Data Engineering Assessment - Sales Dashboard Documentation

1. Introduction

Project Overview

This project involves designing and implementing a **Sales Data Pipeline** for an e-commerce company. The goal is to **extract, transform, and load (ETL) sales data**, store it in a **MySQL database**, and create an **interactive Power BI dashboard** for insights.

Objectives

- Process raw sales data and clean inconsistencies.
- Store data in a structured MySQL database.
- Visualize key sales metrics using Power BI.
- Enhance decision-making through data-driven insights.

2. Data Processing Pipeline

2.1 Data Source

- **Input:** sales_data.json
- **Final Cleaned Data:** clean_sales_data.csv
- **Database:** MySQL

2.2 Data Cleaning & Transformation (Python ETL Process)

The raw sales data (sales_data.json) was processed using a Python script to clean and transform it into a structured format before loading into MySQL. Below are the key steps performed:

✅ **Flattened Nested JSON Structure** (Extracted product_id, product_name, category, and price fields into separate columns).
✅ **Standardized Date Format** (Converted inconsistent date formats to YYYY-MM-DD).
✅ **Handled Missing Values** (Replaced null customer IDs with 'Unknown').
✅ **Removed Negative Quantities** (Converted negative quantity values to 0).
✅ **Computed Total Sales Value** (total_value = quantity * price).
✅ **Stored Cleaned Data in CSV** (clean_sales_data.csv) for database loading.

Python ETL Script (Used for Cleaning & Transformation)

```
import pandas as pd

import json
```

```
# Load JSON data
f = open('sales_data_large.json', 'r')
data = json.load(f)
f.close()

# Convert to DataFrame
df = pd.DataFrame(data)

# Extract nested fields
df['product_id'] = df['product'].apply(lambda x: x['id'])
df['product_name'] = df['product'].apply(lambda x: x['name'])
df['category'] = df['product'].apply(lambda x: x['category'])
df['price'] = df['product'].apply(lambda x: x['price'])
df.drop(columns=['product'], inplace=True)

# Standardize date format
df['date'] = df['date'].apply(lambda x: x + 'T00:00:00Z' if len(x) == 10 else x)
df['date'] = pd.to_datetime(df['date'], errors='coerce').dt.strftime('%Y-%m-%d')

# Handle missing values and incorrect data
df['customer_id'].fillna('Unknown', inplace=True)
df['quantity'] = df['quantity'].apply(lambda x: max(x, 0))
df['total_value'] = df['quantity'] * df['price']
df['price'].fillna(0, inplace=True)

# Save cleaned data
```

df.to_csv('clean_sales_data.csv', index=False, quoting=1)

💡 **The cleaned CSV file was then loaded into MySQL using:**

from sqlalchemy import create_engine

 # Create a database connection

engine = create_engine('mysql+pymysql://user:password@localhost/dbname')

 # Load data in batches

df.to_sql('sales', con=engine, if_exists='append', index=False, chunksize=40)

OR

LOAD DATA INFILE 'C:/path_to_file/clean_sales_data.csv'

INTO TABLE sales

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;

3. Database Design & Querying

3.1 MySQL Table Schema (sales table)

CREATE TABLE sales (

transaction_id VARCHAR(10) PRIMARY KEY,

customer_id VARCHAR(10),

quantity INT,

date DATE,

region VARCHAR(20),

product_id VARCHAR(10),

product_name VARCHAR(50),

category VARCHAR(20),

price DECIMAL(10,2),

```
    total_value DECIMAL(10,2)
);
```

3.2 SQL Queries for Analysis

Total Sales by Region

```
SELECT region, SUM(total_value) AS total_sales
FROM sales
GROUP BY region
ORDER BY total_sales DESC;
```

Top 5 Best-Selling Products

```
SELECT product_name, SUM(total_value) AS total_sales
FROM sales
GROUP BY product_name
ORDER BY total_sales DESC
LIMIT 5;
```

Monthly Sales Trend

```
SELECT DATE_FORMAT(date, '%Y-%m') AS month, SUM(total_value) AS total_sales
FROM sales
GROUP BY month
ORDER BY month;
```

4. Power BI Dashboard

4.1 Visualizations Created

1. **Bar Chart**- Total Sales by Region
   - **X-Axis:** Region
   - **Y-Axis:** Total Sales
2. **Line Chart**- Monthly Sales Trend
   - **X-Axis:** Date (Month)
   - **Y-Axis:** Total Sales
3. **Table**- Top 5 Products by Sales
   - **Columns:** Product Name, Total Sales

4. **Card**- Total Sales Summary
    - **Displays:** Total Sales in $
5. **Slicers**- Interactive Filters
    - **Filters by:** Category, Region, Date

4.2 DAX Measures Used

Total Sales

```
Total Sales = SUM('company_task sales'[total_value])
```

Average Sales Per Transaction

```
Avg Sales = AVERAGE('company_task sales'[total_value])
```

Year-to-Date (YTD) Sales

```
YTD Sales = TOTALYTD(SUM(Sales[total_value]), Sales[date])
```

Count of High-Value Transactions

```
High Value Transactions = COUNTROWS(FILTER('company_task sales','company_task sales'[total_value]>1000))
```

Total Sales for Electronics Category

```
Total Sales for Electronics = CALCULATE(SUM('company_task sales'[total_value]),'company_task sales'[category]="Electronics")
```

5. Dashboard Enhancements

- **Conditional Formatting** for highlighting high/low sales.
- **Tooltips & Drill-Through** for additional insights.
- **Bookmarks & Navigation** for interactive filtering.
- **KPI Cards** for quick sales metrics.

6. Conclusion

This project successfully **processed sales data, optimized storage in MySQL, and built a Power BI dashboard** for data-driven decision-making. Future improvements could include:

- **Integrating real-time data updates.**
- **Implementing predictive analytics using Power BI AI features.**

Prepared by: MUHAMMED ABDUL HAYY NT
Date: 14-03-2025
Tools Used: MySQL, Power BI, Python (for ETL), Windows