PROJECT – PIZZA SALES ANALYSIS

This project is designed to demonstrate SQL and Advanced Excel skills and techniques typically used by data analysts to explore, clean, and analyse sales data. The project involves setting up a Pizza sales database, performing exploratory data analysis (EDA), and answering specific business questions through SQL queries and visualising through Excel dashboards.

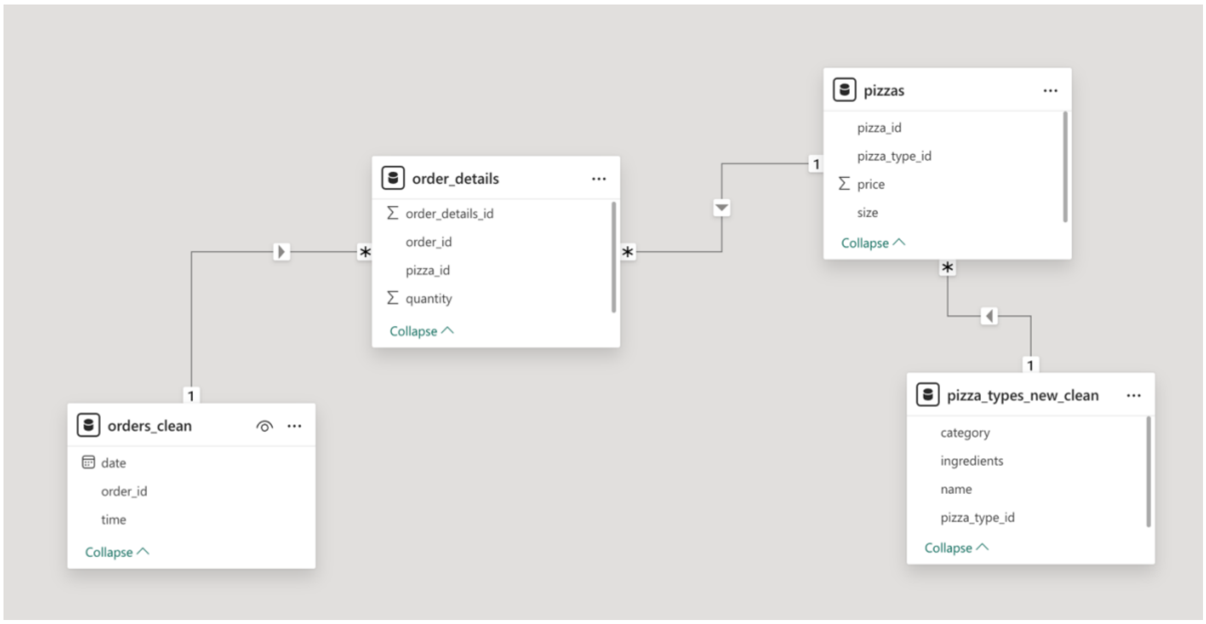
TOOLS USED – MySQL Workbench, Excel (Power tools and Visualisation)

Objectives-

1. **Set up a Pizza sales database**: Create and populate a Pizza 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.
5. **Data Visualisation**: Use Excel to create a Dashboards to visualise the analysis of sales data.

DATA Details:

Star Schema



We have four tables for the pizza sales analysis.

1. Orders

COLUMN NAME	DATA TYPE	CONSTRAINT
Order_id	INTEGER	PRIMARY KEY
Order_date	DATE	NOT NULL
Order_time	TIME	NOT NULL

2. Order Details

COLUMN NAME	DATA TYPE	CONSTRAINT
Order_details_id	INTEGER	PRIMARY KEY
Order_id	INTEGER	FOREIGN KEY
Pizza_id	VARCHAR(20)	FOREIGN KEY
quantity	INTEGER	NOT NULL

3. Pizzas

COLUMN NAME	DATA TYPE	CONSTRAINT
Pizza_id	VARCHAR(20)	PRIMARY KEY
Pizza_type_id	VARCHAR(20)	FOREIGN KEY
Size	VARCHAR(20)	NOT NULL
Price	DOUBLE	NOT NULL

4. Pizza Types

COLUMN NAME	DATA TYPE	CONSTRAINT
Pizza_type_id	VARCHAR(20)	PRIMARY KEY
Name	VARCHAR(20)	NOT NULL
category	VARCHAR(20)	NOT NULL
Ingredients	VARCHAR(20)	NOT NULL

PROBLEM STATEMENT

The pizza company’s management wants to evaluate overall business performance to understand revenue growth trends, sales drivers, and areas of improvement across time, products, and customer demand patterns.

ANALYSIS ROADMAP –

1. Key Performance Indicators (KPI) Identification
2. Overall Sales Performance Evaluation
3. Product Performance Analysis
4. Customer Demand Analysis
5. Advanced Performance Metrics
6. Visualization & Dashboard Development
7. Insights & Business Recommendations

ANALYSIS USING SQL

Tool – MySQL Workbench

1. **DATABASE Setup**
   Database name – pizza_shop

```
CREATE DATABASE pizza_shop;
USE pizza_shop;
```

- Importing CSV files into new tables pizzas and pizza_types directly
- Creating Tables orders and order_details, then importing data into these Tables to match appropriate data types

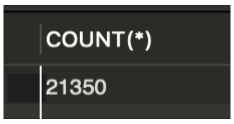
```
CREATE TABLE orders (
   order_id INT NOT NULL,
   order_date DATE NOT NULL,
   order_time TIME NOT NULL,
   PRIMARY KEY (order_id)
);

CREATE TABLE orders_details (
   order_details_id INT NOT NULL,
   order_id INT NOT NULL,
   pizza_id VARCHAR(20) NOT NULL,
   quantity INT NOT NULL,
   PRIMARY KEY (order_details_id)
);
```

2. **DATA EXPLORATION AND CLEANING**

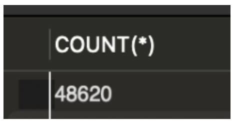
   - Record counts – Total number of rows in each table.

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



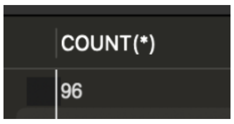
```
COUNT(*)
21350
```

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



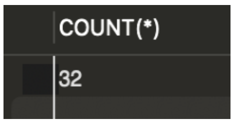
```
COUNT(*)
48620
```

SELECT COUNT(*) FROM pizzas;



```
COUNT(*)
96
```

SELECT COUNT(*) FROM pizza_types_new_clean;

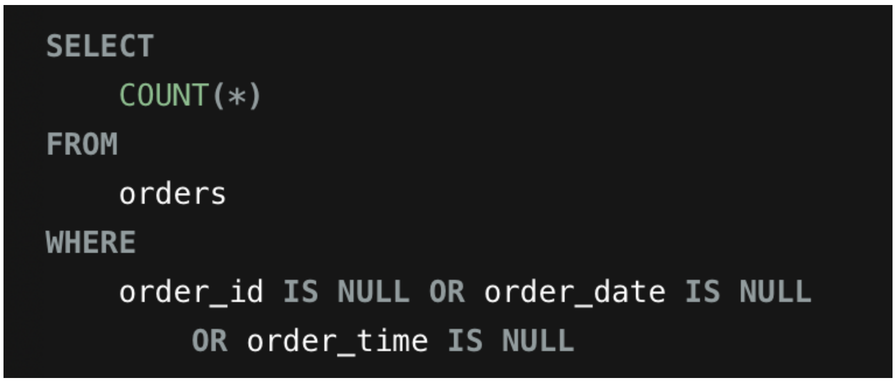


```
COUNT(*)
32
```

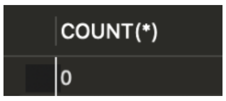
- Table structure – Understand the data types and constraints of all columns in each table.

  DESCRIBE orders;
  DESCRIBE orders_details;
  DESCRIBE pizzas;
  DESCRIBE pizza_types_new_clean;

- NULL Value Check
  Null value check in orders table –

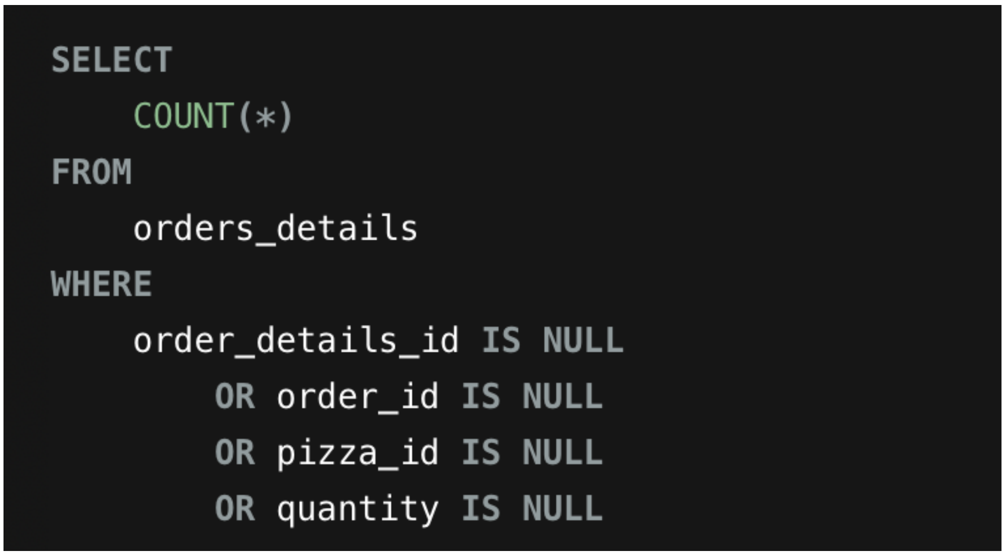


```
SELECT
    COUNT(*)
FROM
    orders
WHERE
    order_id IS NULL OR order_date IS NULL
        OR order_time IS NULL
```

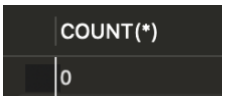


```
COUNT(*)
0
```

Null value check in order_details table –

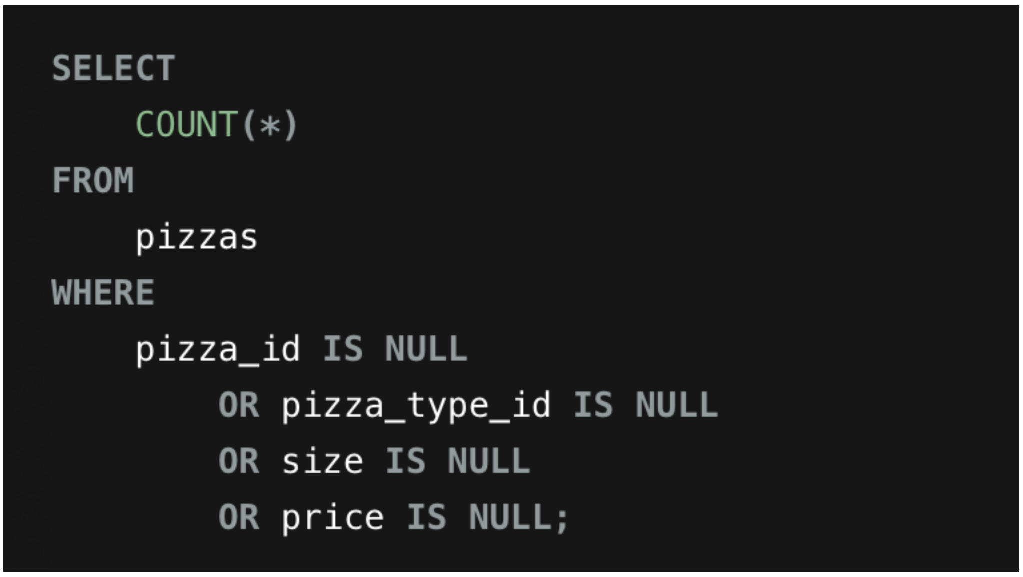


```
SELECT
    COUNT(*)
FROM
    orders_details
WHERE
    order_details_id IS NULL
        OR order_id IS NULL
        OR pizza_id IS NULL
        OR quantity IS NULL
```

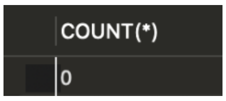


```
COUNT(*)
0
```

Null value check in pizzas table –

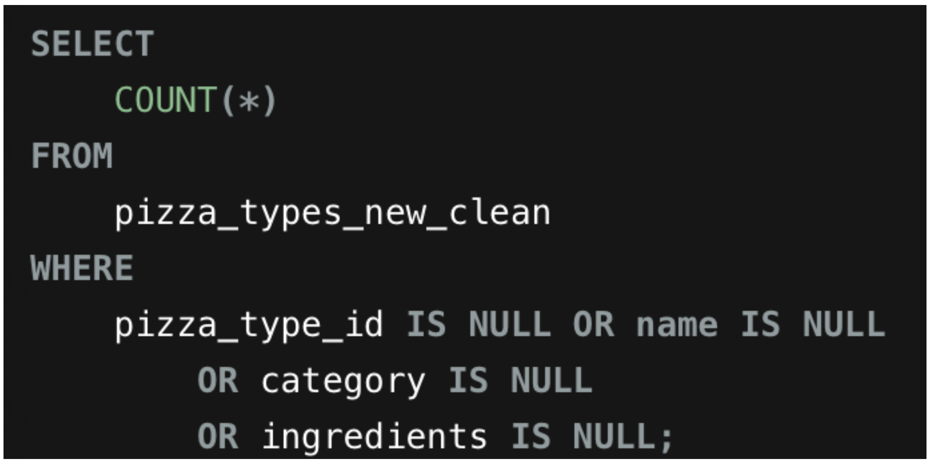


```
SELECT
    COUNT(*)
FROM
    pizzas
WHERE
    pizza_id IS NULL
        OR pizza_type_id IS NULL
        OR size IS NULL
        OR price IS NULL;
```

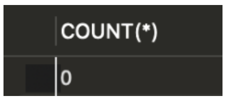


```
COUNT(*)
0
```

Null value check in pizza_types table –



```
SELECT
    COUNT(*)
FROM
    pizza_types_new_clean
WHERE
    pizza_type_id IS NULL OR name IS NULL
        OR category IS NULL
        OR ingredients IS NULL;
```



```
COUNT(*)
0
```

Result – No NULL value present in any of the tables.

If there would have been any NULL values, it will be discussed upon with stakeholders and accordingly removed from table using the following syntax.

```
DELETE FROM table name
        WHERE
                Column1 IS NULL OR
                Column2 IS NULL OR... ;
```

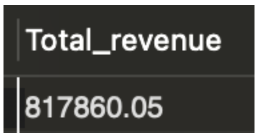
3. DATA ANALYSIS

A. Calculating KPIs

a) TOTAL REVENUE :

I will use Quantity and Price from Order_details and Pizzas table to get total revenue.

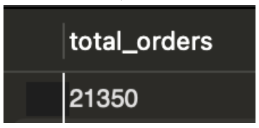
```
SELECT
  ROUND(SUM(OD.quantity * P.price), 2) AS Total_revenue
FROM
  orders_details OD
    JOIN
  pizzas P ON OD.pizza_id = P.pizza_id;
```



```
Total_revenue
817860.05
```

b) TOTAL ORDERS PLACED

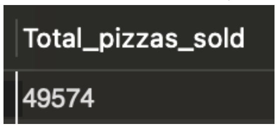
```
SELECT
   COUNT(DISTINCT order_id) AS total_orders
FROM
   orders;
```



```
total_orders
21350
```

c) TOTAL PIZZAS SOLD

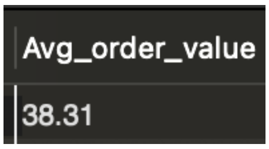
```
SELECT
   SUM(quantity) AS Total_pizzas_sold
FROM
   orders_details;
```



```
Total_pizzas_sold
49574
```

d) AVERAGE ORDER VALUE

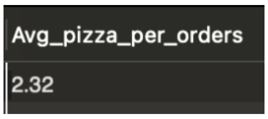
```
SELECT
   ROUND(SUM(OD.quantity * P.price) / COUNT(DISTINCT O.order_id),
         2) AS Avg_order_value
FROM
   orders_details OD
     JOIN
   pizzas P ON OD.pizza_id = P.pizza_id
     JOIN
   orders O ON O.order_id = OD.order_id;
```



```
Avg_order_value
38.31
```

e) AVERAGE PIZZAS PER ORDER

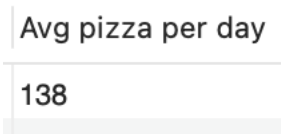
```
SELECT
   ROUND(SUM(quantity) / COUNT(DISTINCT O.order_id),
         2) AS Avg_pizza_per_orders
FROM
   orders_details OD
     JOIN
   orders O ON OD.order_id = O.order_id;
```



```
Avg_pizza_per_orders
2.32
```

f) AVERAGE NUMBER OF PIZZAS ORDERED PER DAY

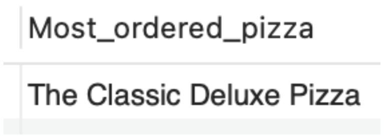
```
WITH Daywise_pizzas AS
(SELECT order_date, SUM(OD.quantity) AS total FROM orders O
JOIN orders_details OD ON O.order_id = OD.order_id
GROUP BY order_date
)
SELECT ROUND(AVG(total),0) AS "Avg pizza per day" FROM Daywise_pizzas;
```



Avg pizza per day
138

g) MOST ORDERED PIZZA

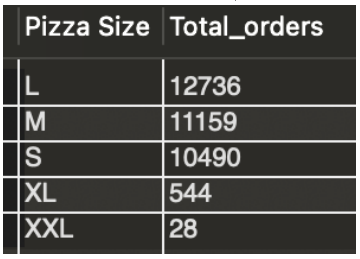
```
SELECT
  PT.name AS Most_ordered_pizza
FROM
  orders_details OD
    JOIN
  pizzas P ON OD.pizza_id = P.pizza_id
    JOIN
  pizza_types_new_clean PT ON PT.pizza_type_id = P.pizza_type_id
GROUP BY PT.name
ORDER BY SUM(OD.quantity) DESC
LIMIT 1;
```



Most_ordered_pizza
The Classic Deluxe Pizza

B. NUMBER OF ORDERS ACCORIDNG TO PIZZA SIZES

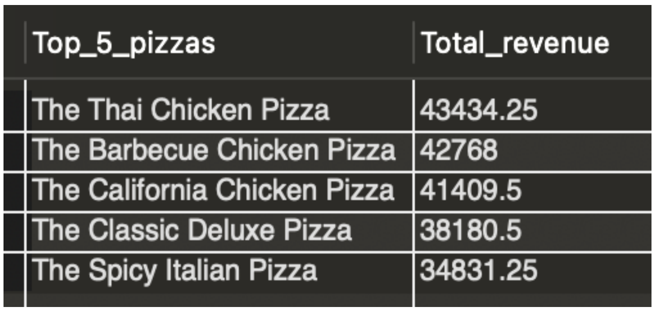
```
SELECT
  P.size, COUNT(DISTINCT O.order_id)  AS Total_orders
FROM
  pizzas P
    JOIN
  orders_details OD ON P.pizza_id = OD.pizza_id
    JOIN
  orders O ON O.order_id = OD.order_id
GROUP BY P.size;
```



Pizza Size	Total_orders
L	12736
M	11159
S	10490
XL	544
XXL	28

C. TOP 5 MOST ORDERED PIZZA AS PER REVENUE

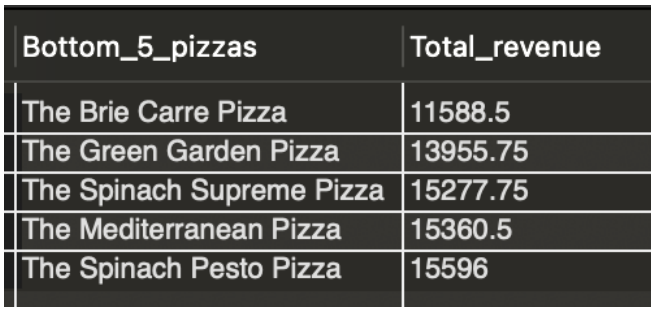
```
SELECT PT.name AS Top_5_pizzas, ROUND(SUM(OD.quantity*P.price),2) AS
Total_revenue
FROM orders_details OD
    JOIN
  pizzas P ON OD.pizza_id = P.pizza_id
    JOIN
  pizza_types_new_clean PT ON PT.pizza_type_id = P.pizza_type_id
GROUP BY PT.name
ORDER BY Total_revenue DESC
LIMIT 5;
```



Top_5_pizzas	Total_revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5
The Classic Deluxe Pizza	38180.5
The Spicy Italian Pizza	34831.25

D. BOTTOM 5 MOST ORDERED PIZZA AS PER REVENUE

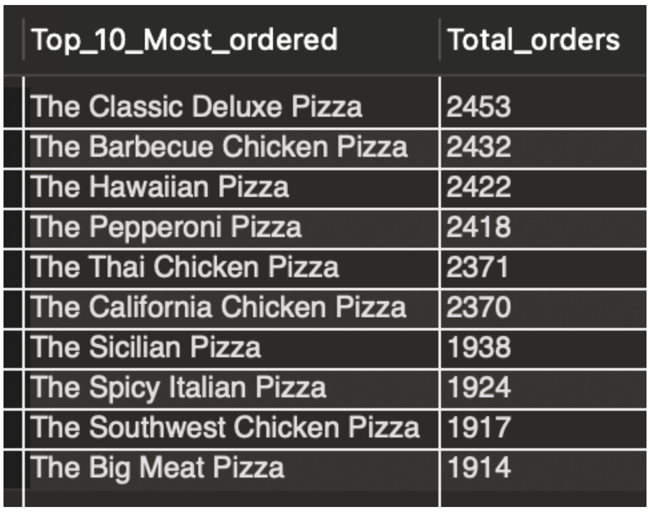
```
SELECT PT.name AS Bottom_5_pizzas, ROUND(SUM(OD.quantity*P.price),2)
AS Total_revenue
FROM orders_details OD
    JOIN
  pizzas P ON OD.pizza_id = P.pizza_id
    JOIN
  pizza_types_new_clean PT ON PT.pizza_type_id = P.pizza_type_id
GROUP BY PT.name
ORDER BY Total_revenue
LIMIT 5;
```



Bottom_5_pizzas	Total_revenue
The Brie Carre Pizza	11588.5
The Green Garden Pizza	13955.75
The Spinach Supreme Pizza	15277.75
The Mediterranean Pizza	15360.5
The Spinach Pesto Pizza	15596

E. TOP 10 MOST ORDERED PIZZA

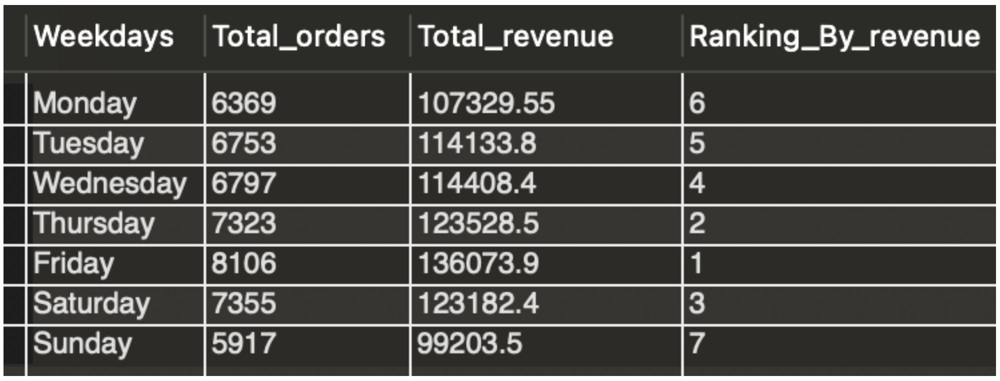
```
SELECT PT.name AS Top_10_Most_ordered, ROUND(SUM(OD.quantity),2) AS
Total_orders
FROM orders_details OD
    JOIN
  pizzas P ON OD.pizza_id = P.pizza_id
    JOIN
  pizza_types_new_clean PT ON PT.pizza_type_id = P.pizza_type_id
GROUP BY PT.name
ORDER BY Total_orders DESC
LIMIT 10;
```



Top_10_Most_ordered	Total_orders
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371
The California Chicken Pizza	2370
The Sicilian Pizza	1938
The Spicy Italian Pizza	1924
The Southwest Chicken Pizza	1917
The Big Meat Pizza	1914

F. DAILY TREND FOR TOTAL ORDERS & REVENUE

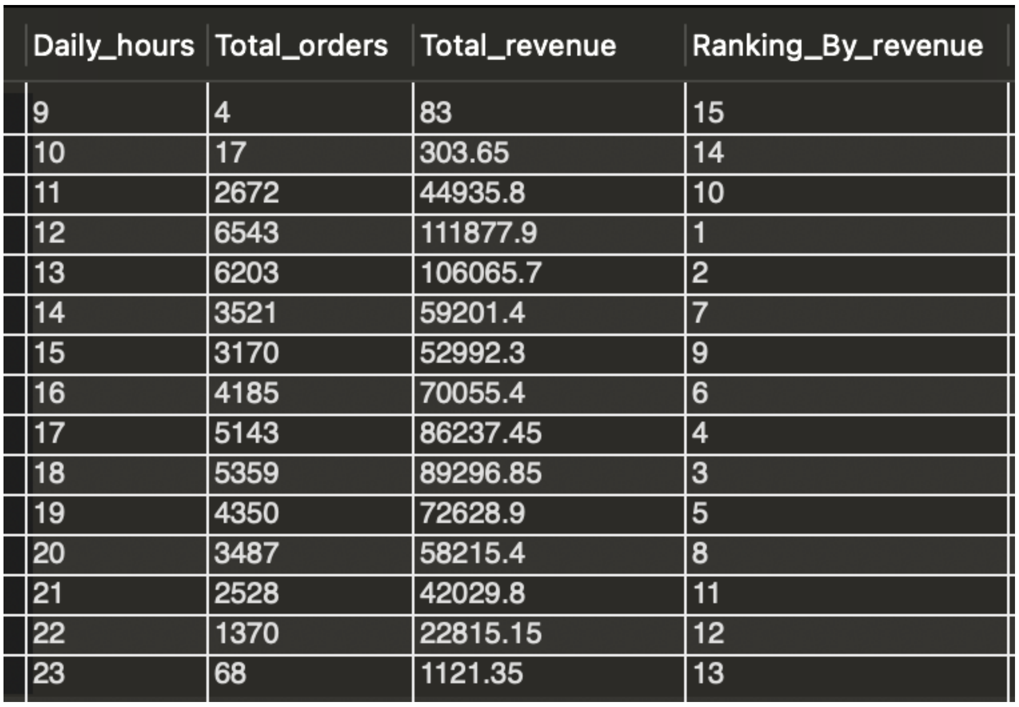
```
SELECT
  DAYNAME(O.order_date) AS Weekdays,
  COUNT(O.order_id) AS Total_orders,
  ROUND(SUM(P.price*OD.quantity),2) AS Total_revenue,
  RANK() OVER(ORDER BY SUM(P.price*OD.quantity) DESC) AS
Ranking_By_revenue
FROM
  orders O
  JOIN orders_details OD
  ON O.order_id = OD.order_id
  JOIN pizzas P
  ON P.pizza_id = OD.pizza_id
GROUP BY Weekdays , WEEKDAY(O.order_date)
ORDER BY WEEKDAY(O.order_date);
```



Weekdays	Total_orders	Total_revenue	Ranking_By_revenue
Monday	6369	107329.55	6
Tuesday	6753	114133.8	5
Wednesday	6797	114408.4	4
Thursday	7323	123528.5	2
Friday	8106	136073.9	1
Saturday	7355	123182.4	3
Sunday	5917	99203.5	7

G. HOURLY TREND FOR ORDERS & REVENUE

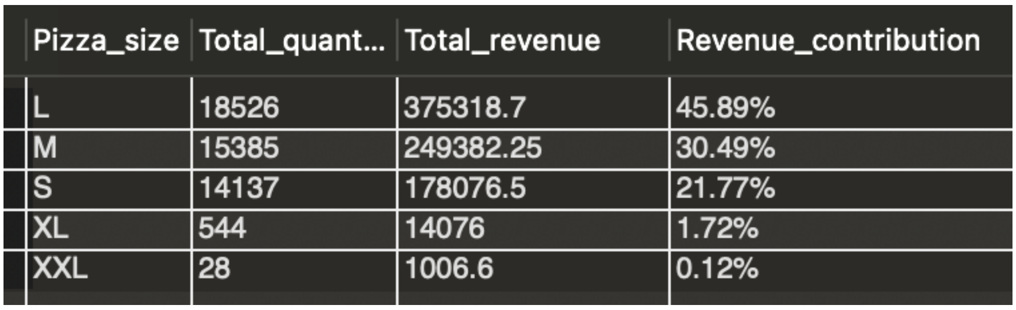
```
SELECT HOUR(O.order_time) AS Daily_hours,
   COUNT(O.order_id) AS Total_orders,
   ROUND(SUM(P.price*OD.quantity),2) AS Total_revenue,
   RANK() OVER(ORDER BY SUM(P.price*OD.quantity) DESC) AS
Ranking_By_revenue
FROM
   orders O
   JOIN orders_details OD
   ON O.order_id = OD.order_id
   JOIN pizzas P
   ON P.pizza_id = OD.pizza_id
GROUP BY Daily_hours
ORDER BY Daily_hours;
```



Daily_hours	Total_orders	Total_revenue	Ranking_By_revenue
9	4	83	15
10	17	303.65	14
11	2672	44935.8	10
12	6543	111877.9	1
13	6203	106065.7	2
14	3521	59201.4	7
15	3170	52992.3	9
16	4185	70055.4	6
17	5143	86237.45	4
18	5359	89296.85	3
19	4350	72628.9	5
20	3487	58215.4	8
21	2528	42029.8	11
22	1370	22815.15	12
23	68	1121.35	13

H. % OF SALES, REVENUE AND QUANTITY BY PIZZA SIZE

```
SELECT
  P.size AS Pizza_size,
  COUNT(OD.quantity) AS Total_quantity,
  ROUND(SUM(OD.quantity * P.price), 2) AS Total_revenue,
  CONCAT(CAST(ROUND((SUM(OD.quantity * P.price) / (SELECT
                SUM(P.price * OD.quantity)
              FROM
                pizzas P
                  JOIN
                orders_details OD ON OD.pizza_id = P.pizza_id)) * 100,
            2)
        AS CHAR),
      '%') AS Revenue_contribution
FROM
  pizzas P
    JOIN
  orders_details OD ON OD.pizza_id = P.pizza_id
GROUP BY P.size
ORDER BY SUM(OD.quantity * P.price) DESC;
```

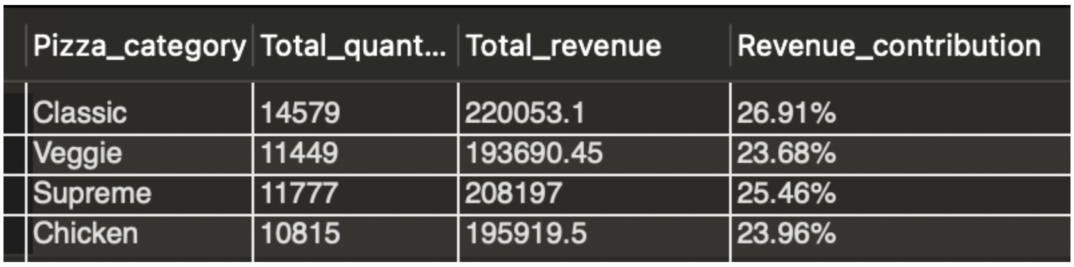


Pizza_size	Total_quant...	Total_revenue	Revenue_contribution
L	18526	375318.7	45.89%
M	15385	249382.25	30.49%
S	14137	178076.5	21.77%
XL	544	14076	1.72%
XXL	28	1006.6	0.12%

I. % OF SALES, REVENUE AND QUANTITY BY PIZZA CATEGORY

```
SELECT
  PT.category AS Pizza_category,
  COUNT(OD.quantity) AS Total_quantity,
  ROUND(SUM(OD.quantity * P.price), 2) AS Total_revenue,
  CONCAT(CAST(ROUND((SUM(OD.quantity * P.price) / (SELECT
                SUM(P.price * OD.quantity)
              FROM
                pizzas P
                  JOIN
                orders_details OD ON OD.pizza_id = P.pizza_id)) * 100,
            2)
        AS CHAR),
      '%') AS Revenue_contribution
FROM
```

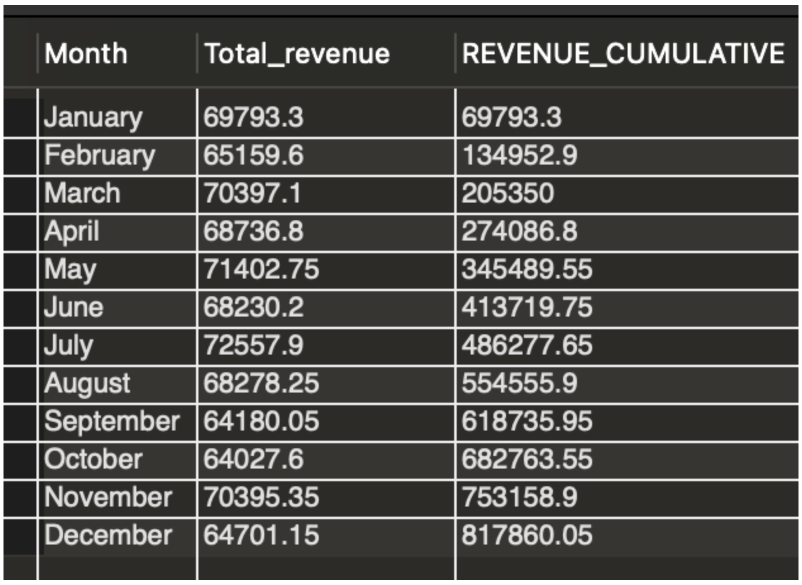
```
   pizza_types_new_clean PT
     JOIN
   pizzas P ON P.pizza_type_id = PT.pizza_type_id
     JOIN
   orders_details OD ON OD.pizza_id = P.pizza_id
GROUP BY PT.category;
```



Pizza_category	Total_quant...	Total_revenue	Revenue_contribution
Classic	14579	220053.1	26.91%
Veggie	11449	193690.45	23.68%
Supreme	11777	208197	25.46%
Chicken	10815	195919.5	23.96%

J. CUMULATIVE REVENUE GENERATED OVER MONTHS

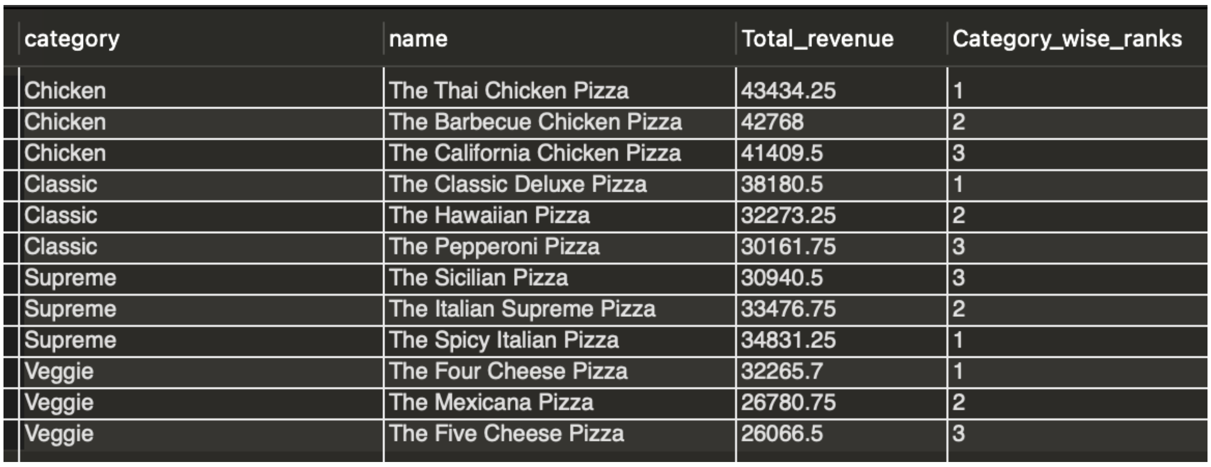
```
SELECT Month,Total_revenue, ROUND(Cumulative_revenue,2) AS
REVENUE_CUMULATIVE FROM
(SELECT MONTHNAME(O.order_date) AS Month,
ROUND(SUM(P.price*OD.quantity),2) AS Total_revenue,
     SUM(SUM(P.price*OD.quantity)) OVER(ORDER BY MONTH(O.order_date))
AS Cumulative_revenue
     FROM
   orders O
     JOIN
   orders_details OD ON O.order_id = OD.order_id
     JOIN
   pizzas P ON OD.pizza_id = P.pizza_id
GROUP BY MONTHNAME(O.order_date), MONTH(O.order_date)
ORDER BY MONTH(O.order_date) ) AS Cumulative
;
```



Month	Total_revenue	REVENUE_CUMULATIVE
January	69793.3	69793.3
February	65159.6	134952.9
March	70397.1	205350
April	68736.8	274086.8
May	71402.75	345489.55
June	68230.2	413719.75
July	72557.9	486277.65
August	68278.25	554555.9
September	64180.05	618735.95
October	64027.6	682763.55
November	70395.35	753158.9
December	64701.15	817860.05

K. **TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY**

```
SELECT category,name, Total_revenue, Category_wise_ranks
FROM
(SELECT PT.category, PT.name, ROUND(SUM(P.price*OD.quantity),2) AS
Total_revenue,
RANK() OVER(PARTITION BY PT.category ORDER BY SUM(P.price*OD.quantity)
DESC ) AS Category_wise_ranks
FROM
pizza_types_new_clean PT
    JOIN
  pizzas P ON P.pizza_type_id = PT.pizza_type_id
    JOIN
  orders_details OD ON OD.pizza_id = P.pizza_id
GROUP BY PT.category, PT.name
ORDER BY PT.category) AS pizza_ranking
WHERE Category_wise_ranks <4 ;
```



category	name	Total_revenue	Category_wise_ranks
Chicken	The Thai Chicken Pizza	43434.25	1
Chicken	The Barbecue Chicken Pizza	42768	2
Chicken	The California Chicken Pizza	41409.5	3
Classic	The Classic Deluxe Pizza	38180.5	1
Classic	The Hawaiian Pizza	32273.25	2
Classic	The Pepperoni Pizza	30161.75	3
Supreme	The Sicilian Pizza	30940.5	3
Supreme	The Italian Supreme Pizza	33476.75	2
Supreme	The Spicy Italian Pizza	34831.25	1
Veggie	The Four Cheese Pizza	32265.7	1
Veggie	The Mexicana Pizza	26780.75	2
Veggie	The Five Cheese Pizza	26066.5	3

ANALYSIS CONCLUSION –

1. Most ordered pizza – The Classic Deluxe Pizza
2. Out of all sizes – XL and XXL doesn't contribute much in number of orders