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

-- Order by transaction date to understand the chronological flow of

-- purchases.

SELECT *

FROM retail;

```
5     -- purchases.
6
7 •   SELECT *
8     FROM retail;
```

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

retail 1 ×

-- 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 `Customer ID`, `Quantity`, `Price Per Unit`, `Total Amount`

FROM retail;

```
9
10    -- Clean the dataset by ensuring that numeric fields like Quantity,
11    -- Price per Unit, and Total Amount are properly formatted.
12    -- Remove duplicates or null values if any exist.
13
14    SELECT DISTINCT `Customer ID`, Quantity, `Price Per Unit`, `Total amount`
15    FROM retail;
16
```

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

Customer ID	Quantity	Price Per Unit	Total amount
CUST001	3	50	150
CUST002	2	500	1000
CUST003	1	30	30
CUST004	1	500	500
CUST005	2	50	100
CUST006	1	30	30
CUST007	2	25	50
CUST008	4	25	100
CUST009	2	300	600

retail 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 Average_revenue

FROM retail

GROUP BY `Product Category`;

```
17      -- Calculate the total and average revenue for each product
18      -- category.
19      -- Which categories bring in the most and least revenue?
20
21  •   SELECT  `Product Category`,
22      SUM(`Total Amount`) AS Total_Revenue,
23      AVG(`Total Amount`) AS Average_Revenue
24      FROM retail
25      GROUP BY `Product Category`;
26
27
```

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

Product Category	Total_Revenue	Average_Revenue
Beauty	143515	467.4756
Clothing	155580	443.2479
Electronics	156905	458.7865

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

-- Summarize total revenue per month and order the results

-- chronologically.

SELECT `Date`, SUM(`Total Amount`) AS Monthly_trend

FROM retail;

```
24      FROM retail
25      GROUP BY `Product Category`;
26
27      -- Analyze the monthly sales trend over the entire dataset period.
28      -- Summarize total revenue per month and order the results
29      -- chronologically.
30
31  •   SELECT `Date`, SUM(`Total Amount`) AS Monthly_Trend
32      FROM retail
33      GROUP BY `Date`;
34
```

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

Date	Monthly_Trend
2023-11-24	1110
2023-02-27	1825
2023-01-13	1930
2023-05-21	2900
2023-05-06	990
2023-04-25	160

-- Identify the top 10 customers by total spending.

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

-- transactions.

SELECT `Customer ID`, `Total Amount` AS Total_spending

FROM retail

ORDER BY `Customer ID`

LIMIT 10;

```
32    -- Identify the top 10 customers by total spending.
33    -- Rank customers based on how much they've spent across all
34    -- transactions.
35
36 •  SELECT `Customer ID`, `Total Amount` AS Total_spending
37    FROM retail
38    ORDER BY `Customer ID`
39    LIMIT 10;
40
41
```

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

Customer ID	Total_spending
CUST006	30
CUST007	50
CUST008	100
CUST009	600
CUST010	200

-- Calculate the average transaction value for each customer.

-- How much does each customer spend per transaction on

-- average?

SELECT `Transaction ID`, AVG(`Total Amount`) AS Average_transaction

FROM retail

GROUP BY `Customer ID`;

```
41      -- Calculate the average transaction value for each customer.
42      -- How much does each customer spend per transaction on
43      -- average?
44
45      SELECT `Transaction ID`, AVG(`Total Amount`) AS Average_transaction
46      FROM retail
47      GROUP BY `Transaction ID`;
48
```

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

Transaction ID	Average_transaction
1	150.0000
2	1000.0000
3	30.0000
4	500.0000
5	100.0000

esult 17 ×

-- 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`, `Age`

FROM retail

where `Age` BETWEEN 26 AND 35;

```
48
49      -- Group customers by gender and age brackets (e.g., 18-25, 26-35,
50      -- 36-50, etc.).
51      -- Summarize total revenue and transaction count for each group.
52
53      SELECT `Gender`, `Age`
54      FROM retail
55      WHERE `Age` BETWEEN 26 AND 35;
56
57
```

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

Gender	Age
Male	34
Female	26
Male	30
Male	30
Male	35
Female	27

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

-- Group customers by purchase frequency to determine repeat

-- behavior.

SELECT `Customer ID`, COUNT(DISTINCT `Transaction ID`) AS Purchase_frequency

FROM retail

GROUP BY `Customer ID`;

```
55
56      -- Compare the number of one-time buyers versus repeat buyers.
57      -- Group customers by purchase frequency to determine repeat
58      -- behavior.
59
60      SELECT `Customer ID`, COUNT(DISTINCT `Transaction ID`) AS Purchase_frequency
61      FROM retail
62      GROUP BY `Customer ID`;
```

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

Customer ID	Purchase_frequency
CUST001	1
CUST002	1
CUST003	1
CUST004	1
CUST005	1

-- 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.

SELECT DISTINCT `Date` AS Busiest_day, SUM(`Total Amount`) AS Revenue

FROM retail

GROUP BY `Date`

ORDER BY SUM(`Total Amount`) DESC;

```
69        --  Identify inactive customers who have not made a purchase in the
70      -- last 6 months.
71      -- Use the most recent date in the dataset as the reference point.
72
73      SELECT  `Customer ID`, `Transaction ID`,`Date`
74      FROM retail
75      WHERE `Date` BETWEEN'2023-06-01' AND '2024-01-01' AND `Transaction ID` IS NULL;
76
77      -- Perform RFM (Recency, Frequency, Monetary) analysis for
78      -- customer segmentation.
```

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

Customer ID	Transaction ID	Date

-- Perform RFM (Recency, Frequency, Monetary) analysis for

-- customer segmentation.

-- Recency: Days since last purchase; Frequency: Number of

-- purchases; Monetary: Total amount spent.

SELECT `Customer ID`, COUNT(`Date`) AS Recency, COUNT(`Transaction ID`) AS Frequency,

SUM(`Total Amount`) AS Monetary

FROM retail

GROUP BY `Customer ID`;

```
77      -- Perform RFM (Recency, Frequency, Monetary) analysis for
78      -- customer segmentation.
79      -- Recency: Days since last purchase; Frequency: Number of
80      -- purchases; Monetary: Total amount spent.
81
82  •   SELECT `Customer ID`, COUNT(`Date`) AS Recency, COUNT(`Transaction ID`) AS Frequeny,
83       SUM(`Total Amount`) AS Monetary_spent
84      FROM retail
85      GROUP BY `Customer ID`;
86
87      -- Find the product categories with the highest average quantity per
```

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

Customer ID	Recency	Frequeny	Monetary_spent
CUST001	1	1	150
CUST002	1	1	1000
CUST003	1	1	30
CUST004	1	1	500
CUST005	1	1	100

-- Find the product categories with the highest average quantity per

-- transaction.

-- Which product types are purchased in bulk?

SELECT `Transaction ID`, AVG(`Quantity`) AS Highest_average

FROM retail

GROUP BY `Transaction ID`

ORDER BY AVG(`Quantity`);

```
86
87      -- Find the product categories with the highest average quantity per
88      -- transaction.
89      -- Which product types are purchased in bulk?
90
91 •    SELECT `Transaction ID`, AVG(`Quantity`) AS Highest_average
92       FROM retail
93       GROUP BY `Transaction ID`
94       ORDER BY AVG(`Quantity`) DESC;
95
```

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

Transaction ID	Highest_average
8	4.0000
10	4.0000
14	4.0000
15	4.0000
17	4.0000

esult 21 ×

-- Identify the busiest sales day of the week.

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

-- revenue?

SELECT DISTINCT `Date` AS Busiest_Day, SUM(`Total Amount`) AS Revenue

FROM retail

GROUP BY `Date`

ORDER BY SUM(`Total Amount`) DESC;

```
95
96      -- Identify the busiest sales day of the week.
97     -- Which day(s) consistently have the highest transaction volume or
98     -- revenue?
99
100    SELECT DISTINCT`Date` AS Busiest_day, SUM(`Total Amount`) AS Revenue
101    FROM retail
102    GROUP BY `Date`
103    ORDER BY SUM(`Total Amount`) DESC;
104
```

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

Busiest_day	Revenue
2023-05-23	8455
2023-05-16	7260
2023-06-24	6220
2023-02-17	5890
2023-08-05	5205
2023-07-14	5125

-- Calculate total revenue and average spend per transaction by

-- gender.

-- Are there differences in spending patterns across genders?

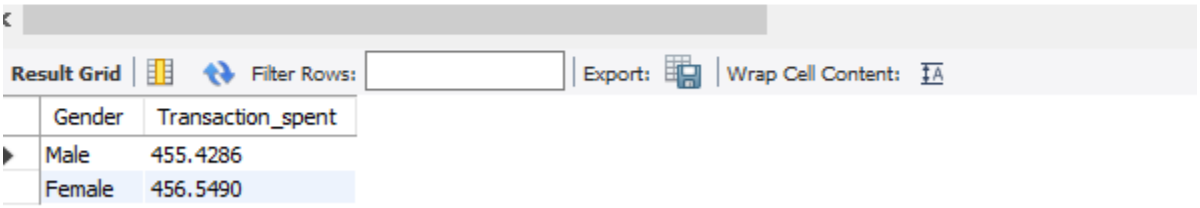
SELECT DISTINCT `Gender`, AVG(`Total Amount`) AS Transaction_spent

FROM retail

GROUP BY `Gender`;

```sql
-- Calculate total revenue and average spend per transaction by
-- gender.
-- Are there differences in spending patterns across genders?

SELECT DISTINCT `Gender`, AVG(`Total Amount`) AS Transaction_spent
FROM retail
GROUP BY `Gender`;
```



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

Gender	Transaction_spent
Male	455.4286
Female	456.5490

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

-- Based on number of transactions involving each category.

```sql
SELECT DISTINCT `Product Category` AS Frequently_purchased, SUM(`Total Amount`) AS Revenue
FROM retail
GROUP BY `Product Category`
ORDER BY `Product Category` DESC
LIMIT 5;
```

```
-- Find the top 5 most frequently purchased product categories.
-- Based on number of transactions involving each category.

SELECT DISTINCT `Product Category` AS Frequently_purchased, SUM(`Total Amount`) AS Revenue
FROM retail
GROUP BY `Product Category`
ORDER BY `Product Category` DESC
```

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

Frequently_purchased	Revenue
Electronics	156905
Clothing	155580
Beauty	143515

```
-- Determine the percentage of total revenue contributed by each
-- age group.
-- Which customer age brackets are most valuable to the business?

WITH AgeGroups AS (
    SELECT
        CASE
            WHEN age < 20 THEN 'Under 20'
            WHEN age BETWEEN 20 AND 35 THEN 'Youth'
            WHEN age BETWEEN 36 AND 50 THEN 'Adults'
            ELSE 'Elderly'
        END AS age_group,
        `Total Amount`
    FROM retail
)
SELECT
    age_group,
    SUM(`Total Amount`) AS total_revenue
```

FROM AgeGroups

GROUP BY age_group

ORDER BY total_revenue DESC;

```
17
18      -- Determine the percentage of total revenue contributed by each
19      -- age group.
20      -- Which customer age brackets are most valuable to the business?
21
22  •   WITH AgeGroups AS (
23          SELECT
24              CASE
25                  WHEN age < 20 THEN 'Under 20'
26                  WHEN age BETWEEN 20 AND 35 THEN 'Youth'
27                  WHEN age BETWEEN 36 AND 50 THEN 'Adults'
28                  ELSE 'Elderly'
```

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

age_group	total_revenue
Youth	156945
Adults	139660
Elderly	133310
Under 20	26085