Exploratory Data Analysis (EDA) Project

Sales - MySQL Analysis

Project Overview

- Objective:
- Analyze customer behavior, sales trends, and product performance.

- Datasets:
- - Category
- - Order_Details.csv
- - Orders
- - Users

Database Design

Schema Definition:

Category: CategoryID, CategoryName, Description

Order_Details: OrderDetailID, OrderID, ProductID, Quantity, UnitPrice, Total

Orders: OrderID, UserID, OrderDate, OrderStatus

Users: UserID, UserName, Email, RegistrationDate, Location

Relationships:

- Foreign keys link Orders to Users and Order_Details.

Data Exploration - Key Questions

Key Questions:

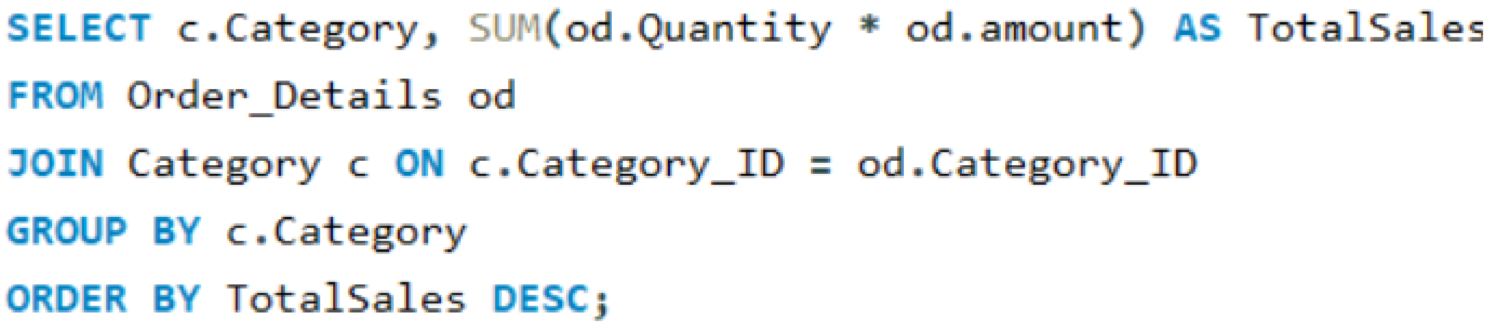
1. Sales Performance:
   - Total sales by category and product.

2. Customer Insights:
   - Active locations and top buyers.

3. Operational Insights:
   - Order status breakdown and trends.

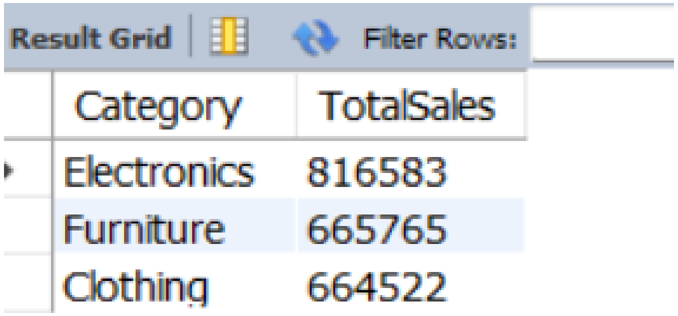
SQL Queries

1. Total Sales by Category:

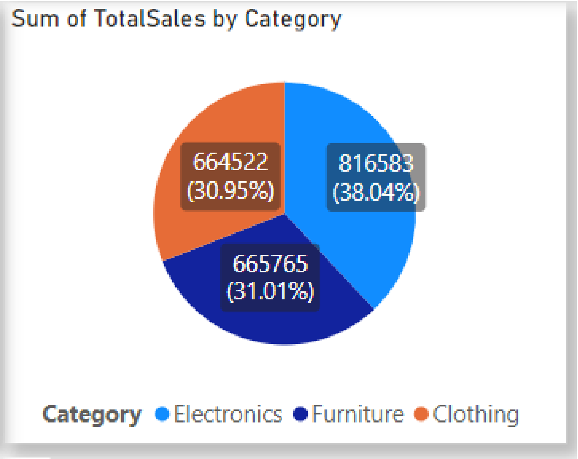


```sql
SELECT c.Category, SUM(od.Quantity * od.amount) AS TotalSales
FROM Order_Details od
JOIN Category c ON c.Category_ID = od.Category_ID
GROUP BY c.Category
ORDER BY TotalSales DESC;
```

Result:



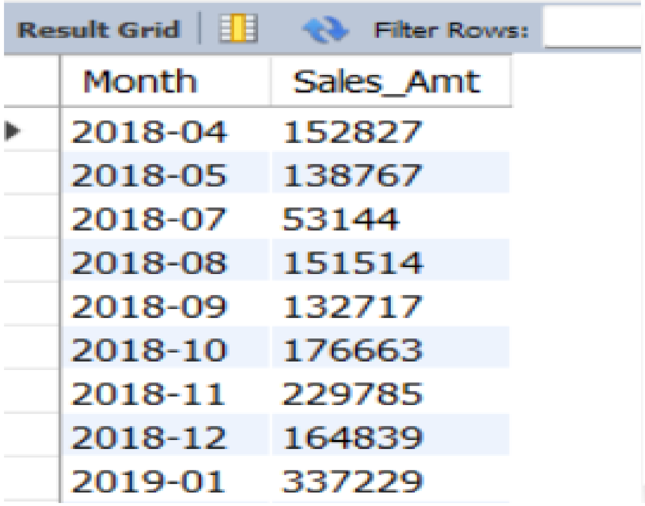
Category	TotalSales
Electronics	816583
Furniture	665765
Clothing	664522



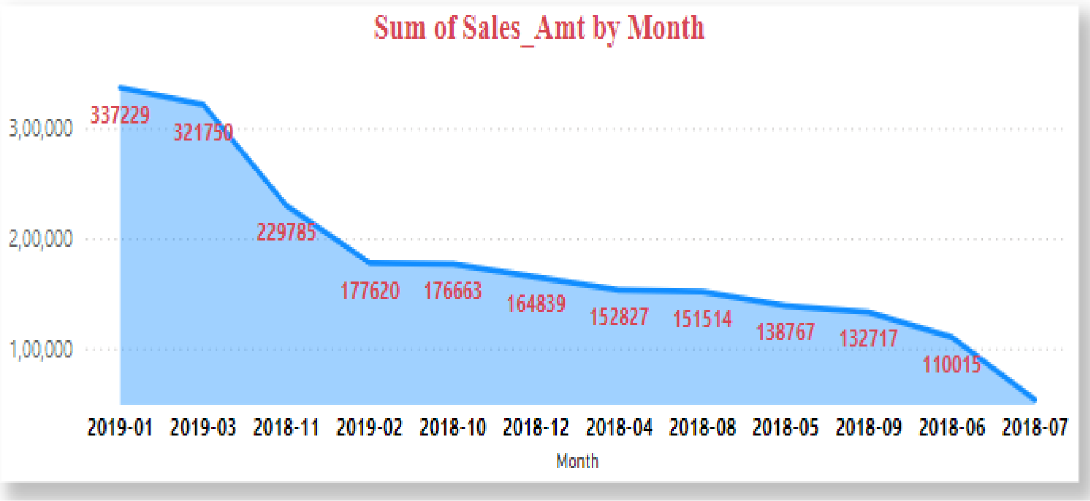
2. Monthly Sales Trends:

```
SELECT
DATE_FORMAT(o.Order_Date, '%Y-%m') AS Month,
SUM(od.Quantity * od.amount) AS Sales_Amt
FROM Order_Details od
JOIN orders o
ON od.order_id = o.order_id
GROUP BY DATE_FORMAT(o.Order_Date, '%Y-%m');
```

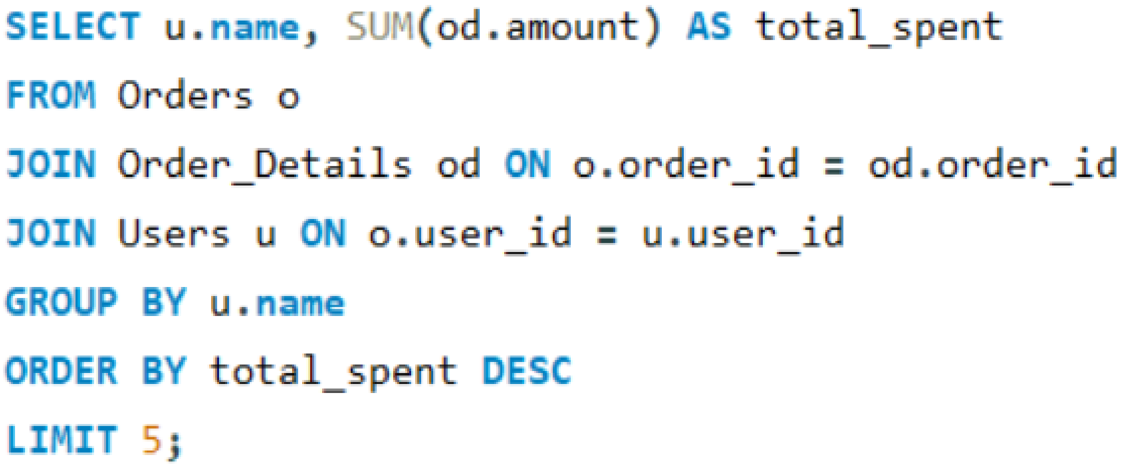
Result:



Month	Sales_Amt
2018-04	152827
2018-05	138767
2018-07	53144
2018-08	151514
2018-09	132717
2018-10	176663
2018-11	229785
2018-12	164839
2019-01	337229

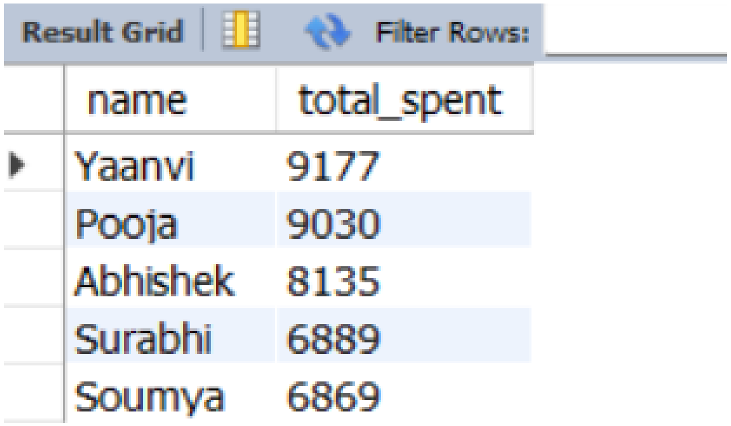


3. Find the top 5 users who have spent the most amount of money.

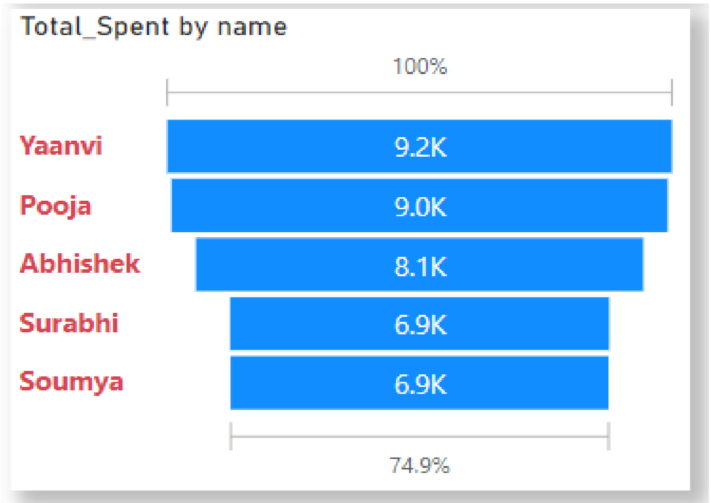


```
SELECT u.name, SUM(od.amount) AS total_spent
FROM Orders o
JOIN Order_Details od ON o.order_id = od.order_id
JOIN Users u ON o.user_id = u.user_id
GROUP BY u.name
ORDER BY total_spent DESC
LIMIT 5;
```

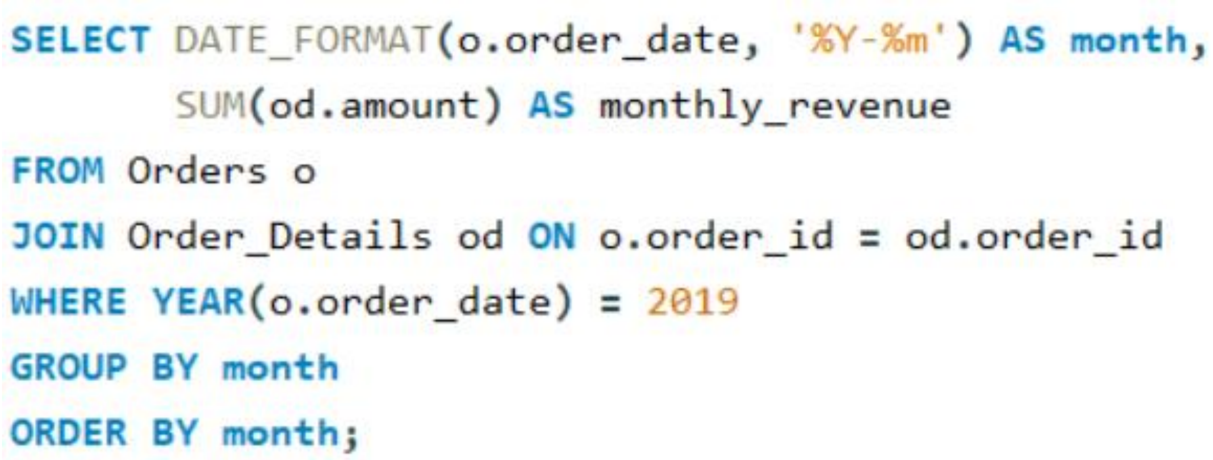
- Result:



name	total_spent
Yaanvi	9177
Pooja	9030
Abhishek	8135
Surabhi	6889
Soumya	6869

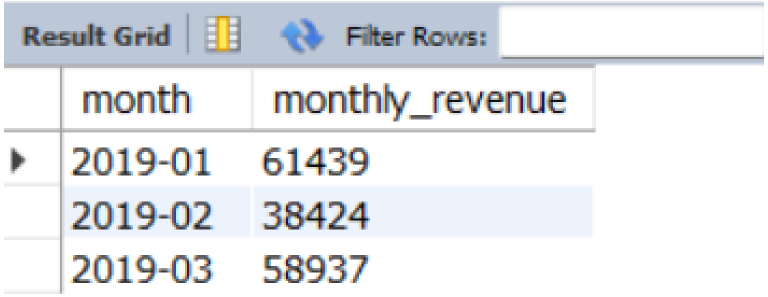


4. Calculate the monthly revenue for the year 2019.

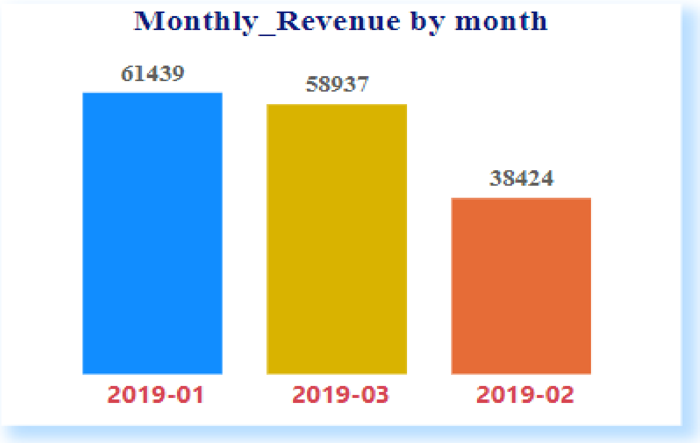


```
SELECT DATE_FORMAT(o.order_date, '%Y-%m') AS month,
       SUM(od.amount) AS monthly_revenue
FROM Orders o
JOIN Order_Details od ON o.order_id = od.order_id
WHERE YEAR(o.order_date) = 2019
GROUP BY month
ORDER BY month;
```

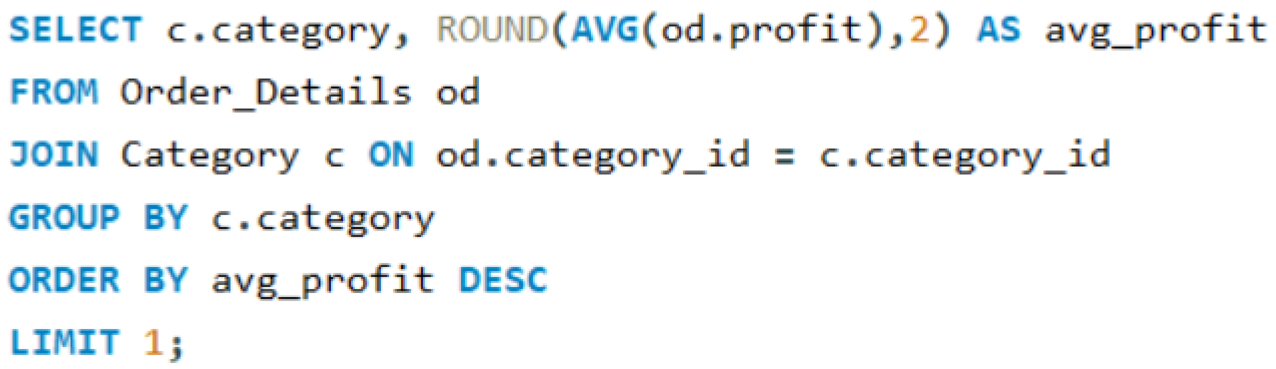
Result:



month	monthly_revenue
2019-01	61439
2019-02	38424
2019-03	58937

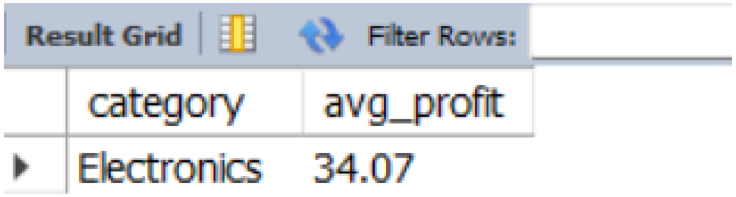


5. Identify the category with the highest average profit per order.



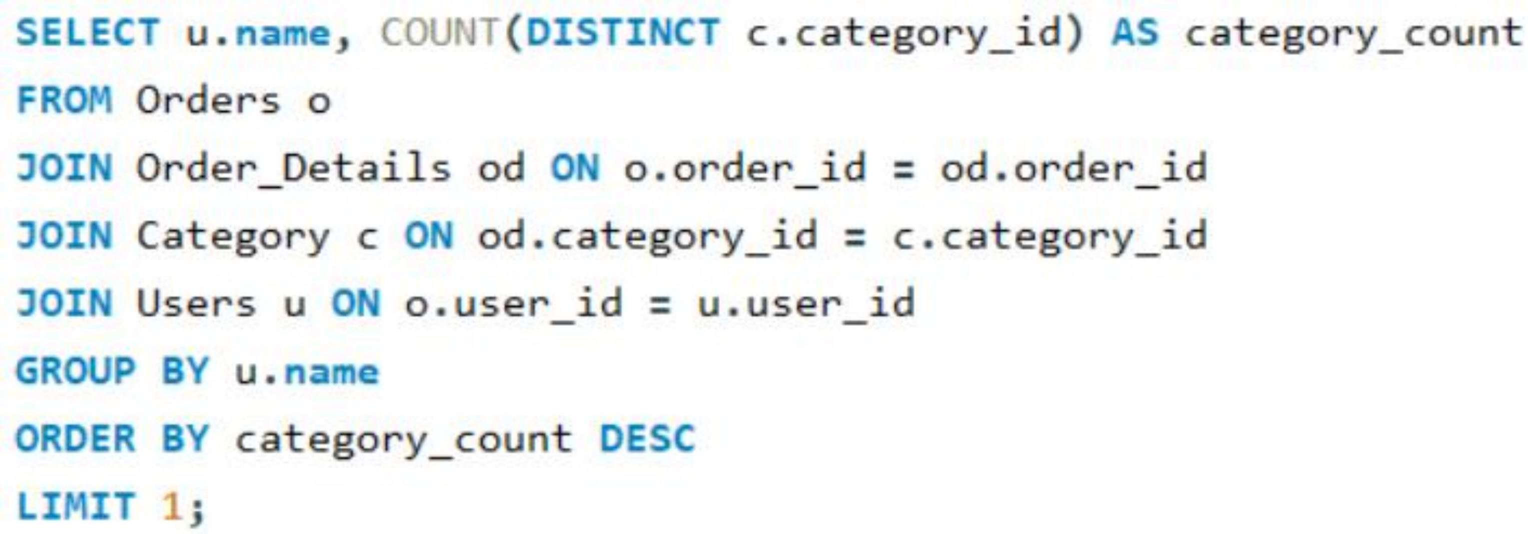
```sql
SELECT c.category, ROUND(AVG(od.profit),2) AS avg_profit
FROM Order_Details od
JOIN Category c ON od.category_id = c.category_id
GROUP BY c.category
ORDER BY avg_profit DESC
LIMIT 1;
```

Result:



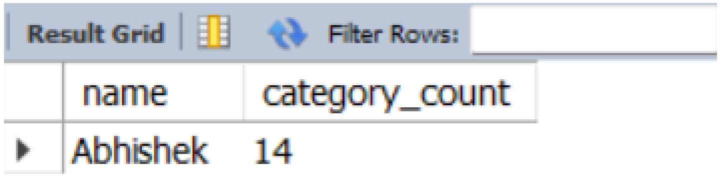
category	avg_profit
Electronics	34.07

6. Find the user who has placed orders in the most number of different categories.



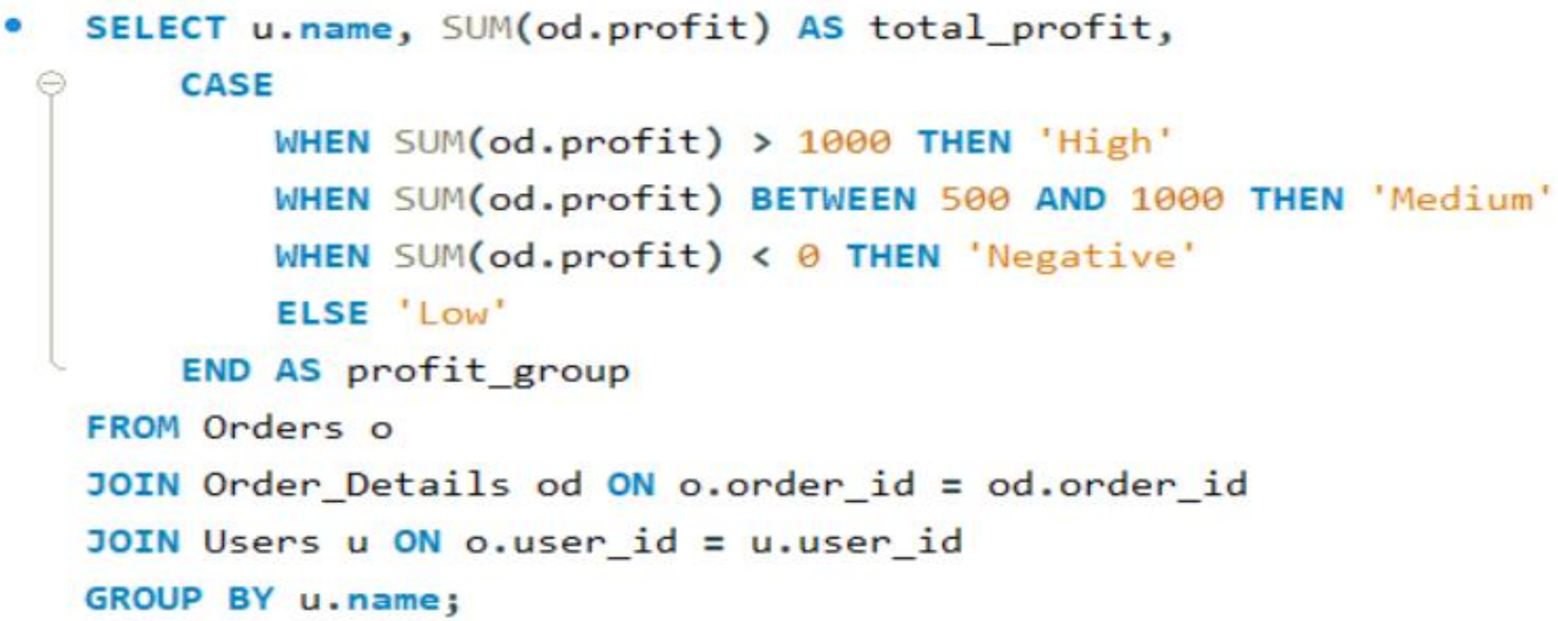
```
SELECT u.name, COUNT(DISTINCT c.category_id) AS category_count
FROM Orders o
JOIN Order_Details od ON o.order_id = od.order_id
JOIN Category c ON od.category_id = c.category_id
JOIN Users u ON o.user_id = u.user_id
GROUP BY u.name
ORDER BY category_count DESC
LIMIT 1;
```

Result:



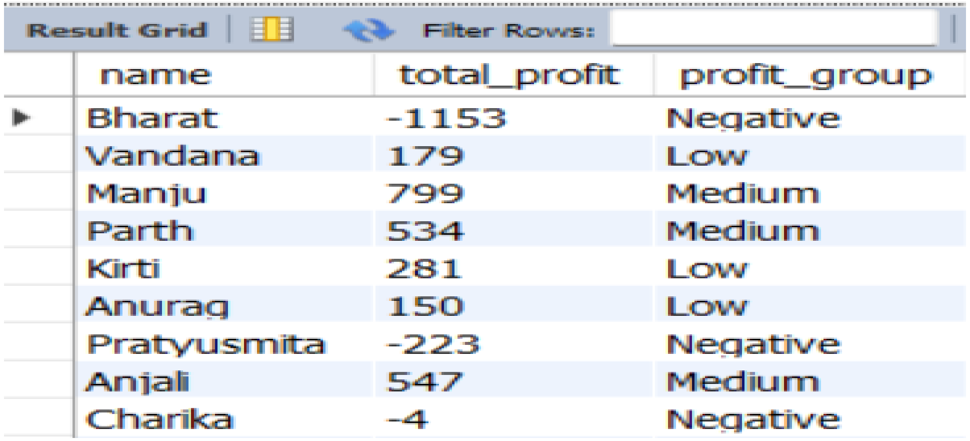
name	category_count
Abhishek	14

7. Calculate the total profit for each user and categorize them into 'High', 'Medium', and 'Low' profit groups.



```sql
SELECT u.name, SUM(od.profit) AS total_profit,
    CASE
        WHEN SUM(od.profit) > 1000 THEN 'High'
        WHEN SUM(od.profit) BETWEEN 500 AND 1000 THEN 'Medium'
        WHEN SUM(od.profit) < 0 THEN 'Negative'
        ELSE 'Low'
    END AS profit_group
FROM Orders o
JOIN Order_Details od ON o.order_id = od.order_id
JOIN Users u ON o.user_id = u.user_id
GROUP BY u.name;
```

Result:

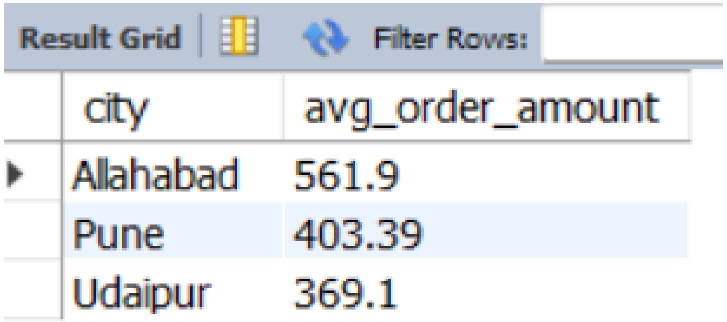


name	total_profit	profit_group
Bharat	-1153	Negative
Vandana	179	Low
Manju	799	Medium
Parth	534	Medium
Kirti	281	Low
Anurag	150	Low
Pratyusmita	-223	Negative
Anjali	547	Medium
Charika	-4	Negative

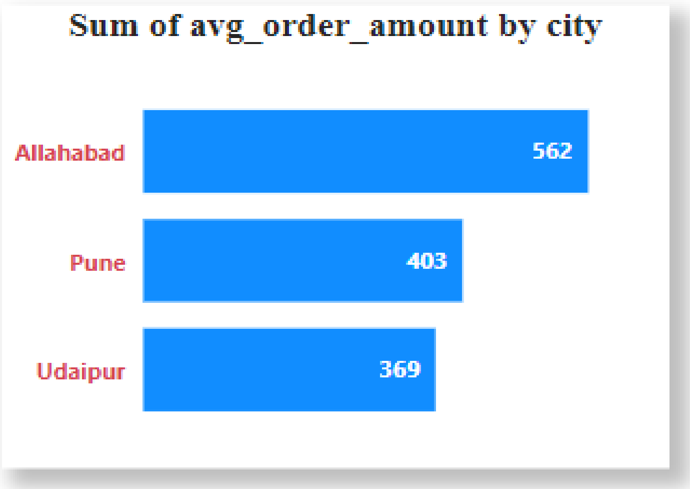
8. Find the top 3 cities with the highest average order amount.

```
SELECT u.city, ROUND(AVG(od.amount),2) AS avg_order_amount
FROM Orders o
JOIN Order_Details od ON o.order_id = od.order_id
JOIN Users u ON o.user_id = u.user_id
GROUP BY u.city
ORDER BY avg_order_amount DESC
LIMIT 3;
```

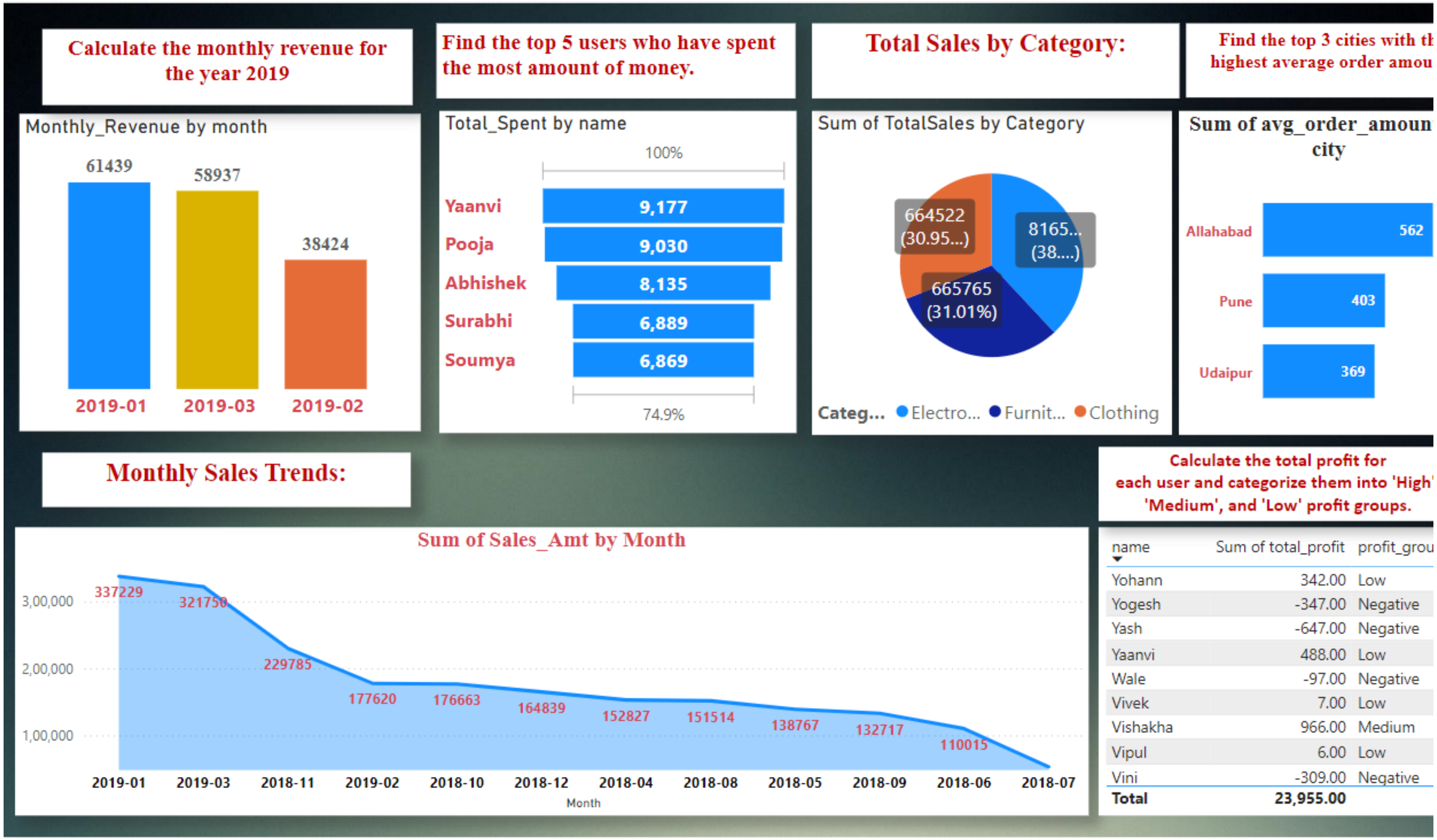
Result:



city	avg_order_amount
Allahabad	561.9
Pune	403.39
Udaipur	369.1



Visualization using Power BI



Calculate the monthly revenue for the year 2019

Find the top 5 users who have spent the most amount of money.

Total Sales by Category:

Find the top 3 cities with th highest average order amou

Monthly Sales Trends:

Calculate the total profit for each user and categorize them into 'High' 'Medium', and 'Low' profit groups.

name	Sum of total_profit	profit_grou
Yohann	342.00	Low
Yogesh	-347.00	Negative
Yash	-647.00	Negative
Yaanvi	488.00	Low
Wale	-97.00	Negative
Vivek	7.00	Low
Vishakha	966.00	Medium
Vipul	6.00	Low
Vini	-309.00	Negative
Total	**23,955.00**	

Insights

1. **Total Sales by Category**: Identifies which categories generate the most revenue.
2. **Monthly Sales Trends**: Shows sales trends over time, highlighting any seasonal patterns.
3. **Top 5 Users by Spending**: Recognizes the top 5 users who have spent the most money.
4. **Monthly Revenue for 2019**: Analyzes financial performance month by month for 2019.
5. **Category with Highest Average Profit per Order**: Identifies the most profitable category.
6. **Monthly Order Count for 2019**: Understands order volume trends throughout 2019.
7. **User with Most Diverse Orders**: Recognizes users with diverse purchasing habits.
8. **Total Profit by User and Profit Group**: Segments users based on their profitability.
9. **Top 3 Cities by Average Order Amount**: Identifies the most lucrative markets.