



PIZZA HUT

HELLO, I AM HARSHAVARDHAN. I HAVE EMPLOYED SQL QUERIES TO ANALYZE AND ADDRESS DATA-RELATED QUESTIONS CONCERNING PIZZA SALES PERFORMANCE.

PIZZA SALES

PRESENTED BY
HARSHAVARDHAN



- 1.RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.
- 2.CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.
- 3.IDENTIFY THE HIGHEST-PRICED PIZZA.
- 4.IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.
- 5.LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES.
- 6.JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED.
- 7.DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.
- 8.JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.
- 9.GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY.
- 10.DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.

- 11.CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.
- 12.ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.
- 13.DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY.

1.RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

The screenshot shows the MySQL Workbench interface. In the SQL editor tab, a query is written:

```
1 -- Retrieve the total number of orders placed.  
2 • select count(order_id) as total_orders from orders;
```

The Result Grid shows the output of the query:

total_orders
21350

The Action Output panel shows the query execution details:

#	Time	Action	Message	Duration / Fetch
1	23:08:58	select count(order_id) as total_orders from orders	1 row(s) returned	0.109 sec / 0.000 sec

2.CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

The screenshot shows the MySQL Workbench interface. The SQL editor tab contains the following query:

```
1 -- Calculate the total revenue generated from pizza sales.
2 SELECT
3     ROUND(SUM(order_details.quantity * pizzas.price),
4           2) AS totalrevenue
5 FROM
6     order_details
7     JOIN
8     pizzas
9 WHERE
10    order_details.pizza_id = pizzas.pizza_id;
```

The Result Grid pane shows the output of the query:

totalrevenue
817860.05

The Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
1	23:08:58	select count(order_id) as total_orders from orders	1 row(s) returned	0.109 sec / 0.000 sec
2	23:12:17	SELECT ROUND(SUM(order_details.quantity * pizzas.price), 2) AS totalrevenue FROM order_... 1 row(s) returned		0.110 sec / 0.000 sec

3. IDENTIFY THE HIGHEST-PRICED PIZZA.

The screenshot shows the MySQL Workbench interface. In the top-left corner, it says "MySQL Workbench" and "Local instance MySQL80". The menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. Below the menu is a toolbar with various icons. The main area has tabs for "Project 1 - SQL" and "SQL File 4*". The SQL code in the editor is:

```
1 -- Identify the highest-priced pizza.
2 SELECT
3     pizza_types.name, pizzas.price
4 FROM
5     pizza_types
6     JOIN
7     pizzas
8 WHERE
9     pizza_types.pizza_type_id = pizzas.pizza_type_id
10 ORDER BY pizzas.price DESC
11 LIMIT 1;
```

Below the code is a "Result Grid" table with columns "name" and "price". It contains one row: "The Greek Pizza" with a price of "35.95". To the right of the result grid is a sidebar with icons for "Result Grid" and "Form Editor".

At the bottom, there is a "Result 3" tab showing the "Action Output" log. It lists two actions:

#	Time	Action	Message	Duration / Fetch
2	23:12:17	SELECT ROUND(SUM(order_details.quantity * pizzas.price), 2) AS totalrevenue FROM order_details JOIN pizzas WHERE pizzas.pizza_type_id = pizza_types.pizza_type_id	1 row(s) returned	0.110 sec / 0.000 sec
3	23:13:54	SELECT pizza_types.name, pizzas.price FROM pizza_types JOIN pizzas WHERE pizza_types.pizza_type_id = pizzas.pizza_type_id	1 row(s) returned	0.000 sec / 0.000 sec

4. IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

The screenshot shows the MySQL Workbench interface. The SQL editor tab contains the following SQL code:

```
-- Identify the most common pizza size ordered.
SELECT pizzas.size, COUNT(order_details.order_details_id)
FROM order_details
JOIN pizzas
WHERE order_details.pizza_id = pizzas.pizza_id
GROUP BY pizzas.size;
```

The results grid displays the following data:

size	COUNT(order_details.order_details_id)
M	15385
L	18526
S	14137
XL	544
XXL	28

The results grid has a toolbar with options like 'Result Grid' and 'Form Editor'. The status bar at the bottom right shows 'Read Only'.

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

The screenshot shows the MySQL Workbench interface. The SQL editor tab contains the following query:

```
1 -- List the top 5 most ordered pizza types along with their quantities.
2
3 • SELECT
4     pizza_types.name, SUM(order_details.quantity) AS quantity
5 FROM
6     pizzahut.pizza_types
7     JOIN
8     pizzahut.pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
9     JOIN
10    pizzahut.order_details ON order_details.pizza_id = pizzas.pizza_id
11 GROUP BY pizza_types.name
12 ORDER BY quantity DESC
13 LIMIT 5;
```

The results grid displays the following data:

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

The output pane shows the message "Result 5" and the status "Read Only".

6.Join the necessary tables to find the total quantity of each pizza category ordered.

The screenshot shows the MySQL Workbench interface. The SQL editor tab contains the following SQL code:

```
1 -- Join the necessary tables to find the total quantity of each pizza category ordered.
2 • SELECT
3     pizza_types.category,
4     SUM(order_details.quantity) AS quantity
5 FROM
6     pizza_types
7     JOIN
8     pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
9     JOIN
10    order_details ON order_details.pizza_id = pizzas.pizza_id
11 GROUP BY pizza_types.category
12 ORDER BY quantity DESC
13 LIMIT 5;
```

The results grid displays the following data:

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

The output pane shows the following message:

Result 6 x Read Only

Action Output

#	Time	Action

Message

Duration / Fetch

7.DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

The screenshot shows the MySQL Workbench interface. The top window is titled "MySQL Workbench" and has a tab bar with "Project 1 - SQL" and "SQL File 4*". The main area contains the following SQL code:

```
1 -- Determine the distribution of orders by hour of the day.
2 SELECT
3     *
4 FROM
5     pizzahut.orders;
6 • SELECT
7     HOUR(order_time) AS hour, COUNT(order_id) AS count
8 FROM
9     orders
10 GROUP BY HOUR(order_time);
11
```

Below this is a "Result Grid" window titled "Result 7" which displays the following data:

hour	count
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336

The "Result Grid" window also includes buttons for "Filter Rows", "Exports", and "Wrap Cell Content". On the right side of the interface, there is a vertical toolbar with icons for "Result Grid" (which is selected), "Form Editor", and "Read Only".

8.JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.

The screenshot shows the MySQL Workbench interface. At the top, there's a menu bar with File, Edit, View, Query, Database, Server, Tools, Scripting, Help. Below the menu is a toolbar with various icons. The main area has tabs for 'Project 1 - SQL' and 'SQL File 4*'. The SQL code in the editor is:

```
1 -- Join relevant tables to find the category-wise distribution of pizzas.
2 SELECT
3     category, COUNT(name)
4 FROM
5     pizza_types
6 GROUP BY category;
```

Below the editor is a 'Result Grid' window showing the output of the query:

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

On the right side of the interface, there's a vertical panel with icons for 'Result Grid', 'Form Editor', and 'Read Only'.

9.GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY.

The screenshot shows the MySQL Workbench interface with a query editor and a result grid.

SQL Editor:

```
1 -- Group the orders by date and calculate the average number of pizzas ordered per day.
2 select * from orders;
3 • SELECT
4 *
5 FROM
6 order_details;
7 • SELECT
8     ROUND(AVG(quantity), 0) as AVG_PIZZAS
9 FROM
10    (SELECT
11        orders.order_date, SUM(order_details.quantity) AS quantity
12     FROM
13        orders
14    JOIN order_details ON orders.order_id = order_details.order_id
15    GROUP BY orders.order_date) AS order_quantity;
```

Result Grid:

order_details_id	order_id	pizza_id	quantity
1	1	hawaiian_m	1
2	2	classic_dx_m	1
3	2	five_cheese_l	1
4	2	ital_supr_l	1
5	2	mexicana_m	1
6	2	thai_dkn_l	1
7	3	ital_supr_m	1

Output:

Action Output	#	Time	Action
Message			

10. Determine the top 3 most ordered pizza types based on revenue.

The screenshot shows the MySQL Workbench interface. The SQL editor tab contains the following query:

```
1 -- Determine the top 3 most ordered pizza types based on revenue.
2 select * from order_details;
3
4 SELECT
5     pizza_types.name,
6     SUM(order_details.quantity * pizzas.price) AS revenue
7
8 FROM
9     pizza_types
10    JOIN
11         pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
12    JOIN
13        order_details ON order_details.pizza_id = pizzas.pizza_id
14
15 GROUP BY pizza_types.name
16
17 ORDER BY revenue DESC;
```

The Result Grid shows the following data:

name	revenue
The Thai Chicken Pizza	43494.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5
The Classic Deluxe Pizza	38180.5
The Spicy Italian Pizza	34831.25
The Southwest Chicken Pizza	34705.75
The Italian Supreme Pizza	33476.75

The Action Output panel shows the following information:

#	Time	Action
1	2023-10-10 14:45:23	Query completed successfully.

Message: Duration / Fetch

11. Calculate the percentage contribution of each pizza type to total revenue.

The screenshot shows the MySQL Workbench interface. The SQL editor tab contains the following query:

```
1 -- Calculate the percentage contribution of each pizza type to total revenue.
2 select pizza_types.category, round(sum(order_details.quantity * pizzas.price) / (select round(sum(order_details.quantity * pizzas.price),2) as total_sales
3 from order_details join pizzas on order_details.pizza_id = pizzas.pizza_id)* 100,0) as revenue from pizza_types join pizzas on pizza_types.pizza_type_id=pizzas.pizza_type_
4 join order_details on order_details.pizza_id=pizzas.pizza_id group by pizza_types.category order by revenue desc;
```

The results grid displays the following data:

category	revenue
Classic	27
Supreme	25
Veggie	24
Chicken	24

The output pane shows the following message:

Action Output

#	Time	Action

Message

Duration / Fetch

12. Analyze the cumulative revenue generated over time.

The screenshot shows the MySQL Workbench interface. The top window displays a SQL query:

```
1 -- Analyze the cumulative revenue generated over time.
2 select order_date, sum(revenue) over (order by order_date) as cum_revenue from
3 (select orders.order_date, sum(order_details.quantity * pizzas.price) as revenue
4 from order_details join pizzas on order_details.pizza_id=pizzas.pizza_id join orders
5 on orders.order_id = order_details.order_id group by orders.order_date) as sales;
```

The bottom window shows the results of the query in a grid format:

category	revenue
Classic	27
Supreme	25
Veggie	24
Chicken	24

The results are also displayed in a "Result 11" tab, which includes an "Action Output" section with columns for #, Time, and Action.

13.Determine the top 3 most ordered pizza types based on revenue for each pizza category.

The screenshot shows the MySQL Workbench interface. The top window displays a SQL query in the editor:

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

The bottom window shows the results of the query in a grid format:

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 results are labeled "Result 12".