

Pizza Hut Sales Analysis – SQL Project

Pizza Hut Sales Analysis

Project Overview:

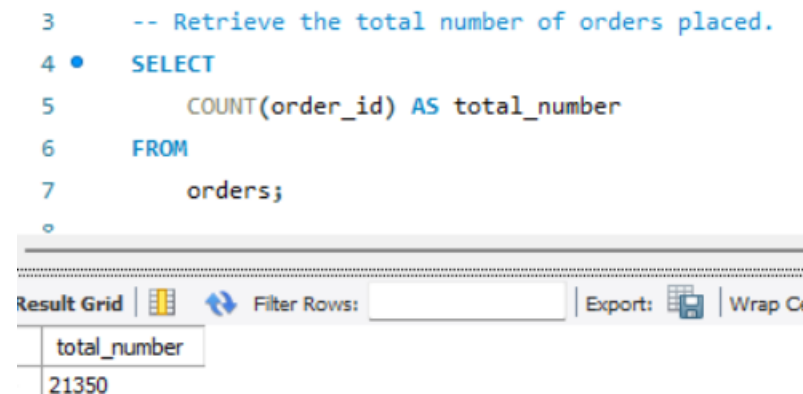
Analysed Pizza Hut sales data using MySQL to uncover trends in orders, revenue, and popular pizzas. Includes category-wise insights, peak order hours, and revenue analysis to support data-driven business decisions.

Q1: Retrieve the total number of orders placed

SQL Query:

```
SELECT COUNT(order_id) AS total_number
FROM orders;
```

```
3      -- Retrieve the total number of orders placed.
4 •    SELECT
5          COUNT(order_id) AS total_number
6      FROM
7          orders;
8
```



total_number
21350

Insight: The total number of orders reflects overall customer engagement. High order volume suggests strong demand and potential for upselling or promotions.

Q2: Calculate the total revenue generated from pizza sales

SQL Query:

```
SELECT ROUND(SUM(o.quantity * p.price), 2) AS total_sales
FROM pizzas AS p
JOIN order_details AS o ON p.pizza_id = o.pizza_id;
```

```

9      -- Calculate the total revenue generated from pizza sales.
10
11  •   SELECT
12      ROUND(SUM(o.quantity * p.price), 2) AS total_sales
13  FROM
14      pizzas AS p
15      JOIN
16      order_details AS o ON p.pizza_id = o.pizza_id;
17

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
total_sales			
817860.05			

Insight: Total revenue indicates store financial performance. This metric serves as a benchmark for growth tracking and profitability evaluation.

Q3: Identify the highest-priced pizza

SQL Query:

```

SELECT pt.name, p.price
FROM pizza_types AS pt
INNER JOIN pizzas AS p ON pt.pizza_type_id = p.pizza_type_id
ORDER BY p.price DESC
LIMIT 1;

```

```

19      -- Identify the highest-priced pizza.
20
21  •   SELECT
22      pt.name, p.price
23  FROM
24      pizza_types AS pt
25      INNER JOIN
26      pizzas AS p ON pt.pizza_type_id = p.pizza_type_id
27  ORDER BY p.price DESC
28  LIMIT 1;
29

```

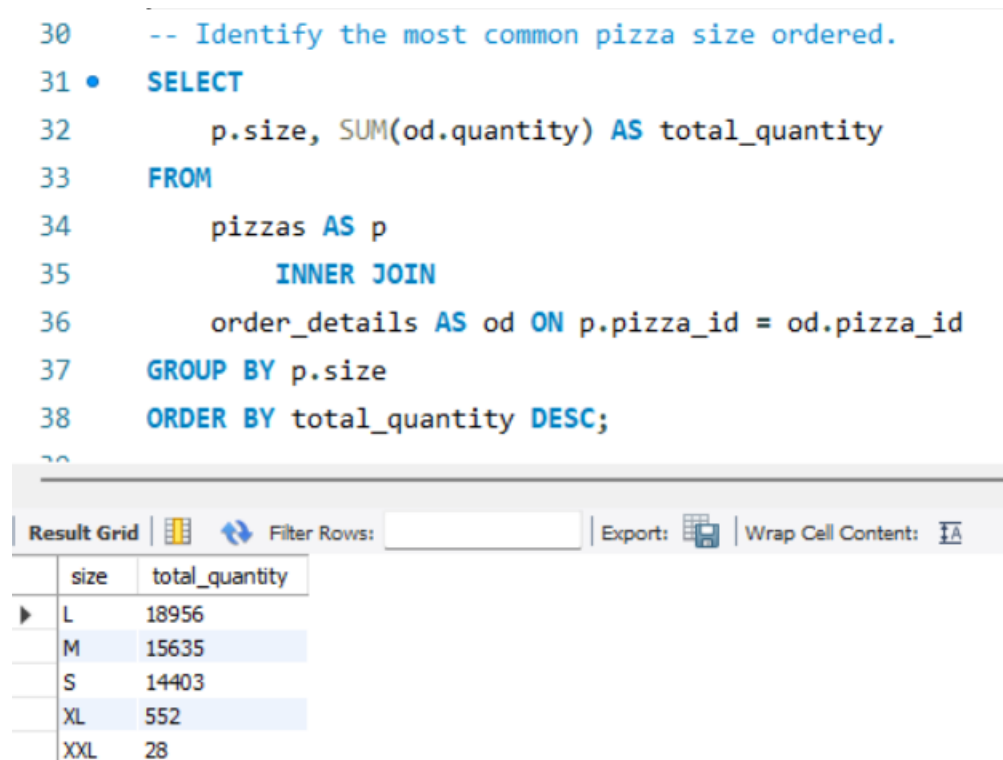
Result Grid	Filter Rows:	Export:	Wrap Cell Content:
total_sales			
▶ 817860.05			

Insight: The highest-priced pizza represents the premium segment. Insights here guide marketing, pricing strategies, and promotions targeting high-value customers.

Q4: Identify the most common pizza size ordered

SQL Query:

```
SELECT p.size, SUM(od.quantity) AS total_quantity
FROM pizzas AS p
INNER JOIN order_details AS od ON p.pizza_id = od.pizza_id
GROUP BY p.size
ORDER BY total_quantity DESC;
```



The screenshot shows a SQL query editor with a query to identify the most common pizza size ordered. Below the query, the results grid displays the output.

```
30  -- Identify the most common pizza size ordered.
31  •  SELECT
32      p.size, SUM(od.quantity) AS total_quantity
33  FROM
34      pizzas AS p
35      INNER JOIN
36      order_details AS od ON p.pizza_id = od.pizza_id
37  GROUP BY p.size
38  ORDER BY total_quantity DESC;
```

size	total_quantity
L	18956
M	15635
S	14403
XL	552
XXL	28

Insight: Shows customer preference for pizza size. Dominance of a particular size, e.g., “Large”, informs inventory planning and marketing focus.

Q5: List the top 5 most ordered pizza types along with their quantities

SQL Query:

```
SELECT pt.name, SUM(od.quantity) AS total_qty_ordered
FROM pizza_types AS pt
INNER JOIN pizzas AS pz ON pt.pizza_type_id = pz.pizza_type_id
```

```

INNER JOIN order_details AS od ON pz.pizza_id = od.pizza_id
GROUP BY pt.name
ORDER BY total_qty_ordered DESC
LIMIT 5;

```

```

40  -- List the top 5 most ordered pizza types along with their quantities.
41 •  SELECT
42      pt.name, SUM(od.quantity) AS total_qty_ordered
43  FROM
44      pizza_types AS pt
45      INNER JOIN
46      pizzas AS pz ON pt.pizza_type_id = pz.pizza_type_id
47      INNER JOIN
48      order_details AS od ON pz.pizza_id = od.pizza_id
49  GROUP BY pt.name
50  ORDER BY total_qty_ordered DESC
51  LIMIT 5;
52

```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
	name	total_qty_ordered			
▶	The Classic Deluxe Pizza	2453			
	The Barbecue Chicken Pizza	2432			
	The Hawaiian Pizza	2422			
	The Pepperoni Pizza	2418			
	The Thai Chicken Pizza	2371			

Insight: Identifies best-selling pizzas driving sales. Key for stock planning, promotional campaigns, and menu optimization.

Q6: Total quantity of each pizza category ordered

SQL Query:

```

SELECT pt.category, SUM(od.quantity) AS total_quantity
FROM pizza_types AS pt
INNER JOIN pizzas AS pz ON pt.pizza_type_id = pz.pizza_type_id
INNER JOIN order_details AS od ON pz.pizza_id = od.pizza_id
GROUP BY pt.category
ORDER BY total_quantity DESC;

```

```

53  -- Join the necessary tables to find the total quantity of each pizza category ordered.
54  • SELECT
55      pt.category, SUM(od.quantity) AS total_quantity
56  FROM
57      pizza_types AS pt
58      INNER JOIN
59      pizzas AS pz ON pt.pizza_type_id = pz.pizza_type_id
60      INNER JOIN
61      order_details AS od ON pz.pizza_id = od.pizza_id
62  GROUP BY pt.category
63  ORDER BY total_quantity DESC;

```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
category	total_quantity			
Classic	14888			
Supreme	11987			
Veggie	11649			
Chicken	11050			

Insight: Category-wise orders highlight the popularity of each pizza category. This supports inventory allocation and category-focused marketing.

Q7: Distribution of orders by hour of the day

SQL Query:

```

SELECT HOUR(order_time) as Order_Hour, COUNT(order_id) AS Order_Count
FROM orders
GROUP BY HOUR(order_time)
ORDER BY Order_Count DESC;

```

```

66  -- Determine the distribution of orders by hour of the day.
67  •  SELECT
68      HOUR(order_time) as Order_Hour, COUNT(order_id) AS Order_Count
69  FROM
70      orders
71  GROUP BY HOUR(order_time)
72  ORDER BY Order_Count DESC;
73

```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	Order_Hour	Order_Count			
▶	12	2520			
	13	2455			
	18	2399			
	17	2336			
	19	2009			
	16	1920			
	20	1642			
	14	1472			

Insight: Peak ordering hours indicate when customers are most active. Useful for staffing, delivery scheduling, and time-based promotions.

Q8: Category-wise distribution of pizzas

SQL Query:

```

SELECT pt.category, COUNT(p.pizza_id) AS total_count
FROM pizza_types AS pt
JOIN pizzas AS p ON pt.pizza_type_id = p.pizza_type_id
GROUP BY pt.category;

```

```

74 -- Join relevant tables to find the category-wise distribution of pizzas.
75 • SELECT
76     pt.category,
77     COUNT(p.pizza_id) AS total_count
78 FROM pizza_types AS pt
79 JOIN pizzas AS p
80     ON pt.pizza_type_id = p.pizza_type_id
81 GROUP BY pt.category;
82

```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
category	total_count			
Chicken	18			
Classic	26			
Supreme	25			
Veggie	27			

Insight: Shows the total number of pizzas in each category, reflecting menu composition and variety.

Q9: Average number of pizzas ordered per day

SQL Query:

```

SELECT ROUND(AVG(daily_quantity), 0) AS avg_pizzas_per_day
FROM (
    SELECT o.order_date, SUM(od.quantity) AS daily_quantity
    FROM orders AS o
    INNER JOIN order_details AS od ON o.order_id = od.order_id
    WHERE od.quantity IS NOT NULL
    GROUP BY o.order_date
) AS daily_totals;

```

```

83 -- Group the orders by date and calculate the average number of pizzas ordered per day.
84 • SELECT
85     ROUND(AVG(daily_quantity), 0) AS avg_pizzas_per_day
86 FROM
87     (SELECT
88         o.order_date, SUM(od.quantity) AS daily_quantity
89     FROM
90         orders AS o
91     INNER JOIN order_details AS od ON o.order_id = od.order_id
92     WHERE
93         od.quantity IS NOT NULL
94     GROUP BY o.order_date) AS daily_totals;

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
avg_pizzas_per_day			
▶ 138			

Insight: Provides a baseline daily demand for inventory and forecasting. Useful for detecting seasonal trends.

Q10: Top 3 most ordered pizza types based on revenue

SQL Query:

```

SELECT pt.name AS pizza_name, ROUND(SUM(od.quantity * p.price), 2) AS total_r
venue
FROM orders AS o
INNER JOIN order_details AS od ON o.order_id = od.order_id
INNER JOIN pizzas AS p ON od.pizza_id = p.pizza_id
INNER JOIN pizza_types AS pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.name
ORDER BY total_revenue DESC
LIMIT 3;

```



```

97  -- Determine the top 3 most ordered pizza types based on revenue.
98  •  SELECT
99      pt.name AS pizza_name,
100      ROUND(SUM(od.quantity * p.price), 2) AS total_revenue
101  FROM
102      orders AS o
103      INNER JOIN
104      order_details AS od ON o.order_id = od.order_id
105      INNER JOIN
106      pizzas AS p ON od.pizza_id = p.pizza_id
107      INNER JOIN
108      pizza_types AS pt ON p.pizza_type_id = pt.pizza_type_id
109  GROUP BY pt.name
110  ORDER BY total_revenue DESC
111  LIMIT 3;
112

```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
	pizza_name	total_revenue			
▶	The Thai Chicken Pizza	43434.25			
	The Barbecue Chicken Pizza	42768			
	The California Chicken Pizza	41409.5			

Insight: Identifies revenue-driving pizzas. These items are crucial for profitability and marketing focus.

Q11: Percentage contribution of each pizza type to total revenue

SQL Query:

```

SELECT pt.category, ROUND(
    SUM(od.quantity * p.price) / (
        SELECT SUM(od.quantity * p.price)
        FROM order_details AS od
        JOIN pizzas AS p ON od.pizza_id = p.pizza_id
    ) * 100, 2) AS pct_of_total
FROM pizza_types AS pt
JOIN pizzas AS p ON pt.pizza_type_id = p.pizza_type_id
JOIN order_details AS od ON od.pizza_id = p.pizza_id
GROUP BY pt.category
ORDER BY pct_of_total DESC;

```

```

113 -- Calculate the percentage contribution of each pizza type to total revenue.
114 • SELECT
115     pt.category,
116     ROUND(
117         SUM(od.quantity * p.price) / (
118             SELECT SUM(od.quantity * p.price)
119             FROM order_details AS od
120             JOIN pizzas AS p
121                 ON od.pizza_id = p.pizza_id) * 100,2) AS pct_of_total
122 FROM
123     pizza_types AS pt
124 JOIN pizzas AS p
125     ON pt.pizza_type_id = p.pizza_type_id
126 JOIN order_details AS od
127     ON od.pizza_id = p.pizza_id
128 GROUP BY pt.category
129 ORDER BY pct_of_total DESC;
130

```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	category	pct_of_total			
▶	Classic	26.91			
	Supreme	25.46			
	Chicken	23.96			
	Veggie	23.68			

Insight: Shows revenue concentration by category. High-percentage categories can be prioritized for promotions and inventory management.

Q12: Analyze cumulative revenue over time (Subquery)

SQL Query:

```

SELECT order_date, SUM(revenue) OVER(ORDER BY order_date) AS total
FROM (
    SELECT o.order_date, SUM(od.quantity * p.price) AS revenue
    FROM order_details AS od
    JOIN pizzas AS p ON od.pizza_id = p.pizza_id
    JOIN orders AS o ON o.order_id = od.order_id
    GROUP BY o.order_date
) AS Sales;

```

```

133  -- Using Subquery
134  • Select order_date, round(sum(revenue) over(order by order_date),2) as total
135  from
136  (Select o.order_date, sum(od.quantity*p.price) as revenue
137   from order_details as od
138   join pizzas as p
139   on od.pizza_id=p.pizza_id
140   join orders as o
141   on o.order_id=od.order_id
142   group by o.order_date) as Sales;
143

```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	order_date	total			
▶	2015-01-01	2713.85			
	2015-01-02	5445.75			
	2015-01-03	8108.15			
	2015-01-04	9863.6			
	2015-01-05	11929.55			

Insight: Tracks growth of revenue cumulatively. Helps monitor trends and measure the impact of campaigns or promotions over time.

Q13: Analyze cumulative revenue over time (CTE)

SQL Query:

```

WITH daily_revenue AS (
    SELECT o.order_date, ROUND(SUM(od.quantity * p.price), 2) AS revenue
    FROM order_details AS od
    JOIN pizzas AS p ON od.pizza_id = p.pizza_id
    JOIN orders AS o ON o.order_id = od.order_id
    GROUP BY o.order_date
)
SELECT order_date, revenue, SUM(revenue) OVER(ORDER BY order_date) AS cumulative_revenue
FROM daily_revenue;

```

```

144 -- Using CTE
145
146 • with daily_revenue as(Select o.order_date, round(sum(od.quantity*p.price),2) as revenue
147   from order_details as od
148   join pizzas as p
149   on od.pizza_id=p.pizza_id
150   join orders as o
151   on o.order_id=od.order_id
152   group by o.order_date)
153
154   Select order_date,revenue,
155   sum(revenue) over(order by order_date) as cumulative_revenue
156   from daily_revenue;

```

	order_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

Insight: Same as above, but modular using CTEs. Cleaner structure for reporting cumulative revenue trends.

Q14: Top 3 most ordered pizza types based on revenue for each pizza category

SQL Query:

```

WITH pizza_revenue AS (
    SELECT pt.name, pt.category, SUM(od.quantity * p.price) AS revenue
    FROM pizzas AS p
    JOIN pizza_types AS pt ON p.pizza_type_id = pt.pizza_type_id
    JOIN order_details AS od ON od.pizza_id = p.pizza_id
    GROUP BY pt.name, pt.category
),
ranked_pizzas AS (
    SELECT name, category, revenue,
           RANK() OVER(PARTITION BY category ORDER BY revenue DESC) AS rn
    FROM pizza_revenue
)
SELECT category, name, ROUND(revenue, 2) AS total_revenue
FROM ranked_pizzas
WHERE rn <= 3
ORDER BY category, revenue DESC;

```

```

159 -- Determine the top 3 most ordered pizza types based on revenue for each pizza category.
160 • WITH pizza_revenue as
161     (Select
162      pt.name,pt.category,
163      sum(od.quantity*p.price)as revenue
164      from pizzas as p
165      join pizza_types as pt
166      on p.pizza_type_id=pt.pizza_type_id
167      join order_details as od on
168      od.pizza_id=p.pizza_id
169      group by pt.name, pt.category),
170 ranked_pizzas as
171     (select
172      name,category,revenue,
173      rank()over(partition by category order by revenue desc) as rn
174      from pizza_revenue)
175 SELECT
176     category, name, ROUND(revenue, 2) AS total_revenue
177 FROM
178     ranked_pizzas
179 WHERE
180     rn <= 3
181 ORDER BY category , revenue DESC;

```

Insight: Provides category-specific top performers. Useful for targeted marketing and inventory optimization per pizza category.
