Report and Queries

Index

- Gender distribution

2.5 Revenue Analysis

- Total revenue

- Revenue by category

- Revenue by gender

2.6 Time-Based EDA

- Daily sales

- Monthly sales

- Peak sales hours

2.7 Customer-Level EDA

- Top customers

- Average sales per customer

---

3. Profitability Analysis

3.1 Profit Analysis

- Profit per transaction

- Profit by category

- Profit by gender

---

4. Outlier Detection

- Identification of very high sales values

---

5. Answering Business Questions

Problem Statement

Retail businesses generate large volumes of transactional data daily, but raw data alone does not provide actionable insights. Without proper cleaning, structuring, and analysis, it becomes difficult to understand customer behaviour, identify top-performing products, track revenue trends, and measure profitability.

The challenge is to transform raw retail sales data into meaningful insights that can support data-driven decision-making, such as improving sales performance, optimizing inventory, and understanding customer purchasing patterns.

Data Cleaning

since data is imported csv file so we finalize data types of all columns to avoid further complications

Renaming 2 columns

```
ALTER TABLE retail_sales
RENAME COLUMN ï»¿transactions_id TO transaction_id;
ALTER TABLE retail_sales
RENAME COLUMN quantiy TO quantity;
```

1) IDs & Numbers

```
ALTER TABLE retail_sales
MODIFY transaction_id INT PRIMARY KEY,
MODIFY customer_id INT;
-- MODIFY age INT NULL,            -- problem for both due to ' ' values
-- MODIFY quantity INT;
```

found 10 blank age, 3 blank quantiy

```
/* SELECT DISTINCT quantiy, count(*)
FROM retail_sales
group by quantiy; */
```

(optional) change setting in mysql workbench

```
SET SQL_SAFE_UPDATES = 0;
```

changing blank values in age to null and then converting age col to int

```sql
UPDATE retail_sales
SET age = NULL
WHERE age = '' OR age NOT REGEXP '^[0-9]+$';
ALTER TABLE retail_sales
MODIFY age INT NULL;
```

changing blank values in quantity to null and then coverting quantity col to int

```sql
UPDATE retail_sales
SET quantity = NULL
WHERE quantity = '' OR quantity NOT REGEXP '^[0-9]+$';
ALTER TABLE retail_sales
MODIFY quantity INT;

SET SQL_SAFE_UPDATES = 1;
```

2) Dates & Time

```sql
ALTER TABLE retail_sales
MODIFY sale_date DATE;
```

fixed formatting issue of time from 19.10.00 to 19:10:00 before changing datatype

```sql
SET SQL_SAFE_UPDATES = 0;
UPDATE retail_sales
SET sale_time = REPLACE(sale_time, '.', ':');
SET SQL_SAFE_UPDATES = 1;

ALTER TABLE retail_sales
MODIFY sale_time TIME;
```

3) Text Columns

```sql
ALTER TABLE retail_sales
MODIFY gender VARCHAR(10),
MODIFY category VARCHAR(50);
```

4) Monetary Columns

```sql
SET SQL_SAFE_UPDATES = 0;
```

changing blank values in price_per_unit to null and then coverting price_per_unit col to decimal

```sql
UPDATE retail_sales
SET  price_per_unit = NULL
WHERE  price_per_unit = '' OR  price_per_unit NOT REGEXP '^[0-9]+(\\.[0-9]+)?$';
ALTER TABLE retail_sales
MODIFY  price_per_unit DECIMAL(10,2);
```

changing blank values in cogs to null and then coverting cogs col to decimal

```sql
UPDATE retail_sales
SET  cogs = NULL
WHERE  cogs = '' OR  cogs NOT REGEXP '^[0-9]+(\\.[0-9]+)?$';
ALTER TABLE retail_sales
MODIFY  cogs DECIMAL(10,2);
```

changing blank values in total_sale to null and then coverting total_sale col to decimal

```sql
UPDATE retail_sales
SET  total_sale = NULL
WHERE  total_sale = '' OR  total_sale NOT REGEXP '^[0-9]+(\\.[0-9]+)?$';
ALTER TABLE retail_sales
MODIFY  total_sale DECIMAL(10,2);
```

```sql
SET SQL_SAFE_UPDATES = 1;
```

EDA

-- (1)--- Overview of data

-- (i)Check uniqueness

```sql
SELECT
    COUNT(*) as total_records,
    COUNT(DISTINCT transaction_id) as total_transactions,
    COUNT(DISTINCT customer_id) as total_customers,
    COUNT(DISTINCT category) as total_category
FROM
    retail_sales;
```

- ■ ***155 customers has done 2000 transactions in total 3 categories***

-- (ii) Checking null values

```sql
SELECT
    SUM(sale_date IS NULL)     AS sale_date_nulls,
    SUM(sale_time IS NULL)     AS sale_time_nulls,
    SUM(customer_id IS NULL)   AS customer_id_nulls,
    SUM(gender IS NULL)        AS gender_nulls,
    SUM(age IS NULL)           AS age_nulls,
    SUM(category IS NULL)      AS category_nulls,
    SUM(quantity IS NULL)      AS quantity_nulls,
    SUM(price_per_unit IS NULL) AS price_per_unit_nulls,
    SUM(cogs IS NULL)          AS cogs_nulls,
    SUM(total_sale IS NULL)    AS total_sale_nulls
FROM retail_sales;
```

- ■ ***null values found: age=10, quantity=3, price_per_unit=3, cogs=3, total_sales=3***

```sql
SELECT *
FROM retail_sales
WHERE quantity IS NULL
    OR price_per_unit IS NULL
    OR cogs IS NULL
    OR total_sale IS NULL;
```

- ■ ***3 rows where quantity, price_per_unit, cogs and total_sale all are null***
- ■ ***therefore dropping those 3 rows***

```sql
SET SQL_SAFE_UPDATES = 0;
```

```sql
DELETE FROM retail_sales
WHERE quantity IS NULL
    OR price_per_unit IS NULL
    OR cogs IS NULL
    OR total_sale IS NULL;

SET SQL_SAFE_UPDATES = 1;
```

-- (iii) Checking duplicats

```sql
SELECT transaction_id, COUNT(*) AS counts
FROM retail_sales
GROUP BY transaction_id
HAVING COUNT(*) > 1;
```
  - ***no duplicate found***

-- (i) MIN, MAX, AVG
SELECT
    MIN(age) AS min_age,
    MAX(age) AS max_age,
    ROUND(AVG(age), 0) AS avg_age,
    ROUND(AVG(price_per_unit), 2) AS avg_price_per_unit,
    ROUND(AVG(cogs), 2) AS avg_cogs,
    ROUND(AVG(quantity), 0) AS avg_quantity,
    ROUND(AVG(total_sale), 2) AS avg_sales
FROM
    retail_sales;

min age	max age	avg age	avg price per unit	avg cogs	avg quantity	avg sales
18	64	41	180.12	95.02	3	456.54

-- (3)-- Categorical Distribution

-- (i) Sales by category

```
SELECT
    category,
    SUM(transaction_id) AS total_tarnsactions,
    ROUND(AVG(total_sale), 2) AS avg_sale
FROM
    retail_sales
GROUP BY category
ORDER BY ROUND(AVG(total_sale), 2) DESC;
```

Category	Total Transactions	Average Sale
Beauty	606,824	468.69
Electronics	693,822	458.79
Clothing	697,704	443.75

-- (ii) Gender distribution

```
SELECT gender, COUNT(*) AS cnt
FROM retail_sales
GROUP BY gender;
```

Gender	Count
Male	980
Female	1017

- ■ *there are more female customers*

```
SELECT gender,
        SUM(quantity) AS total_quantity,
    AVG(total_sale) AS avg_total_sale
FROM retail_sales
GROUP BY gender;
```

Gender	Total Quantity Sold	Average Total Sale
Male	2432	455.43
Female	2586	457.62

- ■ *note that there is no significant difference in sales based on gender*

-- (i) Total Revenue

```
SELECT SUM(total_sale) AS total_revenue
FROM retail_sales;
```

Metric	Value
Total Revenue	911,720.00

-- (ii) Revenue by category

```
SELECT category, SUM(total_sale) AS total_revenue
FROM retail_sales
GROUP BY category;
```

Category	Total Revenue
Beauty	286,840.00
Clothing	311,070.00
Electronics	313,810.00

-- (iii) Revenue by gender

```
SELECT gender, SUM(total_sale) AS total_revenue
FROM retail_sales
GROUP BY gender;
```

Gender	Total Revenue
Male	446,320.00
Female	465,400.00

-- (i) Daily sales

```sql
SELECT day(sale_date) AS dates, SUM(total_sale) AS top_sales
FROM retail_sales
GROUP BY day(sale_date)
ORDER BY SUM(total_sale) DESC
LIMIT 3;
```

Day	Top Sales
10	41,460.00
26	38,580.00
13	38,070.00

- **sales are low in mid of months and hign on starting and endings of months**

```sql
SELECT dayname(sale_date) AS days, SUM(total_sale) AS top_sales
FROM retail_sales
GROUP BY dayname(sale_date)
ORDER BY SUM(total_sale) DESC
LIMIT 3;
```

Day	Top Sales
Sunday	153,800.00
Monday	147,695.00
Saturday	141,400.00

- **sales are high on weekends**

-- (ii) Monthly sales

```sql
SELECT
    YEAR(sale_date) AS yr,
    MONTH(sale_date) AS mn,
    SUM(total_sale) AS revenue
FROM retail_sales
```

```
GROUP BY yr, mn
ORDER BY revenue desc
LIMIT 3;
```

Year	Month	Revenue
2022	12	72,880.00
2023	12	69,145.00
2022	11	68,915.00

- **■ *sales are high in year-end in both year***

-- (iii) Peak Hours

```
SELECT sale_time AS peak_hour, SUM(total_sale) AS total_sales
FROM retail_sales
GROUP BY sale_time
ORDER BY SUM(total_sale) DESC
LIMIT 3;
```

Peak Hour	Total Sales
18:05:00	7,200.00
18:13:00	6,400.00
17:12:00	5,670.00

- **■ *peak hours are evening hours***

-- (6)--- Customer Level EDA

-- (i) Top customer

SELECT customer_id, SUM(total_sale) AS total_spent
FROM retail_sales
GROUP BY customer_id
ORDER BY total_spent DESC
LIMIT 3;

customer_id	total_spent
3	38,440.00
1	30,750.00
5	30,405.00

- **customer 1,3,5 are top customers**

-- (ii) Average sales per customer

SELECT ROUND(AVG(total_spent), 2) AS avg_sales_per_customer
FROM (
    SELECT customer_id, SUM(total_sale) AS total_spent
    FROM retail_sales
    GROUP BY customer_id
) t;

- **avg_sales_per_customer = 5882.06**

cogs = Cost of Goods Sold

-- (i) Profit per transaction
SELECT transaction_id,
           ROUND((total_sale - cogs), 2) AS profit_per_transaction
FROM retail_sales
ORDER BY profit_per_transaction DESC
LIMIT 3;

transaction_id	profit_per_transaction
1416	1875.00
875	1865.00
93	1860.00

- ***max profit_per_transaction = 1875***
- ***min profit_per_transaction = -120 (loss of 120)***

-- (ii) Profit by category
SELECT category,
      ROUND(SUM(total_sale - cogs), 2) AS profit_per_category
FROM retail_sales
GROUP BY category
ORDER BY profit_per_category DESC;

Category	Profit per Category
Clothing	246,679.50
Electronics	246,647.65
Beauty	228,630.15

- ***clothing has highest profit***

-- (iii) Profit by gender

```
SELECT gender,
     ROUND(SUM(total_sale - cogs), 2) AS profit_per_gender
FROM retail_sales
GROUP BY gender
ORDER BY profit_per_gender DESC;
/*
```

Gender	Total Sales
Female	362,034.85
Male	359,922.45

*/

- ■ *females contributes more in profit than male*

-- (8)-- Outlier Detection

-- (i) Very high sales

```
SELECT COUNT(outlier) AS total_outliers
FROM
    (SELECT total_sale AS outlier
    FROM retail_sales
    WHERE total_sale > (
                    SELECT AVG(total_sale) + 3 * STDDEV(total_sale)
        FROM retail_sales
        )
        ) t;
```

- **total_outliers = 0**

Answering Business Questions

1. Write a SQL query to retrieve all columns for sales made on 2022-11-05

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 more than 4 in the month of Nov-2022

SELECT *
FROM retail_sales
WHERE category = 'Clothing'
            AND quantity > 4
    AND MONTH(sale_date) = 11
    AND YEAR(sale_date) = 2022;

3. Write a SQL query to calculate the total sales (total_sale) for each category.

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.

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

5. Write a SQL query to find all transactions where the total_sale is greater than 1000.

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.

SELECT gender, category, COUNT(transaction_id) AS total_transactions
FROM retail_sales
GROUP BY gender , category
ORDER BY gender , category;

7. Write a SQL query to calculate the average sale for each month. Find out best selling month in each year

SELECT YEAR(sale_date) AS years,
    MONTH(sale_date) AS months,
    AVG(total_sale) AS avg_sales
FROM retail_sales
GROUP BY years , months;
        /* find monthly-sales and then rank them and then extract top month*/

with monthly_sales as (
select year(sale_date) as years, month(sale_date) as months, sum(total_sale) as total_sales
from retail_sales
group by years, months
),
ranked_sales as (
select years, months, total_sales, dense_rank() over(partition by years order by total_sales
desc) as rn
from monthly_sales
)
select * from ranked_sales where rn=1;

8. Write a SQL query to find the top 5 customers based on the highest total sales.

select customer_id as top_5_customers , sum(total_sale) as total_spent
from retail_sales
group by customer_id
order by total_spent desc
limit 5;

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

SELECT DISTINCT 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)

SELECT
    CASE
        WHEN HOUR(sale_time) < 12 THEN 'Morning'
        WHEN HOUR(sale_time) > 17 THEN 'Evening'
        ELSE 'Afternoon'
    END AS shift,
    COUNT(transaction_id) AS total_orders
FROM retail_sales
GROUP BY shift;

11. Find Top 2 Customers per Category

select category, customer_id, total_spent, rn as ranks
from (
        select category,
                    customer_id,
        sum(total_sale) as total_spent,
        dense_rank() over(partition by category order by sum(total_sale) desc) as rn
 from retail_sales
 group by category, customer_id
) t
where rn<=2
order by category, rn;

12. Calculate Month-over-Month (MoM) Sales Growth

select yr_month, total_sales,
                lag(total_sales) over(order by yr_month) as prev_month_sales,
                round(((total_sales - lag(total_sales) over(order by yr_month)) /
((lag(total_sales) over(order by yr_month))))*100 , 2) as mom_growth_pct
from (
        select date_format(sale_date, '%Y-%m') as yr_month, sum(total_sale) as total_sales

```
        from retail_sales
        group by yr_month) as t
order by yr_month;
```

13. Find Running Total per Customer

```
select customer_id, sale_date, total_sale,
sum(total_sale) over(partition by customer_id order by sale_date rows between unbounded
preceding and current row) as running_total
from retail_sales;
```