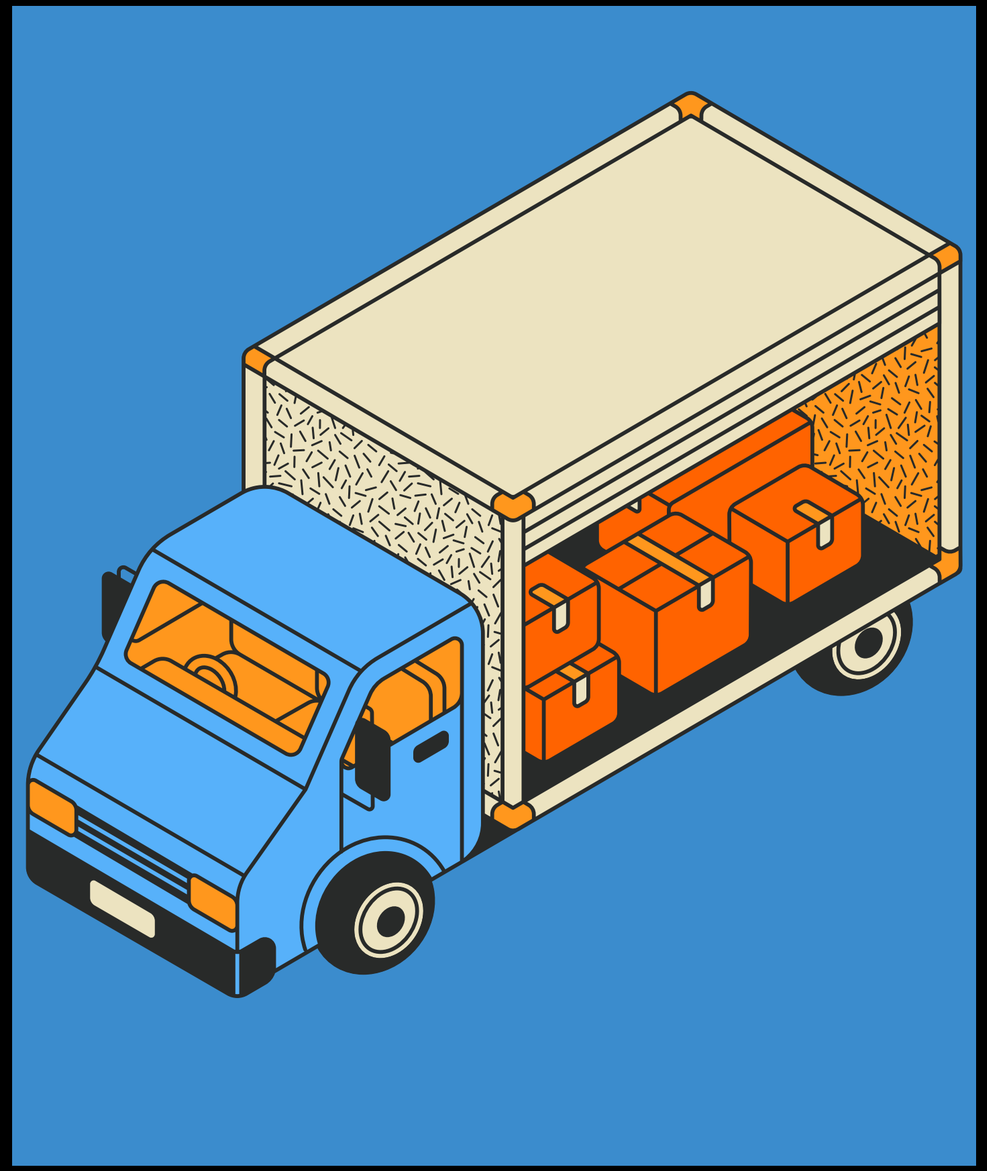
# DATA ANALYSIS OF A FOOD DELIVERY BUISNESS USING SQL



PRESENTED BY-
TISHA RAJ
BTECH/10945/22
BIRLA INSTITUTE OF TECHNOLOGY, MESRA

# ABOUT THIS PROJECT

- In today’s fast-paced digital economy, data-driven insights are crucial for making informed business decisions, especially in customer-centric industries like food delivery. This project focuses on analyzing the core operational and customer data of a food delivery company using Structured Query Language (SQL).
- By examining key datasets related to customers, orders, products, and sales, the project highlights how structured querying can uncover trends in customer behavior, identify high-performing products, reveal location-based revenue differences, and detect patterns that support better business strategy and planning.
- To enhance the interpretability of these insights, an interactive dashboard has also been developed, providing a visual representation of the analyzed data.
- Ultimately, this project demonstrates the power of SQL in transforming raw transactional data into meaningful insights, proving it to be an essential tool in modern business intelligence and decision-making processes.

# DATASET OVERVIEW

- The dataset used in this project was downloaded from GitHub, specifically titled "Food Delivery Sales Dataset". It simulates a transactional database for a food delivery business, making it ideal for applying SQL-based analytical techniques.

## Key Characteristics:

- Publicly available for educational and portfolio projects
- Clean, well-structured format for relational database modeling
- Represents realistic customer-product-order relationships

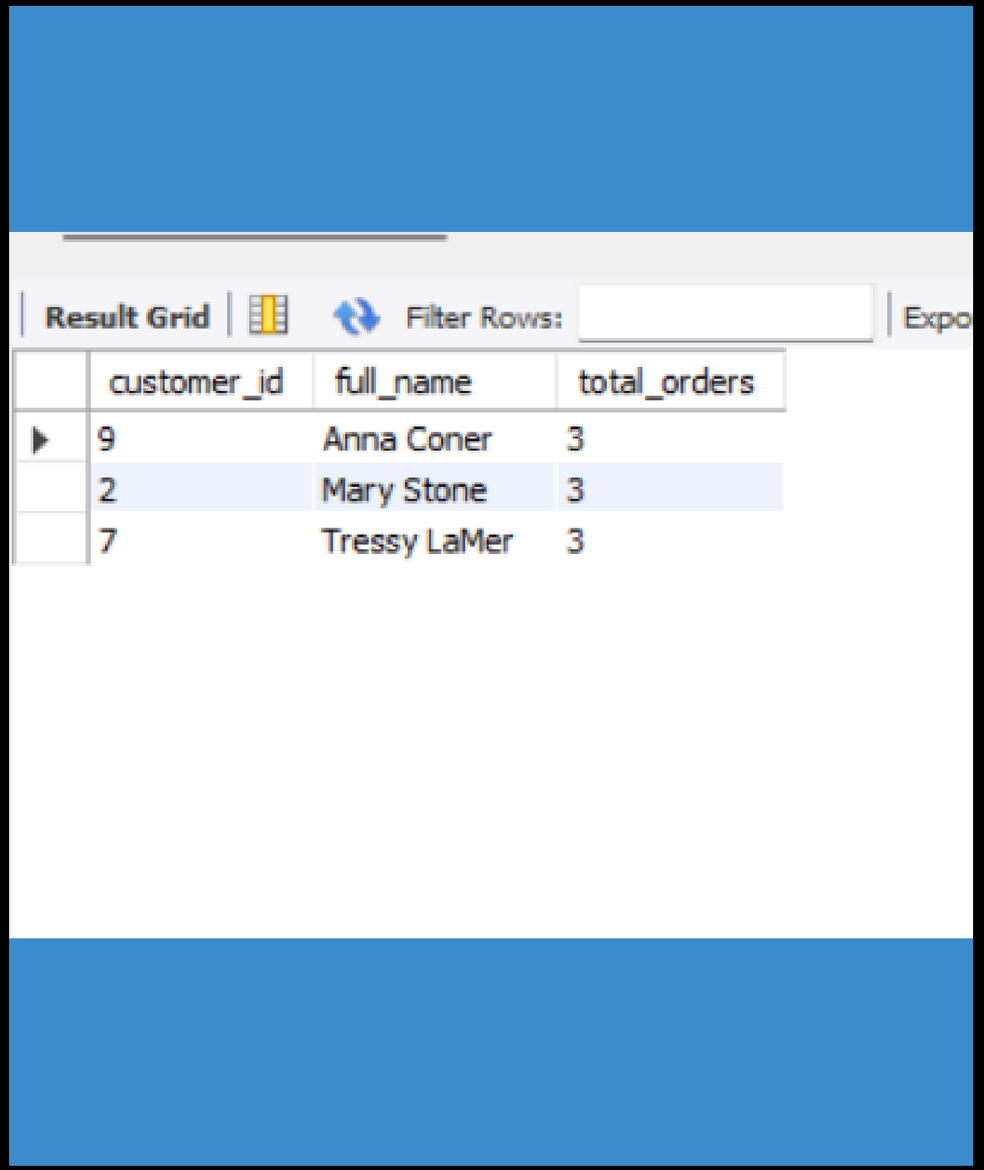
## Files Included:

- customer.csv – Customer ID, names, cities, and other basic details
- product.csv – Product ID, item names, and prices
- orders.csv – Order ID, customer linkage, and dates
- order_item.csv – Order-wise breakdown of products and quantities

Let’s dive into the data with powerful SQL queries using MySQL!

# WHICH CUSTOMERS HAVE PLACED THE HIGHEST NUMBER OF ORDERS?

```sql
SELECT
    c.id AS customer_id,
    CONCAT(c.first_name, ' ', c.last_name) AS full_name,
    COUNT(o.id) AS total_orders
FROM
    customers c
JOIN
    orders o ON c.id = o.customer_id
GROUP BY
    c.id, c.first_name, c.last_name
ORDER BY
    total_orders DESC
```




| customer_id | full_name | total_orders |
|---|---|---|
| 9 | Anna Coner | 3 |
| 2 | Mary Stone | 3 |
| 7 | Tressy LaMer | 3 |

# WHAT IS THE TOTAL NUMBER OF ORDERS MADE IN EACH CITY?

```sql
SELECT
    c.city,
    COUNT(o.id) AS total_orders
FROM orders o
JOIN customers c ON o.customer_id = c.id
GROUP BY c.city
ORDER BY total_orders DESC;
```

Result Grid | Filter Rows:

| city | total_orders |
|---|---|
| San Jose | 21 |
| San Francisco | 8 |
| Santa Clara | 5 |
| Sunnyvale | 3 |
| Palo Alto | 2 |
| Fremont | 2 |
| Mountain View | 1 |

# FIND THE MOST FREQUENTLY ORDERED PRODUCT

```sql
SELECT
    oi.product_id,
    p.name AS product_name,
    SUM(oi.item_quantity) AS total_quantity_ordered
FROM order_item oi
JOIN product p ON oi.product_id = p.id
GROUP BY oi.product_id, p.name
ORDER BY total_quantity_ordered DESC
LIMIT 1;
```

Result Grid | Filter Rows:

| product_id | product_name | total_quantity_ordered |
|---|---|---|
| 4 | Dumplings 1 | 71 |

# WHAT IS THE AVERAGE TIP AMOUNT GIVEN PER ORDER?

```
SELECT
    AVG(tips) AS average_tip
FROM orders
WHERE tips IS NOT NULL;
```

Result Grid

| average_tip |
| --- |
| 3.500000 |

# FIND THE TOP 5 CUSTOMERS WHO GAVE THE HIGHEST AVERAGE RATINGS

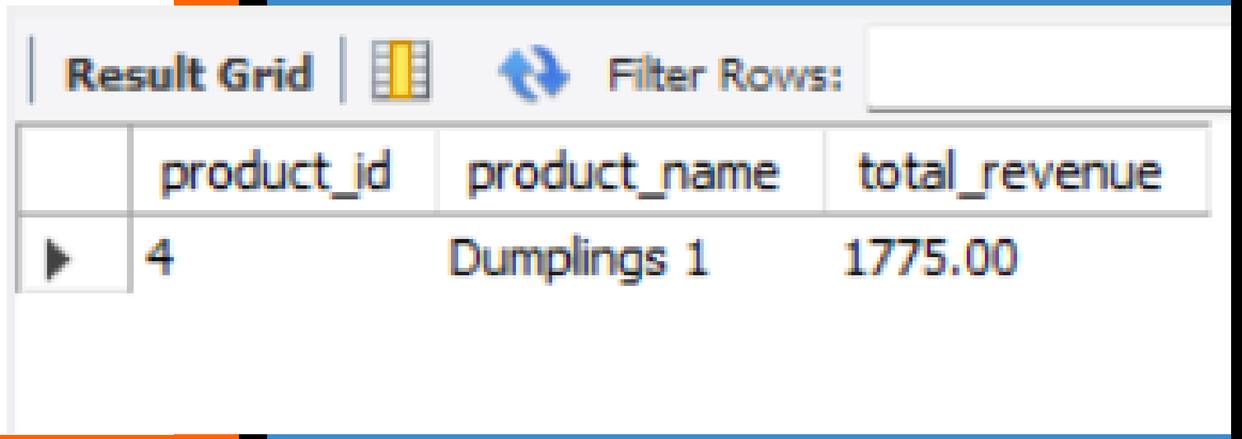
```sql
SELECT
o.customer_id,
c.first_name,
c.last_name,
ROUND(AVG(o.rating)) AS avg_rating,
COUNT(o.id) AS total_orders
FROM orders O
JOIN customers c
ON
o.customer_id=c.id
where o.rating IS NOT NULL
GROUP BY o.customer_id
ORDER BY avg_rating DESC
LIMIT 5;
```

Result Grid | Filter Rows: | Export:

| customer_id | first_name | last_name | avg_rating | total_orders |
|---|---|---|---|---|
| 1 | Dana | Lo | 5 | 1 |
| 9 | Anna | Coner | 5 | 3 |
| 3 | July | Crone | 5 | 1 |
| 5 | Clara | S | 5 | 2 |
| 7 | Tressy | LaMer | 5 | 3 |

# WHICH PRODUCT GENERATES THE HIGHEST REVENUE?

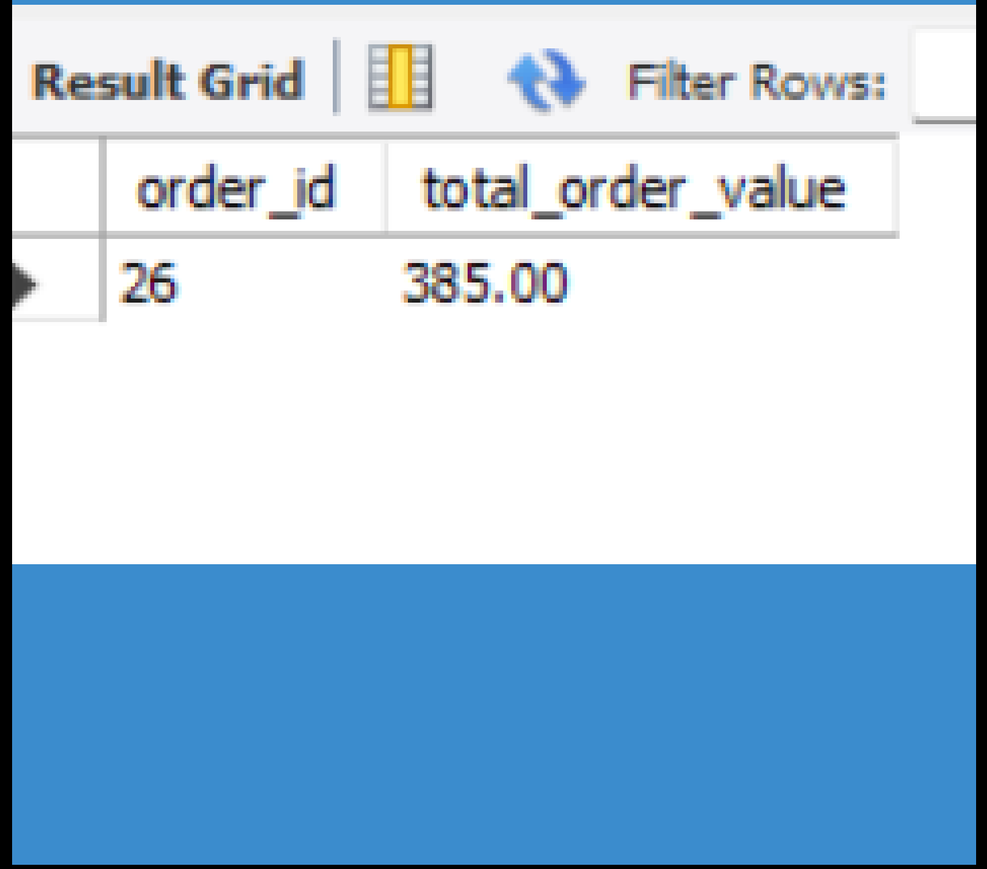
```
SELECT
p.id AS product_id,
p.name AS product_name,
SUM(oi.item_quantity*p.price) AS total_revenue
FROM order_item oi
JOIN product p
ON
oi.product_id=p.id
GROUP BY p.id
ORDER BY total_revenue desc
LIMIT 1;
```




| product_id | product_name | total_revenue |
|---|---|---|
| 4 | Dumplings 1 | 1775.00 |

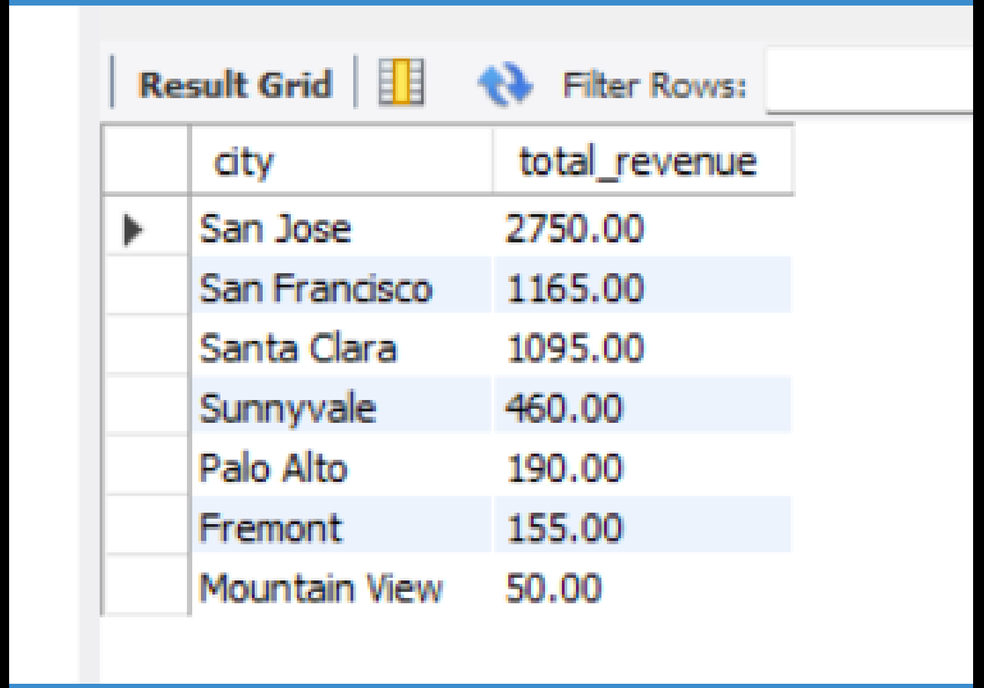
# FIND THE ORDER WITH HIGHEST TOTAL VALUE

```sql
SELECT
    o.id AS order_id,
    SUM(oi.price * oi.item_quantity) + IFNULL(o.tips, 0) AS total_
FROM
    orders o
JOIN
    order_item oi ON o.id = oi.order_id
GROUP BY
    o.id
ORDER BY
    SUM(oi.price * oi.item_quantity) + IFNULL(o.tips, 0) DESC
LIMIT 1;
```



# WHAT IS THE TOTAL REVENUE GENERATED PER CITY?

```sql
SELECT
    c.city,
    SUM(oi.item_quantity * p.price) AS total_revenue
FROM
    order_item oi
JOIN
    orders o ON oi.order_id = o.id
JOIN
    customers c ON o.customer_id = c.id
JOIN
    product p ON oi.product_id = p.id
GROUP BY
    c.city
ORDER BY
    total_revenue DESC;
```




| city | total_revenue |
|---|---|
| San Jose | 2750.00 |
| San Francisco | 1165.00 |
| Santa Clara | 1095.00 |
| Sunnyvale | 460.00 |
| Palo Alto | 190.00 |
| Fremont | 155.00 |
| Mountain View | 50.00 |

# CALCULATE TOTAL REVENUE AND PROFIT PER PRODUCT

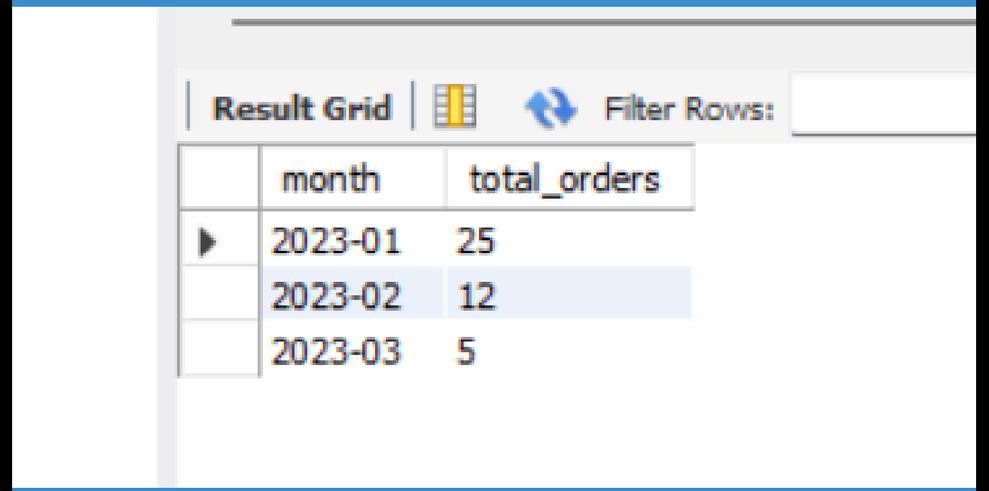
```sql
SELECT
p.id AS product_id,
p.name AS product_name,
SUM(oi.item_quantity * p.price) AS total_revenue,
SUM(oi.item_quantity * (p.price - p.cost)) AS total_profit
FROM order_item oi
JOIN product p
ON
oi.product_id=p.id
GROUP BY p.id
ORDER BY total_revenue ASC;
```

Result Grid | Filter Rows: | Export:

| product_id | product_name | total_revenue | total_profit |
|---|---|---|---|
| 9 | Crepes 0 | 45.00 | 27.00 |
| 8 | Potato pie | 80.00 | 44.00 |
| 11 | Pastry 2 | 140.00 | 77.00 |
| 7 | Cheese pie | 150.00 | 85.00 |
| 2 | Crepes | 240.00 | 112.00 |
| 12 | Vegetable 1 | 275.00 | 143.00 |
| 6 | Cheesecake | 280.00 | 154.00 |
| 3 | Sweet Crepes | 300.00 | 140.00 |
| 10 | Pastry 1 | 480.00 | 312.00 |
| 1 | Cake | 750.00 | 450.00 |
| 5 | Dumplings 2 | 1350.00 | 810.00 |
| 4 | Dumplings 1 | 1775.00 | 1065.00 |

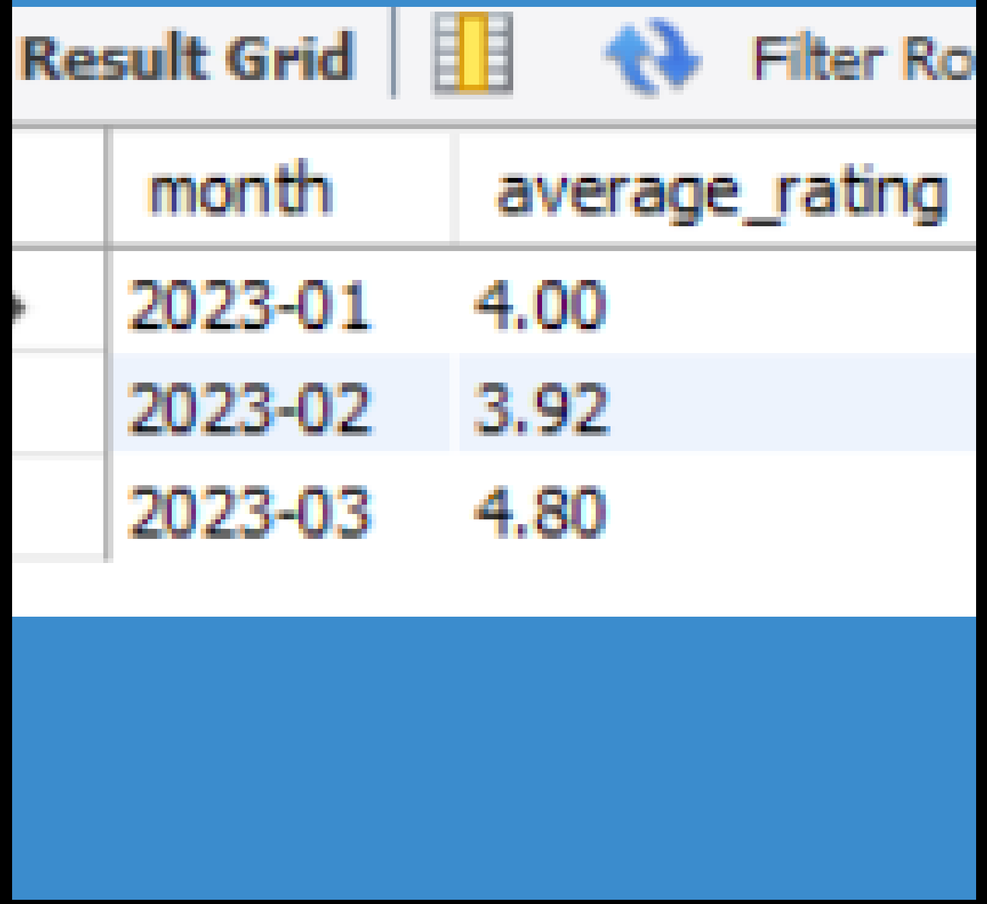
# HOW MANY ORDERS WERE PLACED EACH MONTH?

```sql
  SELECT
    DATE_FORMAT(created_at, '%Y-%m') AS month,
    COUNT(*) AS total_orders
FROM orders
GROUP BY month,
ORDER BY month;
```



# IDENTIFY THE TREND IN AVERAGE RATINGS OVER TIME (MONTHLY)

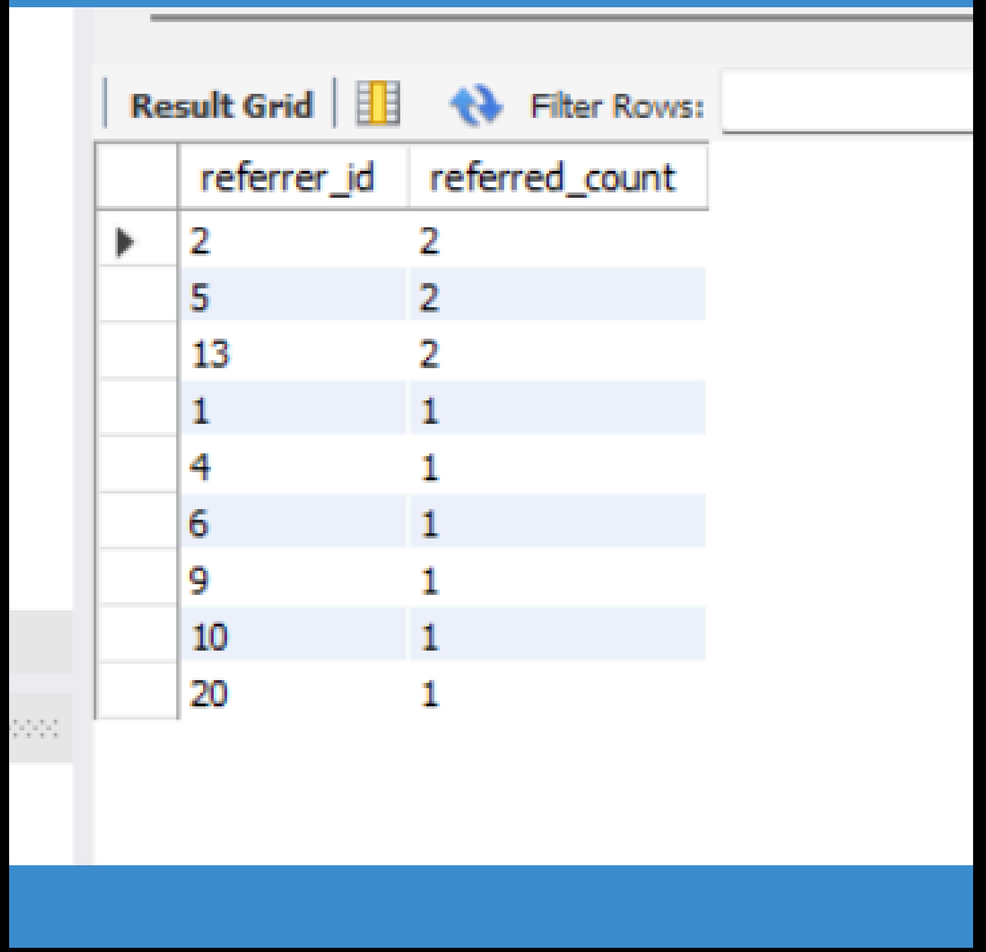
```sql
SELECT
    DATE_FORMAT(created_at, '%Y-%m') AS month,
    ROUND(AVG(rating), 2) AS average_rating
FROM orders
WHERE rating IS NOT NULL
GROUP BY month
ORDER BY month;
```




| month | average_rating |
|---|---|
| 2023-01 | 4.00 |
| 2023-02 | 3.92 |
| 2023-03 | 4.80 |

# HOW MANY CUSTOMERS WERE REFERRED BY EACH EXISTING CUSTOMER?

```sql
SELECT
    referral_customer_id AS referrer_id,
    COUNT(id) AS referred_count
FROM
    customers
WHERE
    referral_customer_id IS NOT NULL
GROUP BY
    referral_customer_id
ORDER BY
    referred_count DESC;
```




| referrer_id | referred_count |
|---|---|
| 2 | 2 |
| 5 | 2 |
| 13 | 2 |
| 1 | 1 |
| 4 | 1 |
| 6 | 1 |
| 9 | 1 |
| 10 | 1 |
| 20 | 1 |

# AVERAGE ORDER VALUE (AOV) PER CUSTOMER

```sql
WITH order_totals AS (
    SELECT
        o.customer_id,
        SUM(oi.item_quantity * oi.price) AS total_order_va
    FROM orders o
    JOIN order_item oi ON o.id = oi.order_id
    GROUP BY o.id
)

SELECT
    customer_id,
    ROUND(AVG(total_order_value), 2) AS avg_order_value
FROM order_totals
GROUP BY customer_id
ORDER BY avg_order_value DESC
LIMIT 10;
```

Result Grid | Filter Rows:

| customer_id | avg_order_value |
|---|---|
| 20 | 340.00 |
| 22 | 280.00 |
| 10 | 215.00 |
| 15 | 210.00 |
| 17 | 195.00 |
| 12 | 162.50 |
| 13 | 160.00 |
| 7 | 151.67 |
| 14 | 146.67 |
| 23 | 145.00 |

# MONTHLY ORDER COUNT AND REVENUE

```sql
WITH monthly_stats AS (
    SELECT
        DATE_FORMAT(o.created_at, '%Y-%m') AS order_month,
        COUNT(DISTINCT o.id) AS total_orders,
        SUM(oi.item_quantity * oi.price) AS total_revenue
    FROM orders o
    JOIN order_item oi ON o.id = oi.order_id
    GROUP BY DATE_FORMAT(o.created_at, '%Y-%m')
)

SELECT
    order_month,
    total_orders,
    total_revenue
FROM monthly_stats
ORDER BY order_month;
```

Result Grid | Filter Rows:

| order_month | total_orders | total_revenue |
|---|---|---|
| 2023-01 | 25 | 2985.00 |
| 2023-02 | 12 | 1995.00 |
| 2023-03 | 5 | 805.00 |

# INTERACTIVE DASHBOARD USING

# FOOD DELIVERY BUISNESS ANALYSIS- SQL PROJECT

**FOOD DELIVERY BUISNESS ANALYSIS**

FILTER PANEL

name: All

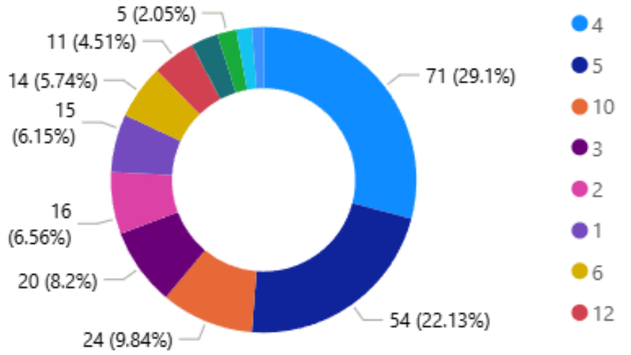
city: All

created_at: 04-01-2023 – 05-03-2023

| 5785 | 137.74 | 42 | 4.07 |
|---|---|---|---|
| Total Revenue | Average Order Value | Total Number of Orders | Average Rating |

| first_name | Total Customer Revenue |
|---|---|
| Aeron | 90 |
| Anna | 195 |
| Anna-Maria | 245 |
| Anthony | 25 |
| Carelia | 70 |
| Clara | 100 |
| Cory | 120 |
| Dana | 55 |
| David | 195 |
| Gregor | 200 |
| **Total** | **3090** |

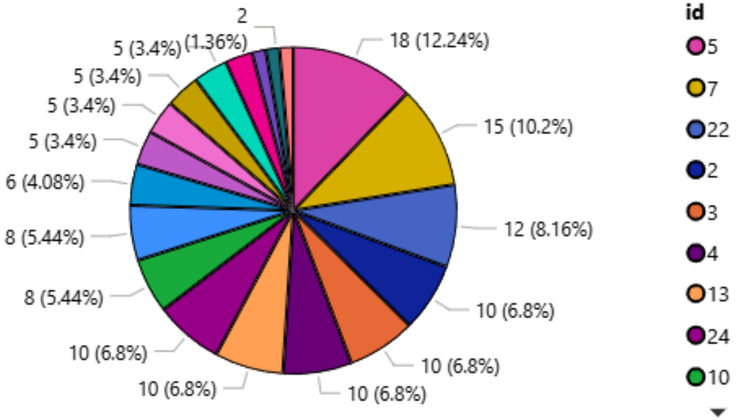
**Total Quantity Sold by id**



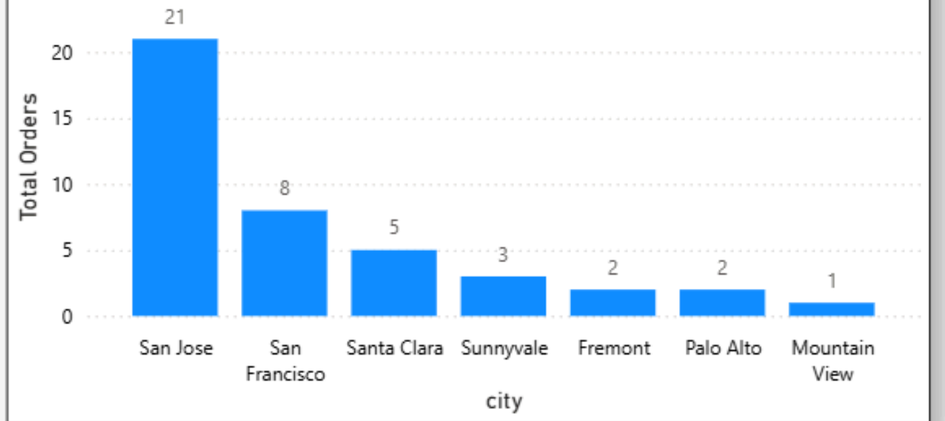

**Top Customers with Most Orders**

| first_name | CustomerRevenue | TotalOrders | city |
|---|---|---|---|
| Vera | 685.00 | 2 | Santa Clara |
| Anthony | 75.00 | 1 | Santa Clara |
| Carelia | 210.00 | 1 | San Jose |
| Dana | 55.00 | 1 | San Francisco |
| Jenna | 280.00 | 1 | San Francisco |
| Julia | 50.00 | 1 | San Francisco |
| July | 50.00 | 1 | Mountain View |
| Kevin | 115.00 | 1 | San Jose |
| Marie | 100.00 | 1 | San Jose |
| Natan | 80.00 | 1 | San Jose |
| **Total** | **5,865.00** | **42** | |

**Total Tip by Customer**




**Total Orders by City**



## TOOLS USED:

- MYSQL WORKBENCH – DATA CLEANING AND ANALYSIS
- POWER BI – INTERACTIVE DASHBOARD CREATION
- CANVA – PROJECT VISUALIZATION AND DOCUMENTATION

# THANK YOU