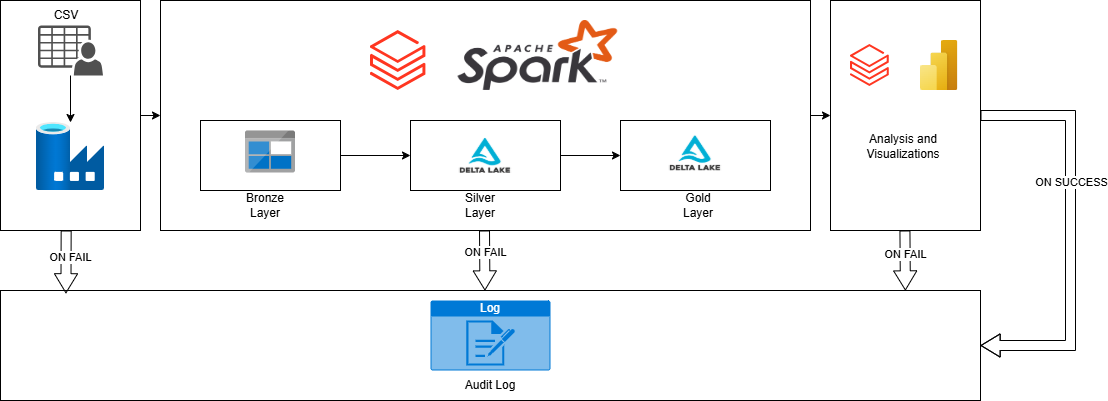
Technical Design Document

**1.       Objectives**

The goal of this project is to develop a robust data pipeline that extracts, transforms, and visualizes data. The solution utilizes Azure Data Factory for data ingestion, Databricks with PySpark for data transformation, Spark SQL for analysis, and visualization using Power BI.

**2.       Project Architecture Diagram**



**3. Project Architecture Overview**The project architecture adopts a **Medallion Architecture with** bronze, silver, gold data layers, a common framework in data warehousing and big data environments. This model organizes data into distinct layers based on their level of processing and refinement.

**3.1. Data Ingestion (Bronze Layer):**

* + Source: Raw data (CSV file).
  + Ingestion Tool: Azure Data Factory.
  + Storage: Azure Data Lake Storage Gen2.
  1. **Data Transformation (Silver Layer):**
  + Processing Tool: Azure Databricks with PySpark.
  + Storage: Delta Lake tables.

**3.3. Data Refinement (Gold Layer):**

* + Final Dataset: Refined, analytical data stored in Delta Lake tables.

**3.4. Data Analysis & Visualization:**

* + Tools for Analysis: Databricks (SQL queries).
  + Visualization Tool: Power BI or Databricks visualizations.

**5. Implementation Plan**

* **5.1 Data Ingestion and Storage**
  + Identify the raw data sources (in this case, the CSV file).
  + Set up an Azure Data Factory pipeline to handle the data ingestion process. Configure activities like the Copy Activity to transfer data from source systems to Azure Data Lake Storage Gen2.
* **5.2 Data Transformation and Cleaning**
  + Establish an Azure Databricks workspace to process and transform the ingested data.
  + Use PySpark to write code for cleaning and transforming the data.
  + Apply performance optimization techniques, such as caching, partitioning, and indexing.
  + Store the transformed data in Delta Lake tables, ensuring efficient querying and updates.
* **5.3 Data Analysis and Visualization**
  + Prepare the transformed data for analysis by performing any additional cleaning and transformations.
  + Use Databricks notebooks to write SQL queries for data analysis.
  + Create interactive dashboards and reports using Power BI or Databricks visualizations.

**6. Tools and Services**

|  |  |
| --- | --- |
| Process | Tools/Services |
| Data Ingestion | Azure Data Factory |
| Data Storage | Azure Data Lake, Azure Delta Tables |
| Data Transformation | Databricks, PySpark |
| Data Analytics | Spark SQL |
| Data Visualization | Power BI |

**7. Schema Design**

**Raw Data Schema (Bronze Layer)**

|  |  |  |
| --- | --- | --- |
| Column | Data Type | Description |
| id | STRING | Unique identifier for each restaurant. |
| name | STRING | Restaurant name. |
| city | STRING | City where the restaurant is located. |
| rating | DOUBLE | Restaurant rating. |
| rating\_count | STRING | Number of people who rated. |
| cost | DOUBLE | Cost per person. |
| cuisine | STRING | Types of cuisines offered. |
| lic\_no | STRING | License number. |
| address | STRING | Full address of the restaurant. |
| menu | STRING | Menu details of each restaurant. |
| link | STRING | Restaurant URLs. |

**Transformed Data Schema (Silver Layer\Gold Layer)**

|  |  |  |
| --- | --- | --- |
| Column | Data Type | Description |
| Restaurant\_ID | STRING | Unique identifier for each restaurant. |
| Name | STRING | Restaurant name. |
| City | STRING | City where the restaurant is located. |
| Cuisine | STRING | Comma-separated cuisines served. |
| Rating | DOUBLE | Numeric rating. |
| Rating\_Count | INT | Number of ratings derived from string. |
| Lic\_no | STRING | License number. |
| Cost | DOUBLE | Average cost per person. |
| Address | STRING | Full address of the restaurant. |
| TimeStamp | DATETIME | Data inserted timestamp. |

**Audit Table:**

|  |  |  |
| --- | --- | --- |
| **Column** | Data Type | Description |
| Id | STRING | Unique ID for the pipeline execution. |
| Process\_Id | STRING | Unique ID for the specific process within the pipeline. |
| Process\_Name | STRING | Name of the process. |
| Start\_Time | TIMESTAMP | Pipeline execution start time. |
| End\_Time | TIMESTAMP | Pipeline execution end time. |
| Status | STRING | Pipeline execution status (Success/Failure). |
| Message | STRING | Log messages, if any. |

**Testing**

**Data Accuracy**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case ID | Test Case Description | Test Steps | Expected Outcome | Negative Scenario |
| DA-01 | Verify if all required fields are present | Check if id, name, city, rating, rating\_count, cost, cuisine, lic\_no, and address exist for each record | All columns should be present for every row | If any field is missing, raise a "Missing Field" error |
| DA-02 | Verify if the id is unique for all entries | Check for duplicates in the id column | No duplicate id values | Duplicate id values result in "Duplicate ID" error |
| DA-03 | Verify if rating values are valid | Check if rating is either numeric 0-5 | Ratings should be between 0 to 5 | Invalid characters, text, or ratings outside the 0-5 range |
| DA-04 | Validate rating\_count format | Ensure rating\_count is in the numeric format like 50, 20, etc., or 0 | Correct format is displayed for all entries | Incorrect formats like "50+ ratings" or "twenty ratings" |
| DA-05 | Verify the correctness of cost field | Ensure cost is numeric and doesn’t have a currency symbol (₹, $, etc.) | **Updated:** Change the cost field to use a float type with a precision of 2 (e.g., $25.75). | Invalid formats such as “₹ 200" or “Rs 200" |
| DA-06 | Validate cuisine values | Check if cuisine is a comma-separated string of food types | Cuisine should be a readable, well-formatted string | Missing or null cuisine values |
| DA-07 | Validate lic\_no format | Check if lic\_no is numeric and 14 digits | All lic\_no should be 14-digit numbers | Any non-numeric or incorrect length lic\_no |

**Data Quality**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case ID | Test Case Description | Test Steps | Expected Outcome | Negative Scenario |
| DQ-01 | Verify if any field contains null, NaN, or undefined values | Check for missing or null values in all columns | No null, undefined, or NaN values | Null, NaN, or undefined values in any field |
| DQ-02 | Check for leading/trailing whitespaces in text fields | Strip whitespaces from name, cuisine, etc. | No leading or trailing spaces in text fields | Presence of unnecessary spaces in fields like name or address |
| DQ-03 | Check for special characters in name, cuisine, and city | Check for invalid characters (like #, %, &, etc.) in text fields | Only alphabets, spaces, and necessary characters allowed | Presence of special characters in name, cuisine, or city |
| DQ-04 | Check for duplicates in name and address | Check for duplicate restaurant entries using name and address combination | No duplicate combinations of name and address | Duplicate restaurants with identical name and address |
| DQ-05 | Validate consistency of cuisine values | Check for duplicates or similar names in cuisine types (e.g., "Pizza" and "Pizzas") | Consistent and properly formatted cuisine types | Duplicate cuisine types due to typos or inconsistencies |
| DQ-06 | Check for cross-field dependencies | Ensure rating\_count is 0 if rating is 0 | If rating is 0, rating\_count should be 0 | rating is 0 but rating\_count has 20 |
| DQ-07 | Validate case consistency of text fields | Check if city, cuisine, address, and name use consistent capitalization | Text fields should follow proper capitalization (e.g., "Pizza" instead of "pizza" or "PIZZA") | Incorrect case usage in text fields |
| DQ-08 | Check if restaurant license number is unique across all restaurants | Validate that no two restaurants share the same license number. | Each restaurant should have a unique license number. | Duplicate license numbers assigned to different restaurants. |
| DQ-09 | Handle null values in the rating column | Calculate the mean and standard deviation for all restaurant ratings and assign accordingly. | Null values in the rating column are replaced accurately using calculated metrics. | Incorrect assignment leading to inaccurate ratings. |

**Data Security**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case ID | Test Case Description | Test Steps | Expected Outcome | Negative Scenario |
| DS-01 | Verify sensitive data exposure | Check if lic\_no is encrypted or masked | lic\_no should be masked or encrypted during transmission and storage | Storing plain-text lic\_no values |
| DS-02 | Check access controls for the file containing the data | Set file permissions to read-only for non-admin users | Non-admin users cannot modify the dataset | Unrestricted read/write access for all users |
| DS-03 | Check for SQL Injection | Attempt to inject SQL queries into name, cuisine, or address fields | The system should reject or sanitize special characters | The system accepts SQL statements as input |
| DS-09 | Test for logging of sensitive data | Check application logs for sensitive data | No sensitive data like lic\_no is present in logs | Logs contain sensitive information like lic\_no or user inputs |

**SQL Requirements**

1. List the names of tables used.
2. Capture the SQL queries/ PySpark code snippets used for data cleansing.
3. Capture the SQL queries/ PySpark code snippets used for data transformations.
4. Create a master table named t\_rating\_desc as follows:

Structure:

|  |  |
| --- | --- |
| COLUMN | DATATYPE |
| RATING | NUMBER |
| RATING\_DESC | STRING |

Data:

|  |  |
| --- | --- |
| RATING | RATING\_DESC |
| 0 | No Rating |
| 1 | Poor |
| 2 | Average |
| 3 | Good |
| 4 | Excellent |
| 5 | Must Try |

1. Write a query that retrieves Restaurant Name, Rating and the corresponding Rating Description. Use the next whole number to get rating description. For the restaurants with blank rating, display the one corresponding to 0.

Example: if the rating of a restaurant is 4.4, display the rating as Must Try.

1. Write a query to get Restaurant Name, City and Rating for Domino's Pizza so that the highest rated is displayed on top. If there are 2 outlets with the same rating, the first on the list should be based on the city in alphabetical order.

Example: If there are 2 Domino's Pizza outlets with rating 3, one in Adityapur and another in Alappuzha then the results should be as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Restaurant\_Name | Rating | Rating\_Description | City |
| Domino's Pizza | 3 | Good | Adityapur |
| Domino's Pizza | 3 | Good | Alappuzha |
| Domino's Pizza | 2 | Average | Adityapur |

1. Find the number of restaurants with rating above 4 for cusine 'North Indian'.
2. Find the number of restaurants with a rating below the 3.
3. Find the number of people who have rated restaurants with cusine 'Fast Food,American'
4. Find the restaurant in LuluMall Kochi that has the highest number of people rated.
5. Identify the city with maximum number of restaurants.
6. Identify the number of Baskin Robbins outlets with a rating of 4.5 and above.
7. Identify the restaurants that sell least expensive Ice cream yet with a rating of 4 and above in Bangalore.
8. Find the top 10 cities as per the number of restaurants listed on Swiggy.
9. Identify the cuisine that has the top rating in this dataset
10. What is the total number of restaurants listed in swiggy in the city Kannur with a rating count of 100+ ratings?
11. Identify the restaurants that serve more than 1 cuisine. Display the Restaurant Name and the Cuisine

Example: Beverages,Pizzas

1. Write a query to list the restaurant names with the letter ‘d’ in the name and also the position of the letter d.
2. Write a query to display the restaurants that have two or more ‘o’ in the name.

Example: Fresh Food Café

1. Write a query to show the restaurant name and cuisine as follows.

Example:

|  |  |
| --- | --- |
| AB FOODS POINT | Beverages | Pizzas |
| Janta Sweet House | Sweets | Bakery |
| theka coffee desi | Beverages |
| Singh Hut | Fast Food | Indian |

1. Write a query to display the restaurant\_name and total cost of restaurants that start with S. The total cost should be based on each city. The list should be sorted based on total cost (least total cost should be on top)
2. Create a new table with the structure:

|  |  |
| --- | --- |
| RESTAURANT\_NAME | STRING |
| LOCALITY | STRING |
| CITY | STRING |
| RATING | DOUBLE |
| RATING\_COUNT | STRING |

Load the data as follows:

Pull the records with city as Electronic City,Bangalore with a rating above 4 and rating count of at least 50+

Split the value into Locality and City using code (Do not hardcode the values).

Example:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RESTAURANT\_NAME | LOCALITY | CITY | Rating | rating\_count |
| Grameen Kulfi | Electronic City | Bangalore | 4.6 | 50+ ratings |

1. Write a query to list the restaurant and city that serve a single cuisine.

Example:

|  |  |  |
| --- | --- | --- |
| Shri Balaji Vaishno Dhaba | Abohar | North Indian |

1. Identify restaurants with anything other than alphabets in the restaurant name. Display distinct values in the output.

**9. Conclusion**By following the outlined design and implementation steps, the pipeline will ensure robust, scalable, and high-quality data processing. This will enable data-driven decision-making and provide valuable insights into the Swiggy dataset.