SQL Project Comprehensive Report

Environment: SQL Server Management Studio (SSMS)
Database: Retail database
Table: sales_data

Project Overview

The goal of this project was to build and analyze a retail sales database using SQL. The main objectives included:

- Setting up the database and table.
- Cleaning the dataset by removing incomplete records.
- Performing Exploratory Data Analysis (EDA) to understand the dataset.
- Answering key business questions to generate useful insights.

The dataset came from a CSV file containing sales transactions, customer details, and product information.

Setup & Data Import

- I created the sales_data table in SQL Server with correct data types to match the CSV columns.
    - Example: transaction_id and customer_id were integers, sale_date was a date type, and price_per_unit was stored as decimal for accuracy.
- The CSV file was imported using the SSMS Import Flat File Wizard.

Data Cleaning

- I checked for **missing or null values** in all columns. Missing data could cause errors or misleading results.

QUERY:

```
SELECT
  SUM(CASE WHEN transactions_id IS NULL THEN 1 ELSE 0 END) AS
missing_transactions_id,
  SUM(CASE WHEN sale_date IS NULL THEN 1 ELSE 0 END) AS
missing_sale_date,
  SUM(CASE WHEN sale_time IS NULL THEN 1 ELSE 0 END) AS
missing_sale_time,
  SUM(CASE WHEN customer_id IS NULL THEN 1 ELSE 0 END) AS
missing_customer_id,
  SUM(CASE WHEN gender IS NULL THEN 1 ELSE 0 END) AS missing_gender,
  SUM(CASE WHEN age IS NULL THEN 1 ELSE 0 END) AS missing_age,
  SUM(CASE WHEN category IS NULL THEN 1 ELSE 0 END) AS
missing_category,
  SUM(CASE WHEN quantity IS NULL THEN 1 ELSE 0 END) AS
missing_quantity,
  SUM(CASE WHEN price_per_unit IS NULL THEN 1 ELSE 0 END) AS
missing_price_per_unit,
  SUM(CASE WHEN cogs IS NULL THEN 1 ELSE 0 END) AS missing_cogs,
  SUM(CASE WHEN total_sale IS NULL THEN 1 ELSE 0 END) AS
missing_total_sale
 FROM sales_data;
```

- Some rows had null values in gender, price_per_unit, quantity, and total_sale.

QUERY:

```
SELECT *
FROM sales_data
 WHERE transactions_id IS NULL
    OR sale_date IS NULL
    OR sale_time IS NULL
    OR customer_id IS NULL
    OR gender IS NULL
    OR age IS NULL
    OR category IS NULL
```

```
OR quantity IS NULL
OR price_per_unit IS NULL
OR cogs IS NULL
OR total_sale IS NULL;
```

- These incomplete rows were removed to maintain accuracy.

QUERY:

```
DELETE FROM sales_data
 WHERE transactions_id IS NULL
    OR sale_date IS NULL
    OR sale_time IS NULL
    OR customer_id IS NULL
    OR gender IS NULL
    OR age IS NULL
    OR category IS NULL
    OR quantity IS NULL
    OR price_per_unit IS NULL`
    OR cogs IS NULL
```

- After cleaning, a final check confirmed **no missing values remained** with the same query that was initially used to identified missing values

Exploratory Data Analysis (EDA)

To better understand the dataset, I performed several exploratory checks:

- **First 10 Records** – Previewed first 10 rows to verify data structure.

```
SELECT TOP 10 * FROM sales_data;
```

- **Total Number of Transactions** – Confirmed dataset size after cleaning: **1,987 records**.

```
SELECT COUNT(transactions_id) AS Total_transactions FROM sales_data;
```

- **Total Sales Amount** – The business generated a total of **₦908,230.00**.

```
SELECT SUM(total_sale) AS Total_sales_amount FROM sales_data;
```

- **Average Sales per Transaction** – Each transaction averaged **₦457.09**.

```
SELECT AVG(total_sale) AS avg_sales_per_transaction FROM sales_data;
```

- **Sales by Gender** – Female customers contributed **₦463,110.00** and male customers **₦445,120.00**.

```
SELECT gender, SUM(total_sale) AS total_sales
FROM sales_data
GROUP BY gender
ORDER BY total_sales DESC;
```

- **Maximum and Minimum Sale Values** – Showed the highest and lowest sales amounts.

```
SELECT MAX(total_sale) AS Max_Sale FROM sales_data;
SELECT MIN(total_sale) AS Min_Sale FROM sales_data;
```

- **Price per Unit Range** – Identified the most expensive and cheapest product.

```
SELECT MAX(price_per_unit) AS Max_Unit_Price FROM sales_data;
SELECT MIN(price_per_unit) AS Min_Unit_Price FROM sales_data;
```

Project Questions

1. **Sales on 2022-11-05**

```
SELECT * FROM sales_data
 WHERE sale_date = '2022-11-05';
```

2. **Clothing Purchases in Nov 2022 with Quantity > 4**

```
SELECT * FROM sales_data
 WHERE category = 'Clothing'
   AND quantity > 4
   AND sale_date >= '2022-11-01'
   AND sale_date < '2022-12-01';
```

iii. **Total Sales by Category**

```
SELECT category, SUM(total_sale) AS total_sales
 FROM sales_data
 GROUP BY category
 ORDER BY total_sales DESC;
```

iv. **Average Age of Beauty Customers**

```
SELECT AVG(age) AS average_age
 FROM sales_data
 WHERE category = 'Beauty';
```

v. **Transactions with Total Sale > 1000**

```
SELECT * FROM sales_data
 WHERE total_sale > 1000;
```

```
SELECT COUNT(*) AS transactions_over_1000
 FROM sales_data
 WHERE total_sale > 1000;
```

vi. **Transactions by Gender per Category**

```
SELECT gender, category, COUNT(transactions_id) AS total_transactions
 FROM sales_data
 GROUP BY gender, category
 ORDER BY gender, category;
```

vii. **Average Sales per Month & Best-Selling Month** – Calculated monthly averages and ranked months.

```
SELECT DATEPART(YEAR,sale_date) AS sale_year,
DATEPART(MONTH,sale_date) AS sale_month,
 SUM(Total_sale) AS total_sales
FROM sales_data
GROUP BY DATEPART(YEAR, sale_date),
 DATEPART(MONTH, sale_date)
ORDER BY sale_year, total_sales DESC;
 WITH monthly_sales AS ( SELECT YEAR(Sale_date) AS sale_year,
MONTH(Sale_date) AS sale_month,
SUM(Total_sale) AS total_monthly_sale
FROM sales_data
GROUP BY YEAR(Sale_date), MONTH(Sale_date) )
SELECT *
FROM monthly_sales m

WHERE total_monthly_sale = ( SELECT MAX(total_monthly_sale)
FROM monthly_sales WHERE sale_year = m.sale_year );
```

viii. **Top 5 Customers by Sales**

```
SELECT TOP 5 customer_id, SUM(total_sale) AS total_sales
 FROM sales_data
 GROUP BY customer_id
 ORDER BY total_sales DESC;
```

ix. **Unique Customers per Category**

```
SELECT category, COUNT(DISTINCT customer_id) AS unique_customers
FROM sales_data
GROUP BY category
ORDER BY unique_customers DESC;
```

x. **Sales by Shifts (Morning, Afternoon, Evening)**

```
SELECT CASE
         WHEN DATEPART(HOUR, sale_time) < 12 THEN 'Morning'
         WHEN DATEPART(HOUR, sale_time) BETWEEN 12 AND 17 THEN
'Afternoon'
         ELSE 'Evening'
       END AS shift,
       COUNT(*) AS number_of_orders
FROM sales_data
GROUP BY CASE
            WHEN DATEPART(HOUR, sale_time) < 12 THEN 'Morning'
            WHEN DATEPART(HOUR, sale_time) BETWEEN 12 AND 17 THEN
'Afternoon'
            ELSE 'Evening'
         END
ORDER BY number_of_orders DESC;
```

Challenges and Solutions

- **Data Quality:** Missing values were removed to ensure accurate analysis.
- **Date and Time Formatting:** Correct handling of date and time fields allowed reliable filtering (e.g., shifts, monthly analysis).
- **Query Performance:** Optimized queries were used for grouping, aggregation, and ranking.

Conclusion

This project successfully demonstrated how to build, clean, and analyze a retail sales database in SQL.

- Data cleaning ensured accuracy and consistency.
- EDA provided a solid understanding of sales, customers, and products.
- Business questions revealed customer demographics, top products, seasonal peaks, and sales patterns.

These insights can guide better decision-making in **marketing, inventory planning, and customer engagement strategies**.