Data Analysis Project

SUPERSHOP SALES REPORT

11-28-2024

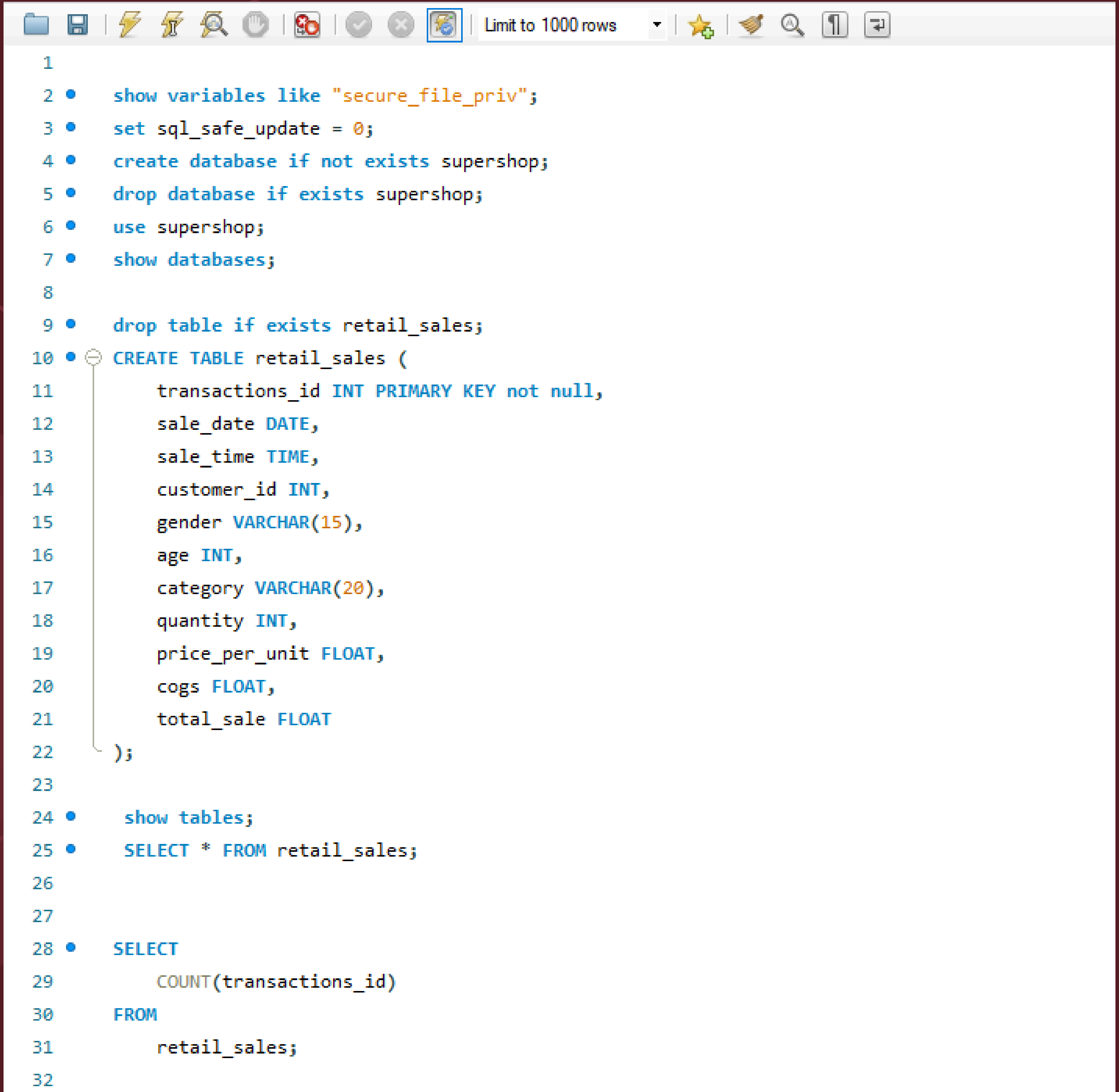
Prepared By: Md Anwarul karim

INTRODUCTION

A Super Shop Sales Analysis project using MySQL involves designing and querying a database to derive insights into retail performance. The database typically includes tables for transactions, products, customers, and sales. Using SQL, one can analyze key metrics such as total sales, average sales per transaction, top-performing categories, and customer demographics. Additional insights include identifying peak shopping hours, best-selling products, and sales trends over time. Advanced queries can group sales by shifts (morning, afternoon, evening) or months to highlight seasonal patterns. This project helps businesses make data-driven decisions, improve inventory management, and optimize customer engagement strategies for higher profitability.

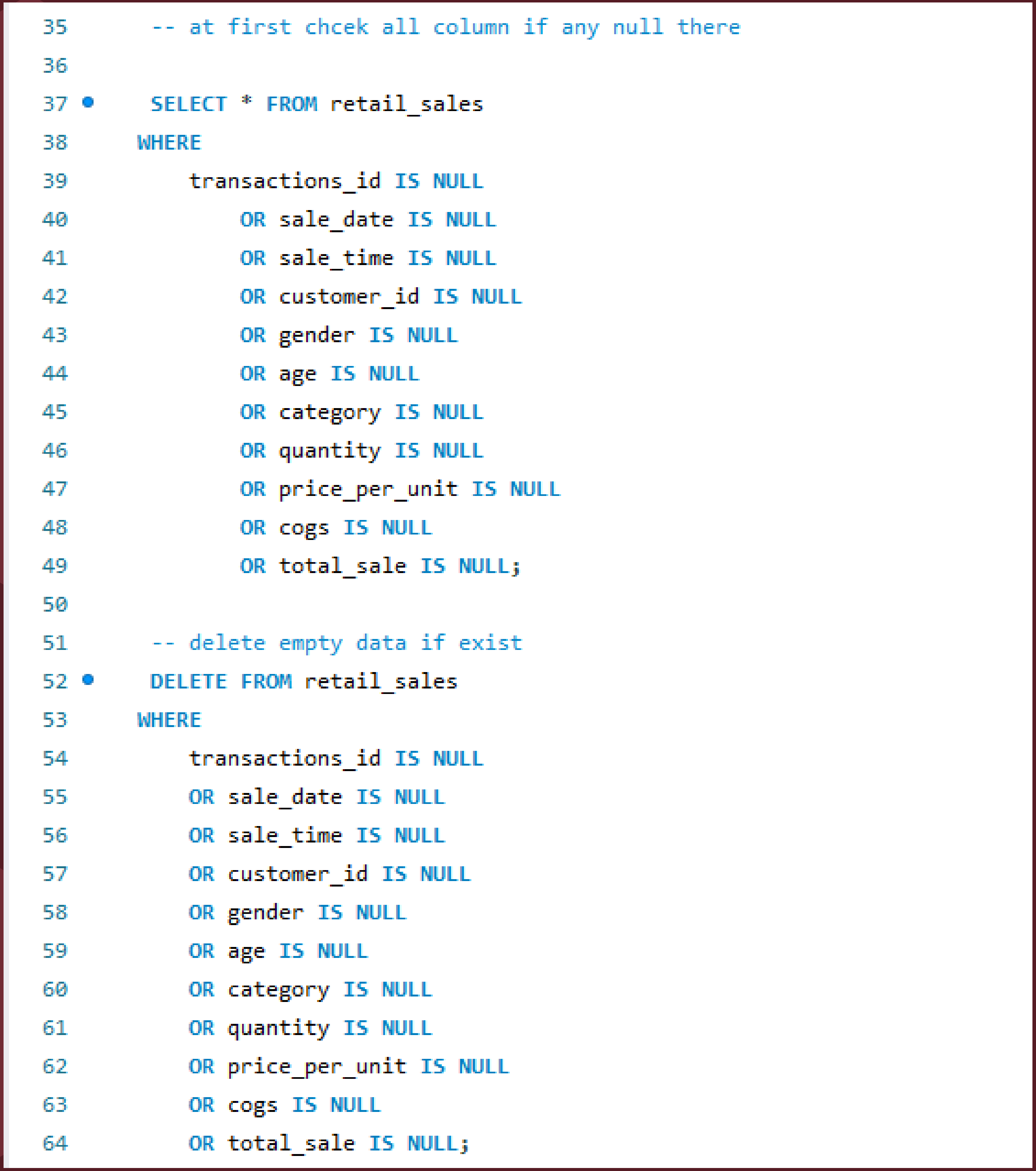
Create Database Super shop

Get the full code from my GitHub repo



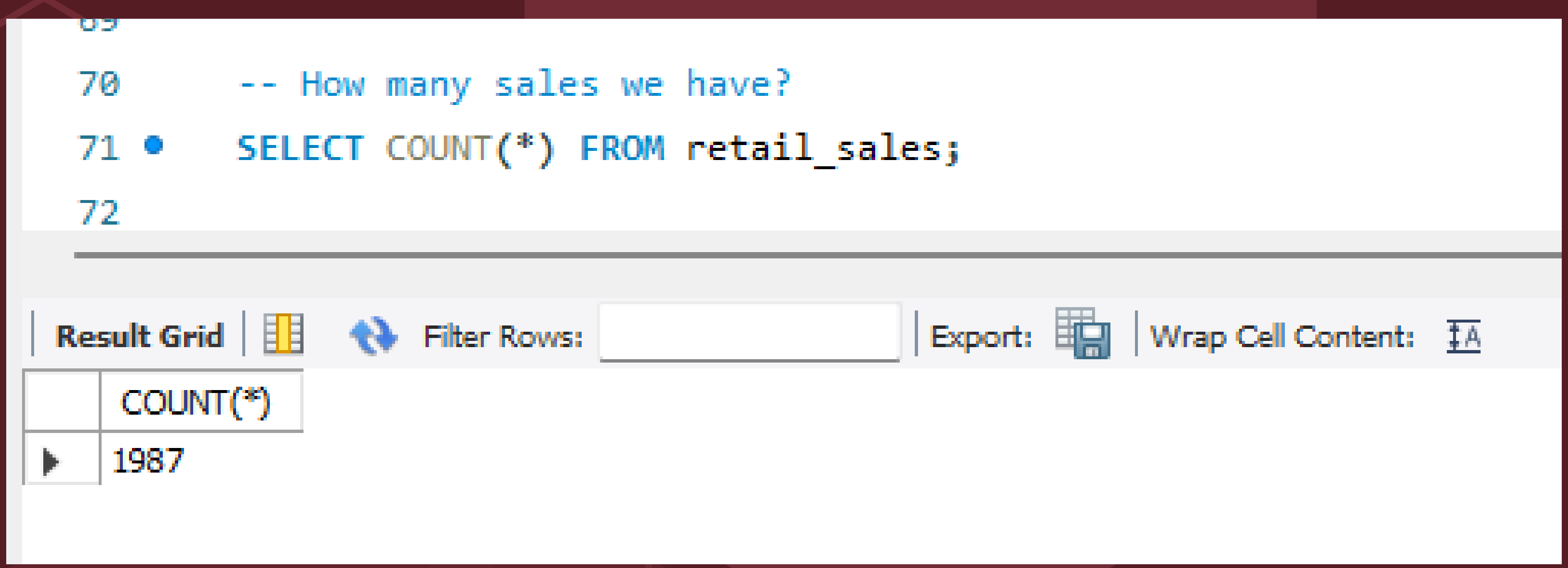
```sql

show variables like "secure_file_priv";
set sql_safe_update = 0;
create database if not exists supershop;
drop database if exists supershop;
use supershop;
show databases;

drop table if exists retail_sales;
CREATE TABLE retail_sales (
    transactions_id INT PRIMARY KEY not null,
    sale_date DATE,
    sale_time TIME,
    customer_id INT,
    gender VARCHAR(15),
    age INT,
    category VARCHAR(20),
    quantity INT,
    price_per_unit FLOAT,
    cogs FLOAT,
    total_sale FLOAT
);

 show tables;
 SELECT * FROM retail_sales;


SELECT
    COUNT(transactions_id)
FROM
    retail_sales;

```

Check the All column if any null value is there then delete the empty value



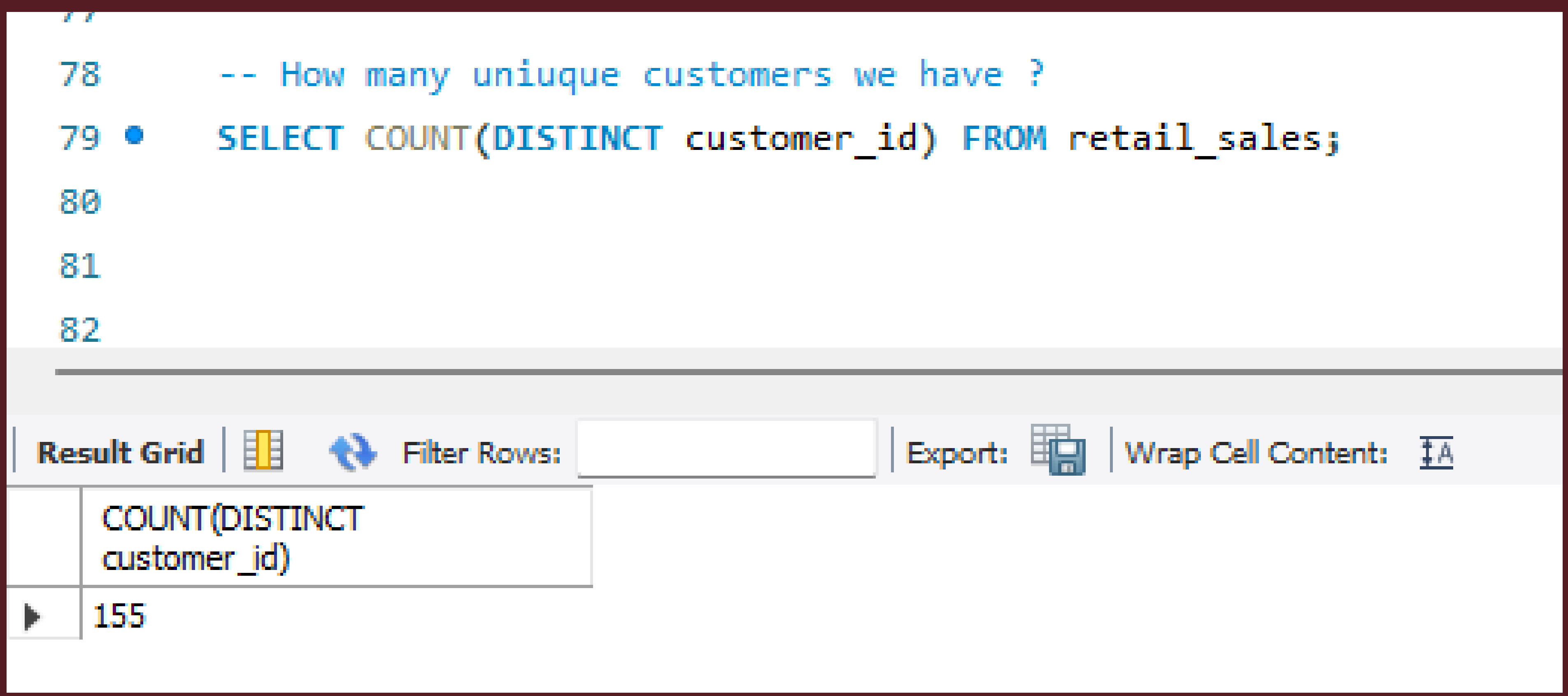
```
 -- at first chcek all column if any null there

 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 quantity IS NULL
        OR price_per_unit IS NULL
        OR cogs IS NULL
        OR total_sale IS NULL;

 -- delete empty data if exist
 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 quantity IS NULL
    OR price_per_unit IS NULL
    OR cogs IS NULL
    OR total_sale IS NULL;
```

Check all unique values



```
-- How many sales we have?
SELECT COUNT(*) FROM retail_sales;
```

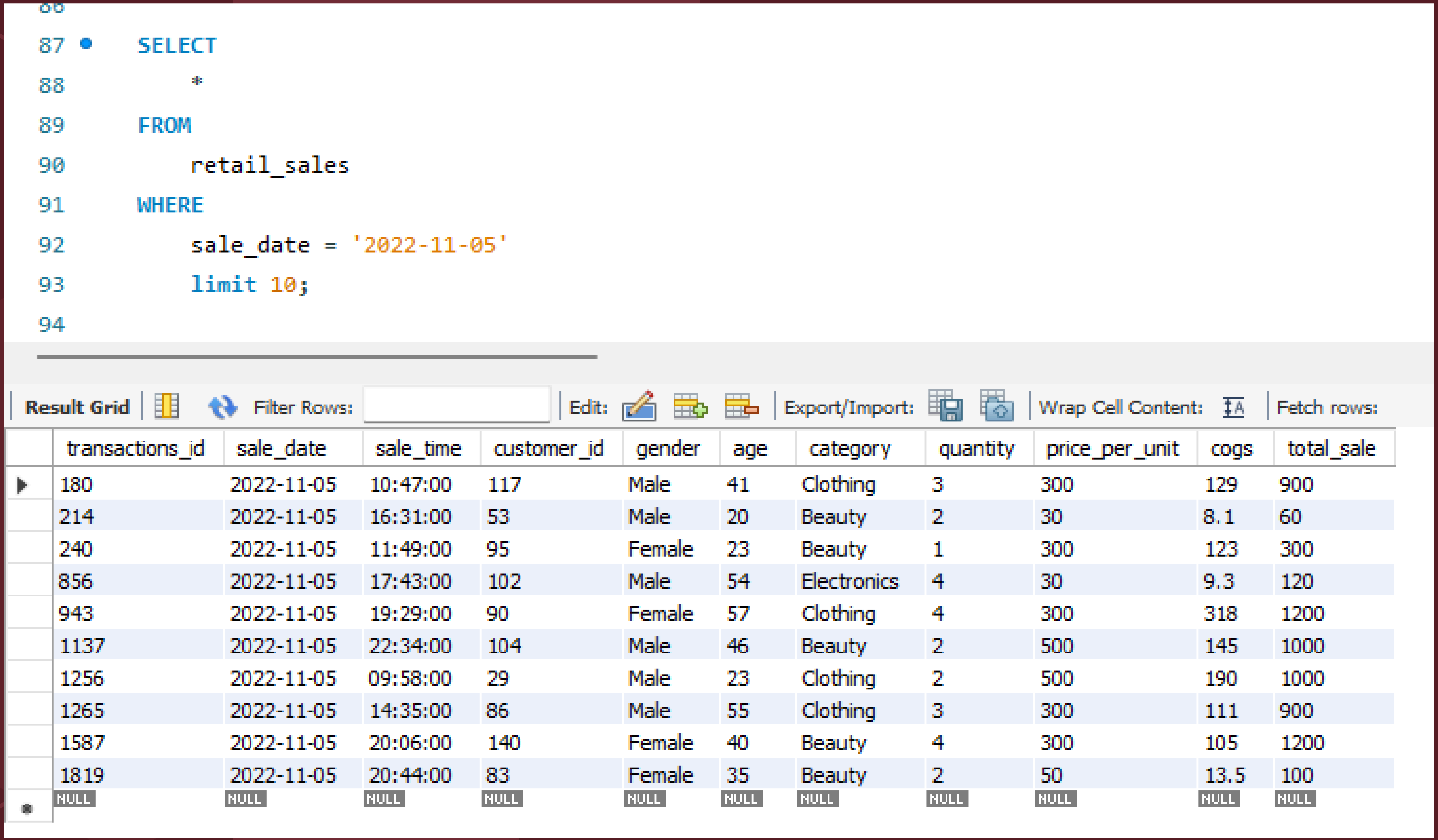
```
COUNT(*)
1987
```



```
-- How many uniuque customers we have ?
SELECT COUNT(DISTINCT customer_id) FROM retail_sales;
```

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

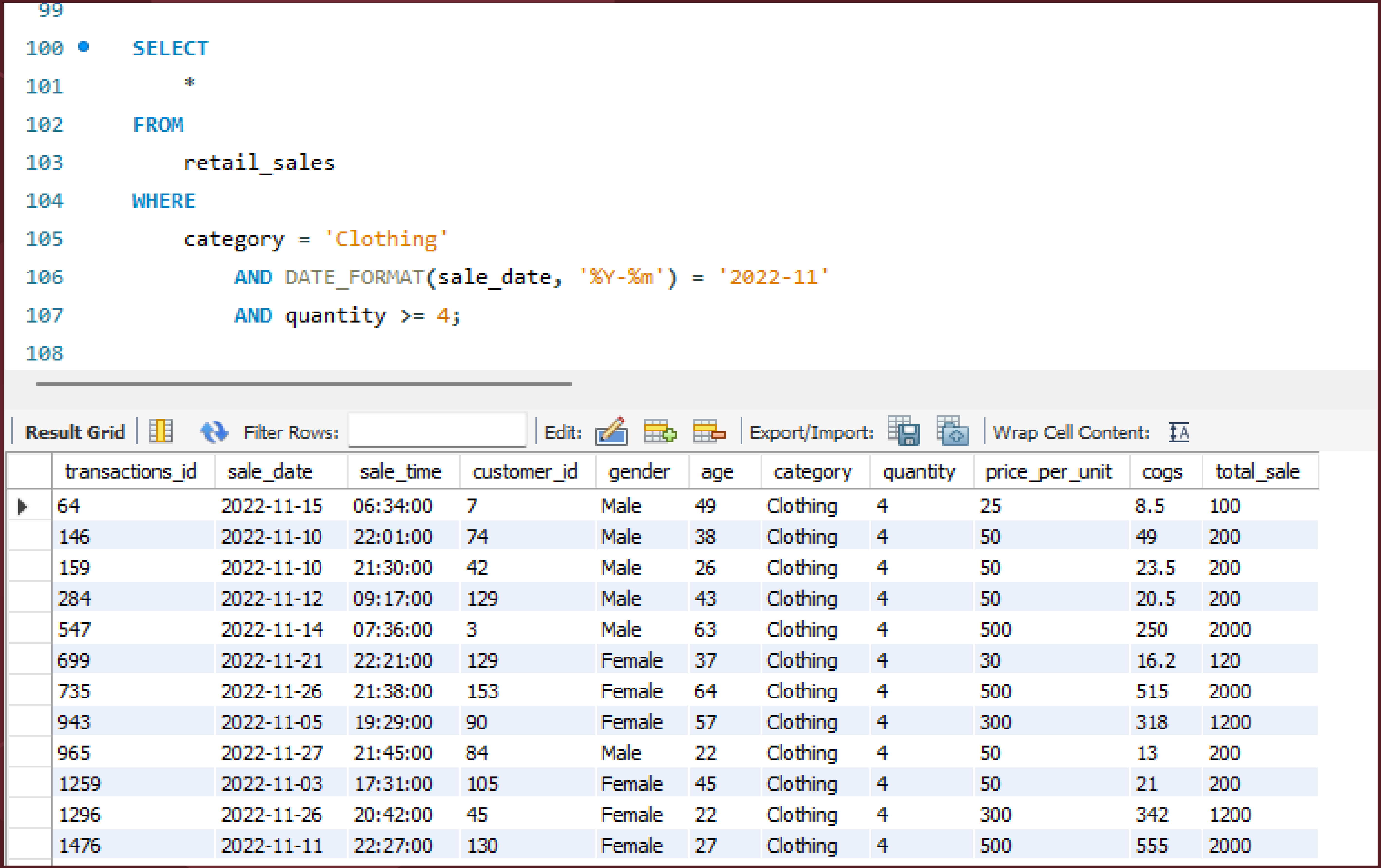
1. Write a SQL query to retrieve all columns for sales made on November 5, 2022.



```
SELECT
    *
FROM
    retail_sales
WHERE
    sale_date = '2022-11-05'
    limit 10;
```

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
214	2022-11-05	16:31:00	53	Male	20	Beauty	2	30	8.1	60
240	2022-11-05	11:49:00	95	Female	23	Beauty	1	300	123	300
856	2022-11-05	17:43:00	102	Male	54	Electronics	4	30	9.3	120
943	2022-11-05	19:29:00	90	Female	57	Clothing	4	300	318	1200
1137	2022-11-05	22:34:00	104	Male	46	Beauty	2	500	145	1000
1256	2022-11-05	09:58:00	29	Male	23	Clothing	2	500	190	1000
1265	2022-11-05	14:35:00	86	Male	55	Clothing	3	300	111	900
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
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2. Write an SQL query to retrieve all transactions in November 2022 in which the category is 'Clothing' and the quantity sold exceeds 10.

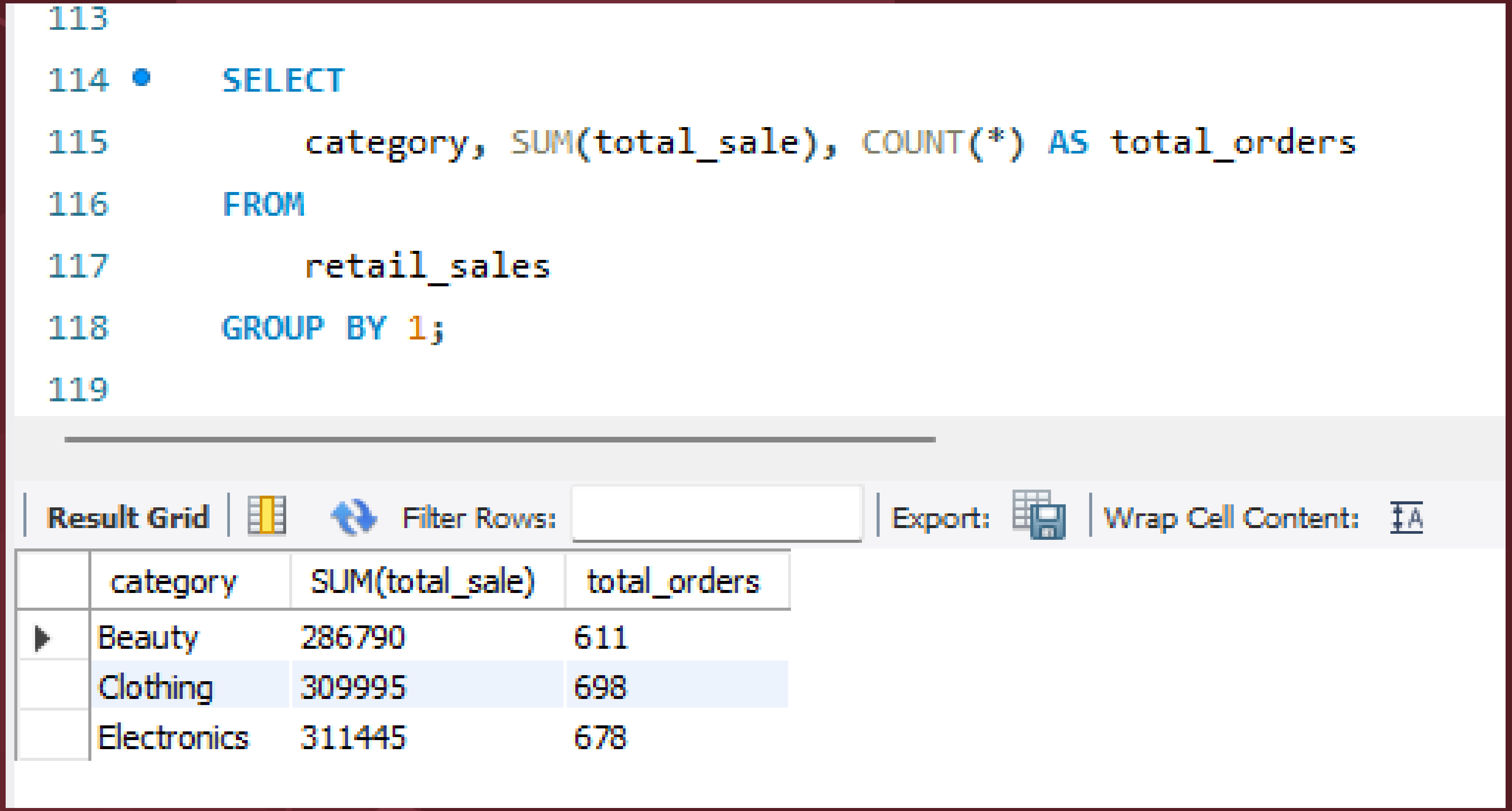


```
SELECT
    *
FROM
    retail_sales
WHERE
    category = 'Clothing'
        AND DATE_FORMAT(sale_date, '%Y-%m') = '2022-11'
        AND quantity >= 4;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

transactions_id	sale_date	sale_time	customer_id	gender	age	category	quantity	price_per_unit	cogs	total_sale
64	2022-11-15	06:34:00	7	Male	49	Clothing	4	25	8.5	100
146	2022-11-10	22:01:00	74	Male	38	Clothing	4	50	49	200
159	2022-11-10	21:30:00	42	Male	26	Clothing	4	50	23.5	200
284	2022-11-12	09:17:00	129	Male	43	Clothing	4	50	20.5	200
547	2022-11-14	07:36:00	3	Male	63	Clothing	4	500	250	2000
699	2022-11-21	22:21:00	129	Female	37	Clothing	4	30	16.2	120
735	2022-11-26	21:38:00	153	Female	64	Clothing	4	500	515	2000
943	2022-11-05	19:29:00	90	Female	57	Clothing	4	300	318	1200
965	2022-11-27	21:45:00	84	Male	22	Clothing	4	50	13	200
1259	2022-11-03	17:31:00	105	Female	45	Clothing	4	50	21	200
1296	2022-11-26	20:42:00	45	Female	22	Clothing	4	300	342	1200
1476	2022-11-11	22:27:00	130	Female	27	Clothing	4	500	555	2000

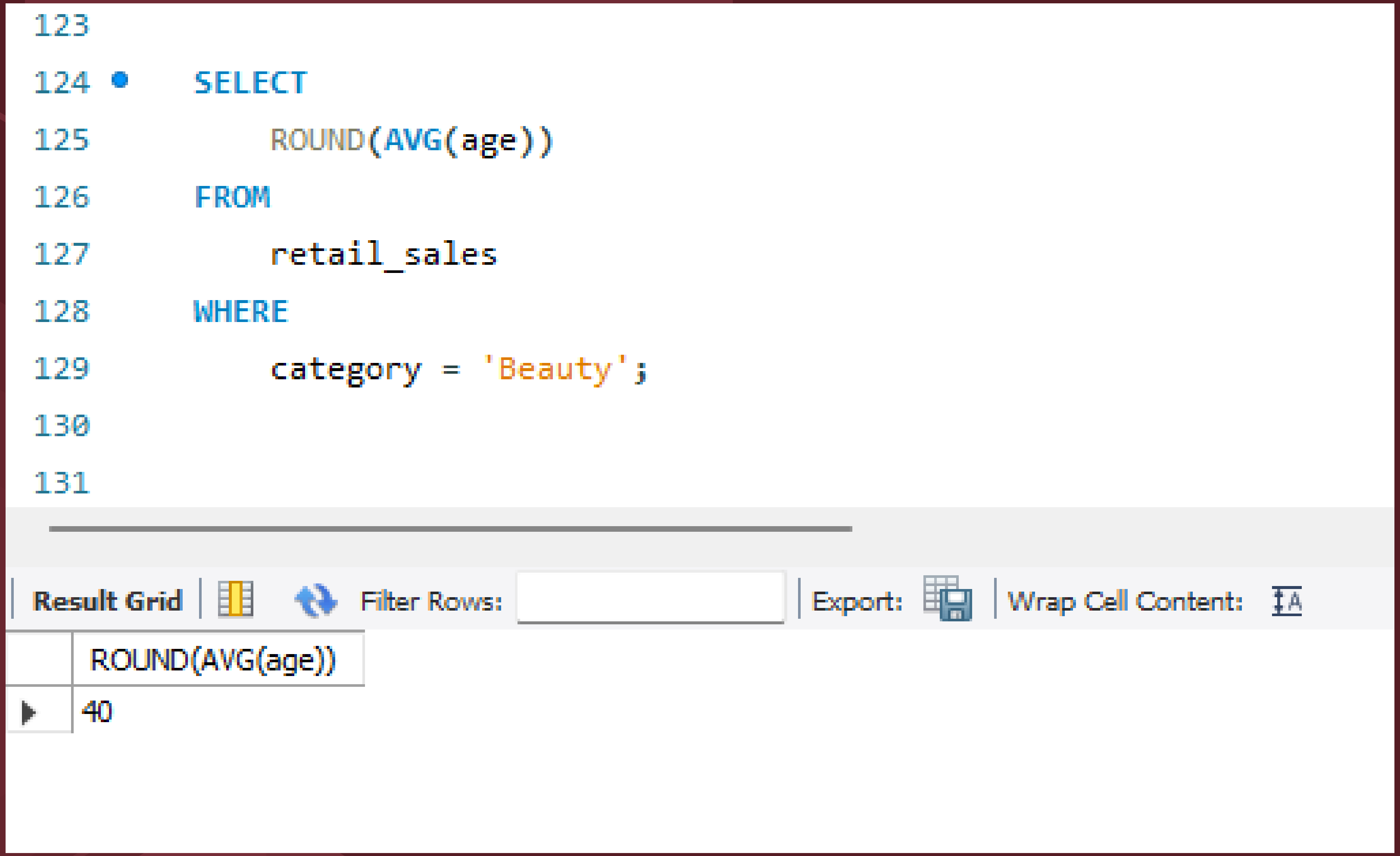
3. Write a SQL query to calculate the total sales (total_sale) for each product category.



```sql
SELECT
    category, SUM(total_sale), COUNT(*) AS total_orders
FROM
    retail_sales
GROUP BY 1;
```

category	SUM(total_sale)	total_orders
Beauty	286790	611
Clothing	309995	698
Electronics	311445	678

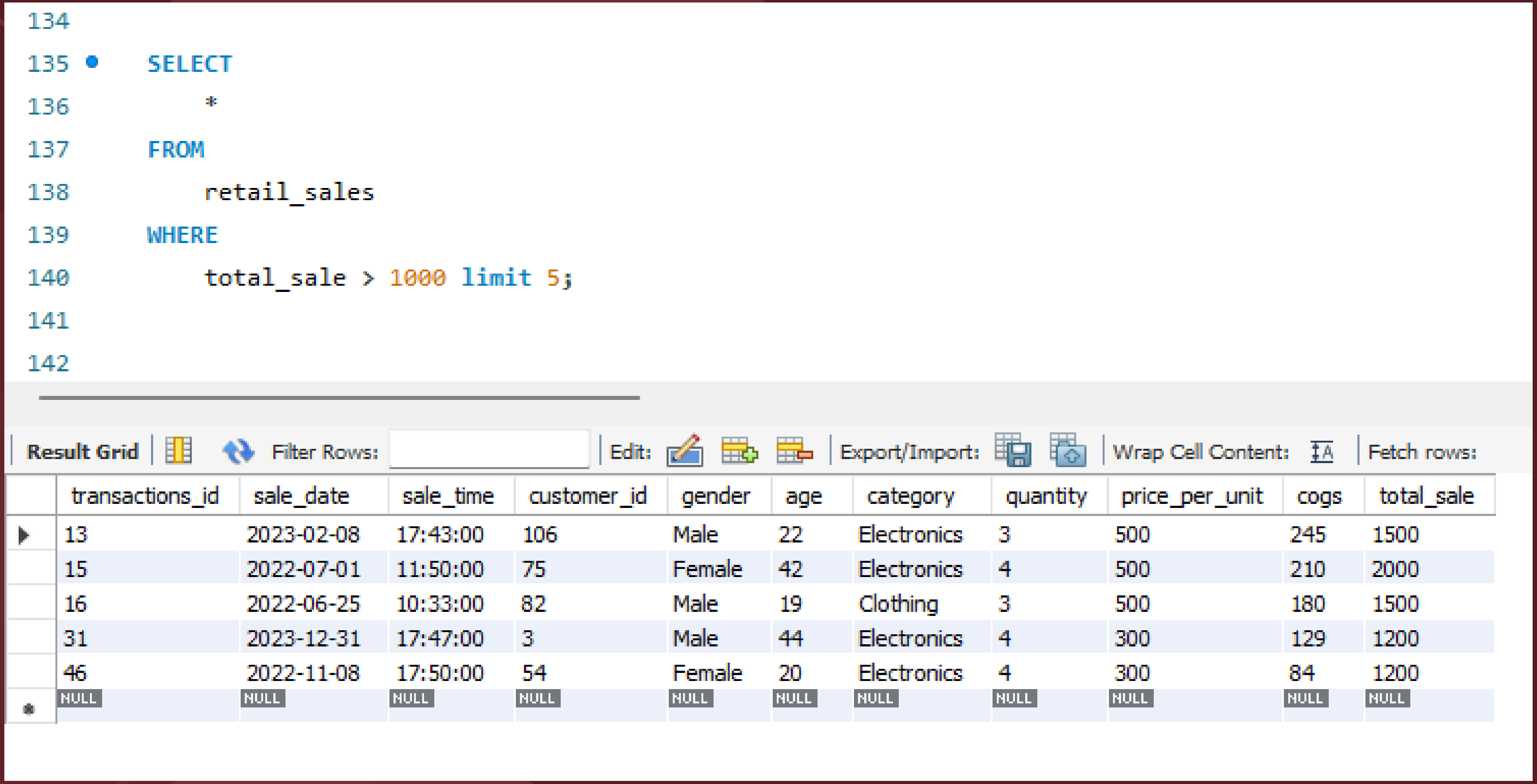
4. Write a SQL query to calculate the average age of customers who purchased items from the 'Beauty' category.



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

```
ROUND(AVG(age))
40
```

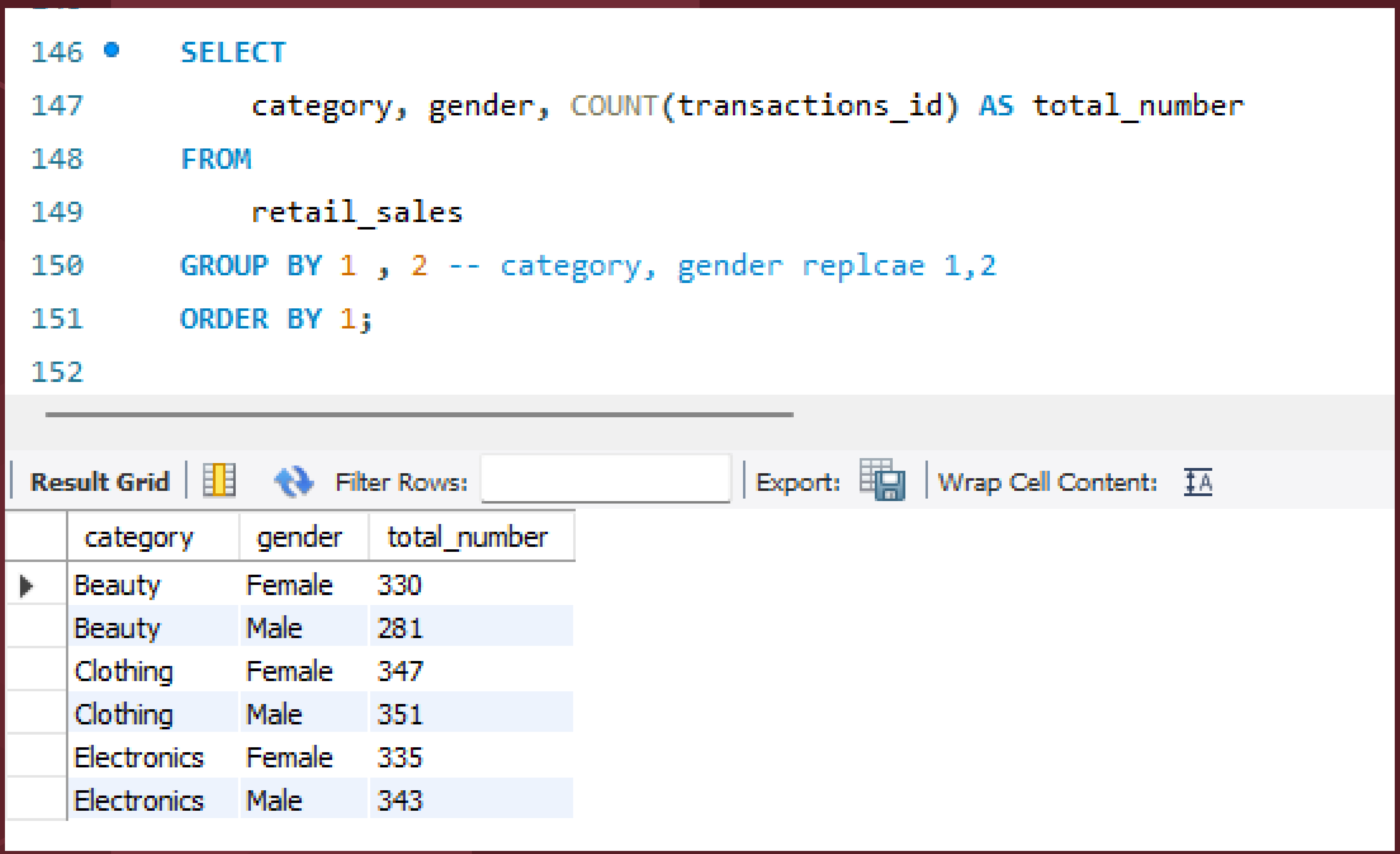
5. Write a SQL query to find all transactions where total_sale exceeds 1000 limit 5.



```
SELECT
    *
FROM
    retail_sales
WHERE
    total_sale > 1000 limit 5;
```

transactions_id	sale_date	sale_time	customer_id	gender	age	category	quantity	price_per_unit	cogs	total_sale
13	2023-02-08	17:43:00	106	Male	22	Electronics	3	500	245	1500
15	2022-07-01	11:50:00	75	Female	42	Electronics	4	500	210	2000
16	2022-06-25	10:33:00	82	Male	19	Clothing	3	500	180	1500
31	2023-12-31	17:47:00	3	Male	44	Electronics	4	300	129	1200
46	2022-11-08	17:50:00	54	Female	20	Electronics	4	300	84	1200
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

6. Write a SQL query to calculate the total number of transactions (transaction_id) made by each gender within each category.

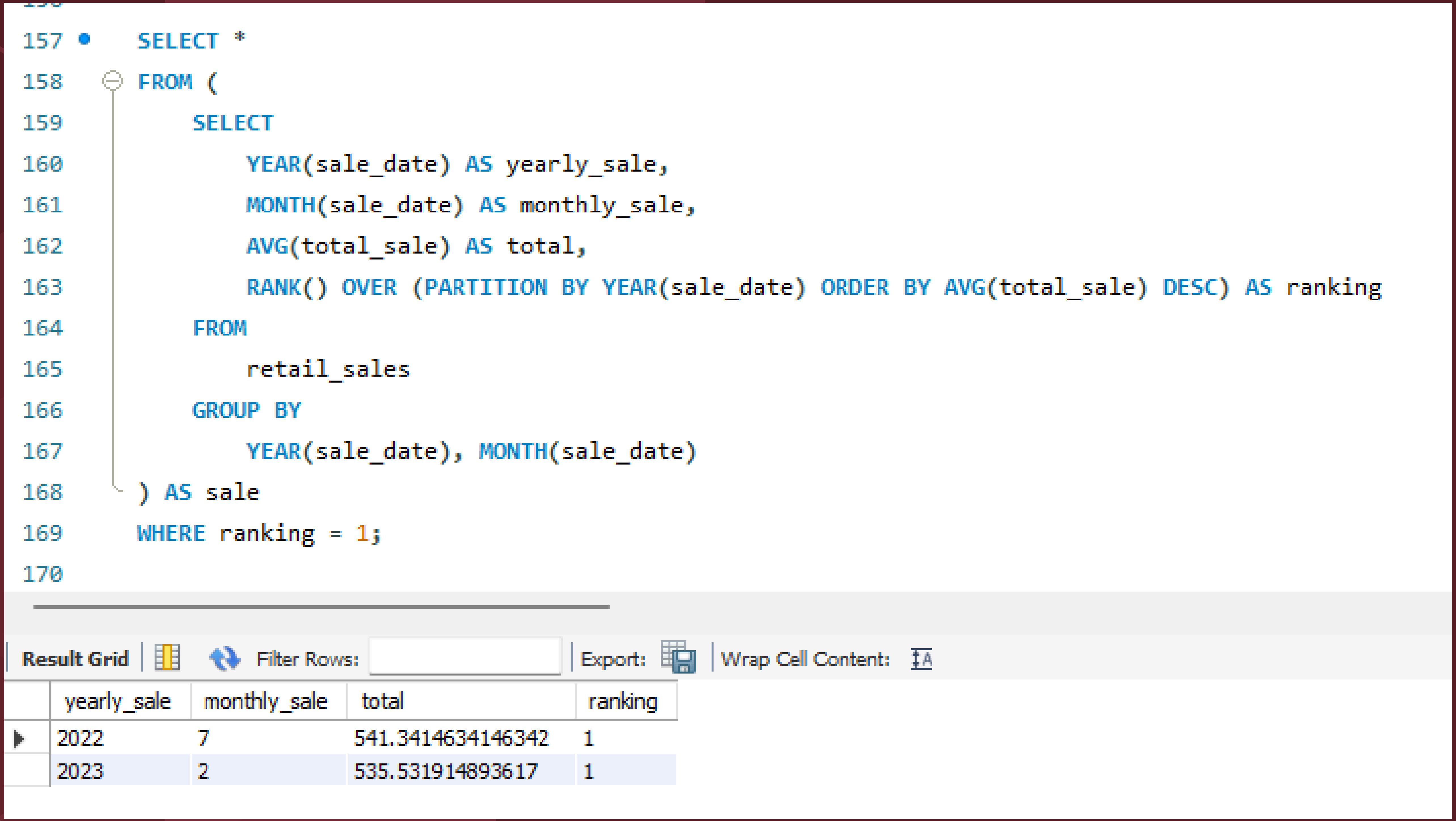


```
SELECT
    category, gender, COUNT(transactions_id) AS total_number
FROM
    retail_sales
GROUP BY 1 , 2 -- category, gender replcae 1,2
ORDER BY 1;
```

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

category	gender	total_number
Beauty	Female	330
Beauty	Male	281
Clothing	Female	347
Clothing	Male	351
Electronics	Female	335
Electronics	Male	343

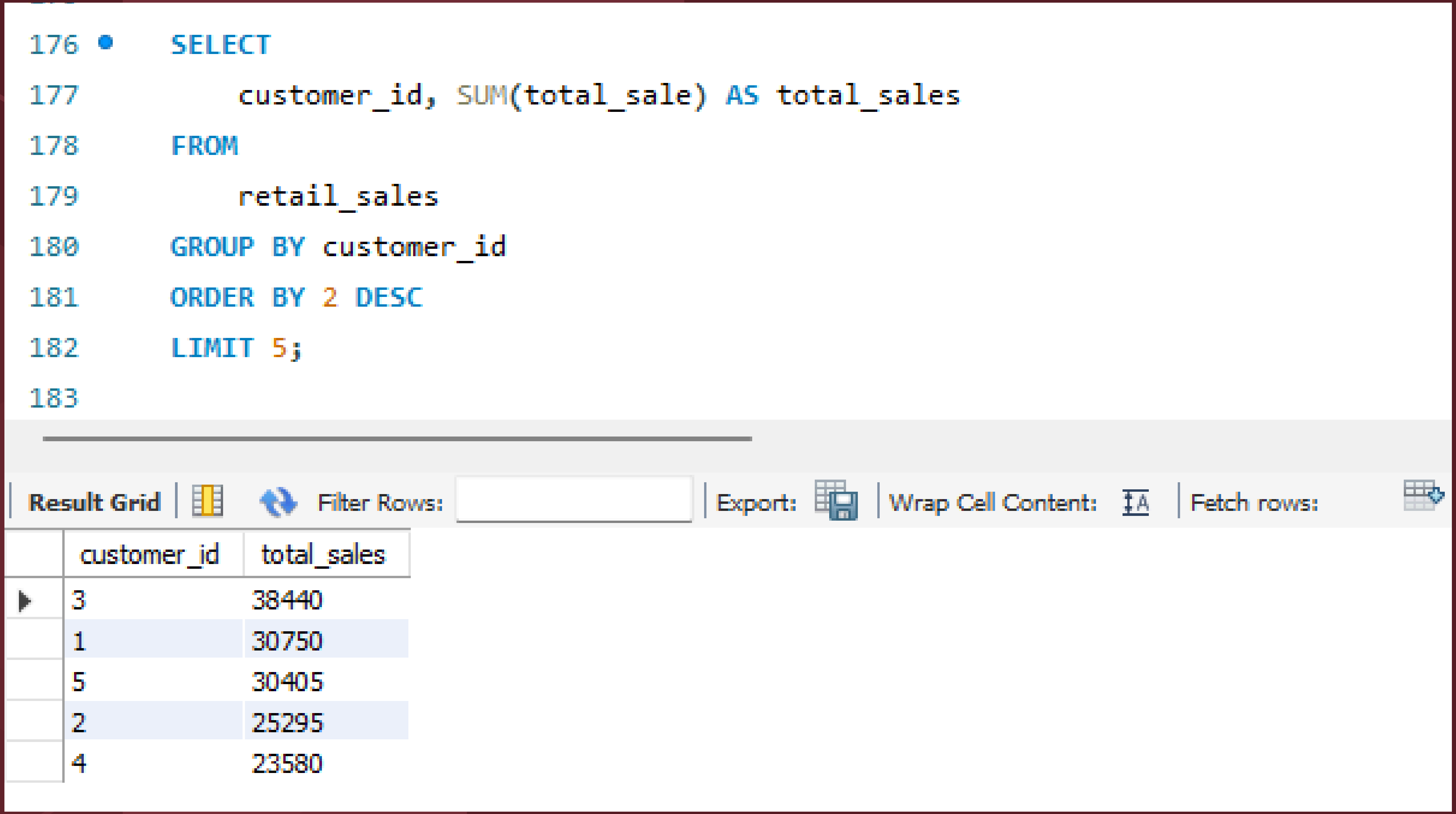
7. Write a SQL query to calculate the average monthly sales. Identify the best-selling month for each year



```
SELECT *
FROM (
    SELECT
        YEAR(sale_date) AS yearly_sale,
        MONTH(sale_date) AS monthly_sale,
        AVG(total_sale) AS total,
        RANK() OVER (PARTITION BY YEAR(sale_date) ORDER BY AVG(total_sale) DESC) AS ranking
    FROM
        retail_sales
    GROUP BY
        YEAR(sale_date), MONTH(sale_date)
) AS sale
WHERE ranking = 1;
```

yearly_sale	monthly_sale	total	ranking
2022	7	541.3414634146342	1
2023	2	535.531914893617	1

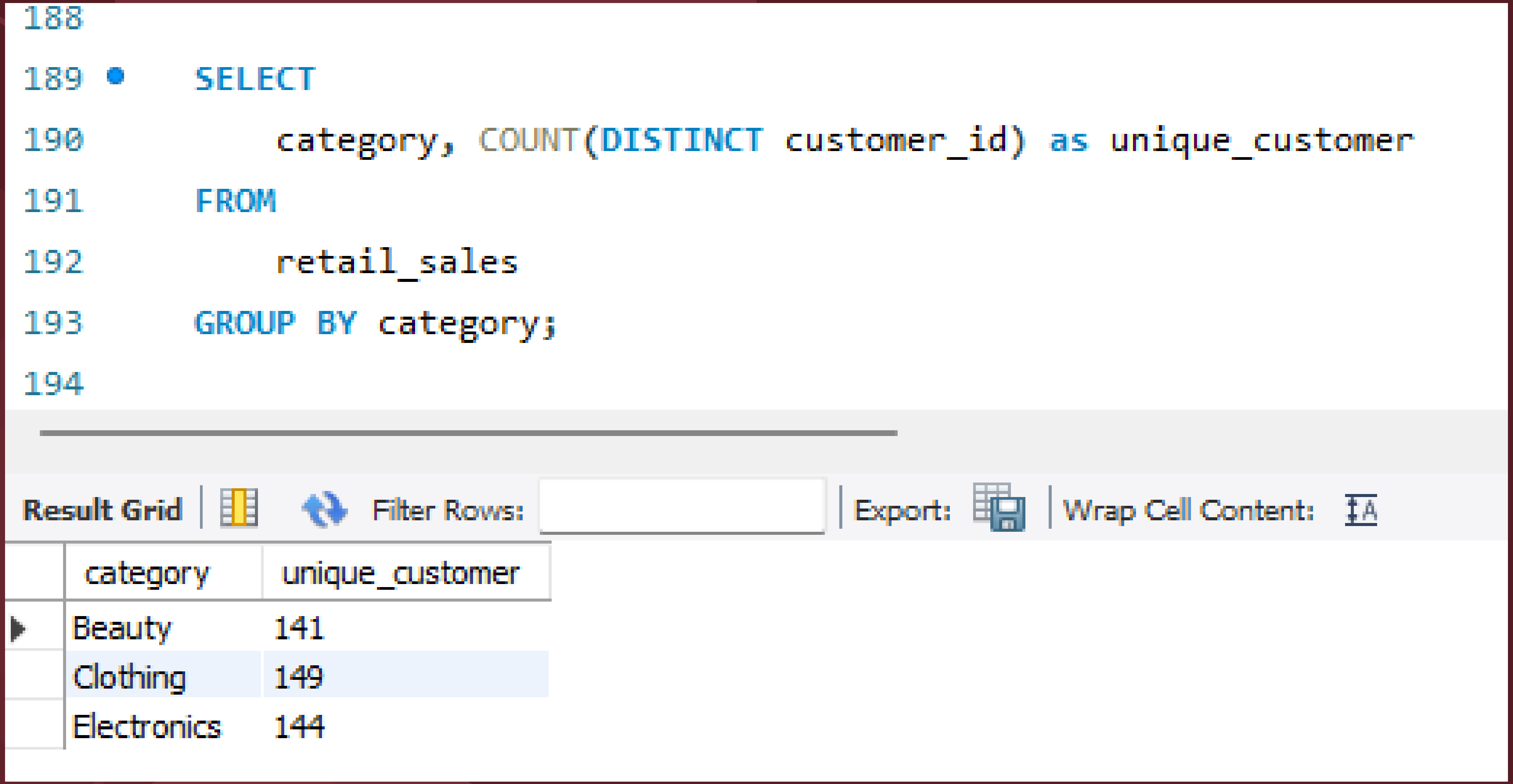
8. Write a SQL query to find the top 5 customers based on the highest total sales.



```
SELECT
    customer_id, SUM(total_sale) AS total_sales
FROM
    retail_sales
GROUP BY customer_id
ORDER BY 2 DESC
LIMIT 5;
```

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

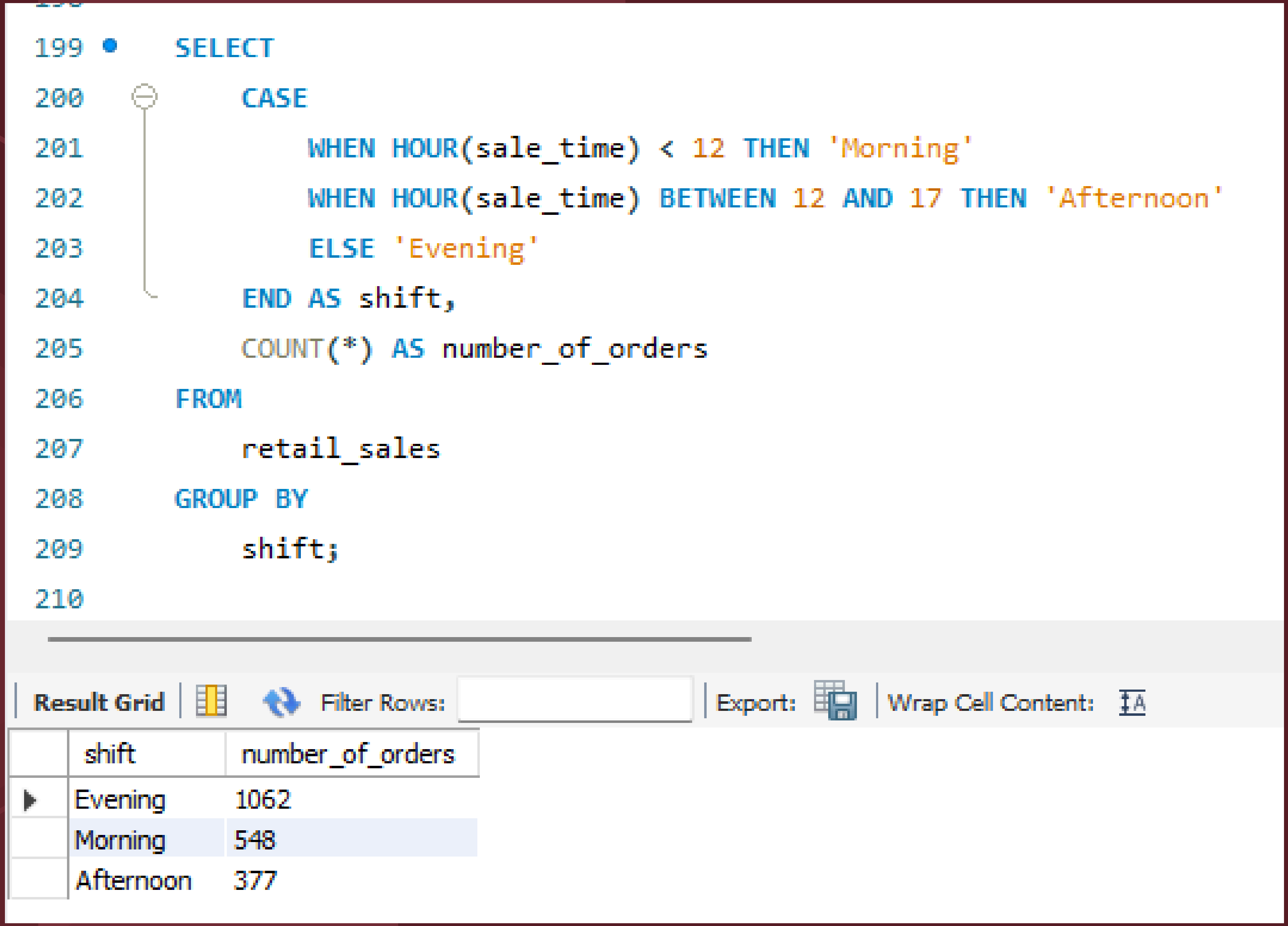
9. Write a SQL query to calculate the number of unique customers who purchased items from each category.



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

category	unique_customer
Beauty	141
Clothing	149
Electronics	144

10. Write a SQL query to classify orders into shifts (Morning: ≤12, Afternoon: 12—17, Evening: >17) and count the number of orders for each shift.



```
SELECT
    CASE
        WHEN HOUR(sale_time) < 12 THEN 'Morning'
        WHEN HOUR(sale_time) BETWEEN 12 AND 17 THEN 'Afternoon'
        ELSE 'Evening'
    END AS shift,
    COUNT(*) AS number_of_orders
FROM
    retail_sales
GROUP BY
    shift;
```

shift	number_of_orders
Evening	1062
Morning	548
Afternoon	377

THANK YOU!

Thank you for taking the time to engage with my presentation. If you have any questions or wish to explore the findings further, please feel free to reach out. Your support and collaboration are greatly valued and appreciated.