5-Days Capstone Synopsis

Enterprise Banking Data Platform Using Azure Cloud

Day 1 – Azure Environment Setup & Data Ingestion Architecture

Objective

To build the foundation of the banking data analytics platform by creating storage resources, ingestion architecture, and initial data flow.

What was Done

- The Azure architecture was designed following real-bank pipeline patterns.
- The primary goal was to ingest ATM transactions, UPI payments, and customer activity into a secure cloud platform.

Execution Steps

1. Created Azure resources:
    - Azure Function App
    - Azure Storage Account
    - Azure Event Grid
    - Azure Service Bus
    - Azure Cosmos DB Account
2. Designed folder structure inside ADLS Gen2 using containers:
    - **raw** → Source data landing zone
    - **bronze** → Validated data
    - **silver** → Cleaned & standardized data
    - **gold** → Analytical-ready tables
3. Ingestion architecture was finalized:
    - When a CSV file is uploaded into ADLS → Blob-Created event triggers Event Grid
    - Event Grid places message → Service Bus Queue
    - Functions consume Queue → Load to Cosmos DB containers

Result

✓ End-to-end ingestion was successfully working
✓ Real-time ingestion pipeline created
✓ Storage and data lake structure implemented

Day 2 – Data Processing Using Python Azure Functions

Objective

To process raw incoming data, validate transactions, apply fraud rules, and store them transaction-wise in Cosmos DB.

What was Done

- Service Bus triggered Python Azure Functions
- CSV files were read dynamically from Blob Storage
- Rows converted into transaction records
- Stored into Cosmos DB based on type

Data Classification Logic Applied

- If filename contains *ATM* → classified as ATM transaction
- If filename contains *UPI* → classified as UPI event

Fraud Rules Applied

Examples:
✓ Withdrawal more than ₹20,000
✓ UPI transfer greater than ₹50,000
✓ Repeated transactions

Data Stored in Following Cosmos DB Containers

Container	**Data Stored**
ATMTransactions	ATM withdrawals & balance transactions
UPIEvents	UPI transfers
FraudAlerts	Real-time fraud triggers

Result

✓ All transactions uploaded through CSV were stored
✓ Suspicious transactions generated events
✓ Real-time alerting was functional

Day 3 – PySpark Data Transformation & Data Warehouse Modeling

Objective

Convert operational data into curated analytics-ready structured tables.

Execution Steps

1. Connected Databricks to ADLS using secret scopes
2. Read documents from Cosmos DB into Spark DataFrames
3. Converted raw format → curated format
4. Created transformation pipelines for:
    - ATMTransactions → Cleaned ATM dataset
    - UPIEvents → Standardized UPI dataset
    - FraudAlerts → Time-based fraud metadata

Bronze Layer Outputs

- Raw schema, no transformations
- Stored in delta format

Silver Layer Outputs

✓ Unified structure applied
✓ Proper datatypes assigned
✓ Columns standardized
✓ Duplicate records removed

Gold Layer Outputs

- Final analytical tables created:
    - FactTransactions
    - FactFraudDetection

Data Warehouse Schema Built

Star Schema created

Fact Tables

- Facts have numeric/measures
- Analytical aggregations possible

Dimension Tables

- DimCustomer
- DimAccount
- DimDate
- DimProduct

Advanced logic—Slowly Changing Dimensions Type-2 was implemented for customer behavior tracking.

Result

✓ All data became analytics-ready
✓ Schema relationships were established

Day 4 – Security, Realtime Alerts & CI/CD Automation

Objective

Industrial-grade security and deployment automation.

1. Real-Time Alerts

- High-value transactions published to Event Grid
- Trigger Function consumed alerts
- Sends fraud notifications
- Fraud stored inside Cosmos DB

2. Security Implementation

I have implemented:

✓ Restrict-public-access on Cosmos DB
✓ Firewall applied on SQL Server
✓ Virtual Network subnetting
✓ Only trusted Function subnet access allowed

Achieved:

Least-privilege access
No unauthorized external connection
Audit-based monitoring

3. CI/CD Pipelines Execution

I have executed:

Azure Function Deployment Pipeline

- Source changes pushed to GitHub
- Auto build + publish to Azure App

Databricks Deployment Pipeline

- Notebooks sync via API

SQL Migration Pipeline

- SQL schema deployed using script execution

Result

✓ Fully automated deployment achieved

Day 5 – Reporting, Customer 360 Analytics & Business Dashboards

Objective

Convert processed data into banking intelligence reports.

Customer 360 was generated using 3 systems:

Source

(Aggregated)

➞ Cosmos DB
➞ SQL Fact and Dim Tables
➞ Gold Layer Data

Insights I have Built:

Customer Insights

✓ Total account balance
✓ Monthly debit-credit patterns
✓ Average UPI spending
✓ Fraud score

Transaction Behavior

✓ ATM vs UPI usage
✓ Peak transaction hours
✓ Account-wise lifecycle

Power BI Reporting Models created:

Dashboard 1 - Branch Performance Dashboard

- Metrics: total transaction amount, number of customers, etc.
- Visuals: bar charts by branch (or you described branch dimension conceptually).

Dashboard 2 – Daily Transaction Volume

- Shows trends over time: daily transaction counts and amounts.
- Visuals: line charts, date slicers, filters by channel (ATM / UPI).

Dashboard 3 – ATM/UPI Channel Analysis

- Success vs failure ratios.
- Peak transaction hours.
- Geo-based distribution of ATM and UPI activity.

Dashboard 4 – Fraud Analytics

- Number of fraud alerts by type (high-value, unusual UPI, etc.).
- Total potential loss amount.
- Top risky customers/accounts.
- Timeline of alerts.

Dashboard 5 – RBI Compliance / Regulatory View

- Summary tables and charts that can be used for compliance reports – e.g., large cash transactions, suspicious activity counts per period.

Final Outcome Delivered

I have successfully built a banking-grade data platform covering:

✓ Real-time ingestion
✓ Fraud analytics system
✓ PySpark data lake architecture
✓ Gold layer analytics
✓ SQL Data Warehouse
✓ Secured network
✓ CI/CD deployment
✓ Customer 360 Analytics
✓ Business dashboards