PREPARED BY ANURAG SHARMA

RETAIL SALES SQL PROJECT

Enhancing Product Sales Strategy

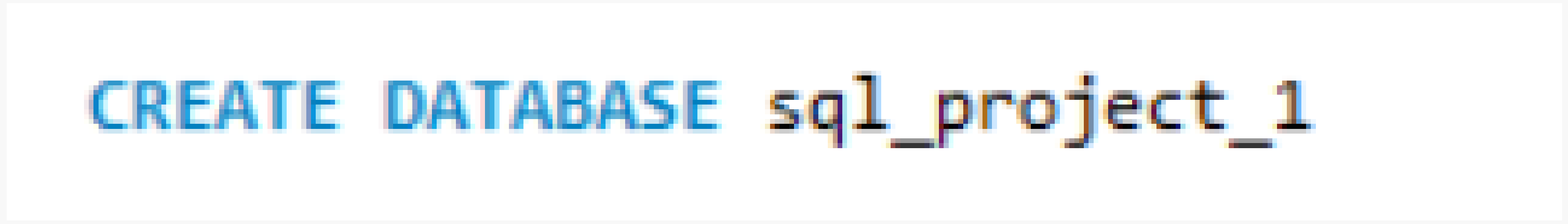
Project Overview

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

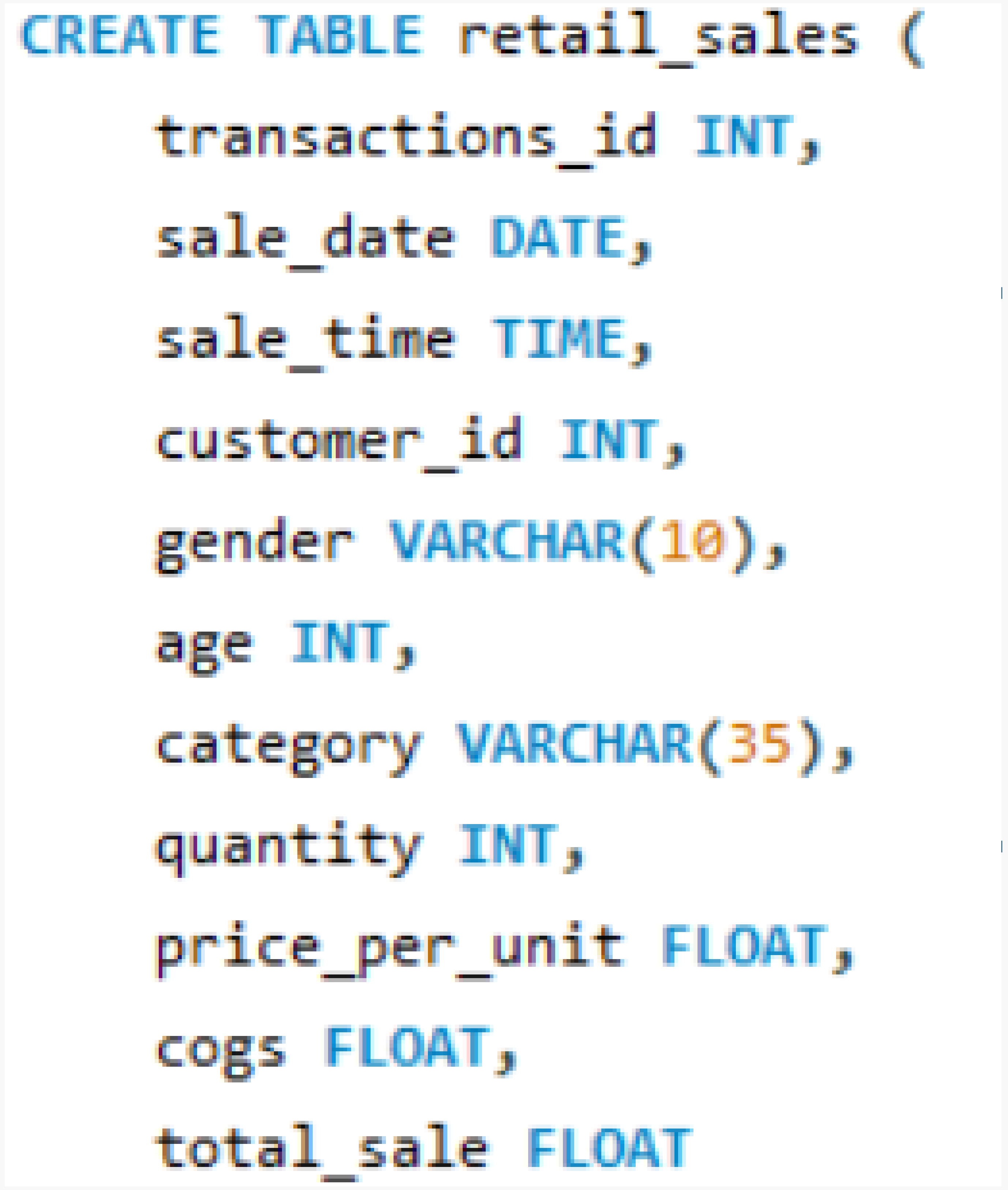
1. Set up a retail sales database: Create and populate a retail sales database with the provided sales data.
2. Data Cleaning: Identify and remove any records with missing or null values.
3. Exploratory Data Analysis (EDA): Perform basic exploratory data analysis to understand the dataset.
4. Business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

Database Setup



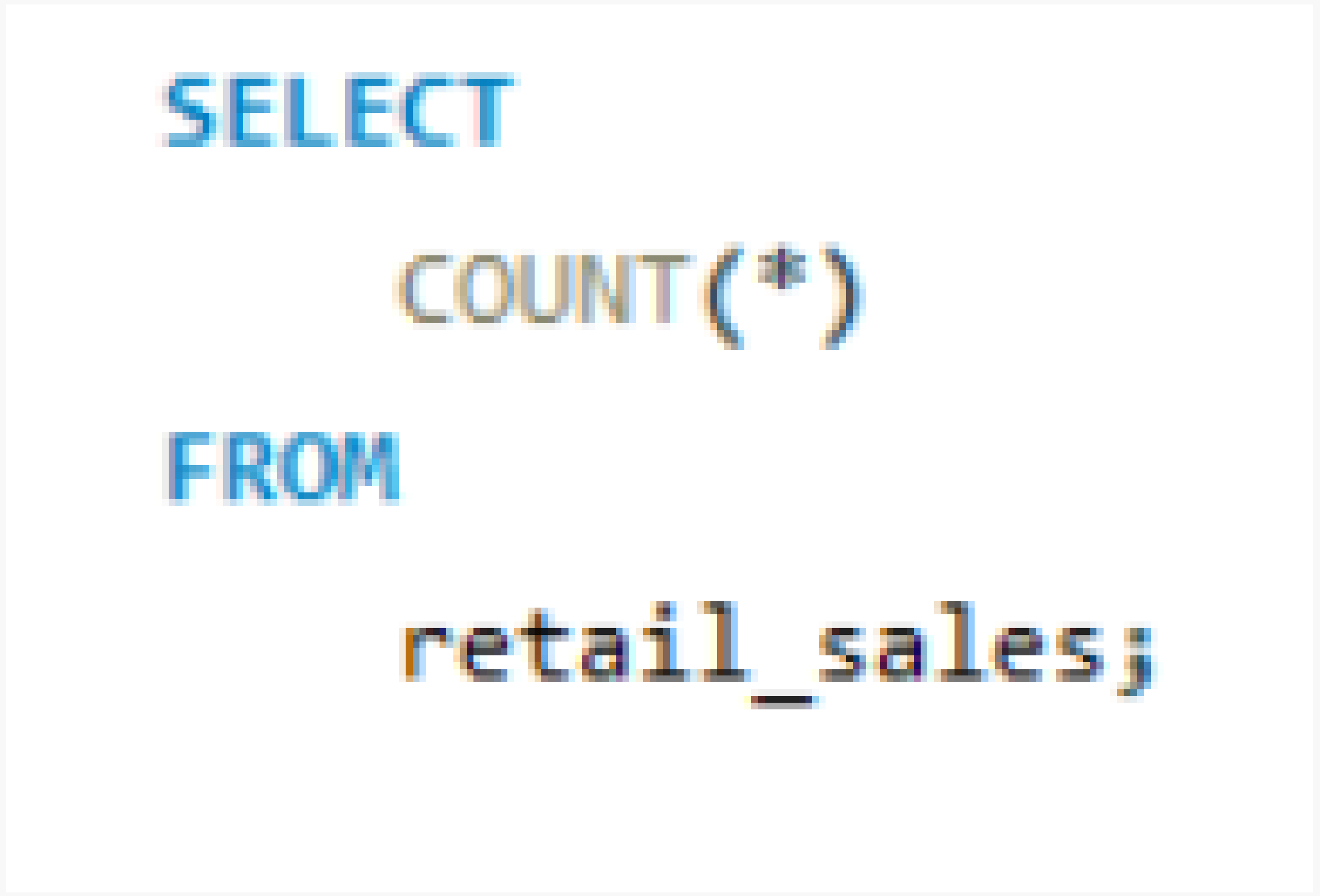
```sql
CREATE DATABASE sql_project_1
```

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.

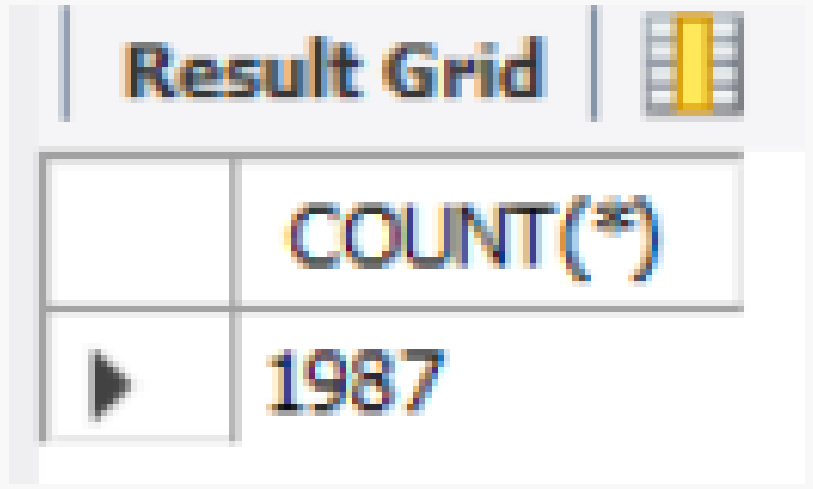


```
CREATE TABLE retail_sales (
    transactions_id INT,
    sale_date DATE,
    sale_time TIME,
    customer_id INT,
    gender VARCHAR(10),
    age INT,
    category VARCHAR(35),
    quantity INT,
    price_per_unit FLOAT,
    cogs FLOAT,
    total_sale FLOAT
```

Record Count: Determine the total number of records in the dataset.

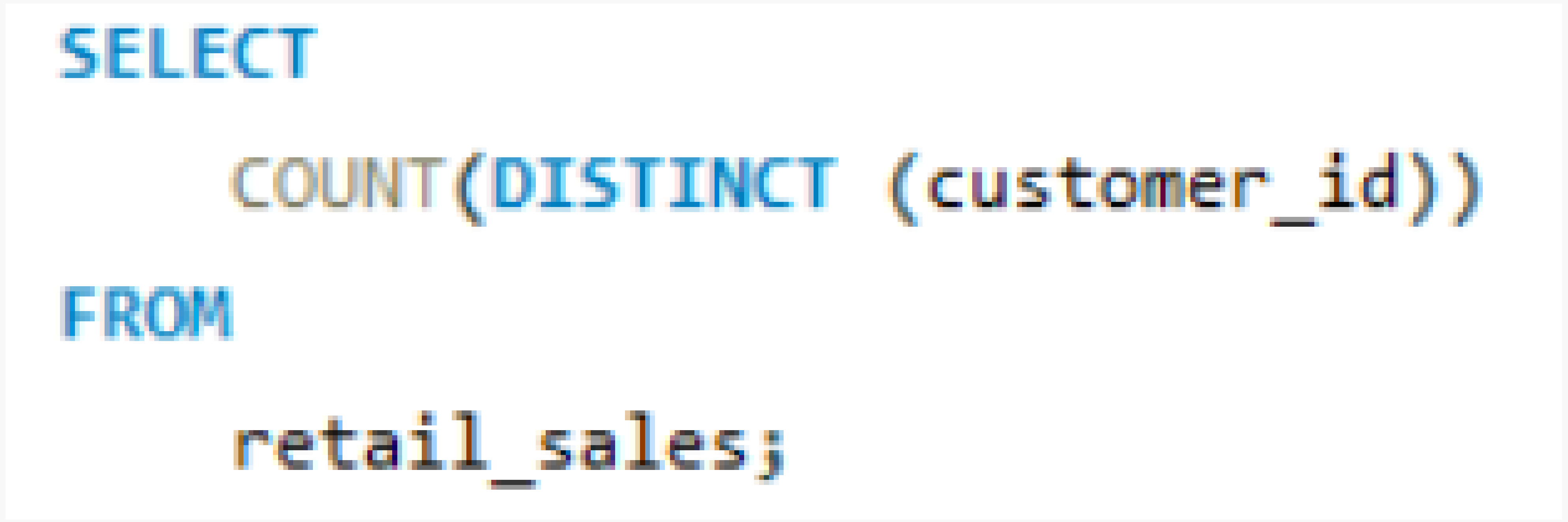


```
SELECT
    COUNT(*)
FROM
    retail_sales;
```

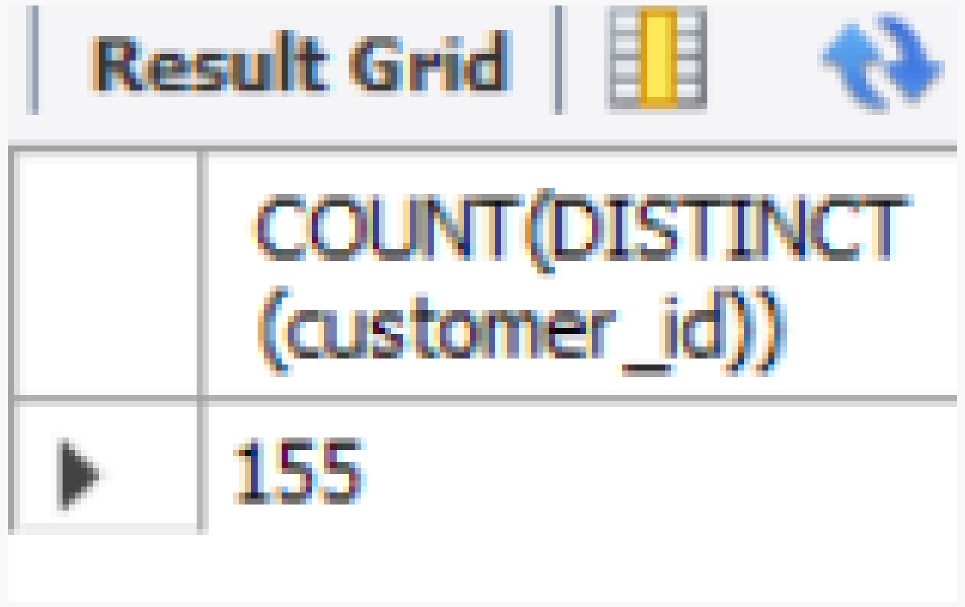


```
COUNT(*)
1987
```

Customer Count: Find out how many unique customers are in the dataset.

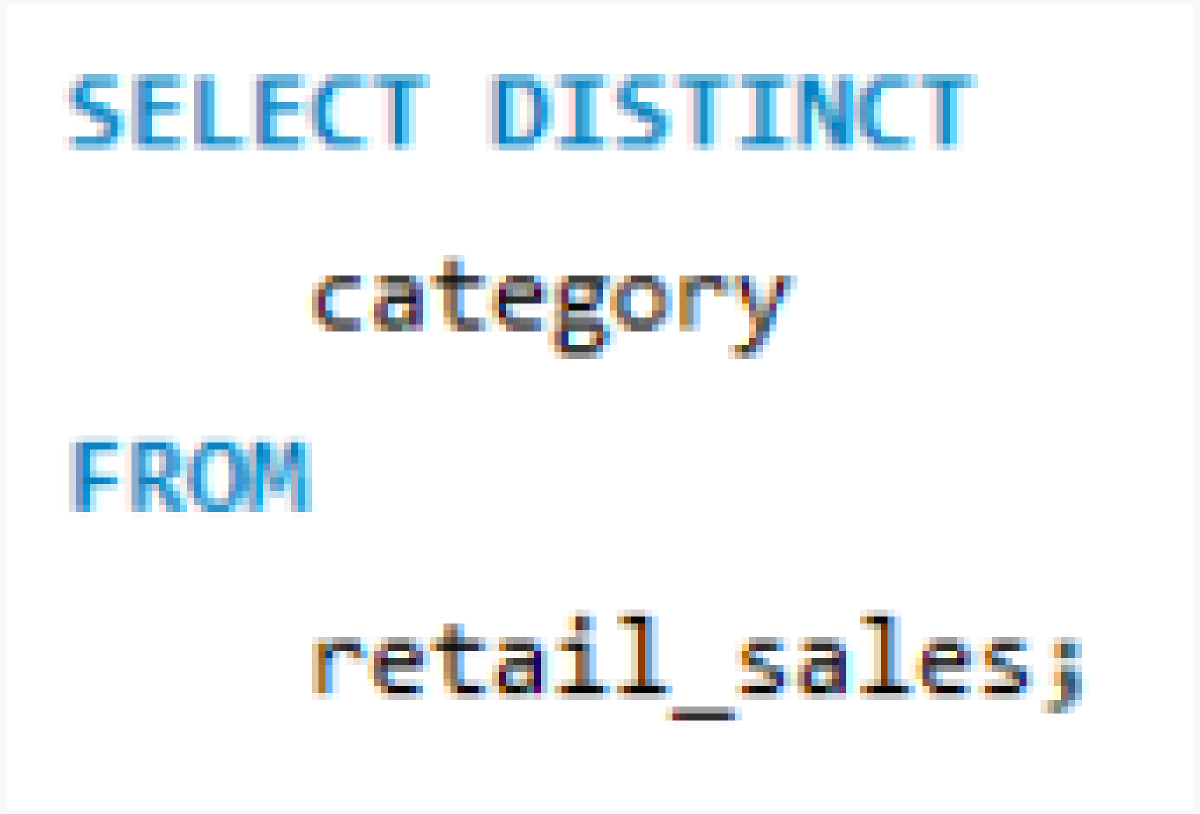


```
SELECT
    COUNT(DISTINCT (customer_id))
FROM
    retail_sales;
```

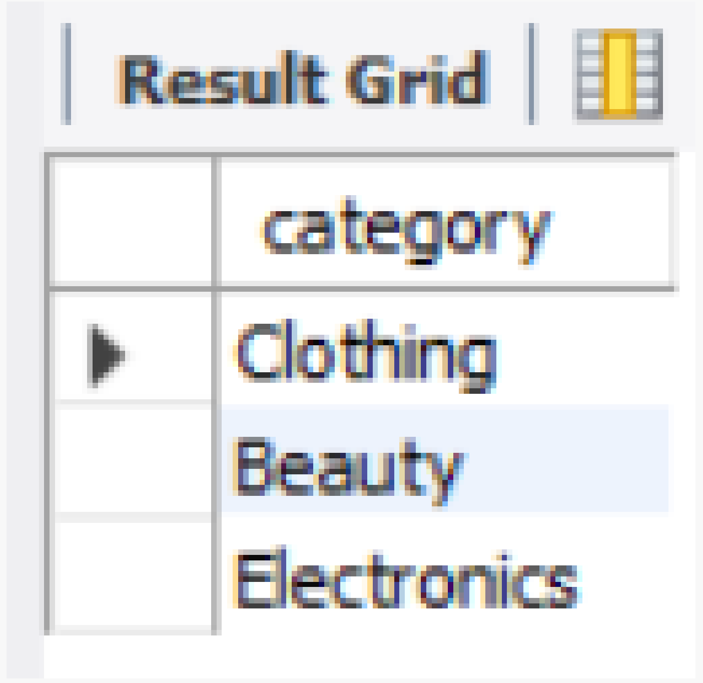


```
COUNT(DISTINCT
(customer_id))
155
```

Category Count: Identify all unique product categories in the dataset.



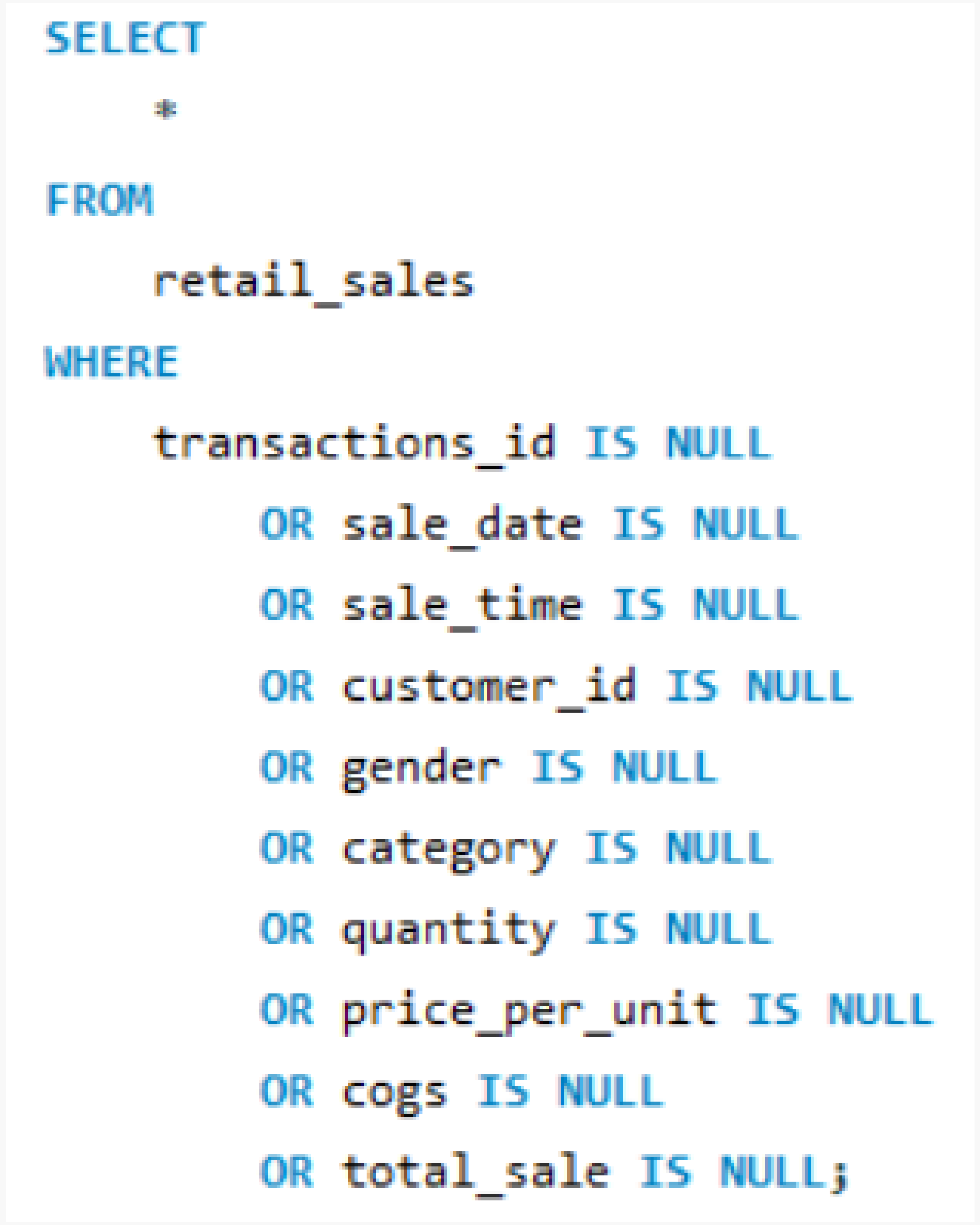
```
SELECT DISTINCT
    category
FROM
    retail_sales;
```



Result Grid

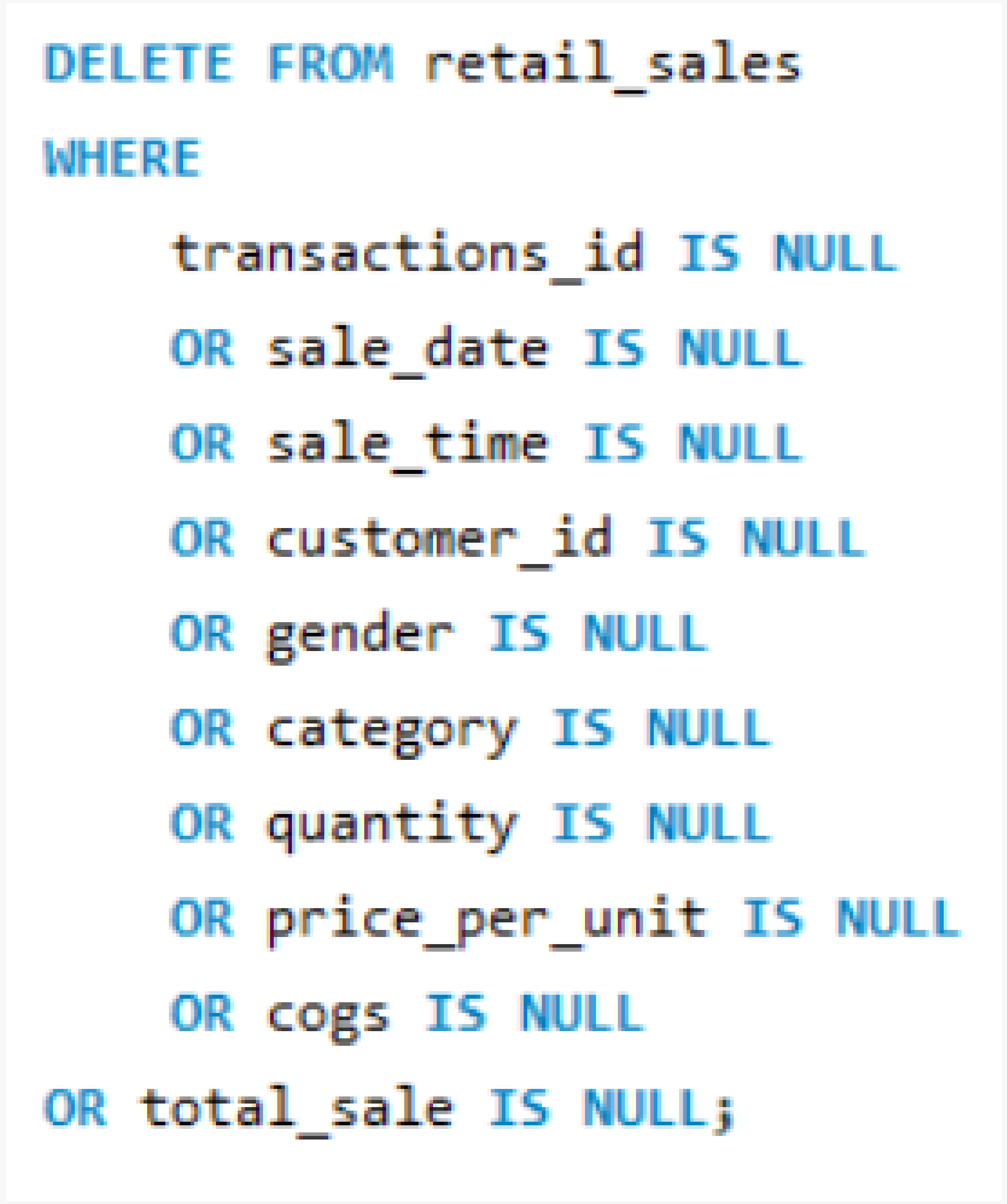
	category
▶	Clothing
	Beauty
	Electronics

Null Value Check: Check for any null values in the dataset and delete records with missing data



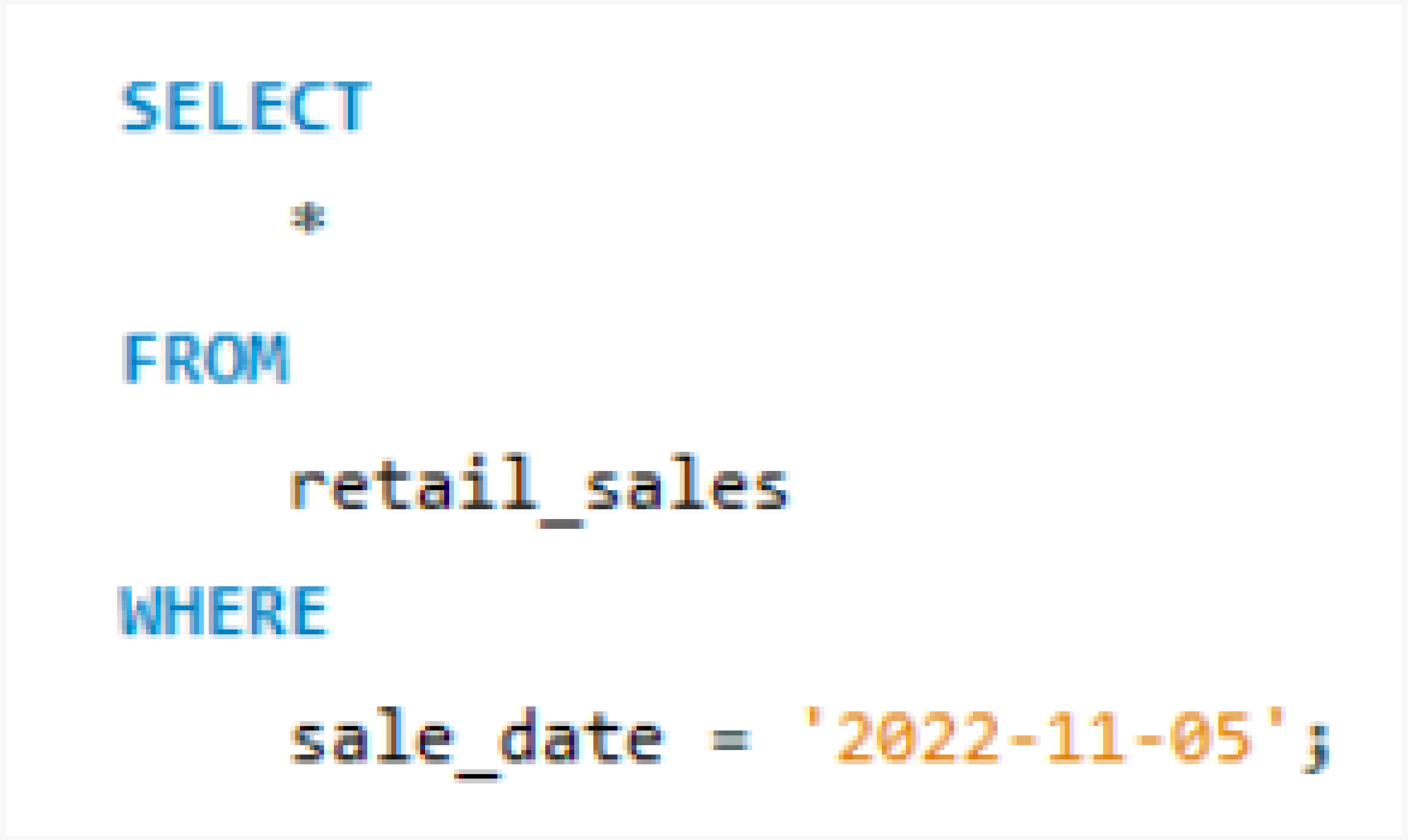
```
SELECT
    *
FROM
    retail_sales
WHERE
    transactions_id IS NULL
        OR sale_date IS NULL
        OR sale_time IS NULL
        OR customer_id IS NULL
        OR gender IS NULL
        OR category IS NULL
        OR quantity IS NULL
        OR price_per_unit IS NULL
        OR cogs IS NULL
        OR total_sale IS NULL;
```

DELETING NULL VALUES

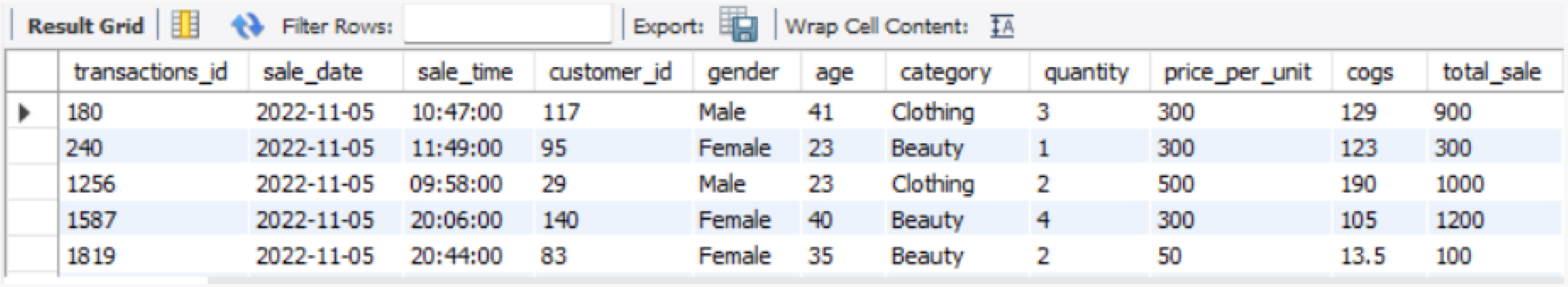


```
DELETE FROM retail_sales
WHERE
    transactions_id IS NULL
    OR sale_date IS NULL
    OR sale_time IS NULL
    OR customer_id IS NULL
    OR gender IS NULL
    OR category IS NULL
    OR quantity IS NULL
    OR price_per_unit IS NULL
    OR cogs IS NULL
OR total_sale IS NULL;
```

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

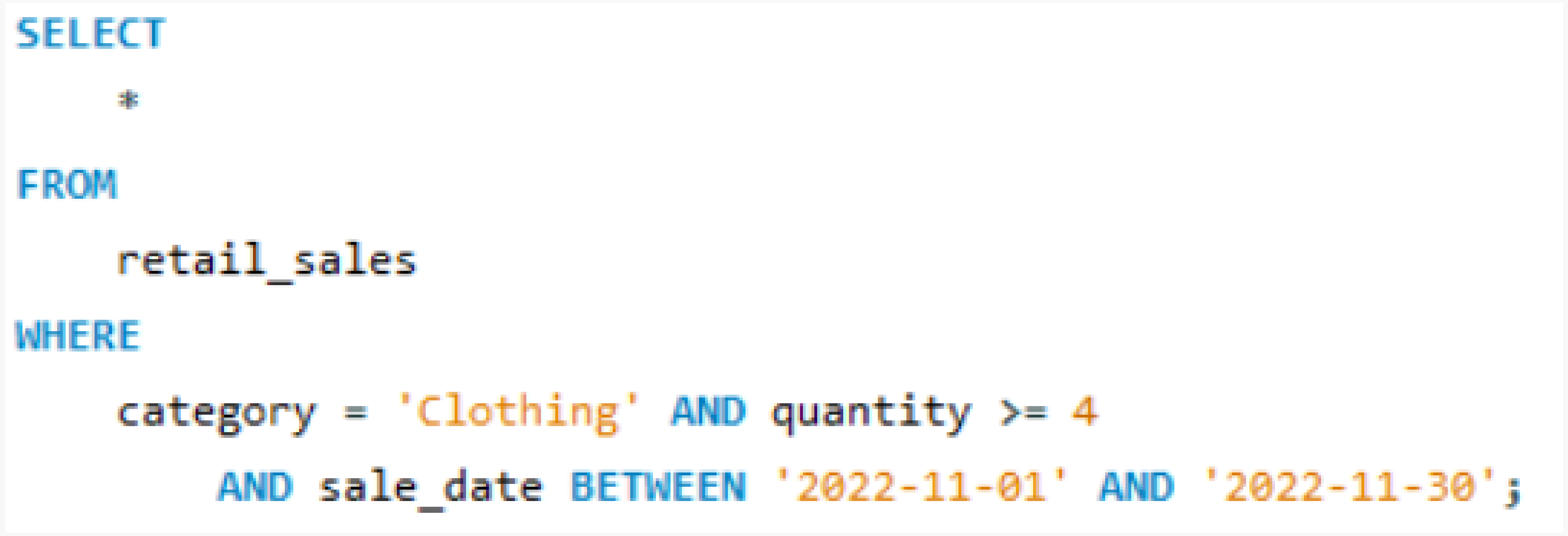


```
SELECT
    *
FROM
    retail_sales
WHERE
    sale_date = '2022-11-05';
```

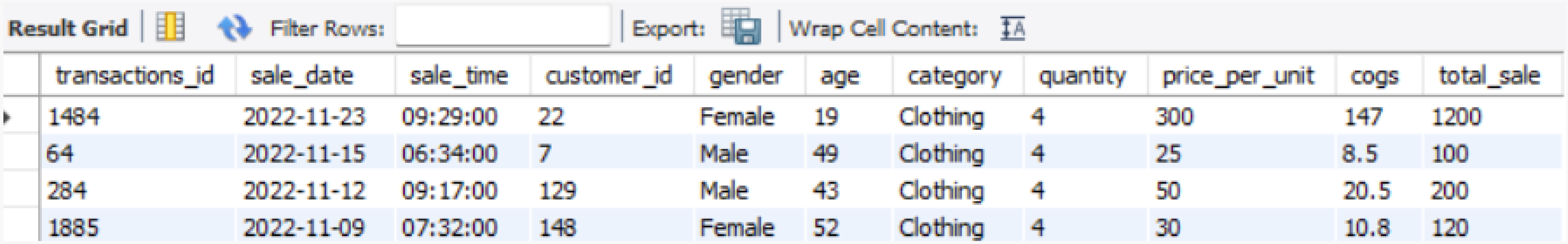


transactions_id	sale_date	sale_time	customer_id	gender	age	category	quantity	price_per_unit	cogs	total_sale
180	2022-11-05	10:47:00	117	Male	41	Clothing	3	300	129	900
240	2022-11-05	11:49:00	95	Female	23	Beauty	1	300	123	300
1256	2022-11-05	09:58:00	29	Male	23	Clothing	2	500	190	1000
1587	2022-11-05	20:06:00	140	Female	40	Beauty	4	300	105	1200
1819	2022-11-05	20:44:00	83	Female	35	Beauty	2	50	13.5	100

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



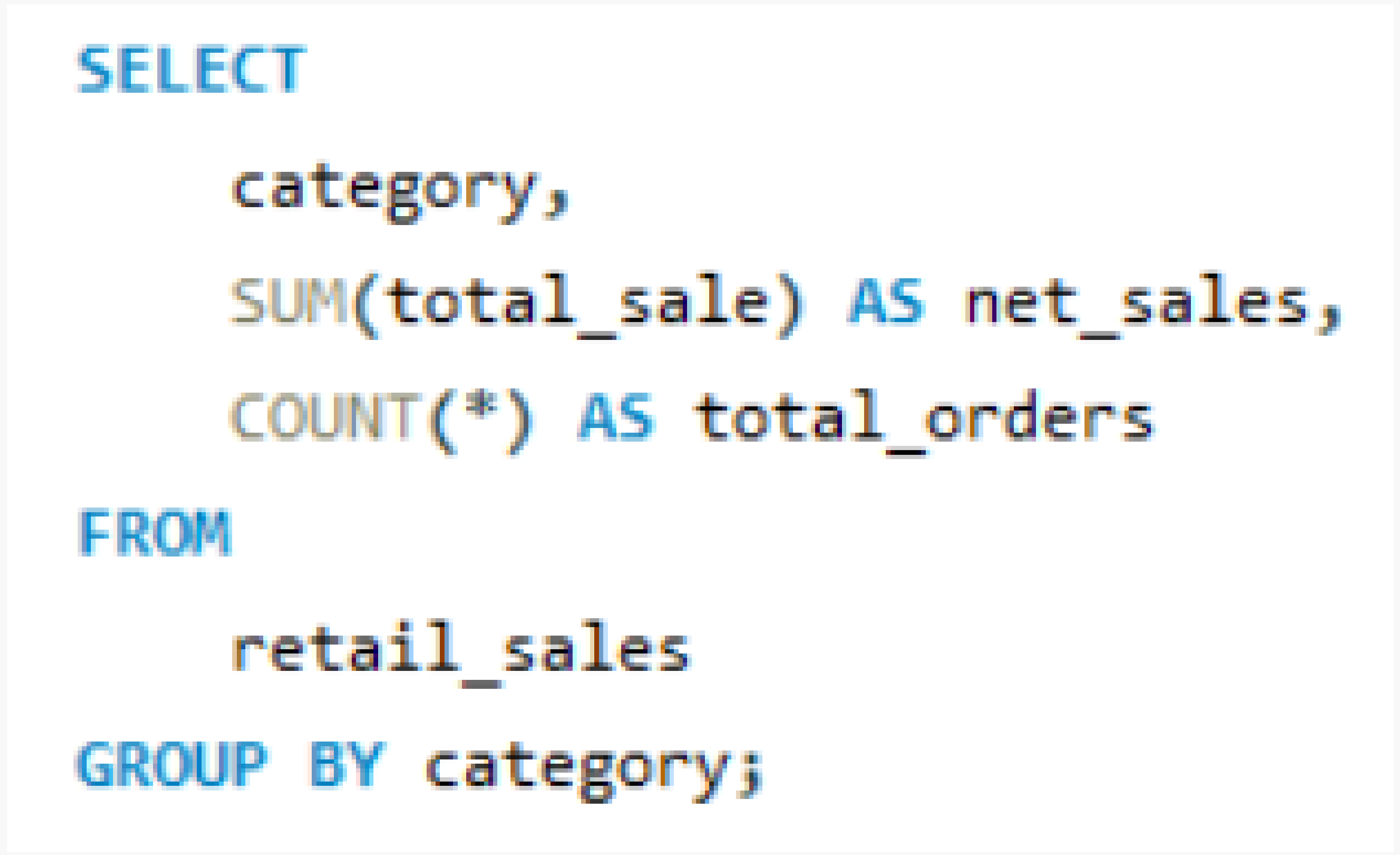
```
SELECT
    *
FROM
    retail_sales
WHERE
    category = 'Clothing' AND quantity >= 4
        AND sale_date BETWEEN '2022-11-01' AND '2022-11-30';
```



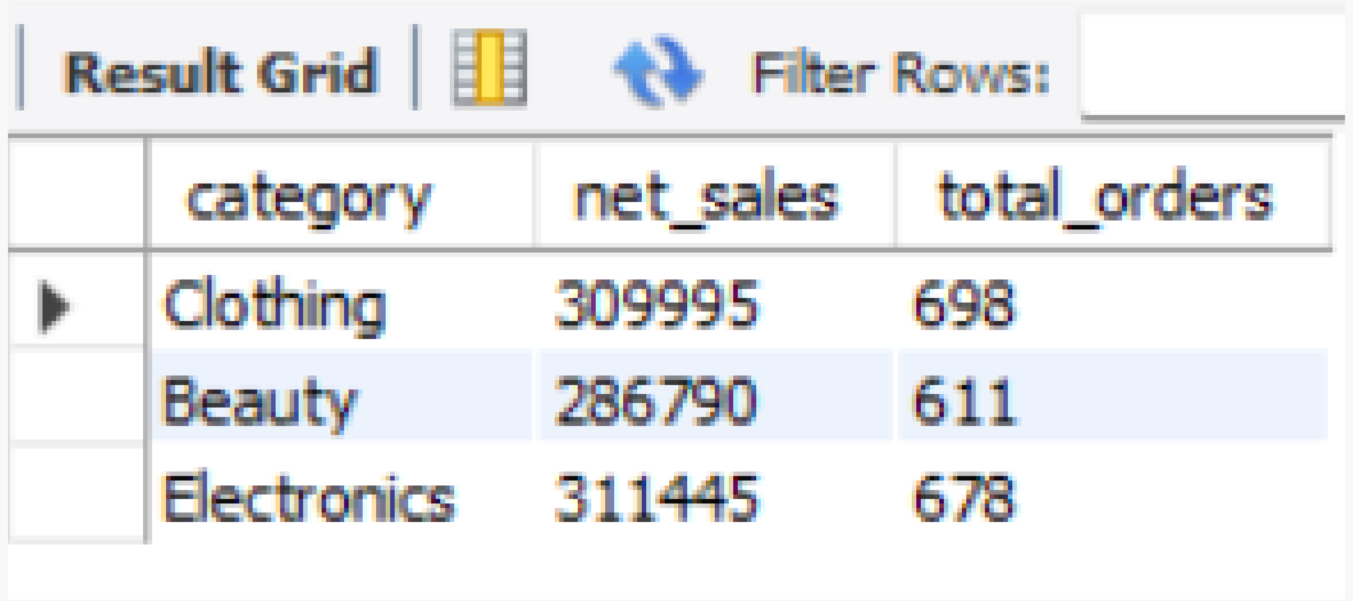
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

transactions_id	sale_date	sale_time	customer_id	gender	age	category	quantity	price_per_unit	cogs	total_sale
1484	2022-11-23	09:29:00	22	Female	19	Clothing	4	300	147	1200
64	2022-11-15	06:34:00	7	Male	49	Clothing	4	25	8.5	100
284	2022-11-12	09:17:00	129	Male	43	Clothing	4	50	20.5	200
1885	2022-11-09	07:32:00	148	Female	52	Clothing	4	30	10.8	120

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

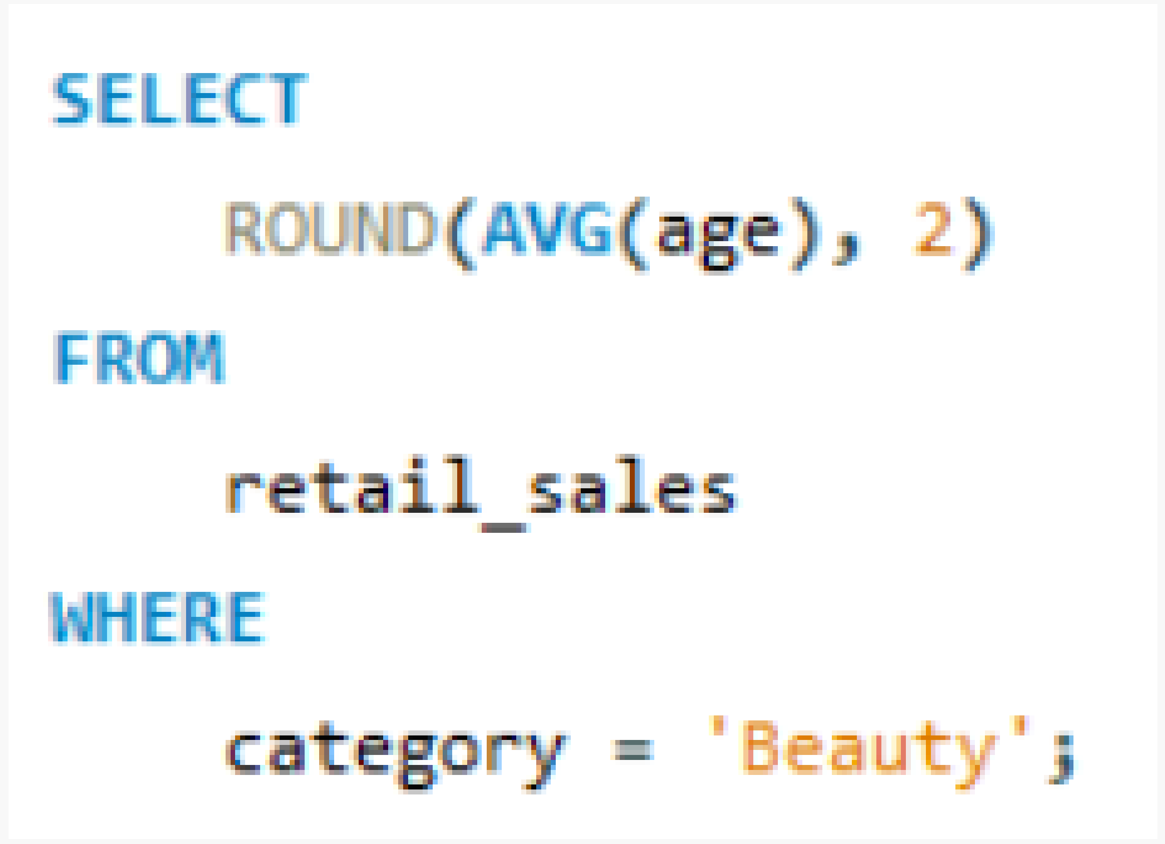


```sql
SELECT
    category,
    SUM(total_sale) AS net_sales,
    COUNT(*) AS total_orders
FROM
    retail_sales
GROUP BY category;
```

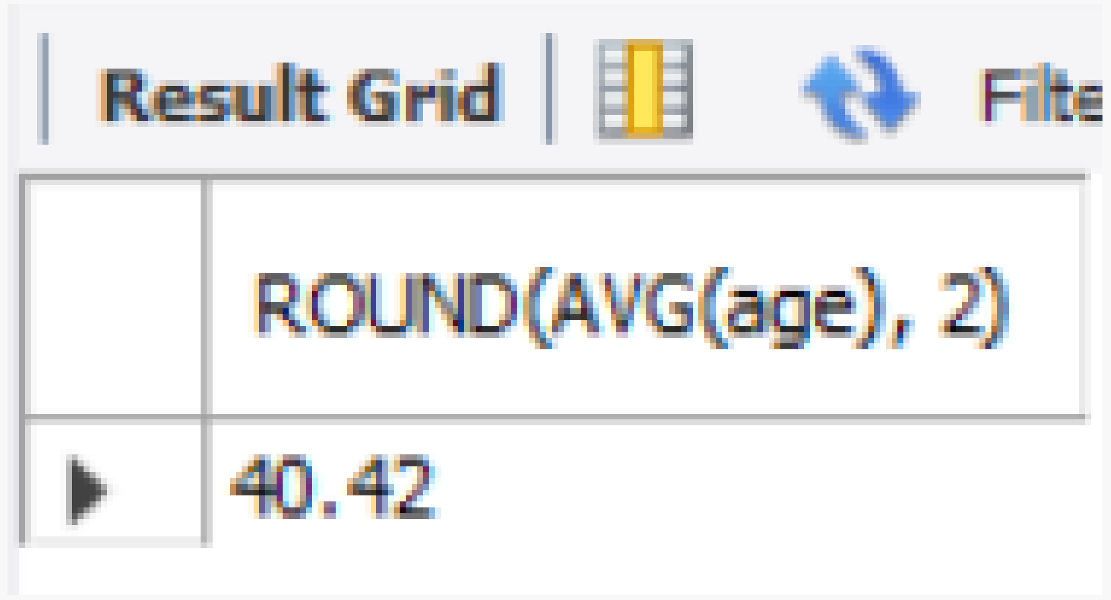


category	net_sales	total_orders
Clothing	309995	698
Beauty	286790	611
Electronics	311445	678

Write a SQL query to find the average age of customers who purchased items from the 'Beauty' category.



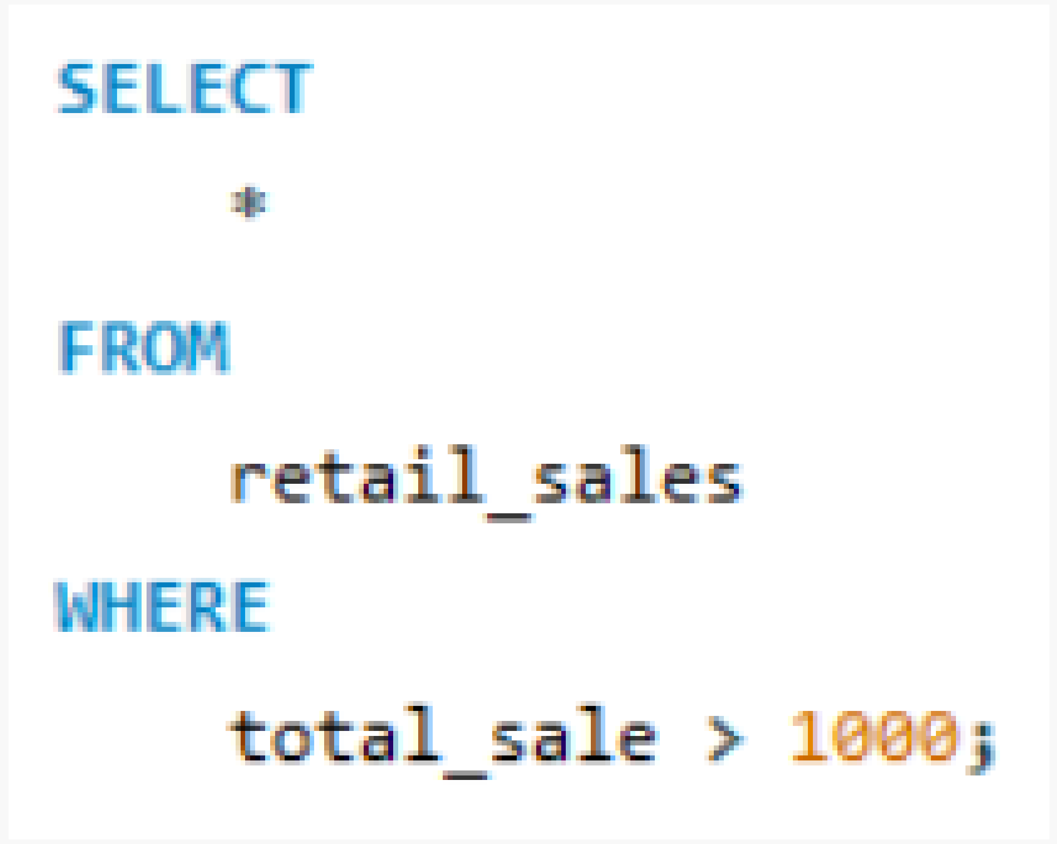
```
SELECT
    ROUND(AVG(age), 2)
FROM
    retail_sales
WHERE
    category = 'Beauty';
```



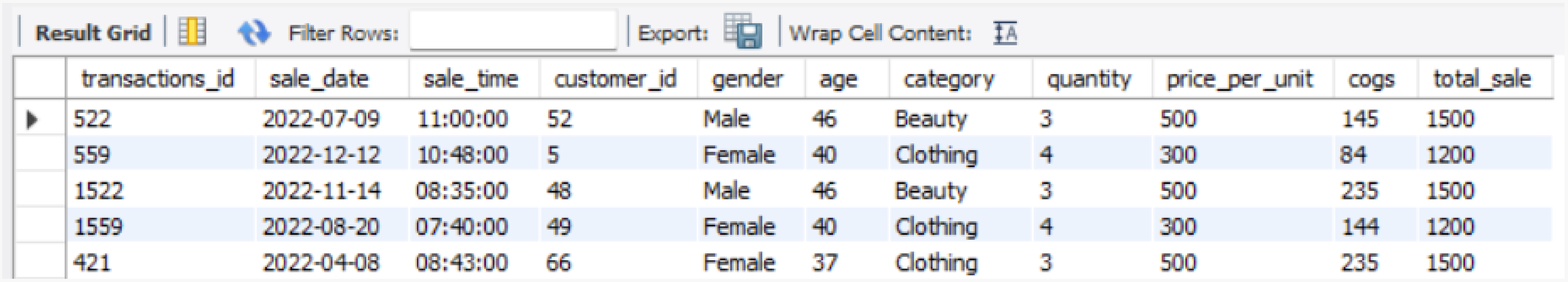
Result Grid | Filte

```
ROUND(AVG(age), 2)
40.42
```

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



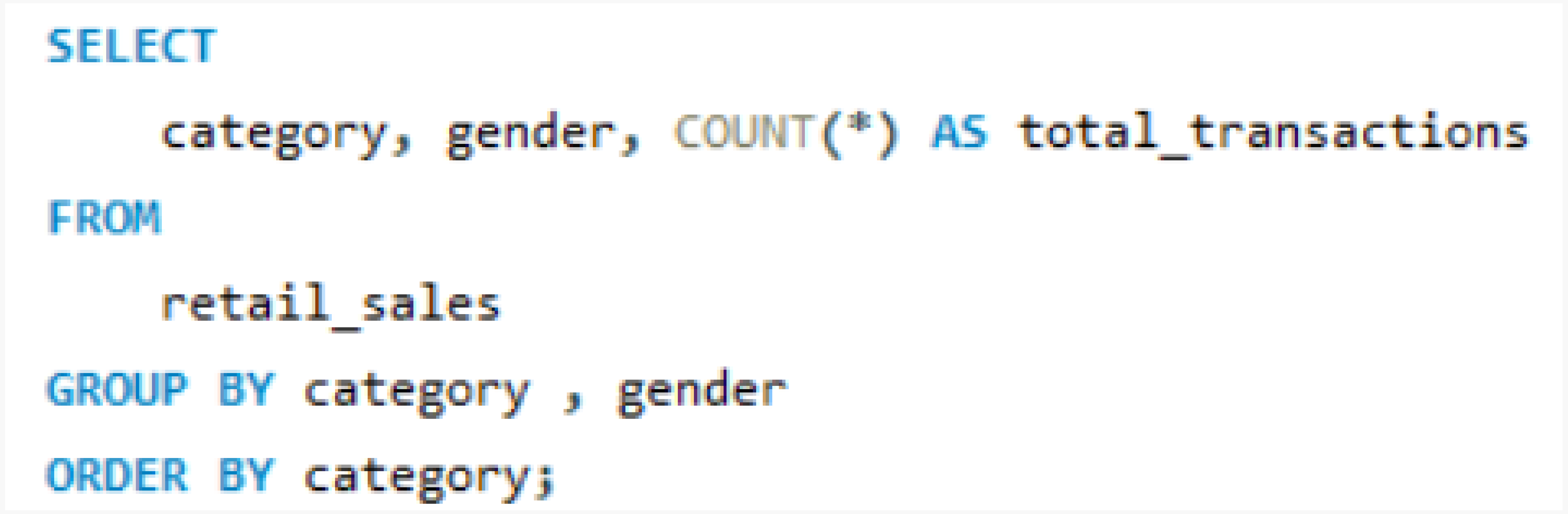
```
SELECT
    *
FROM
    retail_sales
WHERE
    total_sale > 1000;
```



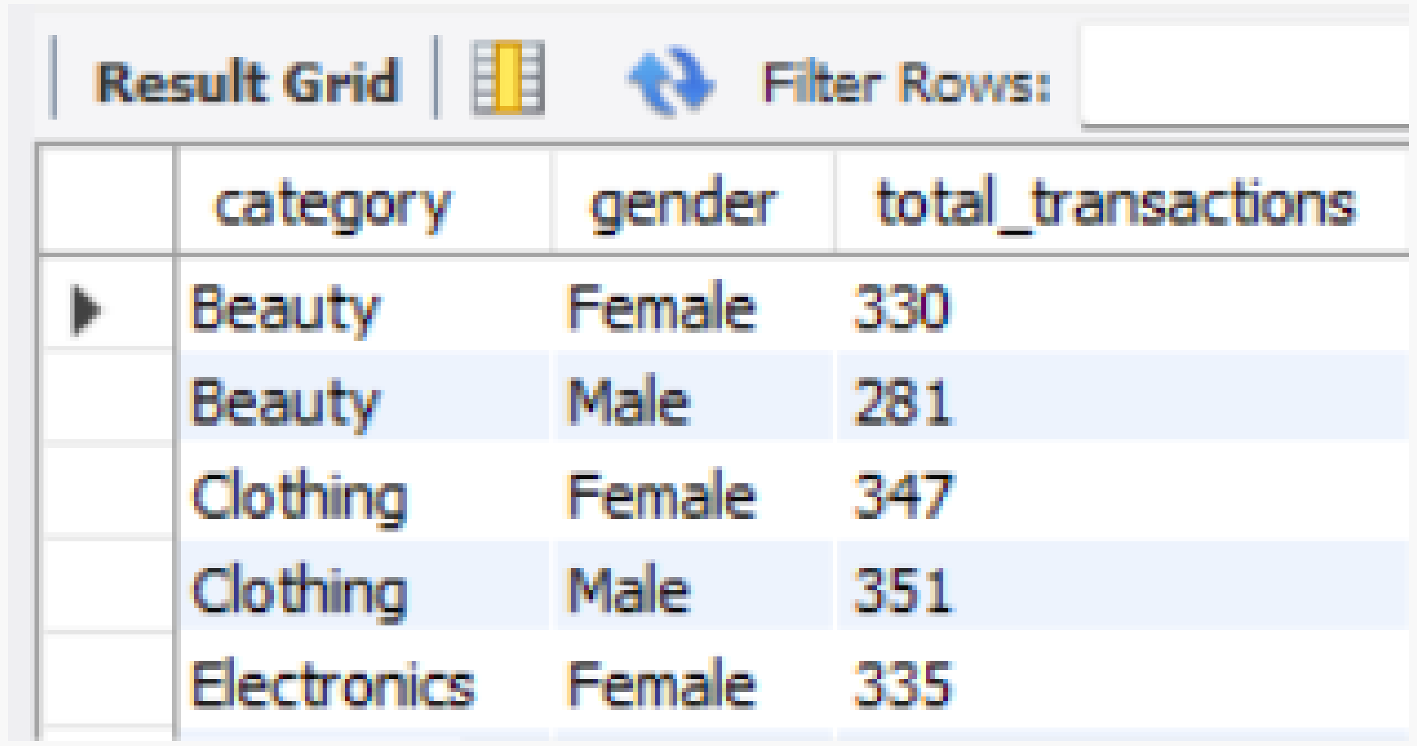
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

transactions_id	sale_date	sale_time	customer_id	gender	age	category	quantity	price_per_unit	cogs	total_sale
522	2022-07-09	11:00:00	52	Male	46	Beauty	3	500	145	1500
559	2022-12-12	10:48:00	5	Female	40	Clothing	4	300	84	1200
1522	2022-11-14	08:35:00	48	Male	46	Beauty	3	500	235	1500
1559	2022-08-20	07:40:00	49	Female	40	Clothing	4	300	144	1200
421	2022-04-08	08:43:00	66	Female	37	Clothing	3	500	235	1500

Write a SQL query to find the total number of transactions (transaction_id) made by each gender in each category.



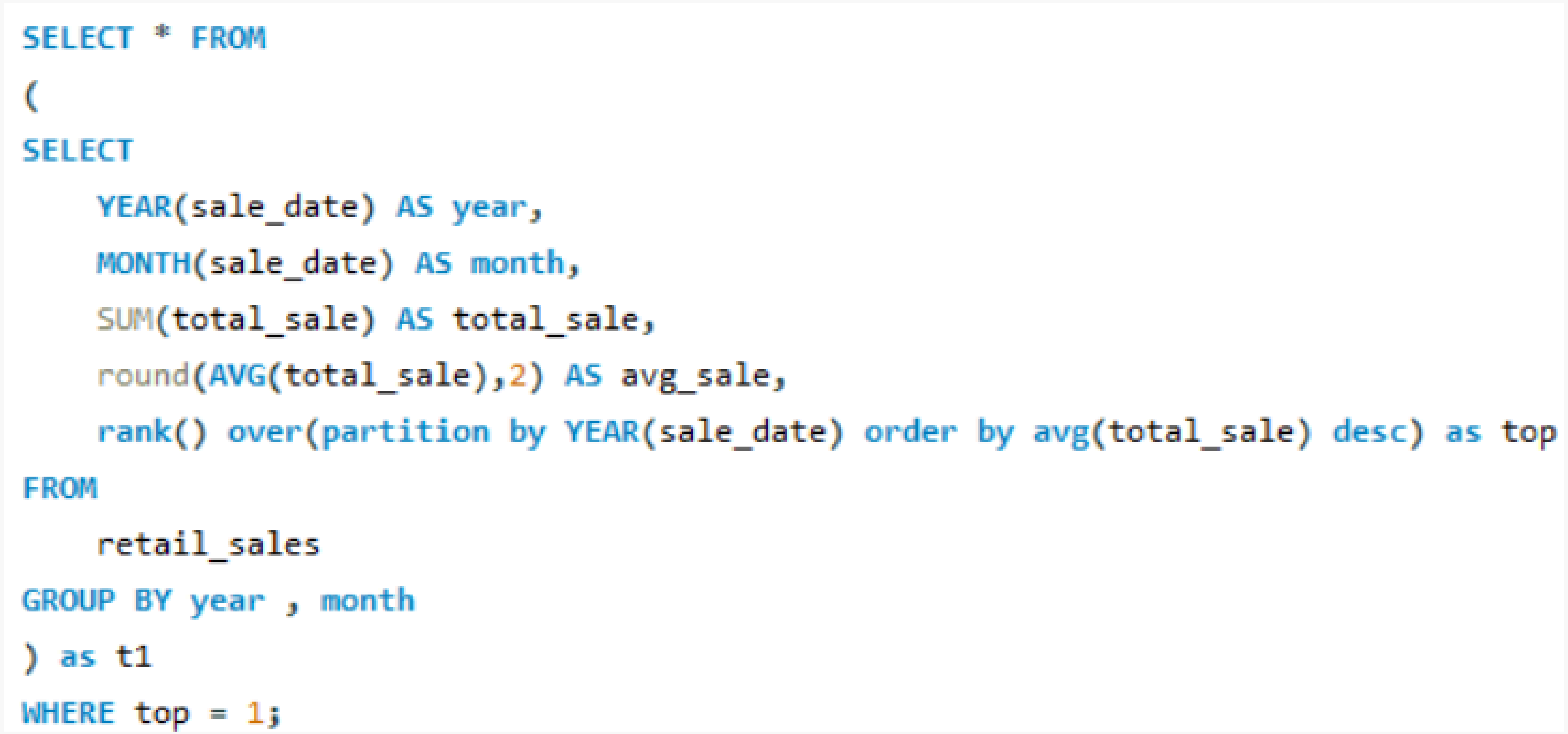
```
SELECT
    category, gender, COUNT(*) AS total_transactions
FROM
    retail_sales
GROUP BY category , gender
ORDER BY category;
```



Result Grid | Filter Rows:

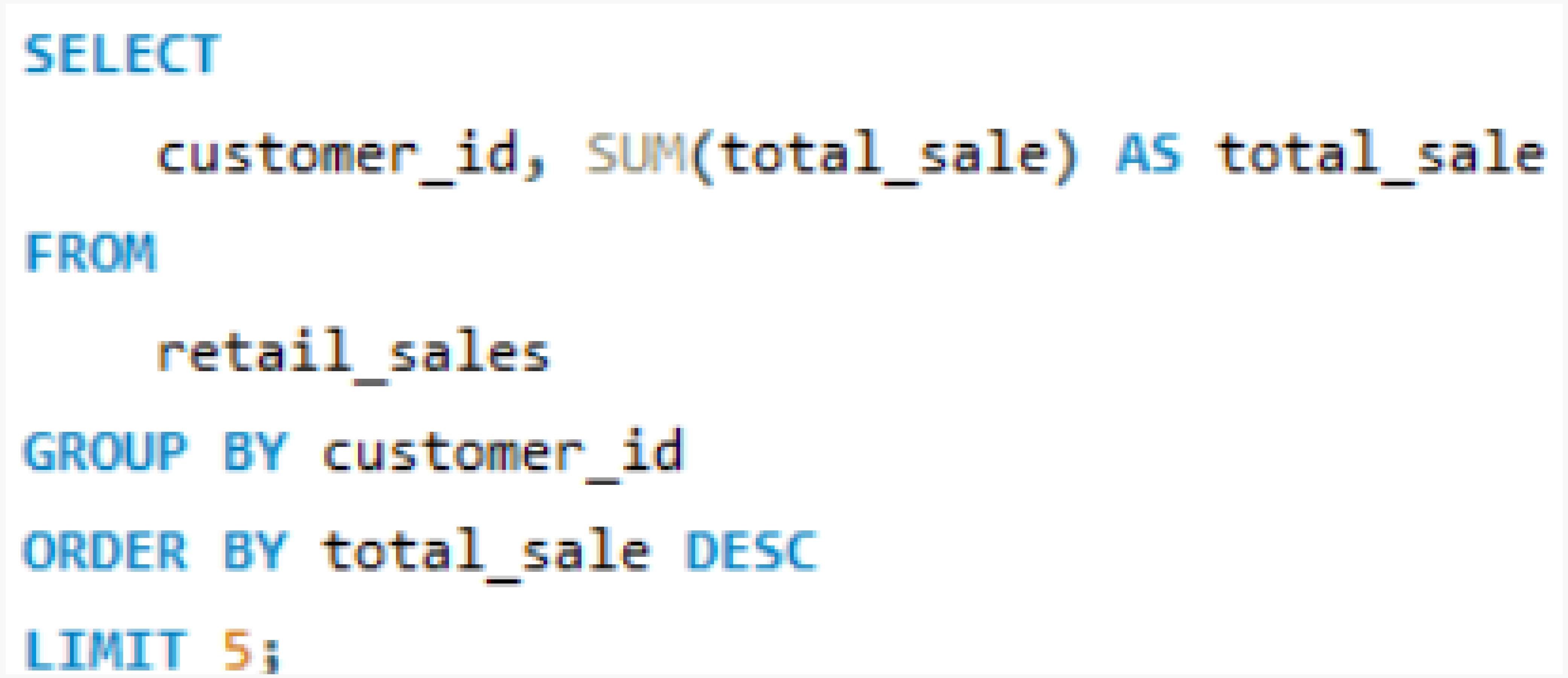
```
category      gender   total_transactions
Beauty        Female   330
Beauty        Male     281
Clothing      Female   347
Clothing      Male     351
Electronics   Female   335
```

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

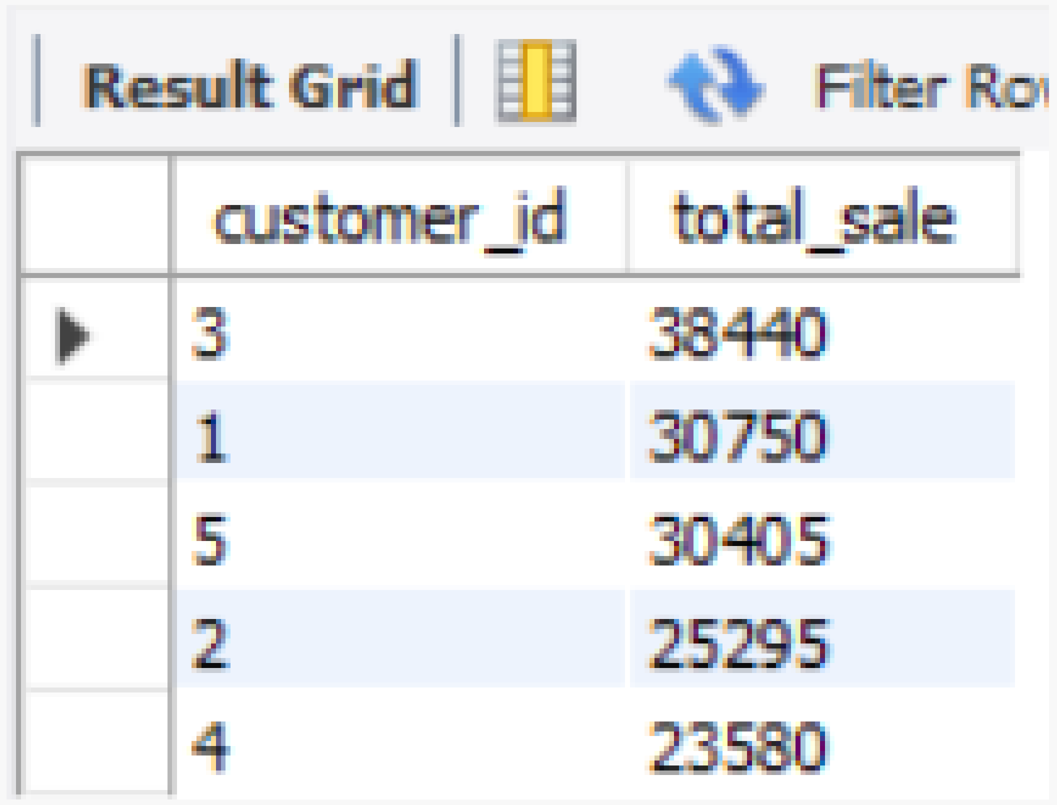


```
SELECT * FROM
(
SELECT
    YEAR(sale_date) AS year,
    MONTH(sale_date) AS month,
    SUM(total_sale) AS total_sale,
    round(AVG(total_sale),2) AS avg_sale,
    rank() over(partition by YEAR(sale_date) order by avg(total_sale) desc) as top
FROM
    retail_sales
GROUP BY year , month
) as t1
WHERE top = 1;
```

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

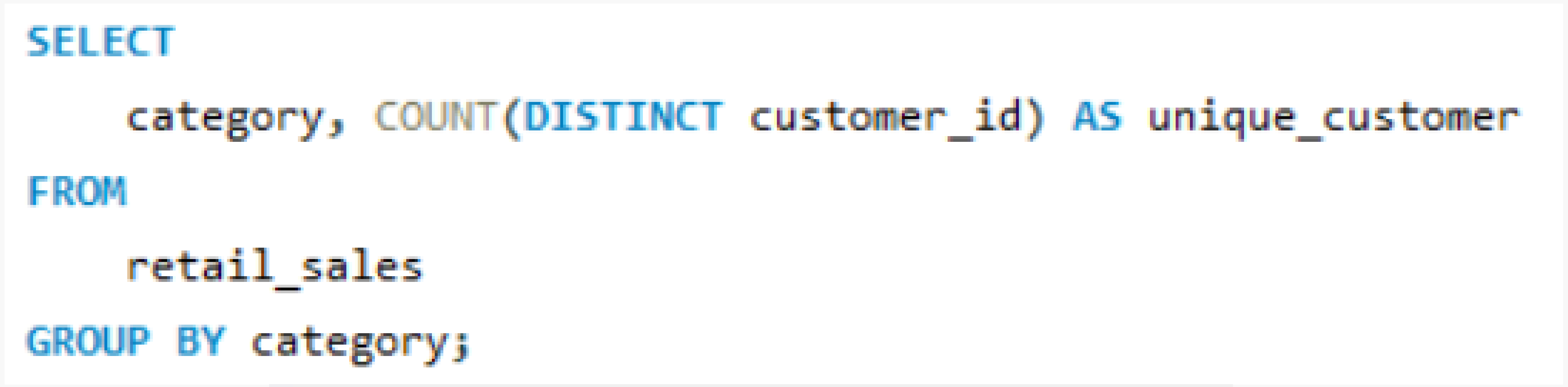


```
SELECT
    customer_id, SUM(total_sale) AS total_sale
FROM
    retail_sales
GROUP BY customer_id
ORDER BY total_sale DESC
LIMIT 5;
```

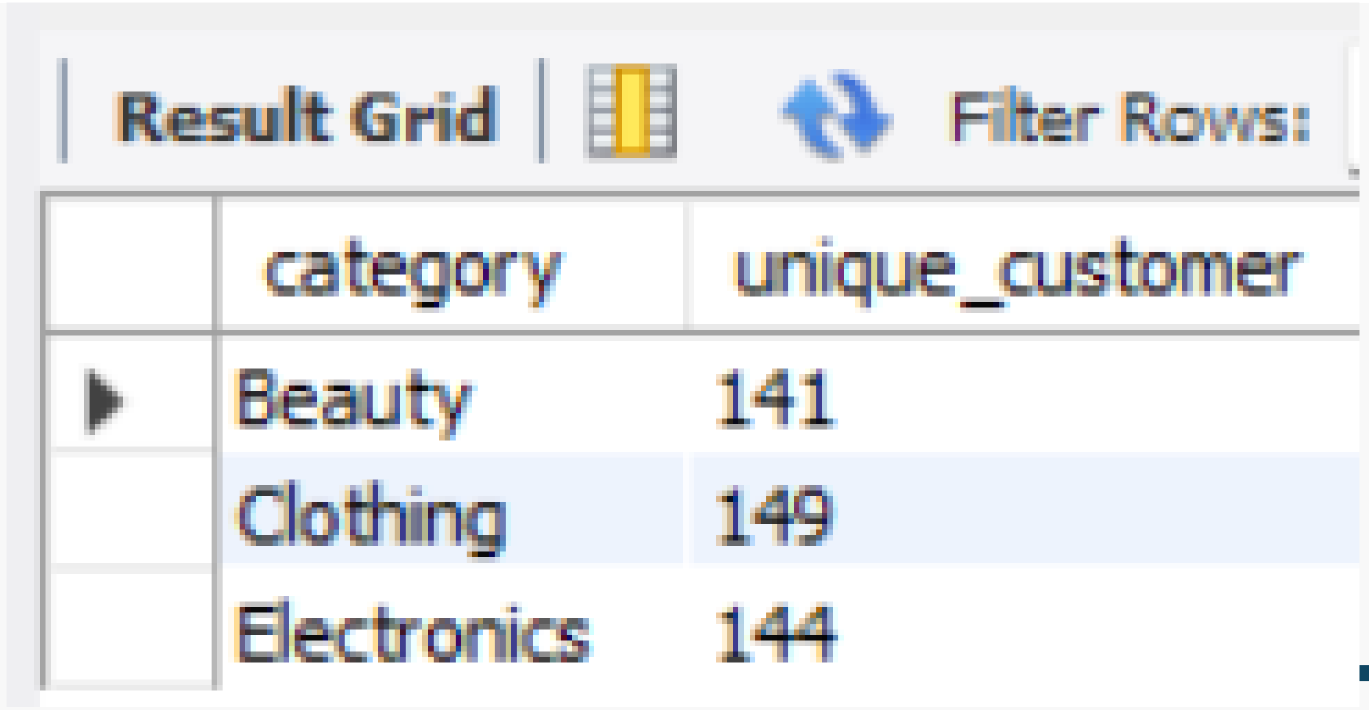


customer_id	total_sale
3	38440
1	30750
5	30405
2	25295
4	23580

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

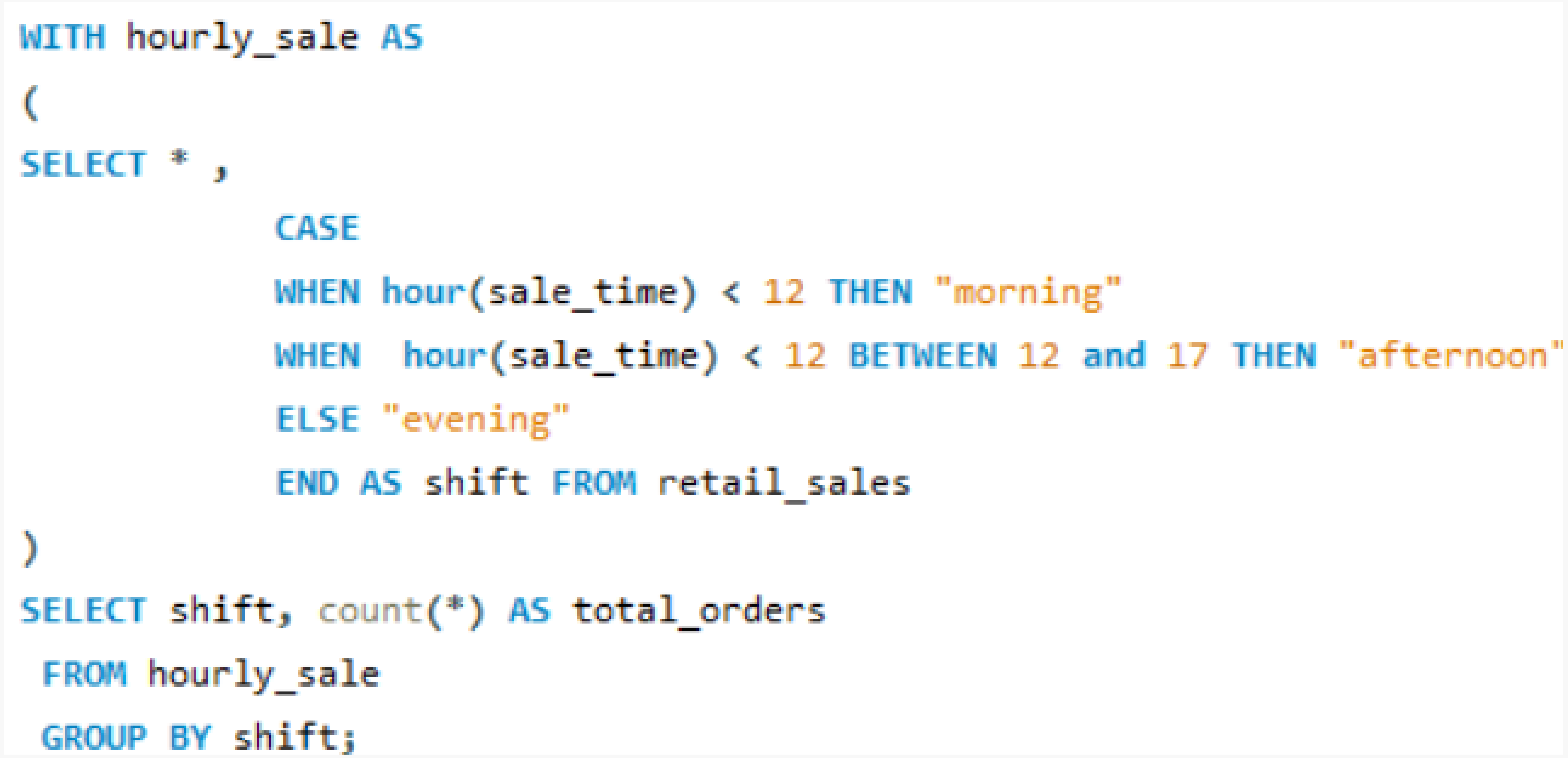


```sql
SELECT
    category, COUNT(DISTINCT customer_id) AS unique_customer
FROM
    retail_sales
GROUP BY category;
```



```
category      unique_customer
Beauty        141
Clothing      149
Electronics   144
```

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 hour(sale_time) < 12 THEN "morning"
            WHEN  hour(sale_time) < 12 BETWEEN 12 and 17 THEN "afternoon"
            ELSE "evening"
            END AS shift FROM retail_sales
)
SELECT shift, count(*) AS total_orders
 FROM hourly_sale
 GROUP BY shift;
```

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.

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.

Thank you