ATER DENG MAYEN

SQL PROJECT WORDS REPORT

1. Retrieve all transactions with valid customer and product data.

Order by transaction date to understand the chronological flow of

purchases.

SELECT *

FROM retail

WHERE "Customer ID" IS NOT NULL AND "Product Category" IS NOT NULL

ORDER BY DATE("Date");

```
2 •  SELECT *
3    FROM retail
4    WHERE "Customer ID" IS NOT NULL AND "Product Category" IS NOT NULL
5    ORDER BY DATE("Date");
6
7
8 •  SELECT DISTINCT *
```

esult Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
1	2023-11-24	CUST001	Male	34	Beauty	3	50	150
2	2023-02-27	CUST002	Female	26	Clothing	2	500	1000
3	2023-01-13	CUST003	Male	50	Electronics	1	30	30
4	2023-05-21	CUST004	Male	37	Clothing	1	500	500
5	2023-05-06	CUST005	Male	30	Beauty	2	50	100
6	2023-04-25	CUST006	Female	45	Beauty	1	30	30
7	2023-03-13	CUST007	Male	46	Clothing	2	25	50
8	2023-02-22	CUST008	Male	30	Electronics	4	25	100
9	2023-12-13	CUST009	Male	63	Electronics	2	300	600
10	2023-10-07	CUST010	Female	52	Clothing	4	50	200
11	2023-02-14	CUST011	Male	23	Clothing	2	50	100
12	2023-10-30	CUST012	Male	35	Beauty	3	25	75
13	2023-08-05	CUST013	Male	22	Electronics	3	500	1500
14	2023-01-17	CUST014	Male	64	Clothing	4	30	120
15	2023-01-16	CUST015	Female	42	Electronics	4	500	2000
16	2023-02-17	CUST016	Male	19	Clothing	3	500	1500

2. Clean the dataset by ensuring that numeric fields like Quantity,

Price per Unit, and Total Amount are properly formatted.

Remove duplicates or null values if any exist.

SELECT DISTINCT *

FROM retail

WHERE Quantity IS NOT NULL AND "Price per Unit" IS NOT NULL AND "Total Amount" IS NOT NULL;

```
7   -- Clean the dataset by ensuring that numeric fields like Quantity,  Price per Unit, and Total Amount are properly formatted.
8   -- Remove duplicates or null values if any exist.
9
10  SELECT DISTINCT *
11  FROM retail
12  WHERE Quantity IS NOT NULL AND "Price per Unit" IS NOT NULL AND "Total Amount" IS NOT NULL;
13
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
1	2023-11-24	CUST001	Male	34	Beauty	3	50	150
2	2023-02-27	CUST002	Female	26	Clothing	2	500	1000
3	2023-01-13	CUST003	Male	50	Electronics	1	30	30
4	2023-05-21	CUST004	Male	37	Clothing	1	500	500
5	2023-05-06	CUST005	Male	30	Beauty	2	50	100
6	2023-04-25	CUST006	Female	45	Beauty	1	30	30
7	2023-03-13	CUST007	Male	46	Clothing	2	25	50
8	2023-02-22	CUST008	Male	30	Electronics	4	25	100
9	2023-12-13	CUST009	Male	63	Electronics	2	300	600
10	2023-10-07	CUST010	Female	52	Clothing	4	50	200
11	2023-02-14	CUST011	Male	23	Clothing	2	50	100
12	2023-10-30	CUST012	Male	35	Beauty	3	25	75
13	2023-08-05	CUST013	Male	22	Electronics	3	500	1500
14	2023-01-17	CUST014	Male	64	Clothing	4	30	120
15	2023-01-16	CUST015	Female	42	Electronics	4	500	2000
16	2023-02-17	CUST016	Male	19	Clothing	3	500	1500

etail 3 ×

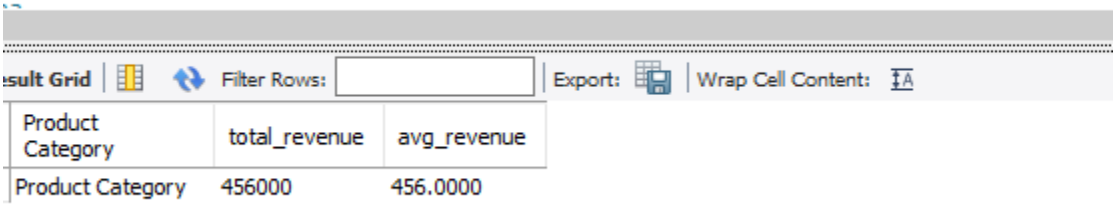
3. Calculate the total and average revenue for each product

category.

Which categories bring in the most and least revenue?

```
SELECT "Product Category",
    SUM(`Total Amount`) AS total_revenue,
    AVG(`Total Amount`) AS avg_revenue
FROM retail
GROUP BY "Product Category"
ORDER BY total_revenue DESC;
```

```
14      --  3Calculate the total and average revenue for each product  category.
15      -- Which categories bring in the most and least revenue
16
17  •   SELECT "Product Category",
18             SUM(`Total Amount`) AS total_revenue,
19             AVG(`Total Amount`) AS avg_revenue
20      FROM retail
21      GROUP BY "Product Category"
22      ORDER BY total_revenue DESC;
```



Product Category	total_revenue	avg_revenue
Product Category	456000	456.0000

4. Analyze the monthly sales trend over the entire dataset period.

Summarize total revenue per month and order the results

chronologically.

```
SELECT monthname(`Date`) AS month,
    SUM(`Total Amount`) AS total_revenue
FROM retail
GROUP BY month
ORDER BY month;
```

```
5       -- 4. Analyze the monthly sales trend over the entire dataset period.
6       -- Summarize total revenue per month and order the results  chronologically.
7
8 •     SELECT monthname(`Date`) AS month,
9              SUM(`Total Amount`) AS total_revenue
0       FROM retail
1       GROUP BY month
2       ORDER BY month;
```

sult Grid | Filter Rows: | Export: | Wrap Cell Content:

month	total_revenue
April	33870
August	36960
December	44690
February	44060
January	36980
July	35465
June	36715
March	28990
May	53150
November	34920
October	46580

ult 8 ×

5. Identify the top 10 customers by total spending.

Rank customers based on how much they've spent across all

transactions.

```
SELECT "Customer ID",
    SUM(`Total Amount`) AS total_spent
FROM retail
GROUP BY "Customer ID"
ORDER BY total_spent DESC
LIMIT 10;
```

```
33
34      -- 5 Identify the top 10 customers by total spending.
35      -- Rank customers based on how much they've spent across all  transactions.
36
37      SELECT "Customer ID",
38             SUM(`Total Amount`) AS total_spent
39      FROM retail
40      GROUP BY "Customer ID"
41      ORDER BY total_spent DESC
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Customer ID	total_spent
Customer ID	456000

6. Calculate the average transaction value for each customer.

How much does each customer spend per transaction on

average?

```
SELECT "Customer ID",
    AVG(`Total Amount`) AS avg_transaction_value
FROM retail
GROUP BY "Customer ID"
ORDER BY avg_transaction_value DESC;
```

```
43      -- 6 Calculate the average transaction value for each customer. How much does each customer spend per transaction on  average?
44
45      SELECT "Customer ID",
46             AVG(`Total Amount`) AS avg_transaction_value
47      FROM retail
48      GROUP BY "Customer ID"
49      ORDER BY avg_transaction_value DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Customer ID	avg_transaction_value
Customer ID	456.0000

Result 14

7. Group customers by gender and age brackets (e.g., 18–25, 26–35,

36–50, etc.).

Summarize total revenue and transaction count for each group.

```
SELECT Gender,
    CASE
      WHEN Age BETWEEN 18 AND 25 THEN '18–25'
      WHEN Age BETWEEN 26 AND 35 THEN '26–35'
      WHEN Age BETWEEN 36 AND 50 THEN '36–50'
      ELSE '51+'
    END AS age_bracket,
    COUNT(*) AS transaction_count,
    SUM(`Total Amount`) AS total_revenue
FROM retail
GROUP BY Gender, age_bracket;
```

```sql
-- 7Group customers by gender and age brackets (e.g., 18-25, 26-35,  36-50, etc.
-- Summarize total revenue and transaction count for each group

SELECT Gender,
       CASE
           WHEN Age BETWEEN 18 AND 25 THEN '18-25'
           WHEN Age BETWEEN 26 AND 35 THEN '26-35'
           WHEN Age BETWEEN 36 AND 50 THEN '36-50'
           ELSE '51+'
       END AS age_bracket,
       COUNT(*) AS transaction_count,
       SUM(`Total Amount`) AS total_revenue
FROM retail
GROUP BY Gender, age_bracket;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Gender	age_bracket	transaction_count	total_revenue
Male	26–35	98	43365
Female	26–35	107	55115
Male	36–50	149	68870
Female	36–50	164	70790
Male	51+	155	65845
Female	51+	158	67465

Result 15

8. Compare the number of one-time buyers versus repeat buyers.

Group customers by purchase frequency to determine repeat

behavior.

```sql
SELECT purchase_count,
    COUNT(*) AS customer_count
FROM (
  SELECT "Customer ID", COUNT(*) AS purchase_count
  FROM retail
  GROUP BY "Customer ID"
) sub
```

GROUP BY purchase_count

ORDER BY purchase_count;

```
64      -- 8. Compare the number of one-time buyers versus repeat buyers.
65      -- Group customers by purchase frequency to determine repeat  behavior.
66
67  •   SELECT purchase_count,
68             COUNT(*) AS customer_count
69      FROM (
70          SELECT "Customer ID", COUNT(*) AS purchase_count
71          FROM retail
72          GROUP BY "Customer ID"
73      ) sub
74      GROUP BY purchase_count
75      ORDER BY purchase_count;
76
77
78
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

purchase_count	customer_count
1000	1

Result 16

9. Identify inactive customers who have not made a purchase in the

last 6 months.

Use the most recent date in the dataset as the reference point.

WITH LatestDate AS (

SELECT MAX(Date) AS MaxDate FROM retail

),

LastPurchase AS (

SELECT `Customer ID`, MAX(Date) AS LastPurchaseDate

FROM retail

GROUP BY `Customer ID`

)

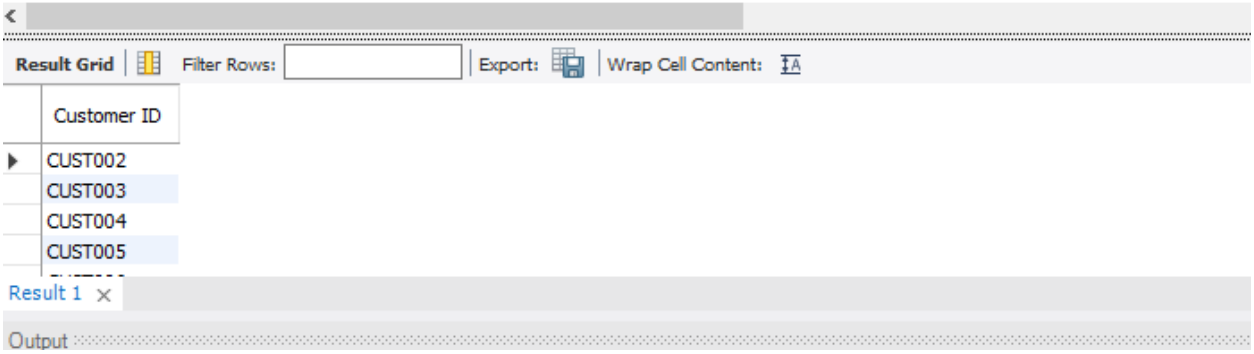
SELECT lp.`Customer ID`

FROM LastPurchase lp

JOIN LatestDate ld

WHERE lp.LastPurchaseDate < DATE_SUB(ld.MaxDate, INTERVAL 6 MONTH);

```
76      -- 9 Identify inactive customers who have not made a purchase in the  last 6 months.
77      -- Use the most recent date  in the dataset as the reference point.
78      WITH LatestDate AS (
79        SELECT MAX(Date) AS MaxDate FROM retail
80      ),
81      LastPurchase AS (
82        SELECT `Customer ID`, MAX(Date) AS LastPurchaseDate
83        FROM retail
84        GROUP BY `Customer ID`
85      )
86      SELECT lp.`Customer ID`
87      FROM LastPurchase lp
88      JOIN LatestDate ld
89      WHERE lp.LastPurchaseDate < DATE_SUB(ld.MaxDate, INTERVAL 6 MONTH);
90
```



10. Perform RFM (Recency, Frequency, Monetary) analysis for

customer segmentation.

Recency: Days since last purchase; Frequency: Number of

purchases; Monetary: Total amount spent.

WITH LatestDate AS (

SELECT MAX(Date) AS MaxDate FROM retail

),

```
CustomerStats AS (
  SELECT
    `Customer ID`,
    MAX(Date) AS LastPurchaseDate,
    COUNT(*) AS Frequency,
    SUM(`Total Amount`) AS Monetary
  FROM retail
  GROUP BY `Customer ID`
)
SELECT
  cs.`Customer ID`,
  DATEDIFF(ld.MaxDate, cs.LastPurchaseDate) AS Recency,
  cs.Frequency,
  cs.Monetary
FROM CustomerStats cs
JOIN LatestDate ld;
```

```
90      -- 10. Perform RFM (Recency, Frequency, Monetary) analysis for
91      -- customer segmentation.
92      -- Recency: Days since last purchase; Frequency: Number of
93      -- purchases; Monetary: Total amount spent.
94      WITH LatestDate AS (
95        SELECT MAX(Date) AS MaxDate FROM retail
96      ),
97      CustomerStats AS (
98        SELECT
99          `Customer ID`,
100         MAX(Date) AS LastPurchaseDate,
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Customer ID	Recency	Frequency	Monetary
CUST001	38	1	150
CUST002	308	1	1000
CUST003	353	1	30
CUST004	225	1	500
CUST005	240	1	100
CUST006	251	1	30
CUST007	294	1	50
CUST008	313	1	100
CUST009	19	1	600

Result 2 ×

Output

11. Find the product categories with the highest average quantity per

transaction.

Which product types are purchased in bulk?

```
SELECT
 `Product Category`,
 AVG(`Quantity`) AS AvgQuantityPerTransaction
FROM retail
GROUP BY `Product Category`
ORDER BY AvgQuantityPerTransaction DESC;
```

```
113     -- 11. Find the product categories with the highest average quantity per
114     -- transaction.
115     -- Which product types are purchased in bulk?
116     SELECT
117       `Product Category`,
118       AVG(`Quantity`) AS AvgQuantityPerTransaction
119     FROM retail
120     GROUP BY `Product Category`
121     ORDER BY AvgQuantityPerTransaction DESC;
122
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Product Category	AvgQuantityPerTransaction
Clothing	2.5470
Beauty	2.5114
Electronics	2.4825

Result 3

12. Identify the busiest sales day of the week.

Which day(s) consistently have the highest transaction volume or

revenue?

```
SELECT
  DAYNAME(`Date`) AS DayOfWeek,
  COUNT(*) AS TransactionCount,
  SUM(`Total Amount`) AS TotalRevenue
FROM retail
GROUP BY DayOfWeek
ORDER BY TransactionCount DESC;
```

```
121     ORDER BY AvgQuantityPerTransaction DESC;
122     -- 12. Identify the busiest sales day of the week.
123     -- Which day(s) consistently have the highest transaction volume or
124     -- revenue?
125     SELECT
126       DAYNAME(`Date`) AS DayOfWeek,
127       COUNT(*) AS TransactionCount,
128       SUM(`Total Amount`) AS TotalRevenue
129     FROM retail
130     GROUP BY DayOfWeek
131     ORDER BY TransactionCount DESC;
132
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

DayOfWeek	TransactionCount	TotalRevenue
Tuesday	161	69440
Saturday	150	78815
Monday	146	70250
Friday	143	66290
Wednesday	139	58770

Result 4

Output

Action Output

#	Time	Action	Message
46	00:58:59	SELECT `Product Category`, AVG(`Quantity`) AS AvgQuantityPerTransaction FROM retail ...	3 row(s) returned
47	01:05:01	SELECT DAYNAME(`Date`) AS DayOfWeek, COUNT(*) AS TransactionCount, SUM(`To...	7 row(s) returned

13. Calculate total revenue and average spend per transaction by

gender.

Are there differences in spending patterns across genders?

```
SELECT
  Gender,
  SUM(`Total Amount`) AS TotalRevenue,
  AVG(`Total Amount`) AS AvgSpendPerTransaction
FROM retail
GROUP BY Gender;
```

```sql
-- 13 Calculate total revenue and average spend per transaction by
-- gender.
-- Are there differences in spending patterns across genders?
SELECT
  Gender,
  SUM(`Total Amount`) AS TotalRevenue,
  AVG(`Total Amount`) AS AvgSpendPerTransaction
FROM retail
GROUP BY Gender;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Gender	TotalRevenue	AvgSpendPerTransaction
Male	223160	455.4286
Female	232840	456.5490

Result 5

Output

Action Output

#	Time	Action	Message
47	01:05:01	SELECT DAYNAME('Date') AS DayOfWeek, COUNT(*) AS TransactionCount, SUM('To...	7 row(s) returned
48	01:10:26	SELECT Gender, SUM('Total Amount') AS TotalRevenue, AVG('Total Amount') AS Avg...	2 row(s) returned

14. Find the top 5 most frequently purchased product categories.

Based on number of transactions involving each category.

```sql
SELECT
 `Product Category`,
 COUNT(*) AS TransactionCount
FROM retail
GROUP BY `Product Category`
ORDER BY TransactionCount DESC
LIMIT 5;
```

```sql
-- 14. Find the top 5 most frequently purchased product categories.
-- Based on number of transactions involving each category.
SELECT
  `Product Category`,
  COUNT(*) AS TransactionCount
FROM retail
GROUP BY `Product Category`
ORDER BY TransactionCount DESC
LIMIT 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Product Category	TransactionCount
Clothing	351
Electronics	342
Beauty	307

15. Determine the percentage of total revenue contributed by each

age group.

Which customer age brackets are most valuable to the business?

```
WITH AgeBracketed AS (
 SELECT *,
   CASE
     WHEN Age BETWEEN 18 AND 25 THEN '18–25'
     WHEN Age BETWEEN 26 AND 35 THEN '26–35'
     WHEN Age BETWEEN 36 AND 50 THEN '36–50'
     WHEN Age BETWEEN 51 AND 65 THEN '51–65'
     ELSE 'Other'
   END AS AgeGroup
 FROM retail
),
GroupRevenue AS (
```

```
 SELECT AgeGroup, SUM(`Total Amount`) AS Revenue
 FROM AgeBracketed
 GROUP BY AgeGroup
),

TotalRevenue AS (
 SELECT SUM(`Total Amount`) AS Total FROM retail
)
SELECT
 gr.AgeGroup,
 gr.Revenue,
 ROUND((gr.Revenue / tr.Total) * 100, 2) AS RevenuePercentage
FROM GroupRevenue gr, TotalRevenue tr;
```

```
155    SELECT *,
156      CASE
157        WHEN Age BETWEEN 18 AND 25 THEN '18-25'
158        WHEN Age BETWEEN 26 AND 35 THEN '26-35'
159        WHEN Age BETWEEN 36 AND 50 THEN '36-50'
160        WHEN Age BETWEEN 51 AND 65 THEN '51-65'
161        ELSE 'Other'
162      END AS AgeGroup
163    FROM retail
164  ),
165  GroupRevenue AS (
166    SELECT AgeGroup, SUM(`Total Amount`) AS Revenue
167    FROM AgeBracketed
168    GROUP BY AgeGroup
169  ),
170
171  TotalRevenue AS (
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

AgeGroup	Revenue	RevenuePercentage
26–35	98480	21.60
36–50	139660	30.63
51–65	133310	29.23
18–25	84550	18.54

Result 7

Output

This project provided valuable insights into customer behavior, product trends, and sales patterns through SQL-based analysis of a retail sales dataset. By performing targeted queries in MySQL Workbench, I uncovered several key findings:

Inactive Customers: A significant number of customers have not made purchases in the last 6 months, highlighting opportunities for re-engagement campaigns.

RFM Analysis: Segmenting customers based on recency, frequency, and monetary value helped identify high-value and loyal customers, as well as those at risk of churn.**Bulk Purchases**: Certain product categories showed high average quantities per transaction, suggesting they are commonly purchased in bulk—ideal for discount or bundle strategies

Sales Patterns: The busiest day of the week was identified, providing insight for resource planning and promotional timing.

Spending by Gender: Revenue and average transaction amounts were analyzed by gender, revealing subtle differences in purchasing behavior that can support targeted marketing.

Top Product Categories: The five most frequently purchased categories were determined, allowing the business to focus on high-demand inventory.

Revenue by Age Group: Younger and middle-aged customer segments contributed the largest share of total revenue, helping refine customer targeting and engagement strategies and many others.

Overall, this analysis supports data-driven decision-making in areas such as marketing, inventory management, and customer relationship management. Future work could incorporate time-series analysis or predictive models to forecast trends and personalize offers more effectively thank.