

# RETAIL SALES ANALYSIS

BY- KARTIK JAIN



# INTRODUCTION

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.

# OBJECTIVE

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





# DATABASE SETUP

- Database Creation: The project starts by creating a database named `retailsalesanalysis`.
- 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.

```
create DATABASE retailsalesanalysis;

-- Create TABLE
DROP TABLE IF EXISTS retail_sales;
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,
    cogs FLOAT,
    total_sale FLOAT
);
```

# DATA 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.

```
-- DATA CLEANING--  
-- CHECKING FOR NULL VALUE --  
SELECT * FROM retail_sales  
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 quantiy IS NULL  
    OR price_per_unit IS NULL  
    OR cogs IS NULL  
    OR total_sale IS NULL;
```

```
-- DELETE ROWS WITH NULL VALUE--  
DELETE FROM retail_sales 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 quantiy IS NULL  
    OR price_per_unit IS NULL  
    OR cogs IS NULL  
    OR total_sale IS NULL;
```



# DATA EXPLORATION

```
-- DATA EXPLORATION--
```

```
-- HOW MANY TOTAL SALE WE HAVE--
```

```
SELECT
```

```
    sum(total_sale) AS total_sales
```

```
FROM retail_sales;
```

```
-- how many unique customer we have? --
```

```
SELECT
```

```
    COUNT(distinct customer_id) AS Total_Customer
```

```
FROM retail_sales;
```

```
-- how many category we have ? --
```

```
SELECT
```

```
    distinct category
```

```
FROM retail_sales;
```

# DATA ANALYSIS & FINDINGS

```
-- Q1 WRITE A SQL QUERY TO RETRIVE  
-- ALL COLUMNS FOR SALES MDAE ON '2022-11-05'  
SELECT  
    *  
FROM  
    retail_sales  
WHERE  
    sale_date = '2022-11-05';
```

```
-- Q2 write a sql query to calculate total sale  
-- and total order for each category  
SELECT  
    category,  
    SUM(total_sale) AS net_sale,  
    COUNT(*) AS total_order  
FROM  
    retail_sales  
GROUP BY category;
```

# DATA ANALYSIS & FINDINGS

```
-- Q3 write a sql query where retrieve all transactions_id where the category is 'clothing'  
-- and the quantity sold is more than 3 and the month of nov-2022  
  
SELECT  
    *  
FROM  
    retail_sales  
WHERE  
    category = 'Clothing'  
    AND quanti >=4  
    AND sale_date BETWEEN '2022-11-01' AND '2022-11-30';
```



# DATA ANALYSIS & FINDINGS

```
-- Q4 write a sql query to find the avg age of customer  
-- who purchased the item from the beauty category
```

```
SELECT  
    AVG(age)  
FROM  
    retail_sales  
WHERE  
    category = 'beauty';
```

```
-- Q5 Write a SQL query to find all transactions  
-- where the total_sale is greater than 1000
```

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

# DATA ANALYSIS & FINDINGS

```
-- Q6 Write a SQL query to find the total number of transaction  
-- (transaction_id) made by each gender in each category.:  
SELECT  
    category, gender, COUNT(*) AS total_transaction  
FROM  
    retail_sales  
GROUP BY category , gender  
ORDER BY category;
```

```
-- Q7 Write a SQL query to find the top 5 customers  
-- based on the highest total sales  
SELECT  
    customer_id, SUM(total_sale) AS total_sales  
FROM  
    retail_sales  
GROUP BY customer_id  
ORDER BY total_sales DESC  
LIMIT 5;
```

# DATA ANALYSIS & FINDINGS

```
-- Q8 Write a SQL query to calculate the average sale for each month.  
-- Find out best selling month in each year:  
select * from (  
  SELECT  
    YEAR(sale_date) AS year,  
    MONTH(sale_date) AS month,  
    AVG(total_sale) AS avg_sale,  
    rank() over (partition by YEAR(sale_date) order by  AVG(total_sale) desc ) as ranking  
  FROM  
    retail_sales  
  GROUP BY year , month  
) as t1  
where ranking =1 ;
```

# DATA ANALYSIS & FINDINGS

```
-- Q9 Write a SQL query to find the number of unique customers  
-- who purchased items from each category  
SELECT  
    COUNT(DISTINCT customer_id) AS total_customer, category  
FROM  
    retail_sales  
GROUP BY category;
```

# DATA ANALYSIS & FINDINGS

```
-- Q10 Write a SQL query to create each shift and number of orders
-- (Example Morning <12, Afternoon Between 12 & 17, Evening >17)
with hourly_sale as
(
  SELECT *,
    CASE
      WHEN sale_time < '12:00:00' THEN 'morning'
      WHEN sale_time BETWEEN '12:00:00' AND '17:00:00' THEN 'afternoon'
      WHEN sale_time > '17:00:00' THEN 'evening'
      ELSE 'no'
    END AS shift
  FROM retail_sales
)
select shift, count(transactions_id) from hourly_sale
group by shift
```

# THANK YOU

28 NOV , 2024