INNOMATICS
RESEARCH LABS

PROJECT ON

Grocery Store Management SQL Project

-by Kranthi Kumar Dande
Batch : 419

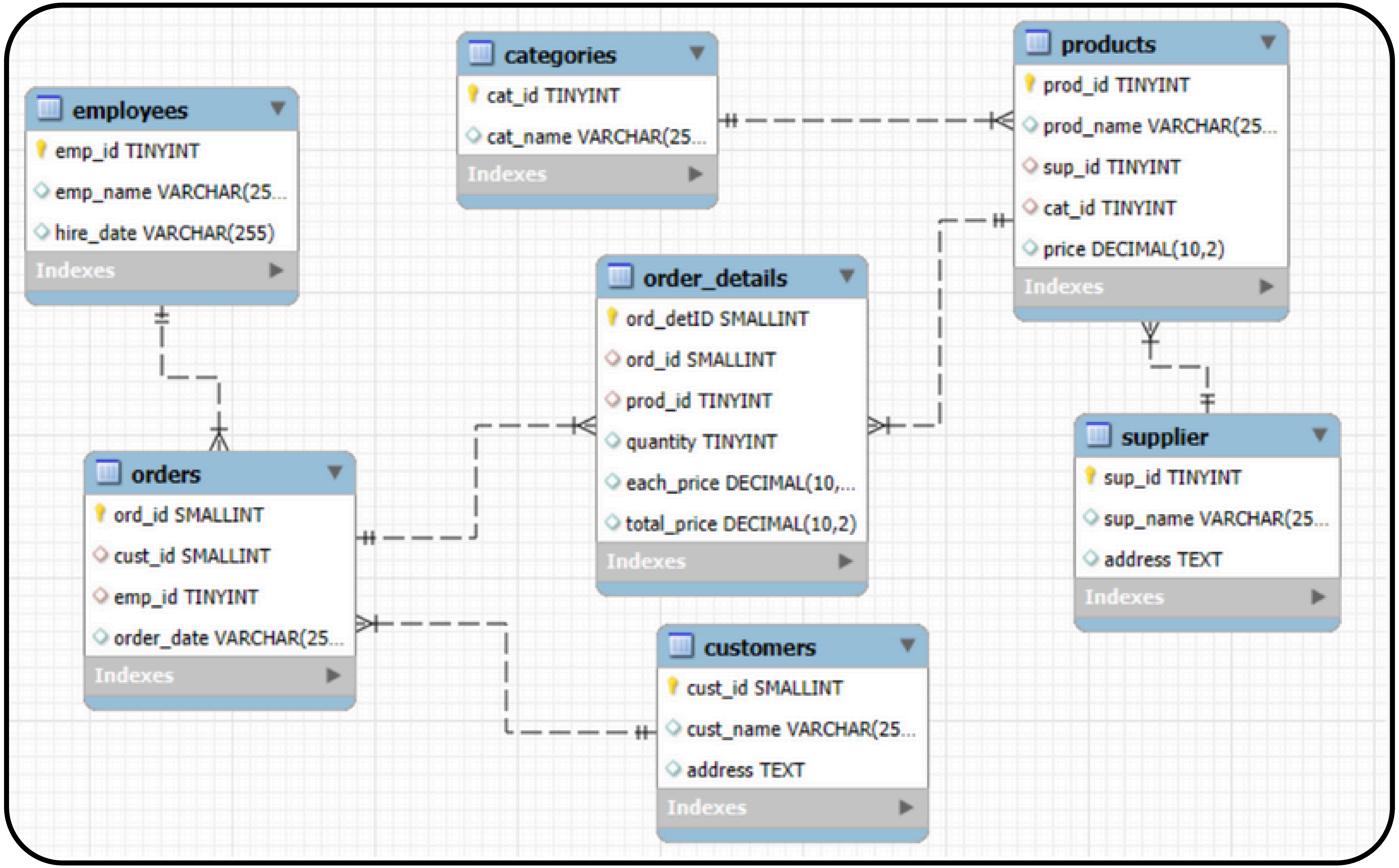
DATASET

- The database contains 7 tables used in a grocery store system.
- These tables store details about products, suppliers, customers, employees, orders, and order items.
- Data includes names, prices, quantities, dates, categories, and transaction details.
- All tables are connected through primary keys and foreign keys for accurate reporting.
- This dataset helps analyze sales patterns, customer purchases, product performance, and overall store activity.

Objectives

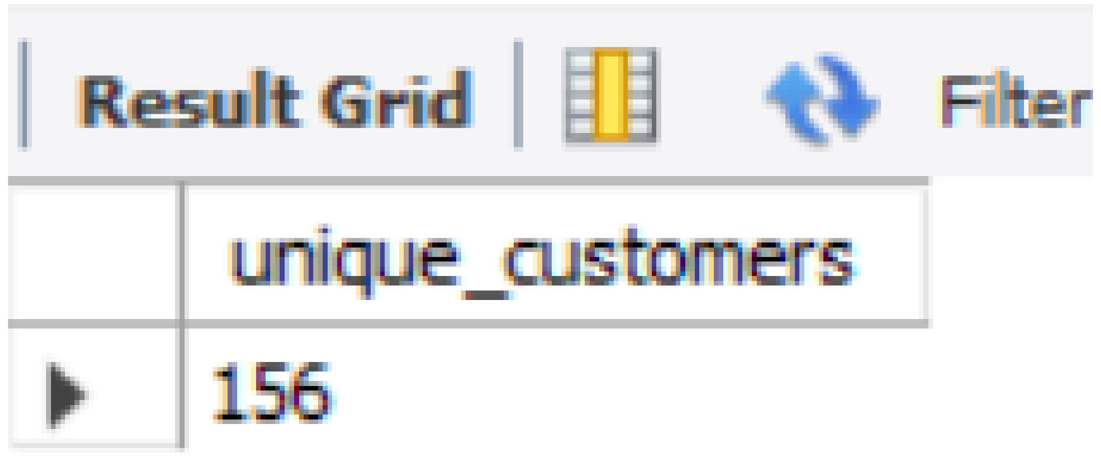
- To design and implement a relational database for a grocery store.
- To retrieve and manipulate data using SQL queries.
- To perform data analysis for business insights such as top customers, best-selling products, and revenue trends.
- To practice using joins, aggregations, subqueries, and filtering techniques.

ER Diagram


categories
cat_id TINYINT
cat_name VARCHAR(25...
Indexes
products
prod_id TINYINT
prod_name VARCHAR(25...
sup_id TINYINT
cat_id TINYINT
price DECIMAL(10,2)
Indexes
employees
emp_id TINYINT
emp_name VARCHAR(25...
hire_date VARCHAR(255)
Indexes
order_details
ord_detID SMALLINT
ord_id SMALLINT
prod_id TINYINT
quantity TINYINT
each_price DECIMAL(10,...
total_price DECIMAL(10,2)
Indexes
supplier
sup_id TINYINT
sup_name VARCHAR(25...
address TEXT
Indexes
orders
ord_id SMALLINT
cust_id SMALLINT
emp_id TINYINT
order_date VARCHAR(25...
Indexes
customers
cust_id SMALLINT
cust_name VARCHAR(25...
address TEXT
Indexes

Customer Insights

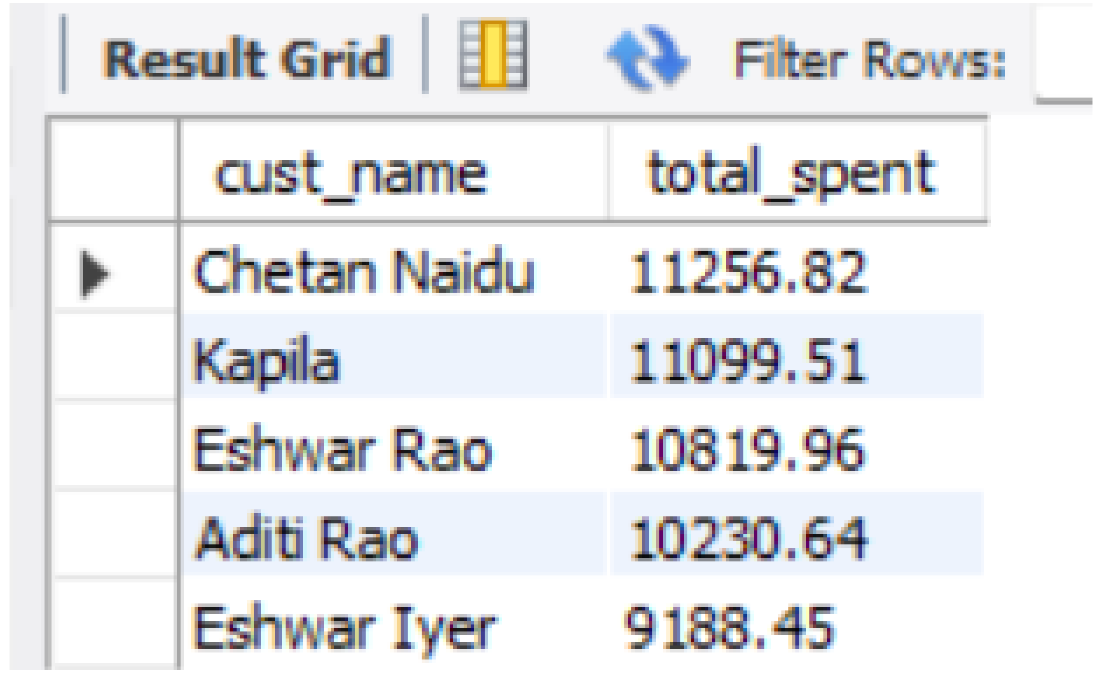
```
SELECT
    COUNT(DISTINCT cust_id) AS unique_customers
FROM orders;
```


Result Grid | Filter

	unique_customers
▶	156

# Total unique Customer 156

```
-- 4. Top 5 customers by total purchase
SELECT c.cust_name, SUM(od.total_price) AS total_spent
FROM customers c
JOIN orders o ON c.cust_id = o.cust_id
JOIN order_details od ON o.ord_id = od.ord_id
GROUP BY c.cust_id
ORDER BY total_spent DESC
LIMIT 5;
```

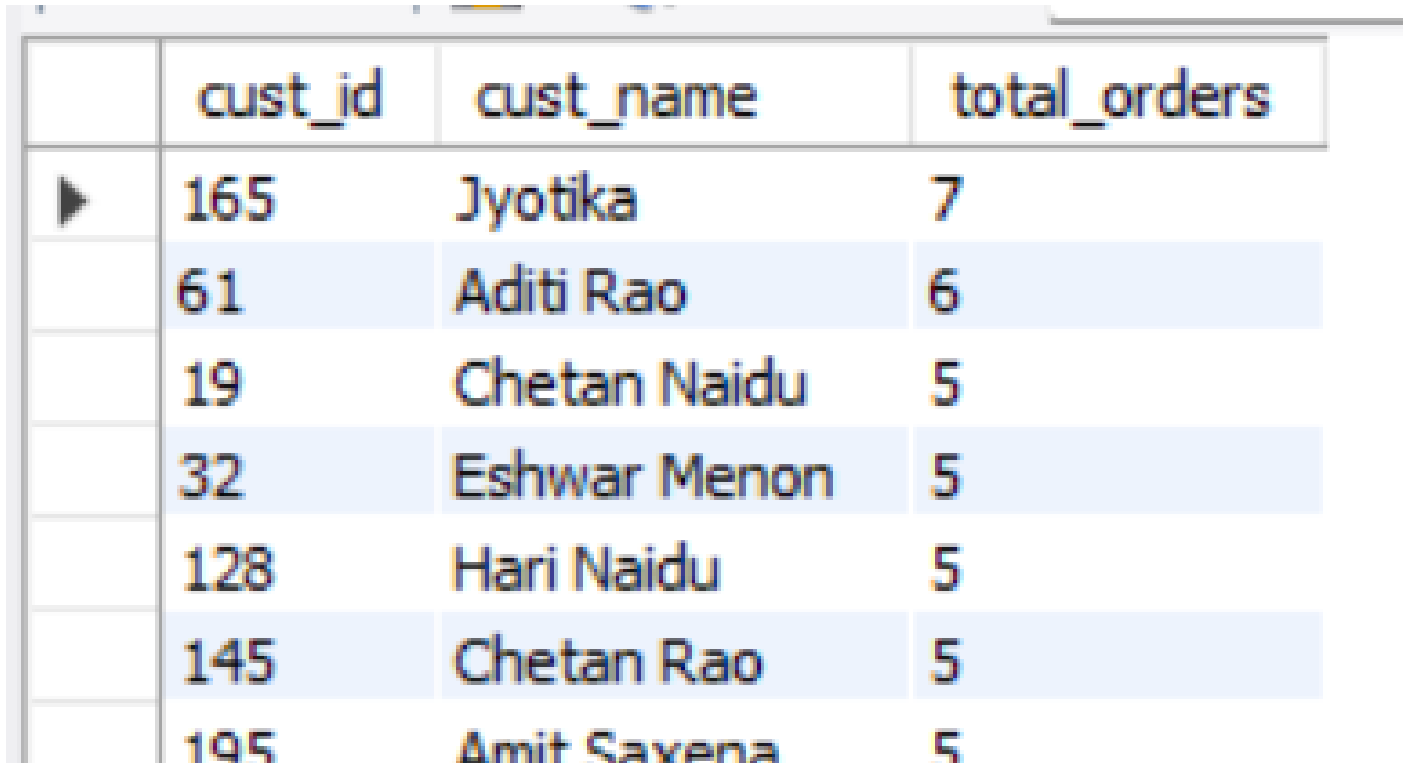

Result Grid | Filter Rows:

	cust_name	total_spent
▶	Chetan Naidu	11256.82
	Kapila	11099.51
	Eshwar Rao	10819.96
	Aditi Rao	10230.64
	Eshwar Iyer	9188.45

# These are Top 5 Customers on Purchases

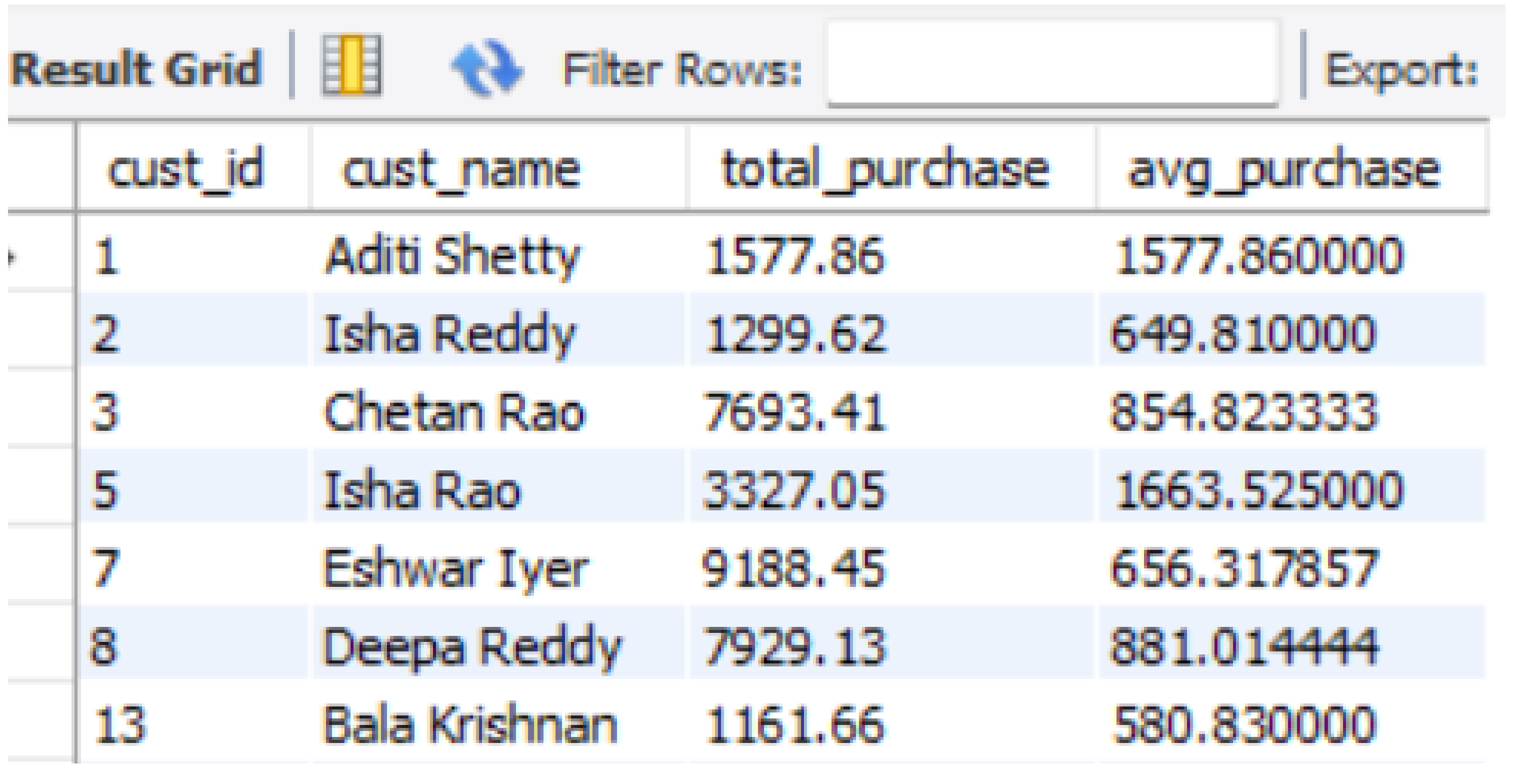
```sql
SELECT c.cust_id, c.cust_name, COUNT(o.ord_id) AS total_orders
FROM customers c
JOIN orders o ON c.cust_id = o.cust_id
GROUP BY c.cust_id
ORDER BY total_orders DESC;
```

```sql
-- 3. Total & average purchase value per customer
SELECT c.cust_id, c.cust_name,
       SUM(od.total_price) AS total_purchase,
       AVG(od.total_price) AS avg_purchase
FROM customers c
JOIN orders o ON c.cust_id = o.cust_id
JOIN order_details od ON o.ord_id = od.ord_id
GROUP BY c.cust_id;
```



cust_id	cust_name	total_orders
165	Jyotika	7
61	Aditi Rao	6
19	Chetan Naidu	5
32	Eshwar Menon	5
128	Hari Naidu	5
145	Chetan Rao	5
195	Amit Saxena	5

# These are customers orders frequency with 7, 6,5,4,3..



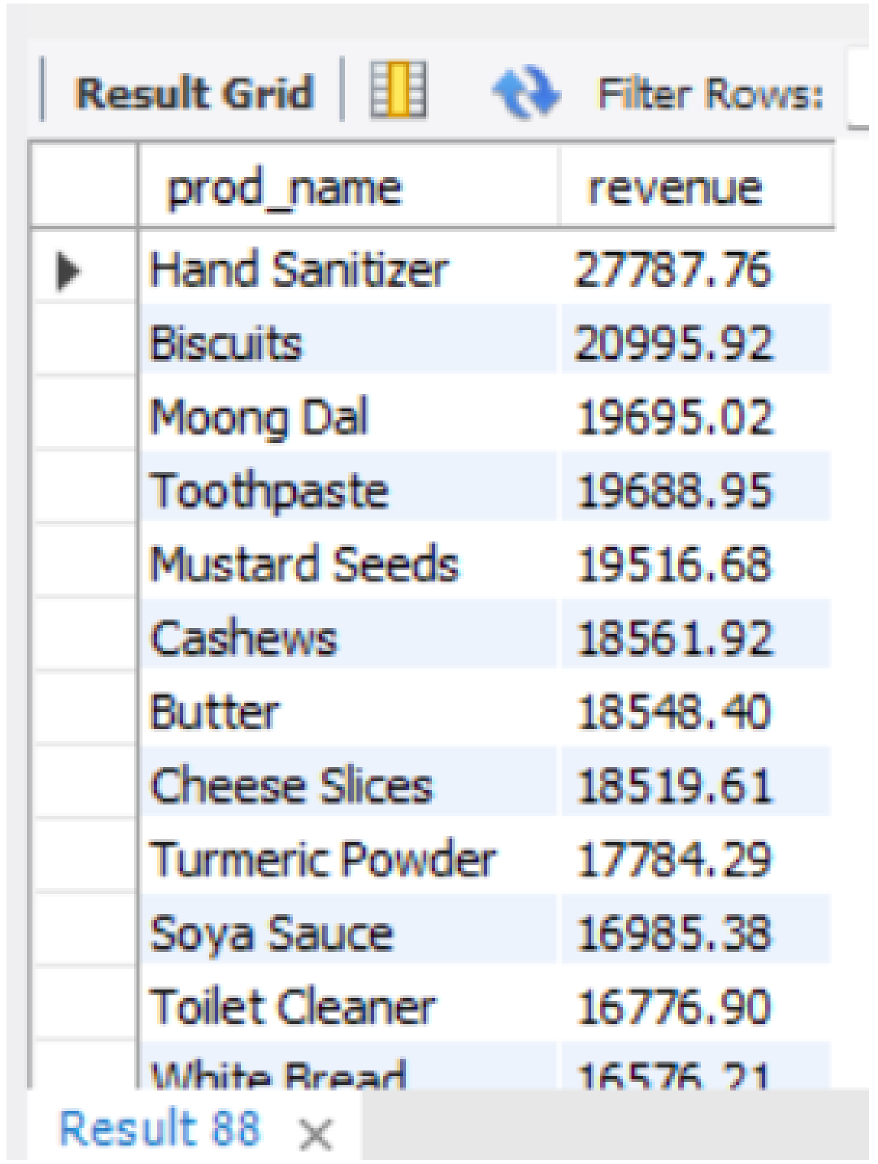
Result Grid | Filter Rows: | Export:

cust_id	cust_name	total_purchase	avg_purchase
1	Aditi Shetty	1577.86	1577.860000
2	Isha Reddy	1299.62	649.810000
3	Chetan Rao	7693.41	854.823333
5	Isha Rao	3327.05	1663.525000
7	Eshwar Iyer	9188.45	656.317857
8	Deepa Reddy	7929.13	881.014444
13	Bala Krishnan	1161.66	580.830000

# Customers with Total Purchases, Avg Purchases

Product Performance

```sql
-- 4. Total revenue generated by each product
SELECT p.prod_name, SUM(od.total_price) AS revenue
FROM products p
JOIN order_details od ON p.prod_id = od.prod_id
GROUP BY p.prod_id
ORDER BY revenue DESC;
```



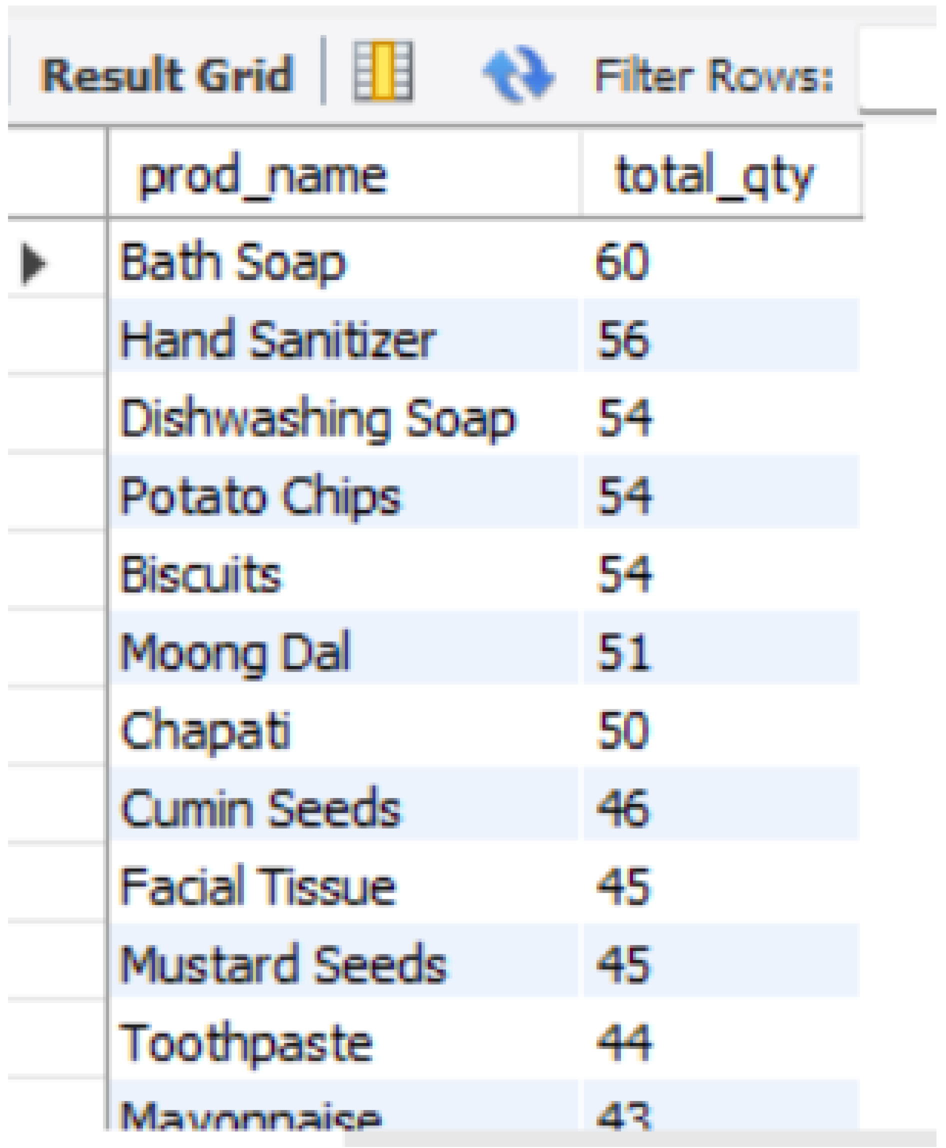
Result Grid | Filter Rows:

prod_name	revenue
Hand Sanitizer	27787.76
Biscuits	20995.92
Moong Dal	19695.02
Toothpaste	19688.95
Mustard Seeds	19516.68
Cashews	18561.92
Butter	18548.40
Cheese Slices	18519.61
Turmeric Powder	17784.29
Soya Sauce	16985.38
Toilet Cleaner	16776.90
White Bread	16576.21

Result 88

# Total Revenye Group by Product

```sql
-- 3. Top products by sales volume
SELECT p.prod_name, SUM(od.quantity) AS total_qty
FROM products p
JOIN order_details od ON p.prod_id = od.prod_id
GROUP BY p.prod_id
ORDER BY total_qty DESC;
```

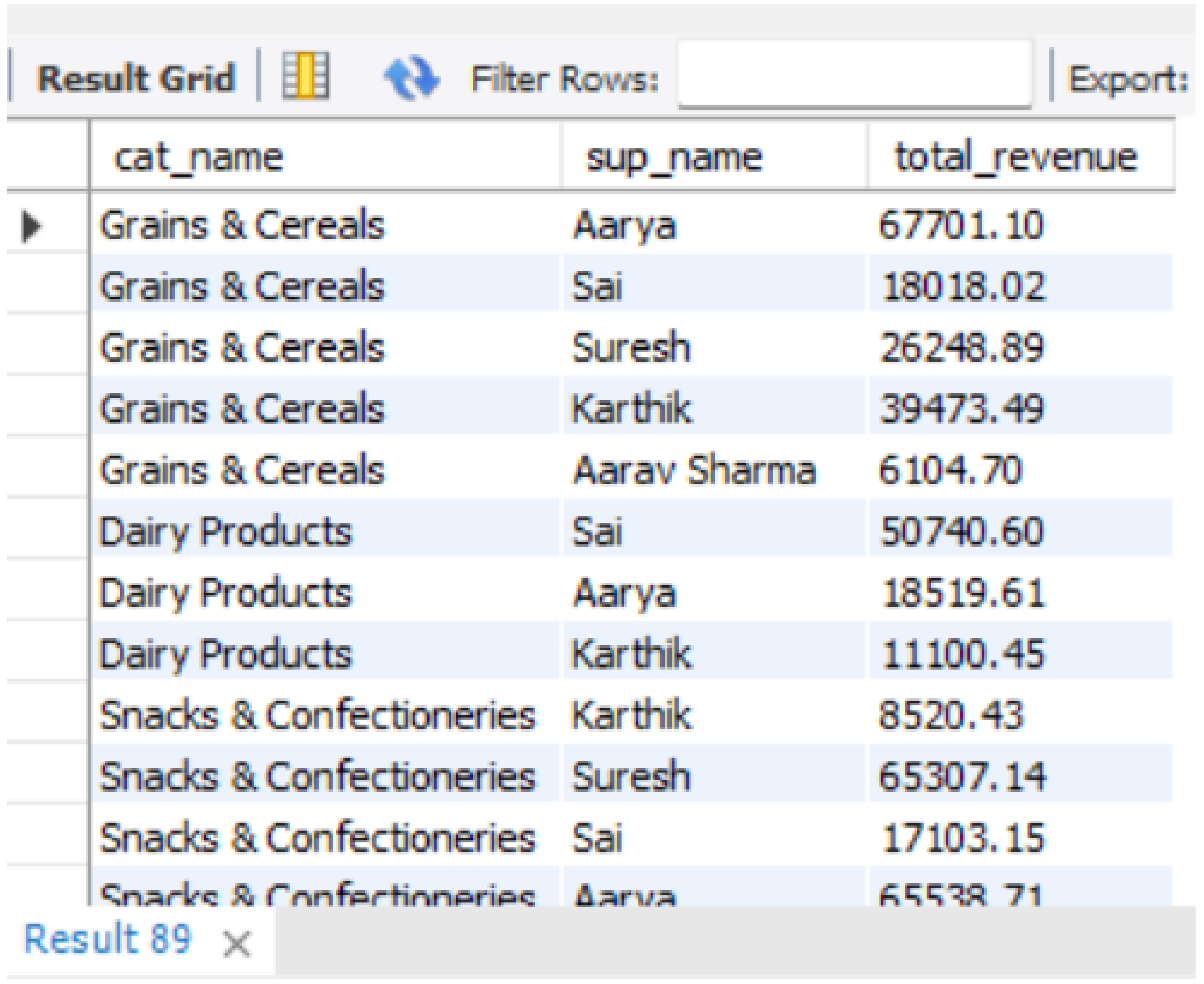


Result Grid | Filter Rows:

prod_name	total_qty
Bath Soap	60
Hand Sanitizer	56
Dishwashing Soap	54
Potato Chips	54
Biscuits	54
Moong Dal	51
Chapati	50
Cumin Seeds	46
Facial Tissue	45
Mustard Seeds	45
Toothpaste	44
Mayonnaise	43

# Total Orders from each product

```sql
-- 5. Sales variation by category & supplier
SELECT cat.cat_name, s.sup_name,
       SUM(od.total_price) AS total_revenue
FROM order_details od
JOIN products p ON od.prod_id = p.prod_id
JOIN categories cat ON p.cat_id = cat.cat_id
JOIN supplier s ON p.sup_id = s.sup_id
GROUP BY cat.cat_id, s.sup_id;
```

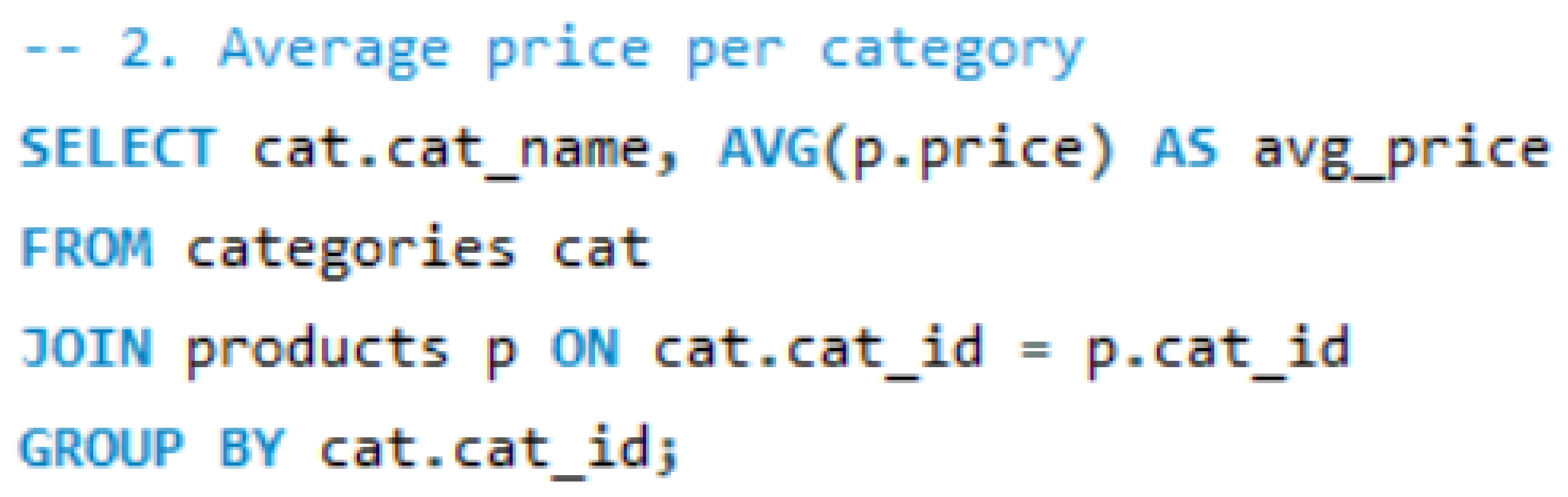


Result Grid | Filter Rows: | Export:

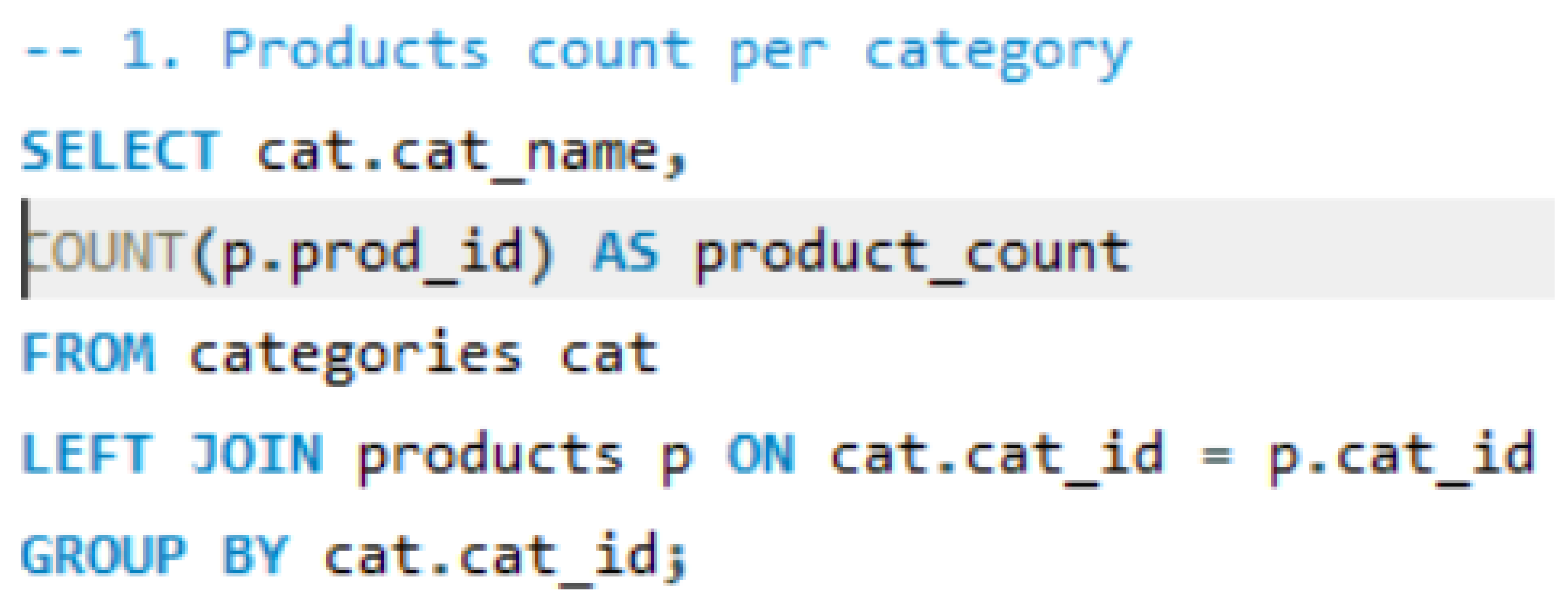
cat_name	sup_name	total_revenue
Grains & Cereals	Aarya	67701.10
Grains & Cereals	Sai	18018.02
Grains & Cereals	Suresh	26248.89
Grains & Cereals	Karthik	39473.49
Grains & Cereals	Aarav Sharma	6104.70
Dairy Products	Sai	50740.60
Dairy Products	Aarya	18519.61
Dairy Products	Karthik	11100.45
Snacks & Confectioneries	Karthik	8520.43
Snacks & Confectioneries	Suresh	65307.14
Snacks & Confectioneries	Sai	17103.15
Snacks & Confectioneries	Aarya	65538.71

Result 89

# Total Revenue from each Categorie and supplier

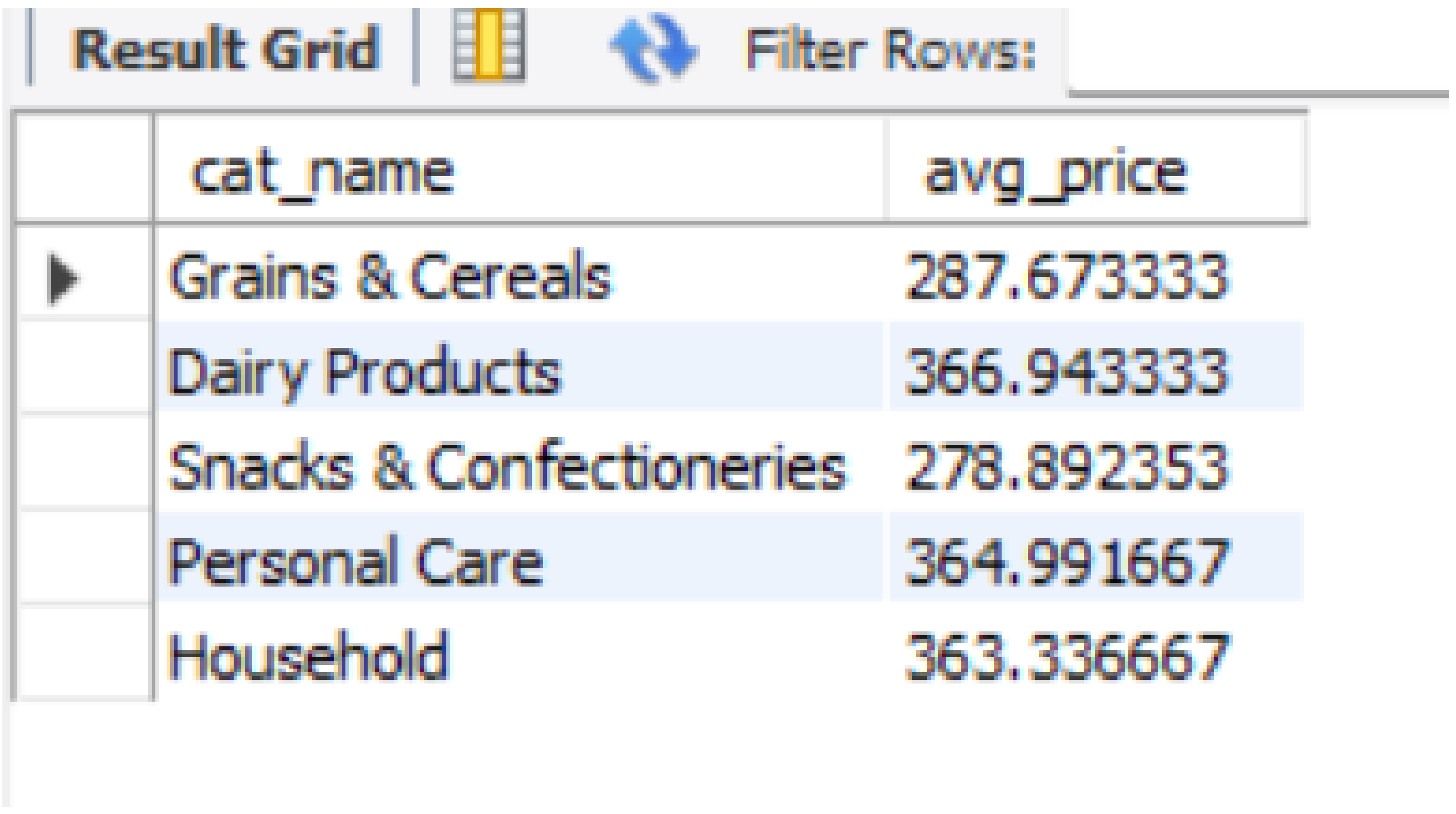


```sql
-- 2. Average price per category
SELECT cat.cat_name, AVG(p.price) AS avg_price
FROM categories cat
JOIN products p ON cat.cat_id = p.cat_id
GROUP BY cat.cat_id;
```



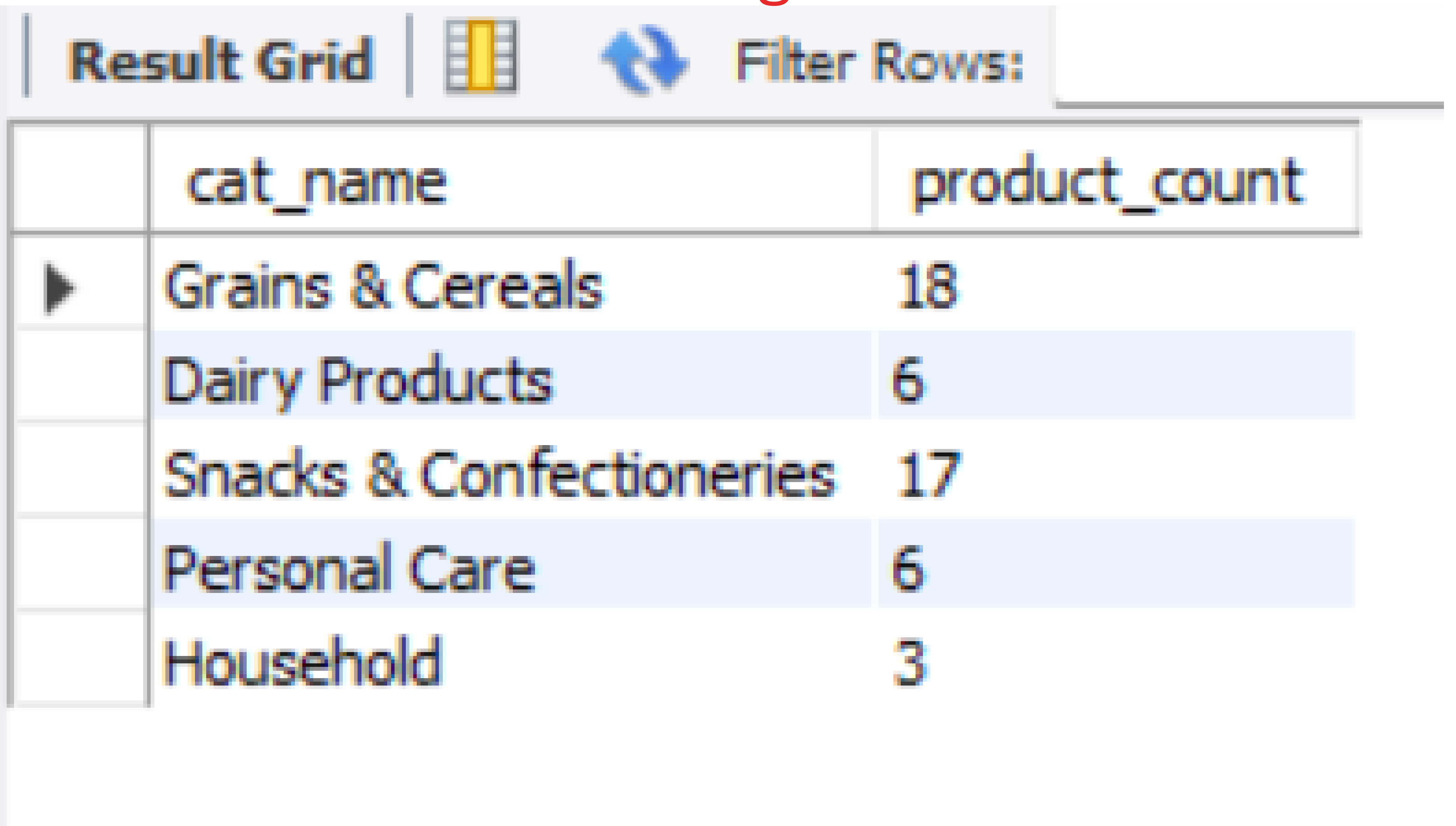
```sql
-- 1. Products count per category
SELECT cat.cat_name,
COUNT(p.prod_id) AS product_count
FROM categories cat
LEFT JOIN products p ON cat.cat_id = p.cat_id
GROUP BY cat.cat_id;
```

# The Quantity of Products in each Categorie



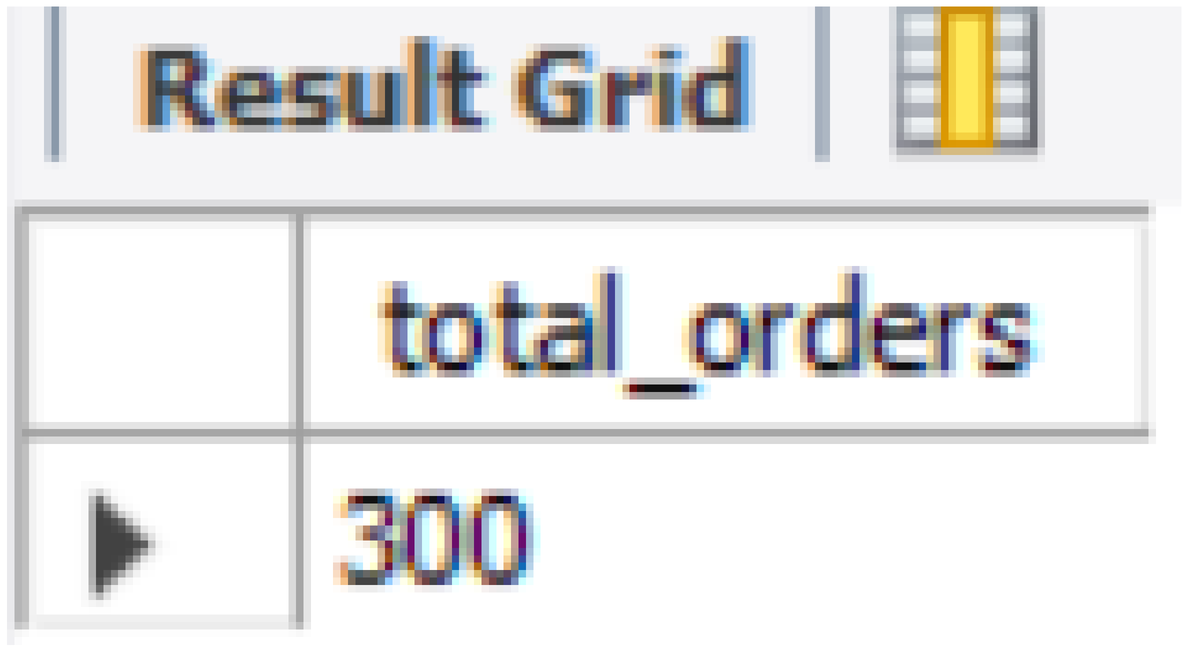
cat_name	avg_price
Grains & Cereals	287.673333
Dairy Products	366.943333
Snacks & Confectioneries	278.892353
Personal Care	364.991667
Household	363.336667

# Average Price from Each Categorie



cat_name	product_count
Grains & Cereals	18
Dairy Products	6
Snacks & Confectioneries	17
Personal Care	6
Household	3

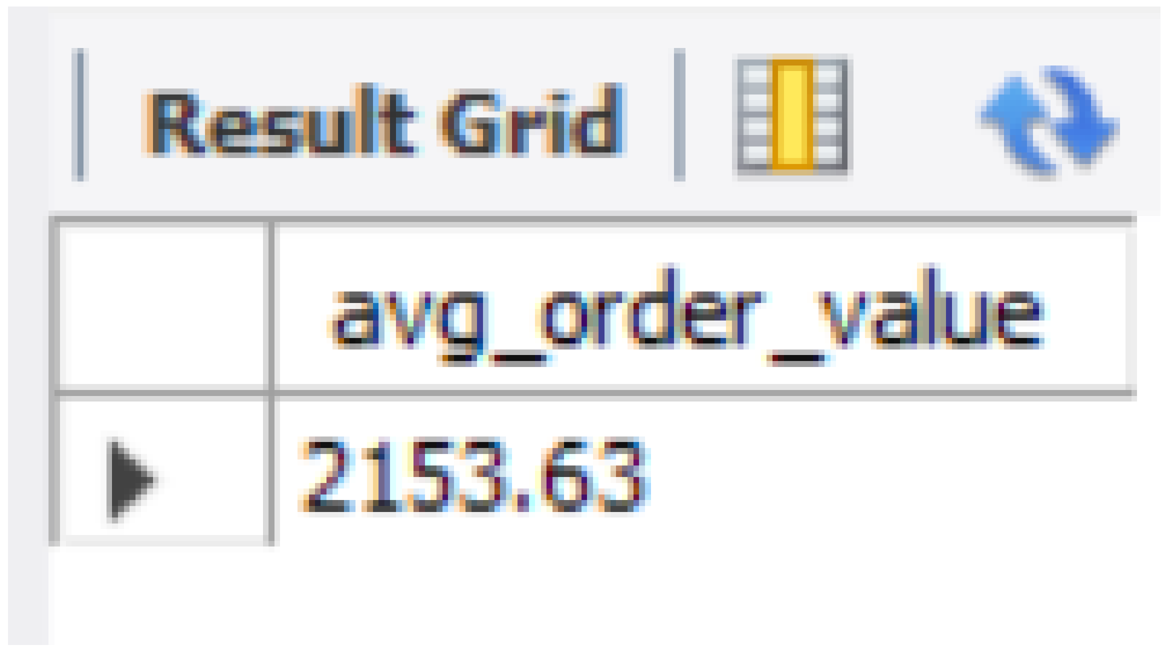
Sales and Order Trends



Result Grid

	total_orders
▶	300

# Total Orders out of 1 year

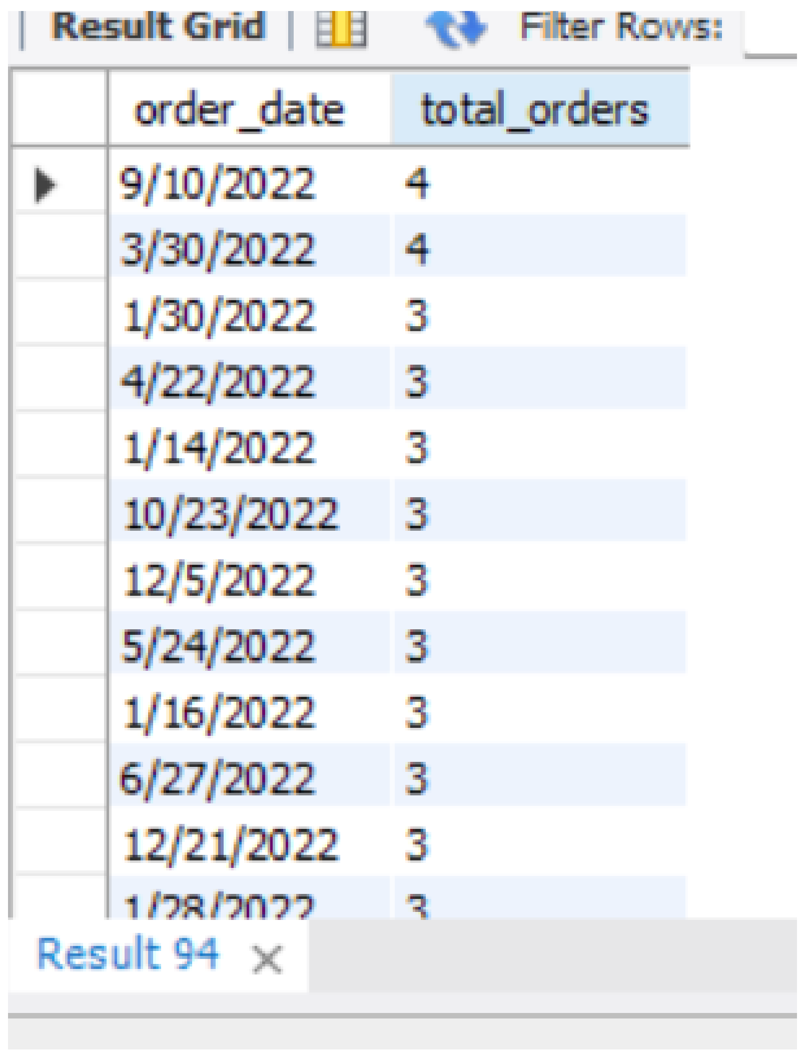


Result Grid

	avg_order_value
▶	2153.63

# Average Order Value

```
-- 3. Dates with most orders
SELECT order_date, COUNT(*) AS total_orders
FROM orders
GROUP BY order_date
ORDER BY total_orders DESC;
```

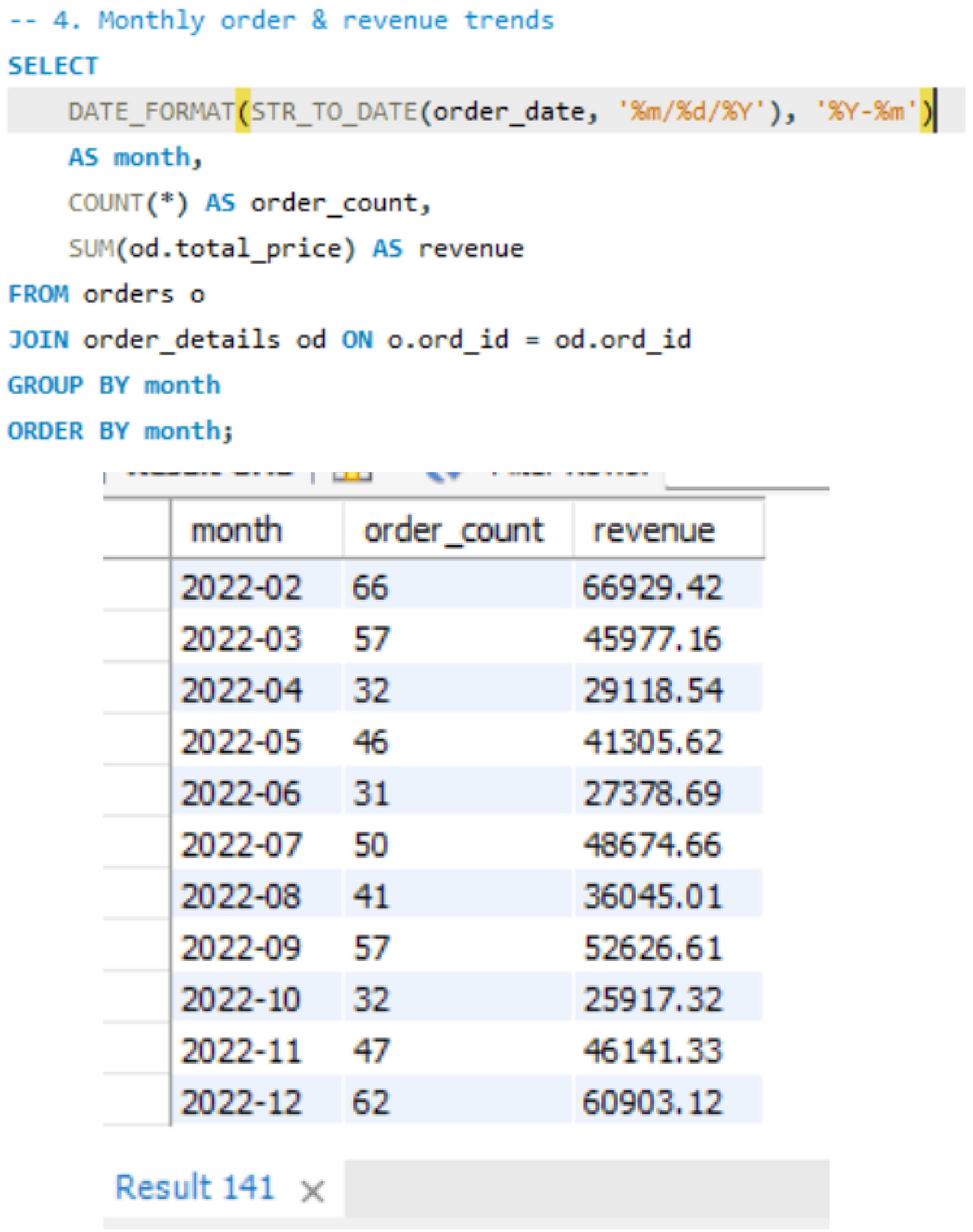


Result Grid | Filter Rows:

	order_date	total_orders
▶	9/10/2022	4
	3/30/2022	4
	1/30/2022	3
	4/22/2022	3
	1/14/2022	3
	10/23/2022	3
	12/5/2022	3
	5/24/2022	3
	1/16/2022	3
	6/27/2022	3
	12/21/2022	3
	1/28/2022	3

Result 94 ×

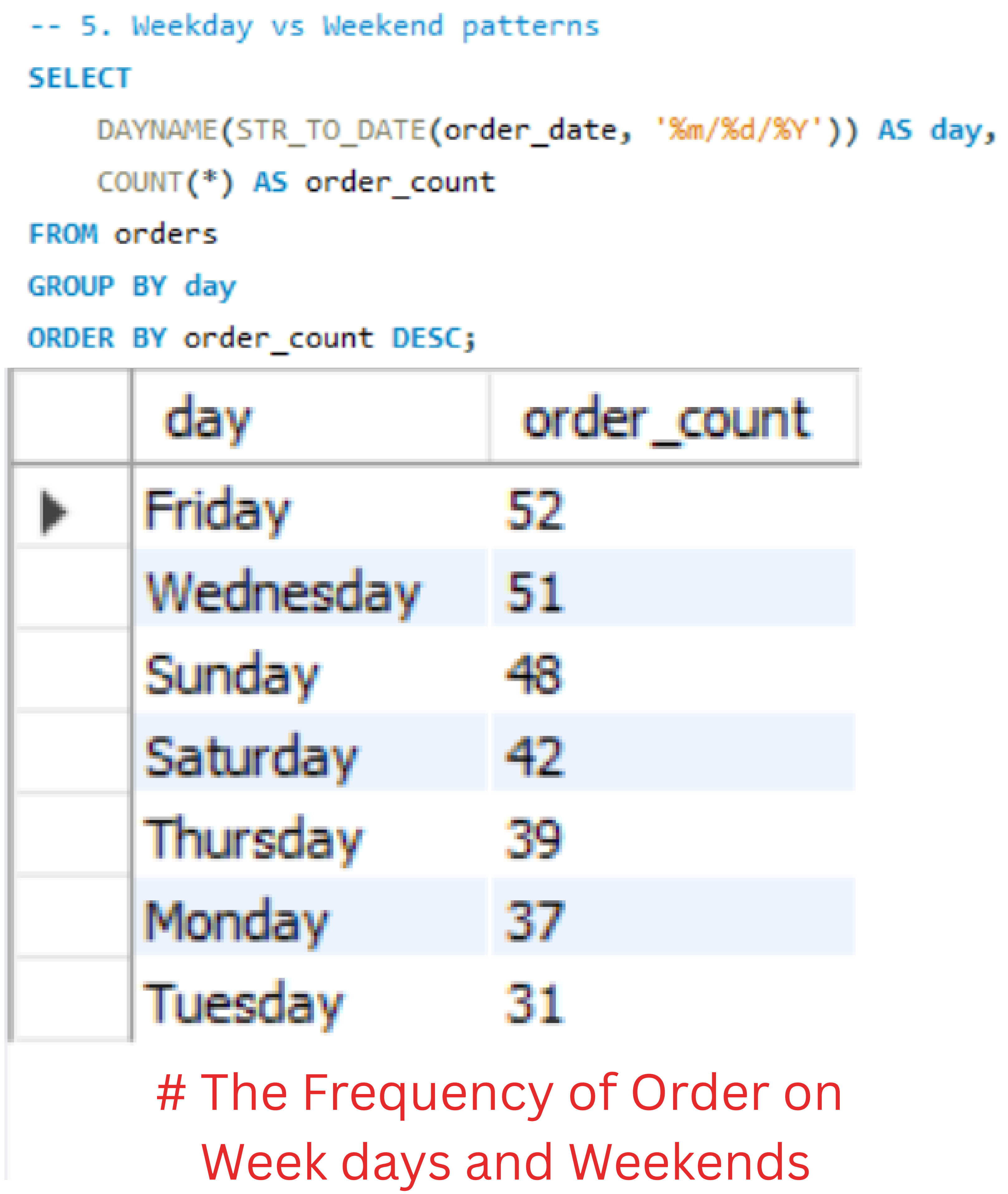
Total Order by
each day ordered by desc



```sql
-- 4. Monthly order & revenue trends
SELECT
    DATE_FORMAT(STR_TO_DATE(order_date, '%m/%d/%Y'), '%Y-%m')
    AS month,
    COUNT(*) AS order_count,
    SUM(od.total_price) AS revenue
FROM orders o
JOIN order_details od ON o.ord_id = od.ord_id
GROUP BY month
ORDER BY month;
```

month	order_count	revenue
2022-02	66	66929.42
2022-03	57	45977.16
2022-04	32	29118.54
2022-05	46	41305.62
2022-06	31	27378.69
2022-07	50	48674.66
2022-08	41	36045.01
2022-09	57	52626.61
2022-10	32	25917.32
2022-11	47	46141.33
2022-12	62	60903.12

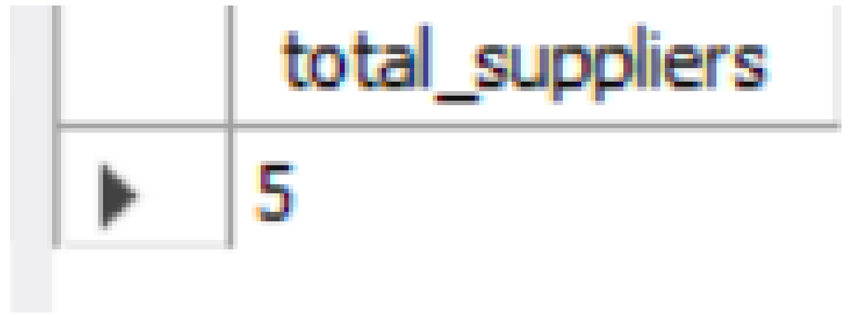
# Revenue generated in each month in the year of 2022



```sql
-- 5. Weekday vs Weekend patterns
SELECT
    DAYNAME(STR_TO_DATE(order_date, '%m/%d/%Y')) AS day,
    COUNT(*) AS order_count
FROM orders
GROUP BY day
ORDER BY order_count DESC;
```

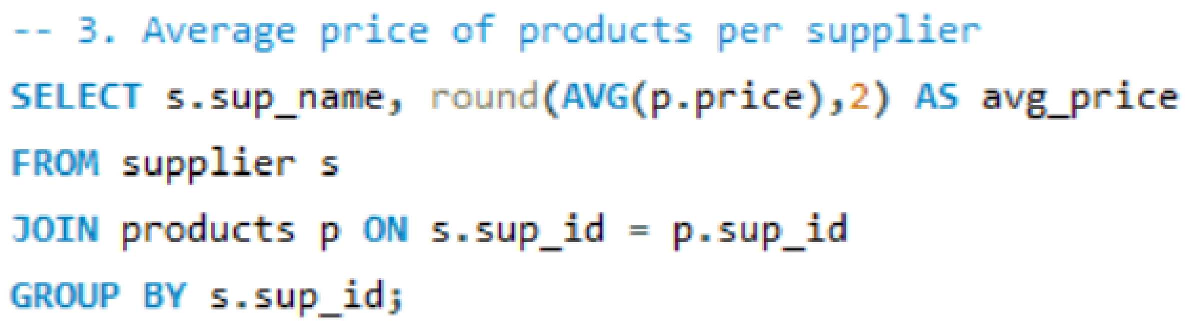
day	order_count
Friday	52
Wednesday	51
Sunday	48
Saturday	42
Thursday	39
Monday	37
Tuesday	31

# The Frequency of Order on Week days and Weekends

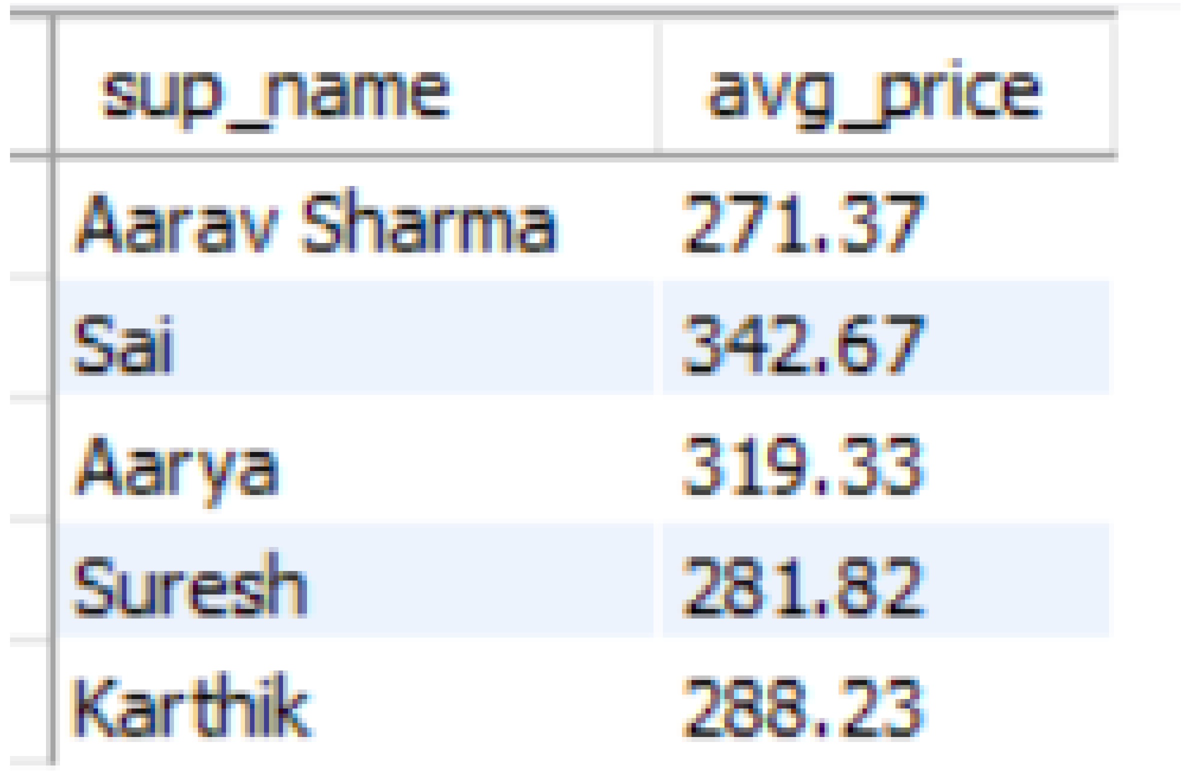


	total_suppliers
▶	5

Supplier Contribution

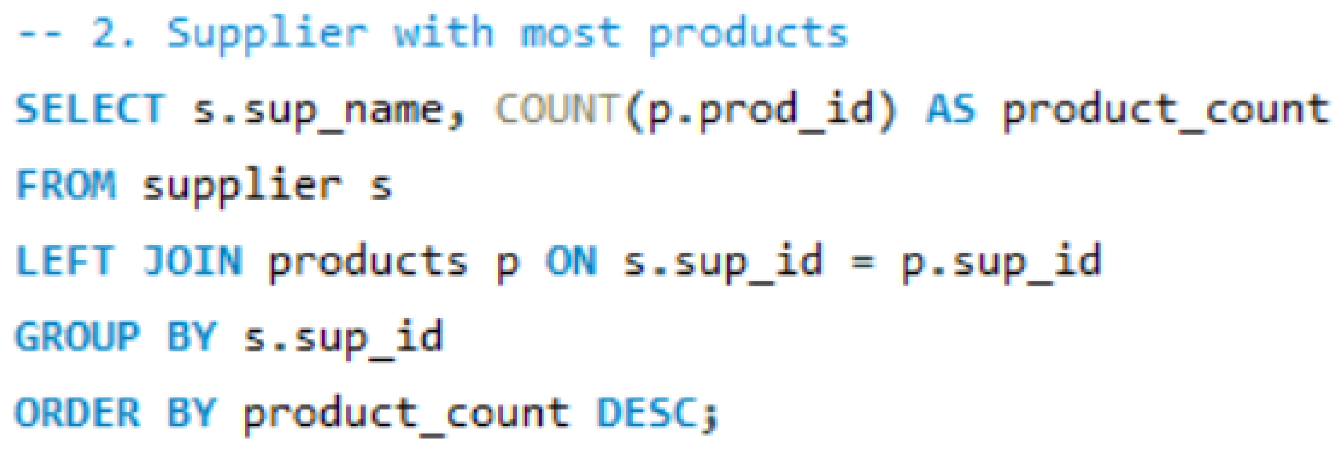


```
-- 3. Average price of products per supplier
SELECT s.sup_name, round(AVG(p.price),2) AS avg_price
FROM supplier s
JOIN products p ON s.sup_id = p.sup_id
GROUP BY s.sup_id;
```

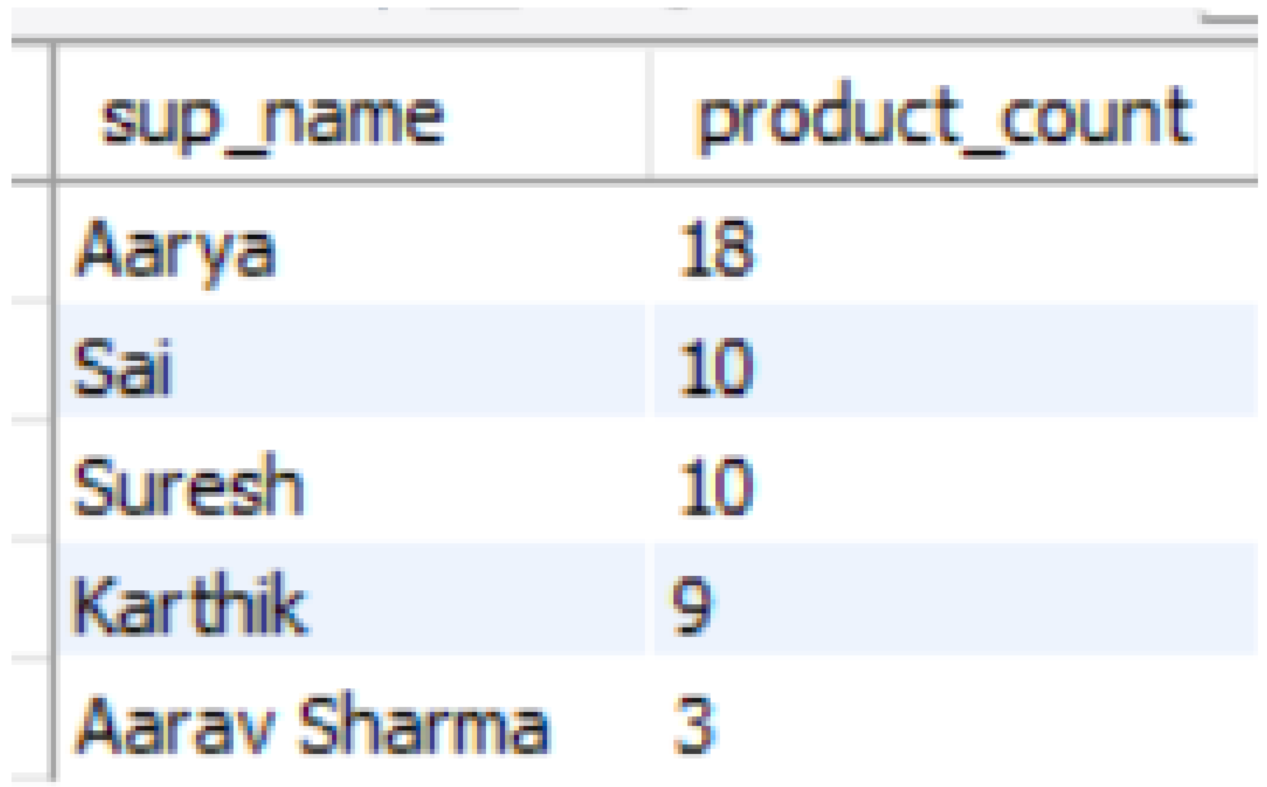


sup_name	avg_price
Aarav Sharma	271.37
Sai	342.67
Aarya	319.33
Suresh	281.82
Karthik	288.23

# Average Price of Each Supllier

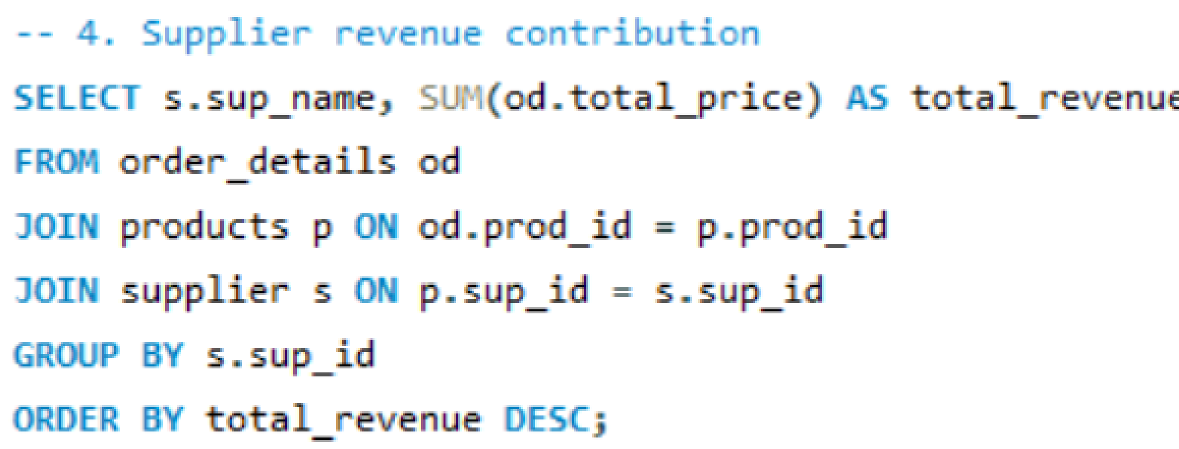


```
-- 2. Supplier with most products
SELECT s.sup_name, COUNT(p.prod_id) AS product_count
FROM supplier s
LEFT JOIN products p ON s.sup_id = p.sup_id
GROUP BY s.sup_id
ORDER BY product_count DESC;
```

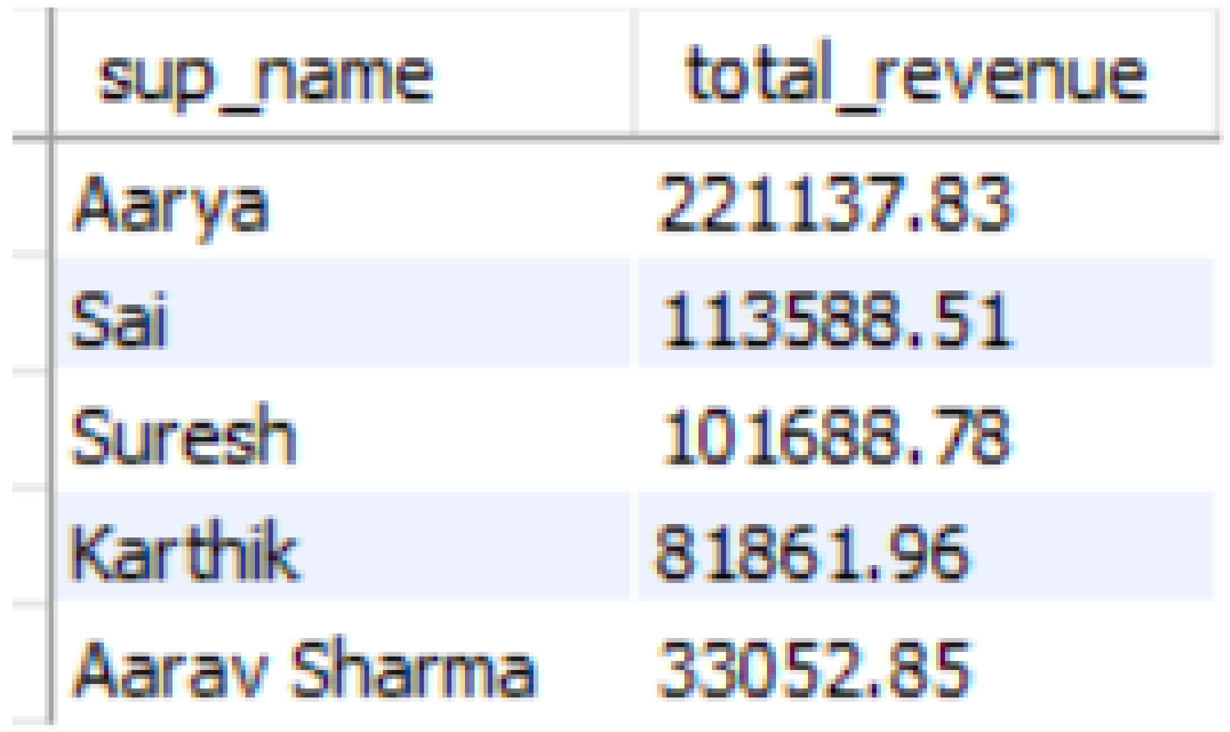


sup_name	product_count
Aarya	18
Sai	10
Suresh	10
Karthik	9
Aarav Sharma	3

# Frequency of Product by Each Supllier



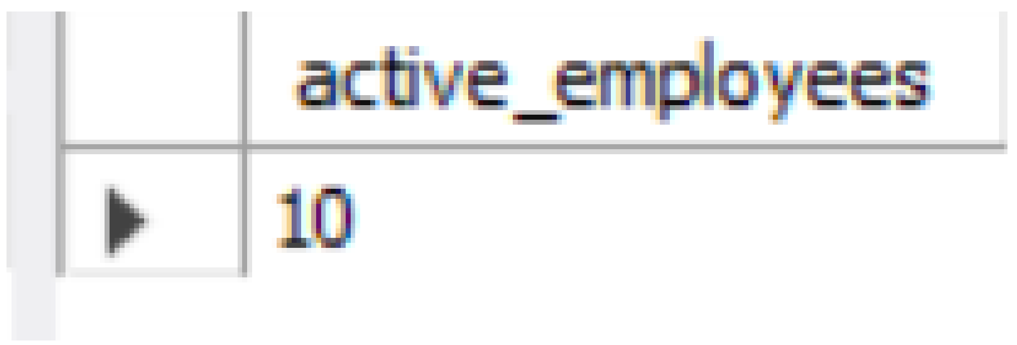
```
-- 4. Supplier revenue contribution
SELECT s.sup_name, SUM(od.total_price) AS total_revenue
FROM order_details od
JOIN products p ON od.prod_id = p.prod_id
JOIN supplier s ON p.sup_id = s.sup_id
GROUP BY s.sup_id
ORDER BY total_revenue DESC;
```



sup_name	total_revenue
Aarya	221137.83
Sai	113588.51
Suresh	101688.78
Karthik	81861.96
Aarav Sharma	33052.85

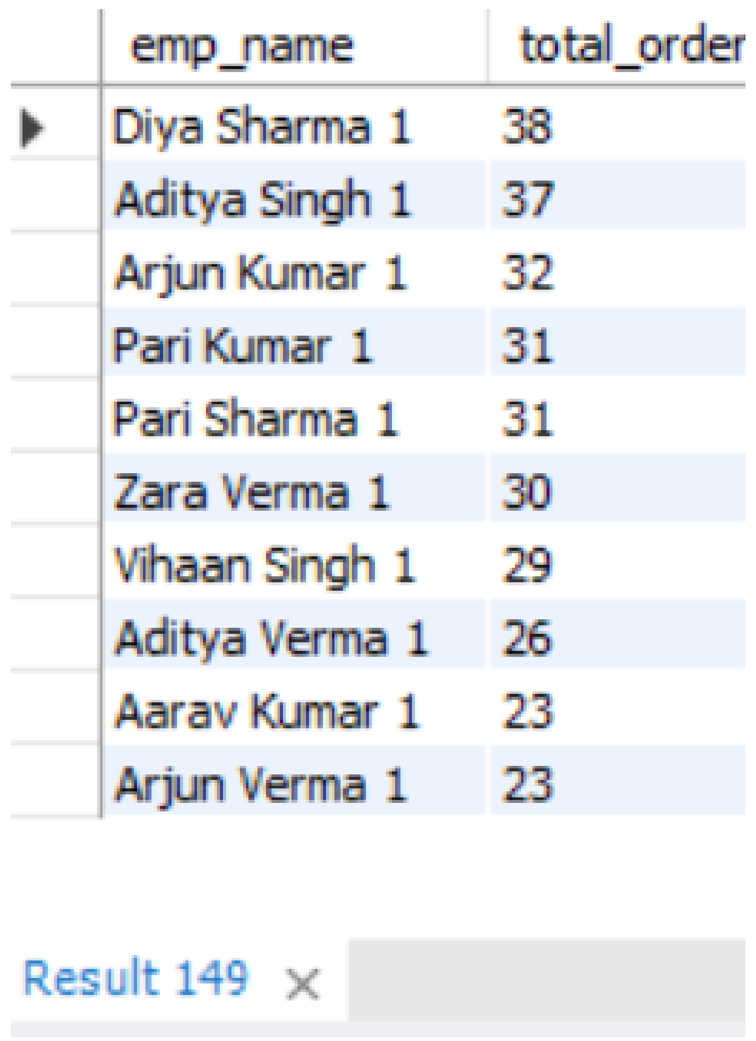
# Total Revenue Generated by Each Supplier

Employee Performance



	active_employees
▶	10

```
-- 2. Employee with most orders
SELECT e.emp_name, COUNT(o.ord_id) AS total_orders
FROM employees e
JOIN orders o ON e.emp_id = o.emp_id
GROUP BY e.emp_id
ORDER BY total_orders DESC;
```

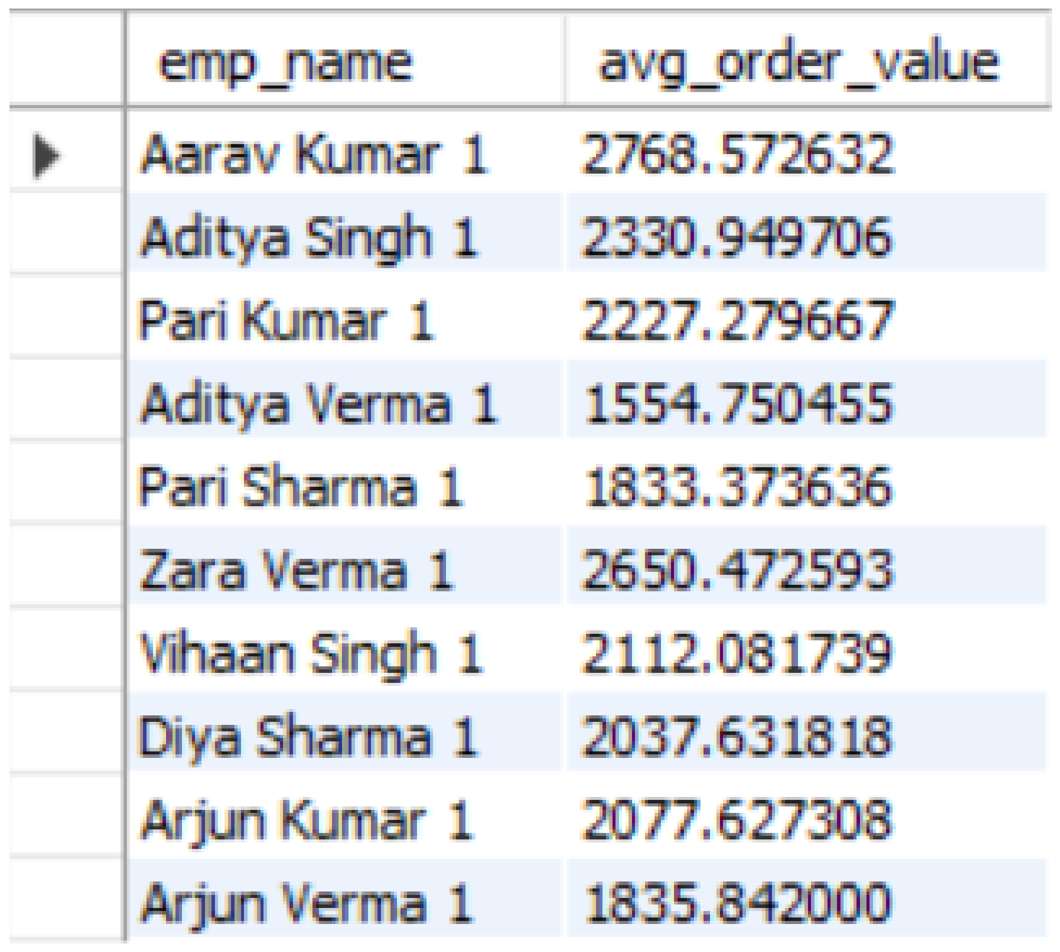


	emp_name	total_order
▶	Diya Sharma 1	38
	Aditya Singh 1	37
	Arjun Kumar 1	32
	Pari Kumar 1	31
	Pari Sharma 1	31
	Zara Verma 1	30
	Vihaan Singh 1	29
	Aditya Verma 1	26
	Aarav Kumar 1	23
	Arjun Verma 1	23

Result 149 ×

# Total Orders by Each Employee Handlied

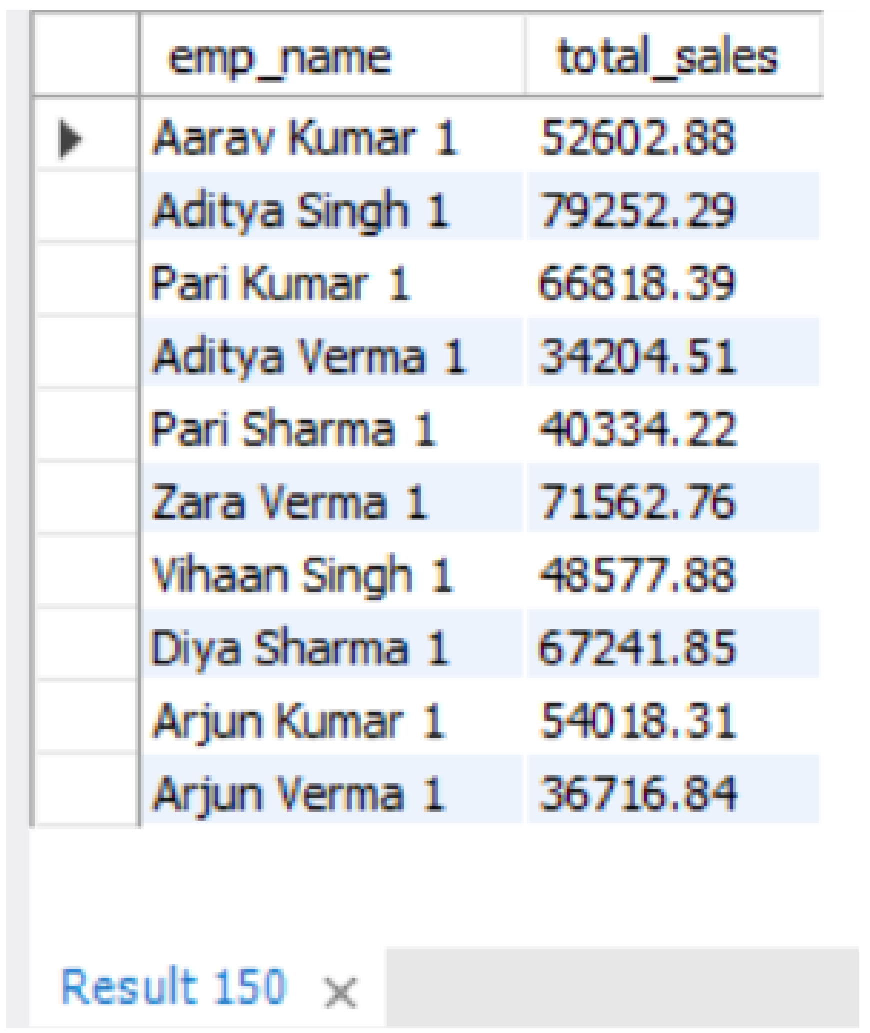
```
-- 4. Average order value per employee
SELECT e.emp_name, AVG(t.order_total) AS avg_order_value
FROM (
    SELECT o.emp_id, o.ord_id, SUM(od.total_price) AS order_total
    FROM orders o
    JOIN order_details od ON o.ord_id = od.ord_id
    GROUP BY o.ord_id
) t
JOIN employees e ON t.emp_id = e.emp_id
GROUP BY e.emp_id;
```



	emp_name	avg_order_value
▶	Aarav Kumar 1	2768.572632
	Aditya Singh 1	2330.949706
	Pari Kumar 1	2227.279667
	Aditya Verma 1	1554.750455
	Pari Sharma 1	1833.373636
	Zara Verma 1	2650.472593
	Vihaan Singh 1	2112.081739
	Diya Sharma 1	2037.631818
	Arjun Kumar 1	2077.627308
	Arjun Verma 1	1835.842000

# Average Order Value By Employee

```
-- 3. Total sales handled by each employee
SELECT e.emp_name, SUM(od.total_price) AS total_sale
FROM employees e
JOIN orders o ON e.emp_id = o.emp_id
JOIN order_details od ON o.ord_id = od.ord_id
GROUP BY e.emp_id;
```



	emp_name	total_sales
▶	Aarav Kumar 1	52602.88
	Aditya Singh 1	79252.29
	Pari Kumar 1	66818.39
	Aditya Verma 1	34204.51
	Pari Sharma 1	40334.22
	Zara Verma 1	71562.76
	Vihaan Singh 1	48577.88
	Diya Sharma 1	67241.85
	Arjun Kumar 1	54018.31
	Arjun Verma 1	36716.84

Result 150 ×

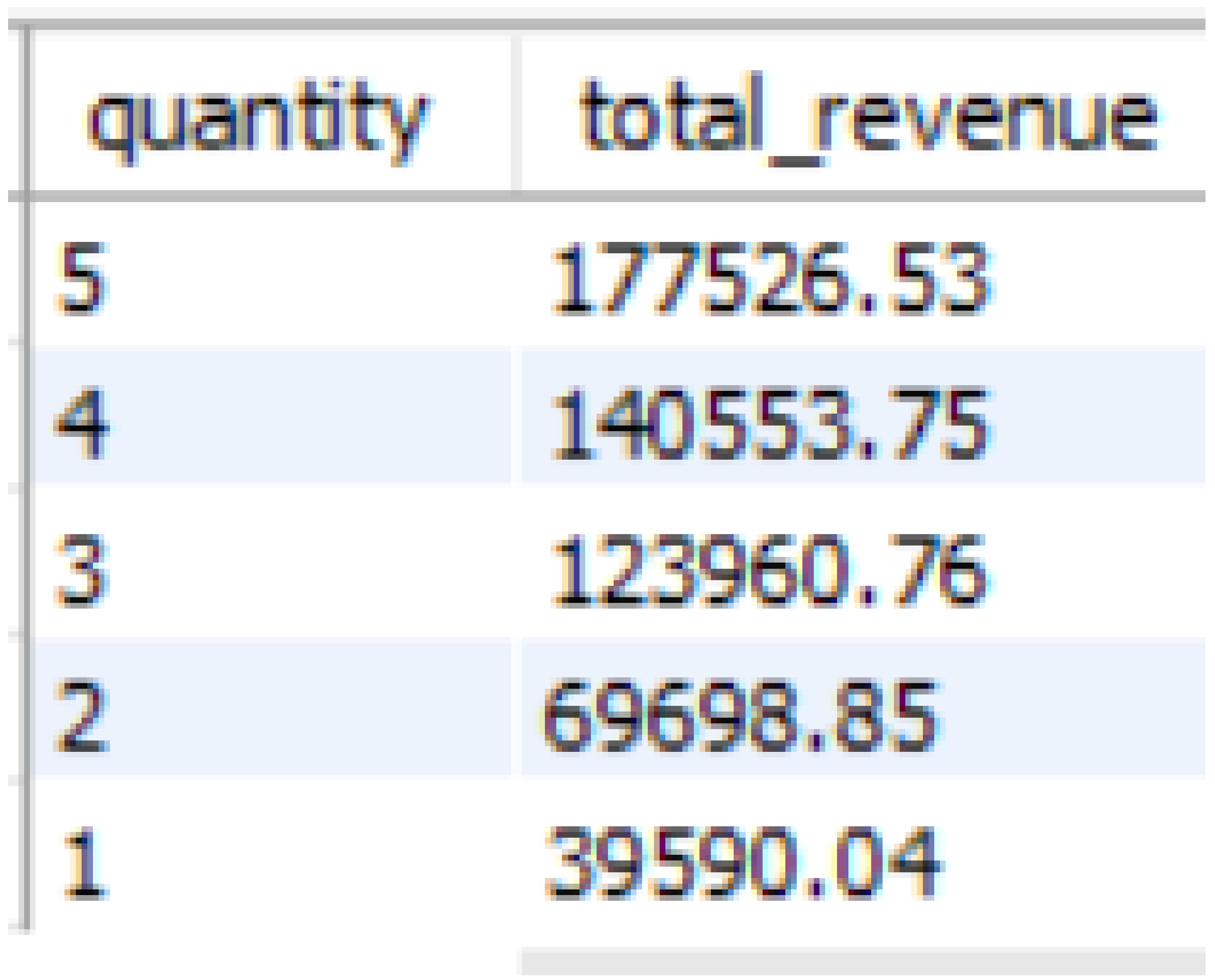
#Toal Sales Generated by Each Employee

Order Details

```sql
SELECT
    quantity,
    SUM(total_price) AS total_revenue
FROM order_details
GROUP BY quantity
ORDER BY total_revenue desc;
```

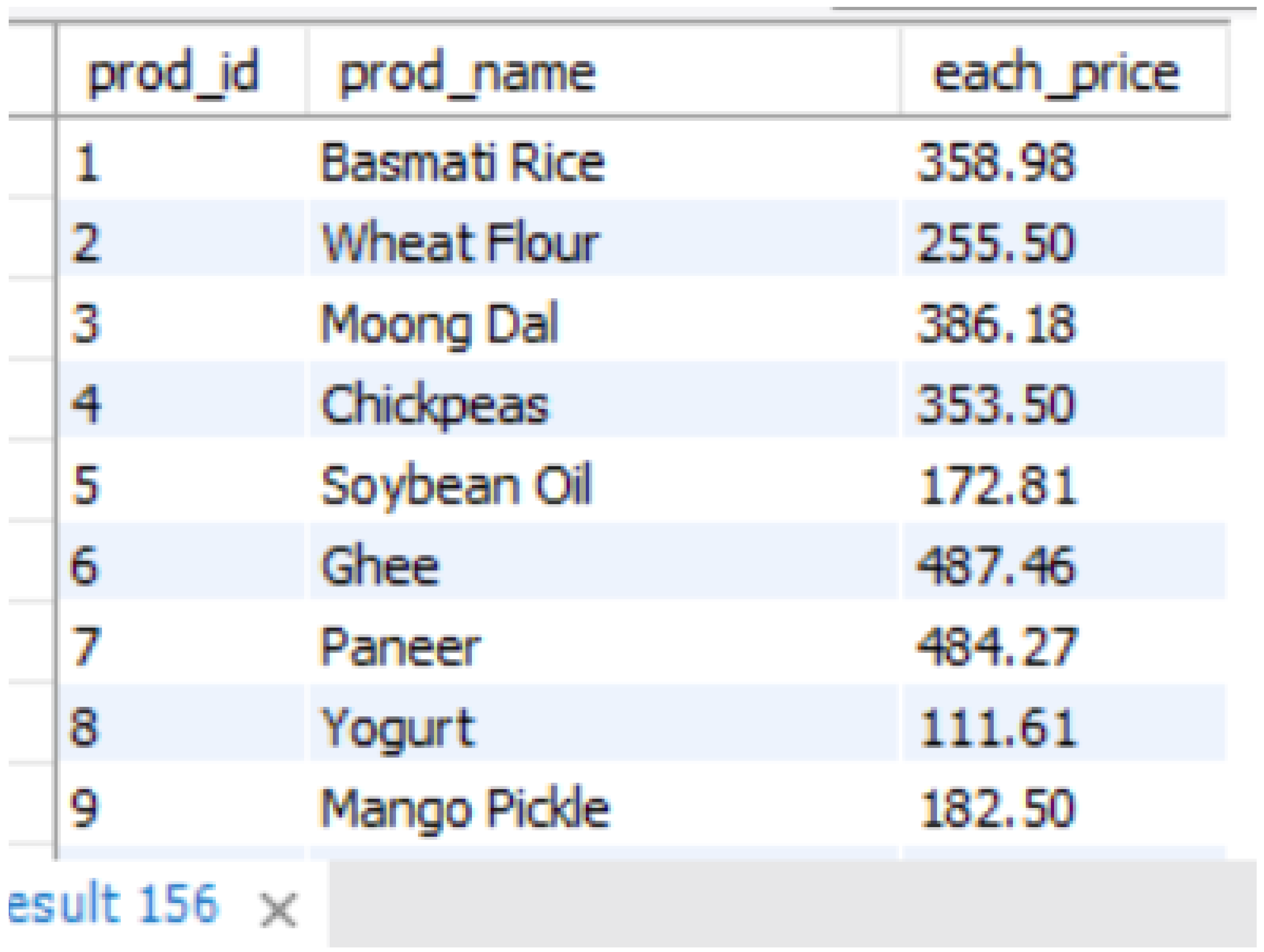
```sql
-- 3. Unit price variation across orders
SELECT DISTINCT p.prod_id, p.prod_name, od.each_price
FROM products p
JOIN order_details od ON p.prod_id = od.prod_id
ORDER BY p.prod_id;
```

```sql
-- 2. Average quantity ordered per product
SELECT p.prod_name, AVG(od.quantity) AS avg_qty
FROM products p
JOIN order_details od ON p.prod_id = od.prod_id
GROUP BY p.prod_id;
```



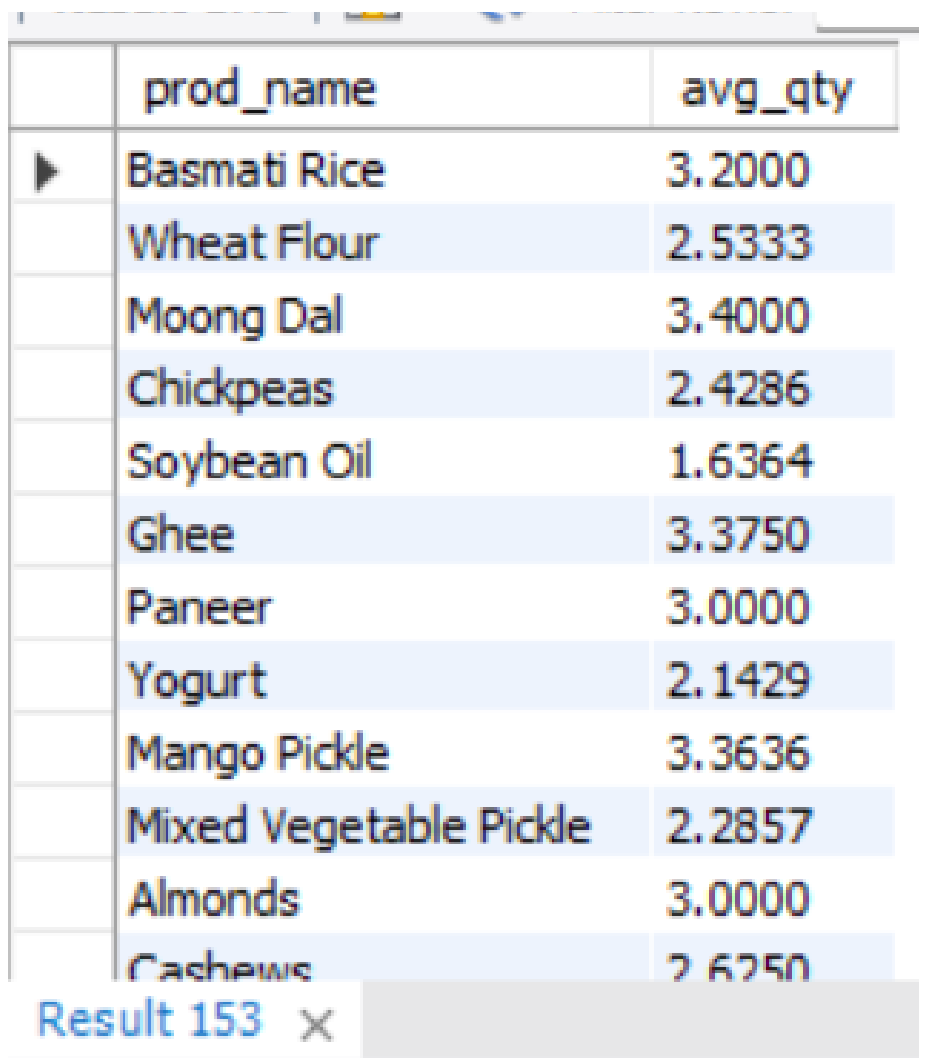
quantity	total_revenue
5	177526.53
4	140553.75
3	123960.76
2	69698.85
1	39590.04

# Total Revenue out Quantity Frequencies



prod_id	prod_name	each_price
1	Basmati Rice	358.98
2	Wheat Flour	255.50
3	Moong Dal	386.18
4	Chickpeas	353.50
5	Soybean Oil	172.81
6	Ghee	487.46
7	Paneer	484.27
8	Yogurt	111.61
9	Mango Pickle	182.50

# Each Product and their Prices



prod_name	avg_qty
Basmati Rice	3.2000
Wheat Flour	2.5333
Moong Dal	3.4000
Chickpeas	2.4286
Soybean Oil	1.6364
Ghee	3.3750
Paneer	3.0000
Yogurt	2.1429
Mango Pickle	3.3636
Mixed Vegetable Pickle	2.2857
Almonds	3.0000
Cashews	2.6250

# Average Quantity of each Product

Challenges

- Understanding how tables are connected using primary and foreign keys.
- Handling data imports where some tables failed due to mismatched column names or invalid foreign key values.
- Managing date fields stored as text, which required conversion before doing time-based analysis.
- Writing correct JOIN queries to combine multiple tables for insights.
- Fixing errors with DISTINCT, ORDER BY, and aggregation rules.
- Ensuring referential integrity while loading and analysing data.
- Cleaning inconsistent values like prices, IDs, and date formats to avoid query errors.

Conclusion

- Demonstrated how structured SQL analysis can reveal clear insights into sales trends, product performance, and customer behaviour.
- Strengthened understanding of database design, relationships, and efficient query writing for real-world datasets.
- Showed the value of clean, well-connected data in supporting accurate reporting and better business decisions.

Thank You