Case Study 1:

The XYZ bank has opted to transition away from utilizing Datawarehouse and is embarking on a journey towards employing centralized cloud-based Azure storage. As part of this shift, mechanisms have been established within On-Prem systems and other sources to transmit incremental daily snapshots of data to Azure Datalake Storage.

Traditionally, the Marketing System (MS) accessed the necessary datasets on a daily basis from the Datawarehouse for the purpose of generating dashboards. However, in light of the bank's strategic pivot, the MS team is now tasked with designing a system to daily ingest the datasets from the centralized storage into their designated storage account. They will also need to execute requisite operations on the datasets to ensure they are readily accessible for dashboarding purposes.

In our capacity as *Senior Azure Engineers*, it is imperative that we collaborate to devise a scalable solution for the MS team's requirements.

Solution:

Here's an outline of the solution:

1. **Azure Data Lake Storage (ADLS)**: Use Azure Data Lake Storage for centralized storage of daily incremental snapshots.
2. **Azure Data Factory (ADF)**: Utilize Azure Data Factory to orchestrate the data ingestion process from ADLS to the MS storage account.
3. **Azure Databricks**: Employ Azure Databricks for processing and transforming the ingested data.
4. **Azure Synapse Analytics or Azure SQL Database**: Store the processed data in a structured format for easy access and query performance for dashboarding purposes.
5. **Power BI**: Use Power BI to connect to the processed data and create dashboards

Detailed Solution

1. **Data Ingestion with Azure Data Factory**:
    - **Create an Azure Data Factory instance**: Set up pipelines to automate the data ingestion from ADLS.
    - **Create Linked Services**: Set up linked services for ADLS (source) and the MS storage account (destination).
    - **Create Datasets**: Define datasets for the source and destination data.

- **Build Pipelines**: Design pipelines to copy data from ADLS to the MS storage account daily. Use triggers to schedule the pipelines to run at specific times.

2. **Data Processing with Azure Databricks**:
   - **Create an Azure Databricks workspace**: Set up clusters and notebooks for data processing.
   - **Ingest Data**: Use Databricks notebooks to read the ingested data from the MS storage account.
   - **Transform Data**: Perform necessary data transformations using Spark.
   - **Load Processed Data**: Write the transformed data to Azure Synapse Analytics or Azure SQL Database.
3. **Data Storage**:
   - **Azure Synapse Analytics or Azure SQL Database**: Choose either Azure Synapse Analytics for large-scale data warehousing or Azure SQL Database for smaller datasets.
   - **Schema Design**: Design a schema optimized for querying and dashboarding.
   - **Data Loading**: Use Azure Databricks to load the transformed data into the chosen storage solution.
4. **Dashboarding with Power BI**:
   - **Connect to Data**: Connect Power BI to Azure Synapse Analytics or Azure SQL Database.
   - **Create Datasets**: Define datasets in Power BI based on the processed data.
   - **Build Dashboards**: Create interactive dashboards and reports in Power BI.

Step-by-Step Implementation

1. **Azure Data Factory**:
   - **Linked Service Setup**: Create linked services for ADLS and the MS storage account.
   - **Pipeline Creation**: Create pipelines with Copy Data activities to move data from ADLS to the MS storage account.
2. **Azure Databricks**:
   - **Cluster Configuration**: Configure clusters with the necessary compute resources.

- **Notebook Development**: Develop notebooks to process the ingested data

```
from pyspark.sql import SparkSession
# Initialize SparkSession
spark = SparkSession.builder.appName("DataProcessing").getOrCreate()
# Read data from MS storage account
df = spark.read.csv("path/to/ms/storage/account/data.csv", header=True)
# Perform transformations
transformed_df = df.withColumn("ProcessedColumn", df["OriginalColumn"] + 1)
# Write transformed data to Azure Synapse or SQL Database
transformed_df.write \
    .format("jdbc") \
    .option("url",
"jdbc:sqlserver://your_sql_server.database.windows.net:1433;database=your_database") \
    .option("dbtable", "your_table") \
    .option("user", "your_username") \
    .option("password", "your_password") \
    .save()
```

3. **Azure Synapse Analytics/Azure SQL Database**:
   - **Schema Definition**: Define the schema for the transformed data tables.
   - **Data Loading**: Use Azure Databricks to load data into the tables.
4. **Power BI**:
   - **Data Connection**: Connect Power BI to the Azure Synapse or SQL Database.
   - **Dashboard Creation**: Create and publish dashboards based on the datasets.

Monitoring and Maintenance

- **Azure Monitor**: Use Azure Monitor to keep track of the data pipelines, cluster performance, and storage usage.
- **Alerting**: Set up alerts to notify the team of any failures or performance issues.
- **Scaling**: Adjust the scale of Databricks clusters and storage based on the data volume and query performance requirements.

By following this approach, the MS team can build a scalable and robust system to ingest, process, and visualize data daily, ensuring that the dashboards are up-to-date and readily accessible.