Expedia Data Engineer Task – Assessment Document

Candidate Name: Tanishka Rajput
Role: Data Engineer

Includes:

- Objective & scenario
- Tech/tools used
- Data cleaning steps
- Business insights (from reports)
- SQL analysis results
- Automation architecture description

Objective:-
This assessment demonstrates my ability to ingest, clean, transform, analyze, and summarize business data to drive actionable insights, simulating a real-world retail sales scenario.

Scenario:-
A retail company captures daily transactional data across multiple stores and stores it as CSV files in a cloud storage bucket. My task was to process this sales data to generate business insights using Python and SQL.

Tech/tools used:

- Programming: Python (Pandas)
- Data Storage: CSV files (as source data)
- IDE: Jupyter Notebook
- Query Language: SQL
- Cloud Automation (Bonus): Proposed using Azure Data Factory

Data Cleaning & Transformation Steps

- Used python library pandas to load the file `sales_data.csv` and `product_master.csv`.
- Use `dropna() to` drop the rows with any **null values** .
- Filtered out the rows where `sale_amount <= 0`.
- Converted `transaction_date` to **datetime** format using to_datetime().
- Derived new columns as given in assessment:

   - `day_of_week` (e.g., Monday, Tuesday)
   - `revenue_per_unit = sale_amount / quantity_sold`

Generated Summary Reports:

- Total sales and quantity sold grouped by store and product
- Top 5 products by total revenue
- The day of the week with the highest overall sales.

These insights help identify business trends related to product performance and provide a comprehensive overview of overall sales activity.

SQL queries analysis

1. Total revenue by each store.

   SELECT store_id, SUM (sale_amount) AS total_revenue
   FROM sales_data
   GROUP BY store_id;

2. Product with the highest quantity sold per store.

   SELECT store_id, product_id, MAX(quantity_sold) as max_quantity
   FROM sales_data
   GROUP BY store_id, product_id;

3. Average revenue per product category (join with product_master.csv).

   SELECT p.product_id, p.category, AVG(s.sale_amount) AS avg_revenue
   FROM sales_data s
   JOIN product_master p ON s.product_id = p.product_id GROUP BY p.product_id;

Proposed architecture for pipeline

The overview of the process is

1. Linked Service & Blob Account (Source)

- Azure Data Factory (ADF) connects to Azure Blob Storage using a linked service to access source files like `sales_data.csv` and `product_master.csv`..

2. Scheduled Trigger

- Automates pipeline execution on a predefined schedule (e.g., daily or hourly).

3. Ingest CSV Files from Blob Storage

- Uses **Copy Activity** to move raw CSV files from Blob Storage to a staging area in **Azure Data Lake** or directly into **Azure SQL DB** for smaller datasets.

4. Data Cleaning & Transformation (ADF Data Flow / Azure Databricks)

- ADF Data Flow (No-code) or Databricks Notebook (Code-based, scalable) to clean and transform the data.

5. Store Results in Data Lake / Azure SQL DB

- Cleaned and transformed data is stored in
  - **Azure Data Lake (ADLS)** for further analytics
  - Or **Azure SQL DB** for reporting and visualization via Power BI

6. Monitoring, Logging & Alerts

- ADF's built-in monitoring tools track pipeline runs, log execution details, and trigger alerts on failure or anomalies.

Architectural Design

This is simple **Architecture diagram**

```
[Scheduled Trigger]

[Ingest CSV Files from Blob Storage]

[Data Cleaning & Transformation (Data Flow / Databricks)]

     │
     ├──────────────► Add Derived Columns
     ├──────────────► Filter Invalid Records
     ├──────────────► Join with Product Master

[Generate Aggregates and Summary Reports]

[Store Results in Data Lake / Azure SQL DB]

[Monitoring, Logging & Alerts]
```

As I am currently working at Innovaccer, where we use a proprietary platform called DAP to build pipelines, the structure I've described is adapted to reflect how we implement them on that system.