**Capstone Project 6. Comprehensive Data Pipeline with Azure Data Factory, Databricks and Dashboard on Restaurants Dataset | Swiggy** //**swiggyadmin// sa@12345 // servernameswiggy**

**Objective:**

This capstone project focuses on building a comprehensive data pipeline for Data Engineering and Analytics. We'll primarily aim to use Azure Data Factory for data ingestion, then will switch to Databricks with PySpark and Spark-SQL for transformation and analytics, and for visualization trainees are independent to use any other options like Dashboard feature of Databricks or Power BI (optional). The architecture includes multiple data sources, a NoSQL landing zone, SQL-based materialized views, and a visualization layer. By the end of this project, we'll have a functional pipeline that provides valuable insights.

**Objectives:**

1. **Data Extraction**: Ingest data from multiple sources using Azure Data Factory.
2. **Landing Zone**: Store raw data in a NoSQL database (e.g., Azure Cosmos DB).
3. **Data Transformation**: Use PySpark in Databricks for data cleaning, transformation, and enrichment.
4. **Materialized Views**: Create SQL-based materialized views for efficient querying and analysis.
5. **Visualization**: Develop a thin visualization layer using Power BI to present the materialized views.

**Architecture:**

1. **Input Data Sources (Multiple):**

* Data sources such as CSV files, Json, APIs, and relational databases.

1. **Landing Zone (NoSQL):**

* Use Azure Cosmos DB to store raw data.
* This zone acts as a staging area for initial data collection.

1. **Materialized Views (SQL):**

* Transform data in Databricks using PySpark.
* Load the transformed data into Azure SQL Database.
* Create materialized views for optimized querying and reporting.

1. **Visualization Layer (Power BI):**

* Create interactive dashboards to visualize key metrics and insights.

**Dataset:**

<https://www.kaggle.com/datasets/ashishjangra27/swiggy-restaurants-dataset>

🆔 id: Unique identifier for each restaurant.

🍽️ name: Name of the restaurant.

🌆 city: City where the restaurant is located.

⭐ rating: Rating of the restaurant.

👥 rating\_count: Number of people who have given the rating.

💵 cost: Cost of eating at the restaurant.

🍲 cuisine: Types of cuisines the restaurant serves.

📝 lic\_no: License number of the restaurant.

🔗 link: Restaurant's link on the Swiggy website.

📍 address: Full address of the restaurant.

**Steps to Implement:**

**1. Setup Azure Environment**

First, we'll set up our Azure environment. This involves creating accounts and setting up services like Azure Data Factory, Databricks, Cosmos DB, and Azure SQL Database.

* **Azure Account**: If you don't already have one, you'll need to create an Azure account.
* **Resource Group**: Organize your resources by creating a new resource group in Azure. (Use project/dataset name for better understanding)
* **Azure Data Factory**: Set up Azure Data Factory to create our data workflows and data pipeline
* **Databricks**: Set up a Databricks workspace for data processing and analytics.
* **Azure Cosmos DB**: Create an instance of Azure Cosmos DB for landing zone.
* **Azure SQL Database**: Set up an Azure SQL Database for storing materialized views.

**2. Data Ingestion with Azure Data Factory**

With our environment ready, we'll use Azure Data Factory to ingest data from various sources.

* **Data Sources**: Identify and connect to multiple data sources such as Azure Data Lake, APIs, and relational databases.
* **Pipelines**: Create data pipelines in Azure Data Factory to automate the data ingestion process.
* **Landing Zone**: Direct ingested data into the NoSQL landing zone, which in our case is Azure Cosmos DB.

**3. Data Transformation with PySpark in Databricks**

Once the data is ingested and stored in the landing zone, it's time to transform it using PySpark in Databricks.

* **Read Data**: Connect Databricks to Azure Cosmos DB to read the raw data.
* **Data Cleaning**: Use PySpark to clean the data, handling missing values, duplicates, and inconsistencies.
* **Data Transformation**: Transform the data into a more usable format. This may include aggregations, joins, and business-specific logic.
* **Enrichment**: Add any additional information or calculations that enhance the value of the data.
* **Write Data**: Load the cleaned and transformed data into Azure SQL Database.

**4. Creation of Materialized Views**

Now that our data is transformed and stored in Azure SQL Database, we will create materialized views for efficient querying.

* **Define Views**: Identify key metrics and insights that the business needs. Define SQL queries to create materialized views based on these requirements.
* **Optimization**: Optimize these views for performance, ensuring they are quick to query and provide accurate results.

**5. Visualization Layer**

Finally Visualize data using Databricks – Spark SQL, Power BI or any other, connecting it to SQL Database.

* **Connect to Azure SQL Database**: Set up a connection between Visualization platform and Azure SQL Database.
* **Dashboard Development**: Create interactive dashboards to visualize the materialized views. Include charts, graphs, and tables to represent key metrics like trends, insights, performances etc.
* **Insights**: Customize the dashboard to highlight actionable insights that can help in decision-making.

**Suggestive Layers for Improvement**

**1. Real-time Data Processing**

To enhance the pipeline, consider integrating real-time data processing capabilities. This can be achieved by leveraging Spark Structured Streaming. This way, you can handle real-time data ingestion and processing, providing up-to-the-minute insights.

**2. Data Quality Management**

Implement data quality management tools and frameworks to ensure the integrity and accuracy of your data. This includes setting up validation rules, data profiling, and cleansing procedures. Azure Data Factory and Databricks both offer features to support this.

**3. Scalability and Performance Tuning**

Regularly monitor the performance of your data pipeline and optimize it for scalability. This may include partitioning strategies, caching mechanisms, and resource scaling in Databricks to handle growing data volumes and increased complexity.