multiple regions worldwide
- **Low latency:** Single-digit millisecond response times
- **Elastic throughput:** Scale up or down based on demand
- **Multi-region writes:** Write data to any region

Global Distribution

Replicate your data across **any number of Azure regions** worldwide. Benefits include:

99.999%	<10ms	<15ms
Availability SLA	Read Latency	Write Latency

Key Features

- **Multi-model:** Support for multiple APIs
- **Scalable:** Auto-scaling throughput
- **Secure:** Enterprise-grade encryption
- **Serverless:** Pay-per-use option

Real-World Use Cases

Netflix Personalization

Uses Cosmos DB to store user preferences and viewing history for personalized recommendations across millions of users worldwide.

IoT Telemetry

Smart devices send telemetry data to Cosmos DB for real-time monitoring and analytics, processing millions of events per second.

E-commerce Catalogs

Product catalogs with flexible schemas, global distribution for low-latency access, and automatic scaling during peak shopping seasons.

Learn more: [Microsoft Learn - Azure Cosmos DB]

4 Azure Cosmos DB APIs - Choose the Right API for Your Data Model

NoSQL API
Native JSON Documents

Best for JSON documents with flexible schema. Native Cosmos DB API with SQL-like query syntax.

Use case:
E-commerce product catalogs with dynamic attributes

JSON | Flexible schema | SQL-like

MongoDB API
MongoDB Compatibility

Compatible with existing MongoDB applications. Use MongoDB drivers and tools.

Use case:
Social media feeds with document-based storage

MongoDB | Compatible | NoSQL

PostgreSQL API
PostgreSQL Compatibility

Full PostgreSQL compatibility with JSONB support and geospatial capabilities.

Use case:
Geospatial applications with JSONB data

PostgreSQL | JSONB | Geo

Table API
Key-Value Storage

Simple key-value storage compatible with Azure Table Storage.

Use case:
IoT device telemetry data storage

Key-value | Simple | Fast

Cassandra API
Column Family Store

Apache Cassandra compatibility for wide-column store patterns.

Use case:
Time-series data with high write throughput

Cassandra | Column | Scale

Gremlin API
Graph Database

Graph database for relationship modeling with vertices and edges.

Use case:
Social network relationship mapping

Graph | Relationships | Traversal

DP-900: Azure Data Fundamentals

Analytics Architecture

Data Analytics Architecture Flow

4-step process

1

Ingest

Extract, Transform, and Load (ETL) or Extract, Load, and Transform (ELT) from various data sources

- Data sources: SQL, NoSQL, files
- Batch or real-time ingestion
- Data validation & cleansing

Example: Retail Sales

Ingest sales data from POS systems, web clicks, and inventory systems

2

Store

Store data in a data lake with flexible, scalable file storage for large-scale analytics

- ADLS Gen2 for data lake
- Blob storage for raw data
- Hierarchical namespace

Example: Data Lake

Store raw sales data in Parquet format for cost-effective analytics

3

Process

Transform and process data in a lakehouse or data warehouse with schema-on-read

- Spark-based processing
- Data transformation
- Schema validation

Example: Data Warehouse

Load processed data into Synapse Analytics for querying

4

Analyze

Query data to produce reports, dashboards, and insights for business decision-making

- OLAP models
- Power BI dashboards
- Real-time analytics

Example: Sales Dashboard

Create interactive dashboards showing sales trends by region

Azure Databricks vs Microsoft Fabric

Azure Databricks

Platform-as-a-Service (PaaS)

- ✓ Spark-first architecture
- ✓ Code-centric (Python, Scala)
- ✓ Data engineering focus
- ✓ Data science capabilities
- ✓ Portable across clouds
- ✓ Requires coding skills

Best For

Data Engineering: ETL pipelines, data processing, ML model training. Example: Processing 1TB of log data daily for analytics.

Microsoft Fabric

Software-as-a-Service (SaaS)

- ✓ Unified OneLake storage
- ✓ End-to-end analytics
- ✓ No-code/low-code friendly
- ✓ Integrated Power BI
- ✓ SaaS - no infrastructure
- ✗ Less control over compute

Best For

Business Analytics: Self-service BI, data warehousing, real-time dashboards. Example: Sales team analyzing regional performance.

When to Choose Databricks

- Complex data engineering tasks
- ML model development & training
- Multi-cloud portability needed
- Deep Spark expertise required

When to Choose Fabric

- End-to-end analytics platform
- Business users need self-service
- Integrated BI & reporting
- No infrastructure management

Both platforms integrate with Azure Data Lake and support Delta Lake format

Azure Databricks Docs

Stream and Real-time Analytics

Compare batch processing vs stream processing for different data scenarios

Batch Processing

Process data at regular intervals

Example: Daily Sales Summary

Process all transactions from the day at midnight to generate reports. Good for historical analysis and periodic reporting.

Scheduled | Nightly

Stream Processing

Process data in near real-time

Example: Fraud Detection

Analyze transactions as they happen to detect suspicious patterns. Essential for real-time monitoring and alerts.

Real-time | Continuous

Fabric Real-Time Intelligence Flow

1. **Eventstream** — Capture streaming data
2. **KQL DB/Lakehouse** — Store & process data
3. **Real-time Dashboards** — Visualize insights

Fabric Real-Time Intelligence
https://learn.microsoft.com/fabric/real-time-intelligence/

Azure Stream Analytics
https://learn.microsoft.com/azure/stream-analytics/

Power BI & Data Visualization

Transform raw data into actionable insights

DP-900

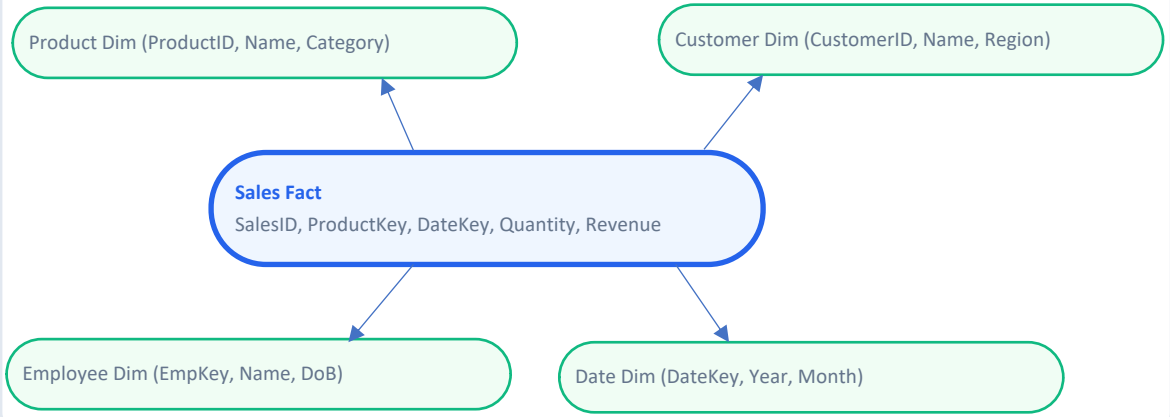
Power BI Workflow

1. **Power BI Desktop:** Import data, define model, create visualizations locally
2. **Publish to Service:** Share reports, schedule refresh, collaborate
3. **Dashboards & Apps:** Create interactive dashboards, distribute to users

Star Schema Modeling

Fact Table: Contains measures (sales, quantity, revenue)

Dimension Tables: Contains descriptive attributes (products, customers, time)



Common Visualizations

Bar Chart
Compare categories

Line Chart
Show trends over time

Pie Chart
Show proportions

Map
Geographic data

Hierarchies & Drill-down

Product Hierarchy: Category → Subcategory → Product

Time Hierarchy: Year → Quarter → Month → Day

Geography: Country → State → City → Store

DAX Measures

Revenue:

```
SUM(Sales[Revenue])
```

YoY Growth:

```
(CY - PY) / PY
```

Learn more: Power BI Documentation

Summary

Complete your DP-900 journey with key takeaways and resources

Key Takeaways

- **Data Types** Structured, semi-structured, unstructured data
- **Relational vs NoSQL** Choose based on schema flexibility
- **Azure SQL Family** SQL VM, Managed Instance, Database
- **Azure Storage** Blob, Files, Tables, Queues
- **Cosmos DB** Multi-model, globally distributed
- **Analytics** Lakehouse, warehouse, Databricks
- **Streaming** Real-time analytics, Event Hubs
- **Power BI** Data visualization, dashboards

All resources are official Microsoft documentation

Microsoft Learn Azure Docs