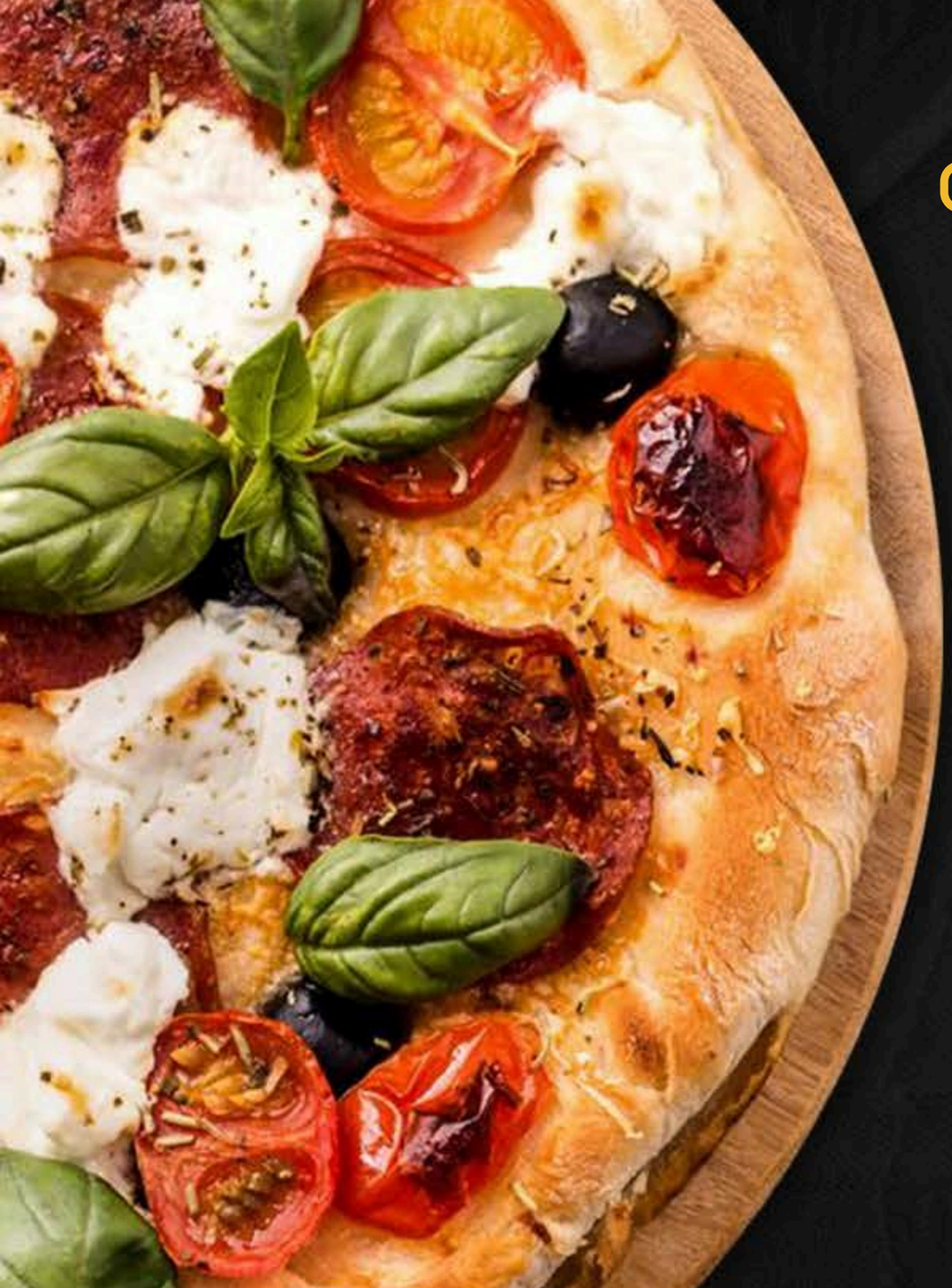
PIZZA SALES ANALYSIS with MYSQL



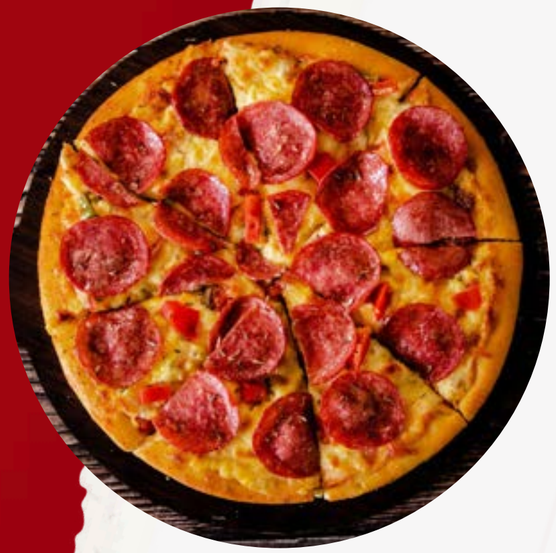
CONDUCTED BY - SHAGUFTA

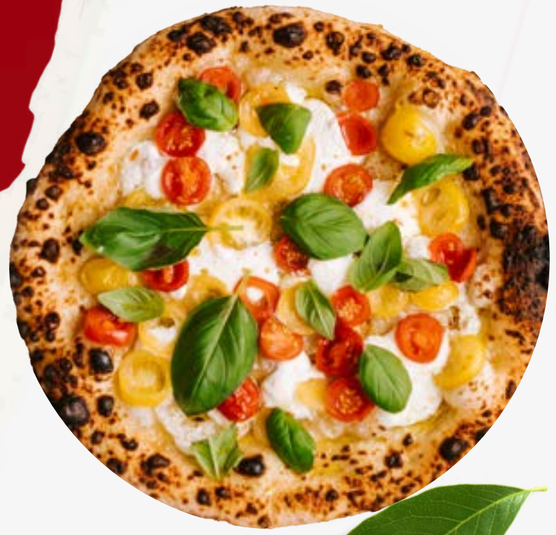
This project was conducted as part of my studies, with the aim of applying data analysis techniques to a real-world business scenario.

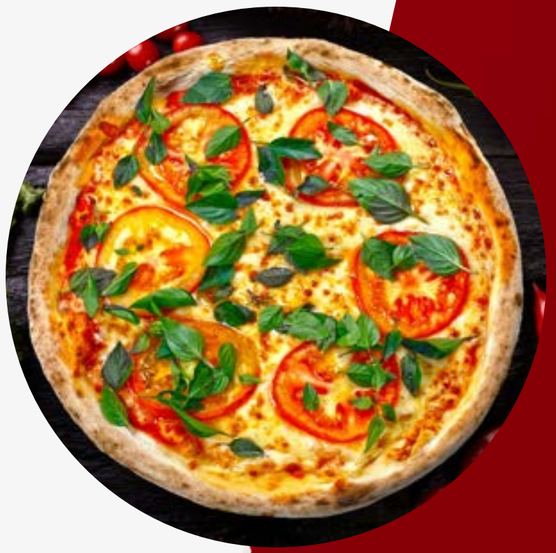
The analysis in this project is carried out through Excel and MySQL. Excel served as the first repository through which I did data cleaning then imported data into MySQL workbench which was used to perform the relevant queries to find out different patterns in the pizza sales.

BUSINESS OBJECTIVES

HOW CAN WE OPTIMISE OUR PROFIT?
HOW CAN WE TAKE THESE INSIGHTS TO BUILD RECOMMENDATIONS?







Using MySQL we will filter out relevant data to find patterns and build recommendations for the business to optimise their sales and profit.

TOTAL REVENUE GENERATED FROM PIZZA SALES

```
2 •   SELECT
3         ROUND(SUM(order_details.quantity * pizzas.price),
4                 2) AS total_revenue_generated
5     FROM
6         order_details
7             JOIN
8         pizzas ON order_details.pizza_id = pizzas.pizza_id;
```

The price and the quantity of pizzas are in two different tables so we calculated to product using INNER JOIN to find the total revenue.

Result Grid | Filter Row

	total_revenue_generated
▶	817860.05

The total revenue generated from pizza sales is $8,17,860. Now this is calculated to see how the business has been doing.

IDENTIFY THE HIGHEST PRICE PIZZA

Again the price and the name of pizzas are in different tables so we used a JOIN to combine the relevant data to find out the result.

```sql
SELECT
    pizza_types.name, pizzas.price
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY price DESC
LIMIT 1;
```

The highest price pizza is THE GREEK PIZZA which costs almost $36. A $36 pizza is on the higher end of pizza prices, indicating the business is positioned as a premium offering catering to wealthier customers.

Result Grid | Filter Ro

name	price
The Greek Pizza	35.95

TOTAL NUMBER OF ORDERS PLACED

```
2 •    SELECT
3          COUNT(order_id) AS total_orders
4      FROM
5          orders;
```

To find the total no of orders placed we used the **COUNT** function to count the no of order_ids which is a primary key (a primary key is unique and not null).

Result Grid

	total_orders
▶	21350

The total orders placed in the 17 days the business has been into place suggests a good business health.

DISTRIBUTION OF ORDERS BY HOUR OF THE DAY

```
2 •   SELECT
3         HOUR(order_time) AS hours, COUNT(order_id)
4     FROM
5         orders
6     GROUP BY hours;
```

To calculate the number of orders per hour we used the **Time Function : Hour** which can filter out the hour from a time series along with count function and group by clause to group the number of orders by hours.

Result Grid | Filter Row

hours	COUNT(order_id)
9	1
10	8
23	28
22	663
21	1198
11	1231
15	1468
14	1472
20	1642
16	1920
19	2009
17	2336
18	2399
13	2455
12	2520

The result grid shows us that the sales are best towards the lunch time and later towards the tea time in the evening and least during the morning and late night times suggsting we may take some actions to improve the sales during these times.

THE MOST COMMON PIZZA SIZE ORDERED

```
2 •  SELECT
3        count(order_details.order_details_id) AS order_count, pizzas.size
4    FROM
5        order_details
6            JOIN
7        pizzas ON order_details.pizza_id = pizzas.pizza_id
8    GROUP BY pizzas.size
9    ORDER BY order_count DESC
10   LIMIT 1;
```

To filter out the most common pizza size ordered we performs a number of functions and clauses such as **JOIN** to combine the data from two diff tables group by and **Order by** to order the results in descending(**DESC**) order of count along with **limit** to limit the no of results to 1.

Result Grid | Filter

	order_count	size
▶	18526	L

The result grid shows that the L size pizzas were ordered the most. To improve the sales on other sizes of pizzas we might create some strategies such as offering a drink or a dessert on discount on purchase of the different sizes of pizzas.

TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES

In this query we took the name from pizza_types table and sum of quantity from order_details table but since they dont have any column in common so we took another table, pizzas, to join the above two tables and then used the group by clause to group the sum of quantity on the basis of pizza names.

```
SELECT
    pizza_types.name AS name,
    SUM(order_details.quantity) AS quantity
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY name
ORDER BY quantity DESC
LIMIT 5;
```

All the top 5 most ordered pizzas are non veg pizzas indicating a meat loving crowd. we can suggest the business to expand the customer variety by introducing some veg specilty pizzas Also offer family meals around these pizzas along with cross selling items.

Result Grid | Filter Rows:

name	quantity
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED

```sql
SELECT
    pizza_types.category AS category,
    SUM(order_details.quantity) AS quantity
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY category;
```

Using the **Select** command we selected the relevant columns needed from both the tables . to sum the quantities of the pizzas from each category using the **Sum Operator** we calculated the quantity of the pizzas ordered from each category.

Result Grid | Filter

category	quantity
Classic	14888
Veggie	11649
Supreme	11987
Chicken	11050

Classics are most popular, it is essential to focus on these by introducing special or combo meals. while others are doing well to only slightly behind we can put limited time offers to let the customers try these new flavors.

THE CATEGORY-WISE DISTRIBUTION OF PIZZAS

Filtered out the category wise distribution of pizzas by using the select command and the group by clause to group the distribution of pizzas on the basis of category.

```
2 •    SELECT
3          category, COUNT(name)
4      FROM
5          pizza_types
6      GROUP BY category;
```

supreme and veggie category have most variety which is good since we have meat loving crowd and veggie needs more variety to grow. The chicken has the least variety which explains why sales in chicken category are low. we need to introduce new varieties considering trends along with offers to optimise sales in this category.

Result Grid | Filter Rows

category	COUNT(name)
Chicken	6
Classic	8
Supreme	9
Veggie	9

THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY

```sql
SELECT
    ROUND(AVG(quantity), 0) AS avg_orders_per_day
FROM
    (SELECT
        orders.order_date AS order_date,
            SUM(order_details.quantity) AS quantity
    FROM
        orders
    JOIN order_details ON orders.order_id = order_details.order_id
    GROUP BY order_date) AS order_quantity_per_day;
```

In this query we selected the sum of quantity and order date using join and grouped the the quantity of orders by order_date through group by clause to get the no of orders per day then we converted the whole query into a **Sub-Query** using parenthesis and did the **average** of the result from the sub query which gave us our final result. we also rounded the decimal point to get the whole number using the **Round function**.

Result Grid | Filter

avg_orders_per_day
138

The average volume of orders per day is 138 which is very well considering the present size and capacity of the business. However to increase the profit and sales we might consider special promotions or delivery partnerships during the off peak hours.

TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE

```sql
SELECT
    pizza_types.name AS name,
    SUM(order_details.quantity * pizzas.price) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY name
ORDER BY revenue DESC
LIMIT 3;
```

Selected the name and sum of product of quantity and price (and named it as revenue using the **As command**) from the relevant tables and joined them together to get the output and grouped the result on the basis revenue then through **Order by** and **limit** we sorted the data in descenfing order to get the top 3 results.

Result Grid | Filter Rows:

name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5

These top sellers can be paired with sides, drinks, or desserts to boost the average order value. Consider combo deals.

THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE

```sql
SELECT
    pizza_types.category AS pizza_type,
    ROUND((SUM(order_details.quantity * pizzas.price) / (SELECT
                    ROUND(SUM(order_details.quantity * pizzas.price),
                                2) AS total_revenue_generated
                FROM
                    order_details
                        JOIN
                    pizzas ON order_details.pizza_id = pizzas.pizza_id) * 100),
            2) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_type
ORDER BY revenue DESC
;
```

To get the contribution of each pizza category we performed an advanced mysql query using multiple joins and functions like round , sum, sub query , group by, and order by.

Result Grid | Filter

pizza_type	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

Since Chicken and Veggie pizzas contribute slightly less to total revenue, offering limited-time offers or introducing new varieties in these categories could help balance the sales mix and increase their contribution.

THE CUMULATIVE REVENUE GENERATED OVER TIME

```
2  select
3  order_date , sum(revenue) over(order by order_date) as cum_revenue
4  from (select orders.order_date, sum(pizzas.price * order_details.quantity)
5  as revenue from order_details
6  join pizzas on order_details.pizza_id = pizzas.pizza_id
7  join orders on order_details.order_id = orders.order_id
8  group by order_date) as sales ;
```

Result Grid | Filter Rows:

order_date	cum_revenue
2015-01-01	2713.8500000000004
2015-01-02	5445.75
2015-01-03	8108.15
2015-01-04	9863.6
2015-01-05	11929.55
2015-01-06	14358.5
2015-01-07	16560.7
2015-01-08	19399.05
2015-01-09	21526.4
2015-01-10	23990.350000000002
2015-01-11	25862.65
2015-01-12	27781.7
2015-01-13	29831.300000000003
2015-01-14	32358.700000000004
2015-01-15	34343.50000000001
2015-01-16	36937.65000000001
2015-01-17	39001.75000000001
2015-01-18	40978.600000000006

In this query we calculated the cumulative revenue through a **sub-query**. Using the sub query we calculated the revenue grouped on the basis of order_date and named the resultant table as sales then using the subquery we calculated sum of revenue over time using the **over command** with order_date.

Now this was calculated to see how sales evolve over time, helping to monitor business progress. Cumulative revenue helps predict future sales and plan resources accordingly. this also Reveals sales trends or spikes, aiding in strategic planning.

THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY

```sql
select name , revenue from
(select category , name , revenue ,
rank() over(partition by category order by revenue desc) as rn
from
(select pizza_types.category , pizza_types.name ,
sum((order_details.quantity) * pizzas.price) as revenue from pizza_types
join pizzas on pizza_types.pizza_type_id = pizzas.pizza_type_id
join order_details on order_details.pizza_id = pizzas.pizza_id
group by pizza_types.category , pizza_types.name) as a) as b
where rn <=3;
```

Result Grid | Filter Rows:

name	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 Hawaiian Pizza	32273.25
The Pepperoni Pizza	30161.75
The Spicy Italian Pizza	34831.25
The Italian Supreme Pizza	33476.75
The Sicilian Pizza	30940.5
The Four Cheese Pizza	32265.70000000065
The Mexicana Pizza	26780.75
The Five Cheese Pizza	26066.5

Calculated the revenue based on category through joining the tables and grouping the results by category and name and converted the query into a subquery named as a. To order the results into a rank we used the **rank command** with over and partition by then again put the whole as sub query to select the final desired columns using where we filtered the rank upto top 3.

The chicken pizza even though has lower order rate than other categories produces the most revenue. Higher-priced chicken pizzas may lead to higher average order values, providing opportunities for upselling or cross-selling. If customers are willing to pay more for chicken pizzas, they might also be interested in sides, drinks, or desserts that complement their orders.

INSIGHTS

- Chicken pizzas are performing very well. Continue expanding this category with **new variations** (e.g., spicy chicken, grilled chicken, or exotic flavors) to maintain customer interest and grow sales keeping the menu fresh.
- **Customer Feedback:** Regularly ask for customer feedback on new menu items or service quality. This will help you refine your offerings while making customers feel valued.
- Introduce new flavors in the best-selling pizzas to offer limited-time offers during late night hours with **combo meals** During the morning **off-peak time offer** a limited-time offer within veggie pizzas along with coffee or other good drinks under the healthy breakfast for office-going people.
- Use **social media** to showcase your top-performing pizzas, share customer reviews, and promote limited-time offers. Contests and giveaways can also drive engagement. Depending on the **season**, sales may fluctuate (holidays, summer, etc.). Plan ahead with **seasonal promotions.**
- Not actively engaging with customers can lead to a lack of loyalty and repeat business. Implement **loyalty programs**, actively solicit customer feedback, and maintain communication through newsletters or social media to **build relationships with customers.**
- Considering the average price of pizzas is quite high; less affordable for students. Introduce smaller, more affordable pizza sizes specifically for students. A personal-sized pizza could fit their budget while providing a quick meal.

THANK YOU!

for your time to go through my project.