Retail Sales Analysis Project - 01

Project Overview

Project Title: Retail Sales Analysis
Level: Beginner
Database: `SQL_Project-01`

This project is designed to demonstrate SQL skills and techniques typically used by data analysts to explore, clean, and analyze retail sales data. The project involves setting up a retail sales database, performing exploratory data analysis (EDA), and answering specific business questions through SQL queries. This project is ideal for those who are starting their journey in data analysis and want to build a solid foundation in SQL.

Objectives

1. **Set up a retail sales database**: Create and populate a retail sales database with the provided sales data.
2. **Data Cleaning**: Identify and remove any records with missing or null values.
3. **Exploratory Data Analysis (EDA)**: Perform basic exploratory data analysis to understand the dataset.
4. **Business Analysis**: Use SQL to answer specific business questions and derive insights from the sales data.

Project Structure

1. Database Setup

- **Database Creation**: The project starts by creating a database named `SQL_Project-01`.
- **Table Creation**: A table named `Retail_Sales` is created to store the sales data. The table structure includes columns for transaction ID, sale date, sale time, customer ID, gender, age, product category, quantity sold, price per unit, cost of goods sold (COGS), and total sale amount.

```sql
CREATE DATABASE SQL_Project-01;

CREATE TABLE Retail_Sales (
    transaction_id INT PRIMARY KEY,
    sale_date DATE,
    sale_time TIME,
    customer_id INT,
        gender VARCHAR(15),
```

    age INT,
    category VARCHAR(15),
    quantity INT,
    price_per_unit FLOAT,
            clogs FLOAT,
    total_sale FLOAT
);
~~~

2. Data Exploration & Cleaning

- **Record Count**: Determine the total number of records in the dataset.
- **Customer Count**: Find out how many unique customers are in the dataset.
- **Category Count**: Identify all unique product categories in the dataset.
- **Null Value Check**: Check for any null values in the dataset and delete records with missing data.

```sql
SELECT COUNT(*) AS total_rows FROM Retail_Sales;
SELECT COUNT(DISTINCT customer_id) AS unique_customers FROM Retail_sales;
SELECT DISTINCT category FROM Retail_sales;

SELECT *
FROM Retail_Sales
WHERE transaction_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 clogs IS NULL
   OR total_sale IS NULL;

DELETE FROM Retail_Sales
WHERE transaction_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 clogs IS NULL
  OR total_sale IS NULL;
```

3. Data Analysis & Findings

The following SQL queries were developed to answer specific business questions:

-- 1. Write a SQL query to retrieve all columns for sales made on '2022-11-05':
``` sql
SELECT *
FROM Retail_Sales
WHERE sale_date = '2022-11-05';
```

-- 2. Write a SQL query to retrieve all transactions where the category is 'Clothing' and the quantity sold is greater or equal to 4 in the month of Nov-2022:
``` sql
SELECT *
FROM Retail_Sales
WHERE category = 'Clothing'
  AND quantity >= 4
  AND sale_date BETWEEN '2022-11-01' AND '2022-11-30';
```

-- 3. Write a SQL query to calculate the total sales (total_sale) for each category.:
``` sql
SELECT category, SUM(total_sale) AS total_sales
FROM Retail_Sales
GROUP BY category;
```

-- 4. Write a SQL query to find the average age of customers who purchased items from the 'Beauty' category.:
``` sql
SELECT AVG(age) AS average_age
FROM Retail_Sales
WHERE category = 'Beauty';
```

-- 5. Write a SQL query to find all transactions where the total_sale is greater than 1000.:
``` sql
SELECT *
FROM Retail_Sales
WHERE total_sale > 1000;
```

-- 6. Write a SQL query to find the total number of transactions (transaction_id) made by each gender in each category.:
``` sql
SELECT category, gender, COUNT(transaction_id) AS total_transactions
FROM Retail_Sales
GROUP BY category, gender;
```

-- 7. Write a SQL query to calculate the average sale for each month. Find out best selling month in each year:
``` sql
SELECT
       year,
       month,
    avg_sale
FROM
(
SELECT
    EXTRACT(YEAR FROM sale_date) as year,
    EXTRACT(MONTH FROM sale_date) as month,
    AVG(total_sale) as avg_sale,
    RANK() OVER(PARTITION BY EXTRACT(YEAR FROM sale_date) ORDER BY
AVG(total_sale) DESC) as rank
FROM Retail_Sales
GROUP BY 1, 2
ORDER BY 1, 2, 3
) as t1
WHERE rank = 1
```

-- 8. Write a SQL query to find the top 5 customers based on the highest total sales:
``` sql
SELECT customer_id, SUM(total_sale) AS total_sales
FROM Retail_Sales
GROUP BY 1
ORDER BY 2 DESC
```

```sql
LIMIT 5;
```

-- 9. Write a SQL query to find the number of unique customers who purchased items from each category.:

```sql
SELECT category, COUNT(DISTINCT customer_id) AS unique_customers
FROM Retail_Sales
GROUP BY category;
```

-- 10. Write a SQL query to create each shift and number of orders (Example Morning <12, Afternoon Between 12 & 17, Evening >17):

```sql
SELECT
   CASE
      WHEN EXTRACT(HOUR FROM sale_time) < 12 THEN 'Morning'
      WHEN EXTRACT(HOUR FROM sale_time) BETWEEN 12 AND 17 THEN 'Afternoon'
      ELSE 'Evening'
   END AS shift,
   COUNT(transaction_id) AS order_count
FROM Retail_Sales
GROUP BY shift
ORDER BY order_count DESC;
```

Findings

- **Customer Demographics**: The dataset includes customers from various age groups, with sales distributed across different categories such as Clothing and Beauty.
- **High-Value Transactions**: Several transactions had a total sale amount greater than 1000, indicating premium purchases.
- **Sales Trends**: Monthly analysis shows variations in sales, helping identify peak seasons.
- **Customer Insights**: The analysis identifies the top-spending customers and the most popular product categories.

Reports

- **Sales Summary**: A detailed report summarizing total sales, customer demographics, and category performance.
- **Trend Analysis**: Insights into sales trends across different months and shifts.
- **Customer Insights**: Reports on top customers and unique customer counts per category.

Conclusion

This project serves as a comprehensive introduction to SQL for data analysts, covering database setup, data cleaning, exploratory data analysis, and business-driven SQL queries. The findings from this project can help drive business decisions by understanding sales patterns, customer behavior, and product performance.

How to Use

1. **Clone the Repository**: Clone this project repository from GitHub.
2. **Set Up the Database**: Run the SQL scripts provided in the `Database Setup Section` file to create and populate the database.
3. **Run the Queries**: Use the SQL queries provided in the `Data Analysis & Findings Section` file to perform your analysis.
4. **Explore and Modify**: Feel free to modify the queries to explore different aspects of the dataset or answer additional business questions.

Author - Pranav Shah

This project is part of my portfolio, showcasing the SQL skills essential for data analyst roles. If you have any questions, feedback, or would like to collaborate, feel free to get in touch!

Thank you for your support, and I look forward to connecting with you!

https://hsbc.taleo.net/careersection/careersection/candidateacquisition/screening/controller/externalServiceController.jsp?sealedRequestId=yxkT8b6SpguQLsEeB8L0Pe-cJUJumK4TqJPc26-RVAEGdL6cEQJoOdh3hKwWBUZzOgZ4dImC_cbNWOl6EbVZ1Nez4GgA2WPOmHHzjC0qtl__zISc_R8fS9IXiUy9WEhOhGD7nx-tWzkE68X60QFTgw==&portal=101430233&lang=en-GB