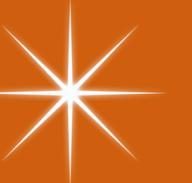




# PIZZA SALES

20  
24

presented by  
**Ranjan  
Pradhan**





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

```
56 •  SELECT
57      pizza_types.category,
58      SUM(order_details.quantity) AS total_quantity
59  FROM
60      pizza_types
61      JOIN
62          pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
63      JOIN
64          order_details ON pizzas.pizza_id = order_details.pizza_id
65  GROUP BY pizza_types.category
66  ORDER BY total_quantity DESC;
```

category	total_quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

Result 37 X

Read Only

Output :::::

02

20  
24

03

# Determine the distribution of orders by hour of the day.

Server Tools Scripting Help

PizzaHut\_sql\* SQL File 3\*

69

70     #Determine the distribution of orders by hour of the day.

71 •    SELECT

72        HOUR(orders.order\_time) AS order\_hour,

73        COUNT(order\_id) AS t\_order

74    FROM

75        orders

76    GROUP BY order\_hour

77    ORDER BY t\_order DESC;

78

79

80

81

Result Grid | Filter Rows: Export: Wrap Cell Content: Result Grid Read Only

order_hour	t_order
12	2520
13	2455
18	2399
17	2336
19	2009

Result 38 x

Output

Action Output

# Time Action Message Duration / Fetch

Join relevant tables to find the category-wise distribution of pizzas.

itabase Server Tools Scripting Help

PizzaHut\_sql\* SQL File 3\*

78  
79  
80  
81  
82     #Join relevant tables to find the category-wise distribution of pizzas.  
83 •    SELECT  
84       category, COUNT(name) AS pizza\_type\_category  
85    FROM  
86       pizza\_types  
87    GROUP BY category;

88  
89  
90

Result Grid | Filter Rows: Export: Wrap Cell Content:

	category	pizza_type_category
▶	Chicken	6
▶	Classic	8
▶	Supreme	9
▶	Veggie	9

Result 39 x

Output

Action Output

#	Time	Action	Message	Duration
56	13:03:23	SELECT HOUR(orders.order_time) AS order_hour, COUNT(order_id) AS t_order FROM...	15 row(s) returned	0.422 sec
57	13:05:57	SELECT category, COUNT(name) AS pizza_type_category FROM pizza_types GROU...	4 row(s) returned	0.000 sec

Group the orders by date and calculate the average number of pizzas ordered per day.

The screenshot shows a SQL database interface with the following details:

- Schemas:** A sidebar lists various schemas including `asian_game`, `class`, `company_db`, `db`, `exam`, `friendsdb`, `hero`, `movie_ratings_db`, `nft_data`, `people_education`, and `pizza`. The `pizza` schema is expanded to show tables: `order_details`, `orders`, `pizza_types`, and `pizzas`.
- SQL Editor:** The main area contains a SQL query:

```
78
79
80
81
82     #Join relevant tables to find the category-wise distribution of pizzas.
83 •  SELECT
84         category, COUNT(name) AS pizza_type_category
85     FROM
86         pizza_types
87     GROUP BY category;
```
- Result Grid:** The results of the query are displayed in a grid:

category	pizza_type_category
Chicken	6
Classic	8
Supreme	9
Veggie	9
- Action Output:** The log shows the execution details:

#	Time	Action	Message	Duration / Fetch
57	13:05:57	SELECT category, COUNT(name) AS pizza_type_category FROM pizza_types GROU...	4 row(s) returned	0.000 sec / 0.000 sec

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

The screenshot shows a SQL development environment with the following interface elements:

- Toolbar:** Includes icons for file operations (New, Open, Save, Print, etc.), search, and various database management functions.
- Navigator:** Shows the current database connection is "PizzaHut\_sql\*" and displays the "SCHEMAS" tree. The "pizza" schema is expanded, showing tables: "order\_details", "orders", "pizza\_types" (which is selected), and "pizzas".
- SQL Editor:** Contains the following SQL code:

```
102
103      #Determine the top 3 most ordered pizza types based on revenue.
104  •  SELECT
105      pizza_types.name,
106      SUM(order_details.quantity * pizzas.price) AS revenue
107  FROM
108      pizza_types
109      JOIN
110      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
111      JOIN
112      order_details ON order_details.pizza_id = pizzas.pizza_id
113  GROUP BY pizza_types.name
114  ORDER BY revenue DESC
```
- Result Grid:** Displays the results of the query:

name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5
- Action Output:** Shows the log entry for the executed query:

#	Time	Action	Message	Duration / Fetch
58	13:08:00	SELECT category, COUNT(name) AS pizza_type_category FROM pizza_types GROU...	4 row(s) returned	0.000 sec / 0.000 sec

20  
24

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

The screenshot shows a SQL development environment with the following interface elements:

- File Edit View Query Database Server Tools Scripting Help**: The top menu bar.
- Navigator**: A sidebar showing database schemas and tables. The **pizza** schema is expanded, showing **order\_details**, **orders**, **pizza\_types**, and **pizzas**.
- PizzaHut\_sql\***: The active tab in the main editor area, containing the following SQL code:

```
117
118      #Calculate the percentage contribution of each pizza type to total revenue.
119 •  SELECT
120      pizza_types.category,
121      ROUND((SUM(order_details.quantity * pizzas.price) / (SELECT
122          ROUND(SUM(order_details.quantity * pizzas.price),
123          0) AS total_sales
124
125      FROM
126          order_details
127          JOIN
128              pizzas ON pizzas.pizza_id = order_details.pizza_id)) * 100,
129      2) AS revenue
130
131  FROM
```

- Result Grid**: A table showing the results of the query:

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

- Result 43 x**: A tab labeled "Output" showing the execution details of the query.
- Action Output**: A table showing the log of actions taken:

#	Time	Action	Message	Duration / Fetch
60	13:13:02	SELECT pizza_types.name, SUM(order_details.quantity * pizzas.price) AS revenue FR...	3 row(s) returned	0.391 sec / 0.000 sec
61	13:13:26	SELECT pizza_types.category, ROUND(SUM(order_details.quantity * pizzas.price) / (...	4 row(s) returned	0.656 sec / 0.000 sec

07

# Analyze the cumulative revenue generated over time.

20  
24

The screenshot shows a SQL development environment with the following interface elements:

- Navigator:** Shows the database schema with the **SCHEMAS** tab selected, displaying databases like asian\_game, class, company\_db, db, exam, friendsdb, hero, movie\_ratings\_db, nft\_data, people\_education, and pizza. The pizza schema is expanded to show Tables (order\_details, orders, pizza\_types, pizzas) and Views.
- PizzaHut\_sql\***: The active SQL editor tab contains the following query:

```
138
139      #Analyze the cumulative revenue generated over time.
140 •   select order_date,revenue,round(sum(revenue) over(order by order_date),2) as cum_revenue
141     from
142     (select orders.order_date,round(sum(order_details.quantity*pizzas.price),2) as revenue
143       from orders
144      join order_details
145        on order_details.order_id=orders.order_id
146      join pizzas on pizzas.pizza_id=order_details.pizza_id
147      group by orders.order_date
148      order by orders.order_date) as sales
149      group by order_date;
150
```

- Result Grid:** A table showing the results of the query:

order_date	revenue	cum_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

- Action Output:** A log of database actions:

#	Time	Action	Message	Duration / Fetch
61	13:13:26	SELECT pizza_types.category, ROUND((SUM(order_details.quantity * pizzas.price) / ...)	4 row(s) returned	0.656 sec / 0.000 sec
62	13:14:59	select order_date,revenue,round(sum(revenue) over(order by order_date),2) as cum_revenu...	358 row(s) returned	0.343 sec / 0.000 sec

08

20  
24



Local instance MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

Navigator PizzaHut\_sql\* SQL File 3\*

SCHEMAS

Filter objects

- asian\_game
- class
- company\_db
- db
- exam
- friendsdb
- hero
- movie\_ratings\_db
- nft\_data
- people\_education
- pizza
  - Tables
    - order\_details
    - orders
    - pizza\_types
    - pizzas
  - Views
  - Stored Procedures

Administration Schemas

Information

Table: pizza\_types

Columns:

pizza_type_id	text
name	text
category	text
ingredients	text

Object Info Session

138  
139    #Analyze the cumulative revenue generated over time.  
140 •    select order\_date,revenue,round(sum(revenue) over(order by order\_date),2) as cum\_revenue  
141    from  
142    (select orders.order\_date,round(sum(order\_details.quantity\*pizzas.price),2) as revenue  
143    from orders  
144    join order\_details  
145    on order\_details.order\_id=orders.order\_id  
146    join pizzas on pizzas.pizza\_id=order\_details.pizza\_id  
147    group by orders.order\_date  
148    order by orders.order\_date) as sales  
149    group by order\_date;  
150

Result Grid | Filter Rows: Export: Wrap Cell Content: Result 47 x Read Only

order_date	revenue	cum_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

Action Output

#	Time	Action	Message	Duration / Fetch
64	13:18:49	select category.name,revenue,rank_pizza from (select category.name,revenue,rank()over(pa...)	12 row(s) returned	0.453 sec / 0.000 sec
65	13:19:48	select order_date,revenue,round(sum(revenue) over(order by order_date),2) as cum_revenu...	358 row(s) returned	0.344 sec / 0.000 sec

09

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

The screenshot shows a SQL development environment with the following interface elements:

- Toolbar:** Includes icons for file operations, database management, and search.
- Navigator:** Shows the current database connection is "PizzaHut\_sql" and the active file is "SQL File 3".
- Schemas:** A tree view of database schemas. The "pizza" schema is expanded, showing tables like "order\_details", "orders", "pizza\_types", and "pizzas".
- SQL Editor:** Displays the following SQL query:

```
159     pizza_types.category,
160     pizza_types.name,
161     ROUND(SUM(order_details.quantity * pizzas.price),
162             2) AS revenue
163
164     FROM
165     pizza_types
166     JOIN
167     pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
168     JOIN
169     order_details ON order_details.pizza_id = pizzas.pizza_id
170     GROUP BY pizza_types.category,pizza_types.name) as p)as d
171     where rank_pizza<=3;
```
- Result Grid:** A table showing the results of the query. The columns are "category", "name", "revenue", and "rank\_pizza". The data is:

category	name	revenue	rank_pizza
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
- Action Output:** Shows the execution log with one entry:

#	Time	Action	Message	Duration / Fetch
63	13:15:57	select category,name,revenue,rank_pizza from (select category,name,revenue,rank()over(pa...)	12 row(s) returned	0.406 sec / 0.000 sec

+123-456-7890

[www.reallygreatsite.com](http://www.reallygreatsite.com)

[hello@reallygreatsite.com](mailto:hello@reallygreatsite.com)

# THANK YOU!

Thank you for your attention to our sales report presentation. If you have any questions or would like to discuss the findings in more detail, please don't hesitate to reach out to our sales team. We appreciate your continued support and partnership.