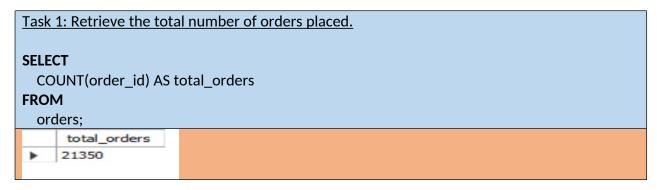
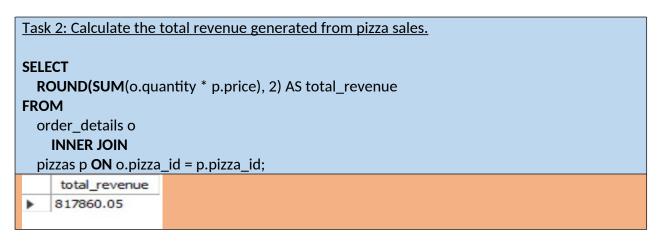
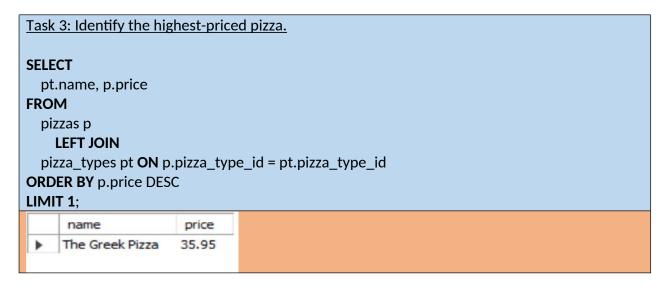
## **DATA EXPLORATION**

This category gives a clear picture of customer demand patterns and sets the foundation for deeper analysis into volume, revenue, pricing, and customer ordering behaviour.







```
Task 4: Identify the most common pizza size ordered.

SELECT
p.size, SUM(od.quantity) AS no_of_size_ordered
FROM
```

```
pizzas p
JOIN
order_details od ON p.pizza_id = od.pizza_id
GROUP BY p.size
ORDER BY no_of_size_ordered DESC;

size no_of_size_ordered
L 18956
M 15635
S 14403
XL 552
XXL 28
```

- 1. The store has received a **large number of total orders**, showing that demand for pizzas is steady and customers are regularly engaged.
- 2. The **total revenue** indicates strong sales performance, reflecting both good customer traffic and effective pricing.
- 3. The **highest-priced pizza** stands out as a premium product. While it may not be the most ordered, it adds significant value when sold.
- 4. The **most common pizza size** is Medium, which shows that customers prefer a balance between value and portion size.

## **SALES ANALYSIS- CRUNCHING THE NUMBERS**

This category helps optimize product offerings and operational efficiency based on demand patterns and revenue impact.

```
Task 1: List the top 5 most ordered pizza types along with their quantities.

SELECT
pt.name, SUM(od.quantity) AS ordered_quantity

FROM
pizza_types pt
LEFT JOIN
pizzas p ON pt.pizza_type_id = p.pizza_type_id
LEFT JOIN
order_details od ON od.pizza_id = p.pizza_id

GROUP BY pt.name
ORDER BY ordered_quantity DESC
LIMIT 5;
```

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

```
Task 2: Determine the distribution (hourly orders/total orders) of orders by hour of the day.
```

# SELECT \* ,sum(hor

,sum(hourly\_orders) over () as total\_orders

,hourly\_orders \*100/sum(hourly\_orders) **over() as** distribution

FROM(

**SELECT** 

HOUR(time) AS hour\_of\_day,

**COUNT(DISTINCT** order\_id) AS hourly\_orders

**FROM** 

orders

#### **GROUP BY HOUR(time)**

) as a;

	hour_of_day	hourly_orders	total_orders	distribution
١	9	1	21350	0.0047
	10	8	21350	0.0375
	11	1231	21350	5.7658
	12	2520	21350	11.8033
	13	2455	21350	11.4988
	14	1472	21350	6.8946
	15	1468	21350	6.8759

#### Task 3: Determine the top 3 most ordered pizza types based on revenue.

#### **SELECT**

```
pt.pizza_type_id,
```

pt.name,

**SUM**(p.price \* od.quantity) **AS** revenue

#### **FROM**

pizza\_types pt

**LEFT JOIN** 

pizzas p **ON** pt.pizza\_type\_id = p.pizza\_type\_id

**LEFT JOIN** 

order\_details od ON p.pizza\_id = od.pizza\_id

**GROUP BY** pt.pizza\_type\_id , pt.name

**ORDER BY** revenue **DESC** 

LIMIT 3;

	pizza_type_id	name	revenue
•	thai_ckn	The Thai Chicken Pizza	43434.25
	bbq_ckn	The Barbecue Chicken Pizza	42768
	cali_ckn	The California Chicken Pizza	41409.5

- 1. The **top 5 most ordered pizzas** account for a major share of total sales. These pizzas are customer favourites and should always be prioritized in inventory and preparation.
- 2. Order trends by **hour of the day** confirm that sales peak during the whole day, highlighting the importance of managing staff and kitchen operations during these hours.
- 3. The **top revenue-generating pizzas** are not always the most ordered ones. Premium pizzas with higher prices contribute strongly to overall earnings.

## **OPERATIONAL INSIGHTS**

This category provides financial health indicators and strategic revenue concentration insights.

```
Task 1: Calculate the percentage contribution of each pizza type to total revenue.
with pizza_type_rev as (
SELECT
  pt.name, ROUND(SUM(p.price * od.quantity), 2) AS revenue
FROM
  pizza_types pt
    LEFT JOIN
  pizzas p ON pt.pizza_type_id = p.pizza_type_id
    LEFT JOIN
  order_details od ON p.pizza_id = od.pizza_id
GROUP BY pt.name
ORDER BY revenue DESC)
SELECT *,
round(sum(revenue) over(),2) as total_revenue,
round(revenue * 100.00/sum(revenue) over(),2) as distribution
from pizza_type_rev;
```

	name	revenue	total_revenue	distribution
•	The Thai Chicken Pizza	43434.25	817860.05	5.31
	The Barbecue Chicken Pizza	42768	817860.05	5.23
	The California Chicken Pizza	41409.5	817860.05	5.06
	The Classic Deluxe Pizza	38180.5	817860.05	4.67
	The Spicy Italian Pizza	34831.25	817860.05	4.26
	The Southwest Chicken Pizza	34705.75	817860.05	4.24
	The Italian Supreme Pizza	33476.75	817860.05	4.09

```
<u>Task 2: Analyze the cumulative revenue generated over time.</u>
with cr_overtime as(
SELECT
  o.date, ROUND(SUM(p.price * od.quantity), 2) AS revenue
FROM
  order_details od
    LEFT JOIN
  pizzas p ON od.pizza_id = p.pizza_id
    LEFT JOIN
  orders o ON od.order_id = o.order_id
group by o.date
order by o.date)
select *,
round(sum(revenue) over( order by date rows between unbounded preceding and current
row),2) as cumulative_revenue
from cr_overtime;
    date
             revenue cumulative_revenue
    2015-01-01 2713.85 2713.85
    2015-01-02 2731.9 5445.75
    2015-01-03 2662.4 8108.15
    2015-01-04 1755.45 9863.6
    2015-01-05 2065.95 11929.55
```

```
Task 3: Determine the top 3 most ordered pizza types based on revenue for each pizza category.

select *
from(
select *,
dense_rank() over(partition by category order by revenue desc) as rank_on_revenue
from(
SELECT
pt.category,
pt.name,
ROUND(SUM(p.price * od.quantity), 2) AS revenue
FROM
```

2015-01-06 2428.95 14358.5 2015-01-07 2202.2 16560.7

```
pizza_types pt
    LEFT JOIN
  pizzas p ON pt.pizza_type_id = p.pizza_type_id
    LEFT JOIN
  order_details od ON p.pizza_id = od.pizza_id
GROUP BY pt.category, pt.name
)as tabl
las tab
where rank_on_revenue<=3;</pre>
     category name
                                               rank_on_revenue
                                      revenue
              The Thai Chicken Pizza
    Chicken
                                      43434.25
    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 Spicy Italian Pizza
                                      34831.25
```

- 1. A few pizzas contribute to the **majority of revenue**, while others contribute very little. This indicates that the menu could be optimized.
- 2. The **cumulative revenue trend** shows steady growth with predictable spikes.
- 3. Within each category, only a handful of pizzas dominate sales. This highlights opportunities to streamline the menu and focus on bestsellers.

# **CATEGORY-WISE ANALYSIS**

```
Task 1: Join the necessary tables to find the total quantity of each pizza category
ordered.
SELECT
  pt.category, SUM(od.quantity) AS total_quantity
FROM
  pizza_types pt
    LEFT JOIN
  pizzas p ON pt.pizza_type_id = p.pizza_type_id
  order_details od ON p.pizza_id = od.pizza_id
GROUP BY pt.category;
      category
                  total_quantity
     Chicken
                 11050
     Classic
                 14888
                 11987
     Supreme
     Veggie 11649
```

```
Task 2: Join relevant tables to find the category-wise distribution of pizzas.
with Pizza_table as
( SELECT
  pt.category, SUM(od.quantity) AS total_quantity
FROM
  pizza_types pt
    LEFT JOIN
  pizzas p ON pt.pizza_type_id = p.pizza_type_id
    LEFT JOIN
  order_details od ON p.pizza_id = od.pizza_id
GROUP BY pt.category
) select *,
round(total_quantity*100/sum(total_quantity) over(),2) as distribution
from pizza_table;
     category total_quantity
                              distribution
    Chicken
               11050
                             22,29
    Classic
              14888
                             30.03
              11987
                             24.18
    Supreme
             11649
                             23.50
    Veggie
```

```
Task 3: Group the orders by the date and calculate the average number of pizzas
ordered per day.
SELECT
  AVG(total_quantity) AS avg_no_pizza
FROM
  (SELECT
    o.date, SUM(od.quantity) AS total_quantity
  FROM
    orders o
  LEFT JOIN order_details od ON o.order_id = od.order_id
  GROUP BY o.date) AS table1;
    avg_no_pizza
 138.4749
Write a query to display each order id along with the order month and year.
SELECT
  order_id,
```

**MONTH**(date) AS order\_month,

YEAR(date) AS order\_year

**FROM** 

```
orders
   order_id order_month
                          order_year
  1
                          2015
  2
                          2015
            1
   3
                          2015
   4
            1
                          2015
   5
            1
                          2015
  6
            1
                          2015
                          2015
```

```
Find the sales according to the hour of the day. (Bussiest hour on the top)
SELECT
  HOUR(o.time) AS busy_hour,
  COUNT(o.order_id) AS total_order,
  ROUND(SUM(od.quantity * p.price), 2) AS total_sales
FROM
  orders o
    LEFT JOIN
  order_details od ON o.order_id = od.order_id
    LEFT JOIN
  pizzas p ON od.pizza_id = p.pizza_id
GROUP BY busy_hour
ORDER BY total_order DESC
    busy_hour
              total order total sales
   12
              6543
                         111877.9
   13
              6203
                         106065.7
   18
              5359
                         89296.85
   17
              5143
                         86237.45
   19
              4350
                         72628.9
   16
              4185
                         70055.4
   14
              3521
                         59201.4
```

```
Compare sales between weekdays and weekends.

SELECT
CASE
WHEN DAYOFWEEK(o.date) IN (1,7) THEN 'Weekend'
ELSE 'Weekday'
END AS Day_sales,
ROUND(SUM(od.quantity * p.price), 2) AS total_sales
FROM
```

```
orders o
LEFT JOIN
order_details od ON o.order_id = od.order_id
LEFT JOIN
pizzas p ON od.pizza_id = p.pizza_id
GROUP BY Day_sales
;

Day_sales total_sales
Weekday 595474.15
Weekend 222385.9
```

```
Find average daily sales.
SELECT
  order_day, ROUND(AVG(total_sales), 2) AS avg_sales
FROM
  (SELECT
    DATE(o.date) AS order_day,
      SUM(od.quantity * p.price) AS total_sales
  FROM
    orders o
  LEFT JOIN order details od ON o.order id = od.order id
  LEFT JOIN pizzas p ON od.pizza_id = p.pizza_id
  GROUP BY order_day
  ORDER BY order_day) AS table1
GROUP BY order_day;
    order_day avg_sales
   2015-01-01 2713.85
   2015-01-02 2731.9
   2015-01-03 2662.4
   2015-01-04 1755.45
   2015-01-05 2065.95
   2015-01-06 2428.95
   2015-01-07 2202.2
```

```
Find the month with the highest number of orders.

SELECT

MONTH(date) AS order_month,

COUNT(DISTINCT order_id) AS total_orders

FROM

orders

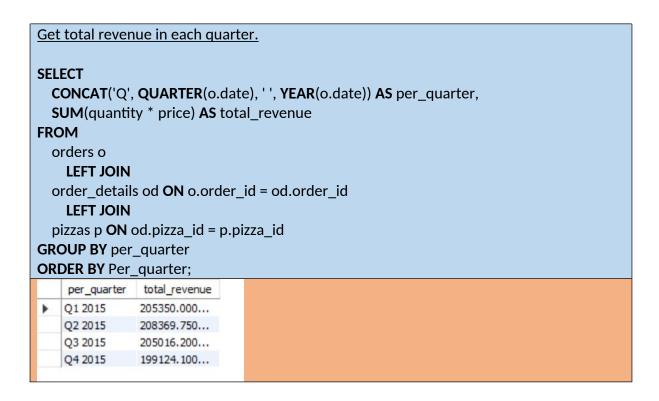
GROUP BY order_month

ORDER BY total_orders DESC

LIMIT 1;
```

```
order_month total_orders

7 1935
```



- 1. The **busiest hours** are during the lunch and in the evenings, matching typical dining patterns.
- 2. **Weekday sales** are higher compared to weekends, showing that customers order more when they are working.
- 3. Certain **months and quarters perform better**, likely due to seasonal demand, holidays, or promotions.
- 4. **Daily sales remain stable overall**, making it easier to forecast and plan inventory.