

SQL Project on Pizza Sale



INTRODUCTION

Hello, I am Abhilasha Singh, a Data Analyst with a background in Computer Science (BE). In this SQL project, I will analyze pizza sales data to uncover trends, insights, and key metrics that drive business decisions.

IDENTIFY HIGHEST PRICE PIZZA

```
1 --- Identify highest price pizza
2 • SELECT
3     pt.name AS pizza_name, p.price
4 FROM
5     pizza_types AS pt
6     JOIN
7     pizzas AS p ON p.pizza_type_id = pt.pizza_type_id
8 ORDER BY p.price DESC
9 LIMIT 1;
```

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

	pizza_name	price
▶	The Greek Pizza	35.95

16. CALCULATE THE TOTAL REVENUE GENERATED BY PIZZA SALES

1 • SELECT

ROUND(SUM(piz.price * d.quantity), 2) AS total_revenue

2 FROM

pizzas AS piz

3 JOIN

orders_detail AS d ON piz.pizza_id = d.pizza_id;

Result Grid



Filter Rows:

Export:



Wrap Cell Content:



	total_revenue
▶	817860.05

IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED



```
1 • SELECT
2     p.size, COUNT(order_detail_id) AS Ordered_by_people, sum(quantity) as total_orders
3
4     FROM
5         pizzas AS p
6
7         JOIN
8             orders_detail AS o ON p.pizza_id = o.pizza_id
9
10        GROUP BY p.size
11
12        ORDER BY Ordered_by_people DESC;
```

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

size	Ordered_by_people	total_orders
L	18526	18956
M	15385	15635
S	14137	14403
XL	544	552
XXL	28	28

Result 1 ×

Output

LIST TOP 5 MOST ORDERED PIZZA TYPE ALONG WITH THEIR QUANTITY

```
1 • | SELECT
2     pizza_types.name,
3         SUM(orders_detail.quantity) AS total_quantity
4     FROM
5         pizza_types
6             JOIN
7                 pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
8             JOIN
9                 orders_detail ON orders_detail.pizza_id = pizzas.pizza_id
10            GROUP BY pizza_types.name
11            ORDER BY total_quantity DESC
12            LIMIT 5;
```

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

	name	total_quantity
▶	The Classic Deluxe Pizza	2453
	The Barbecue Chicken Pizza	2432
	The Hawaiian Pizza	2422
	The Pepperoni Pizza	2418

JOIN THE NECESSARY TABLE TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY

```
1 •   SELECT
2     pizza_types.category,
3     SUM(orders_detail.quantity) AS total_quantity
4   FROM
5     pizza_types
6     JOIN
7     pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
8     JOIN
9     orders_detail ON orders_detail.pizza_id = pizzas.pizza_id
10  GROUP BY pizza_types.category
11  ORDER BY total_quantity DESC;
12
```

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

category	total_quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

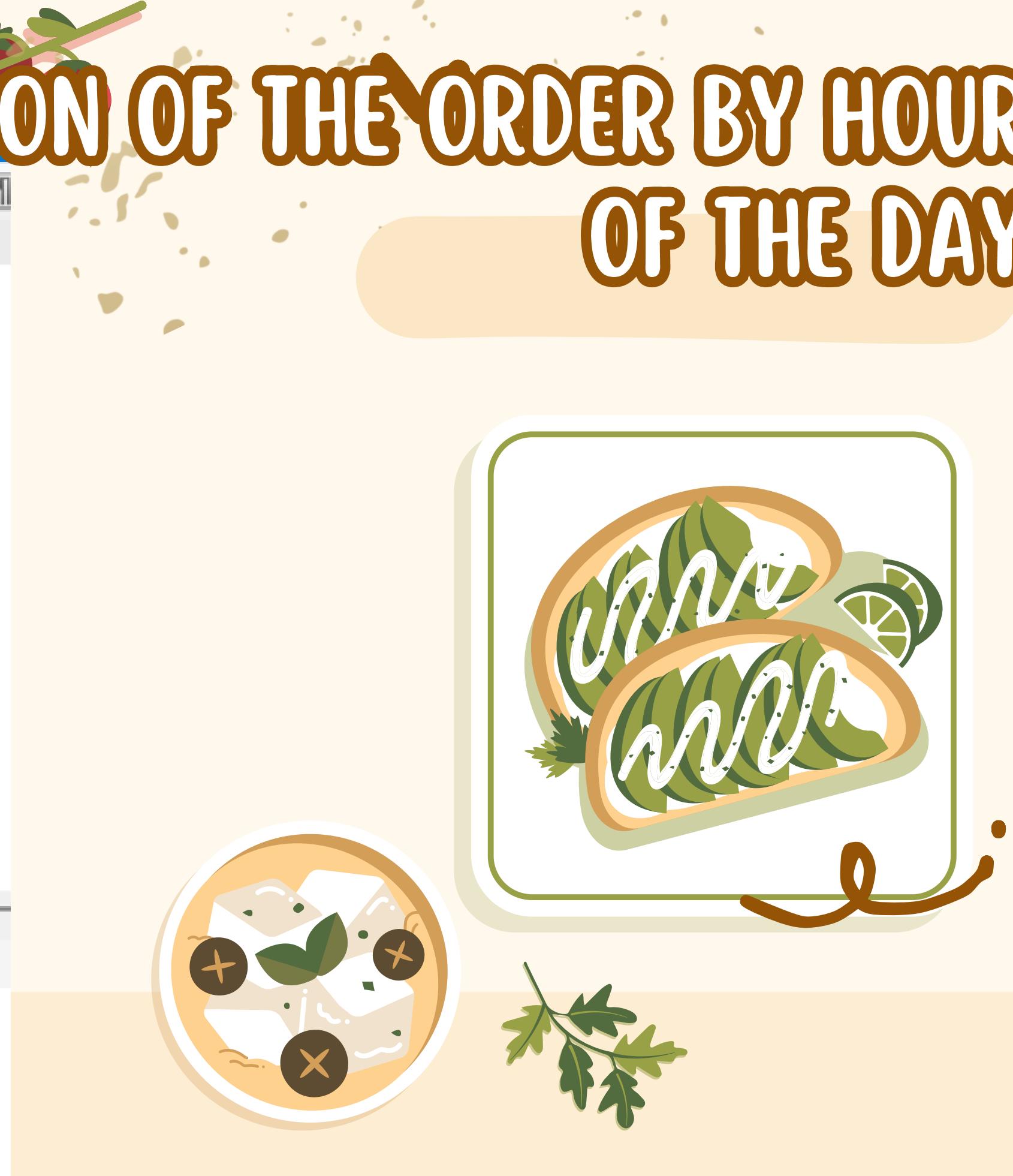
DETERMINE THE DISTRIBUTION OF THE ORDER BY HOUR OF THE DAY

Query 1

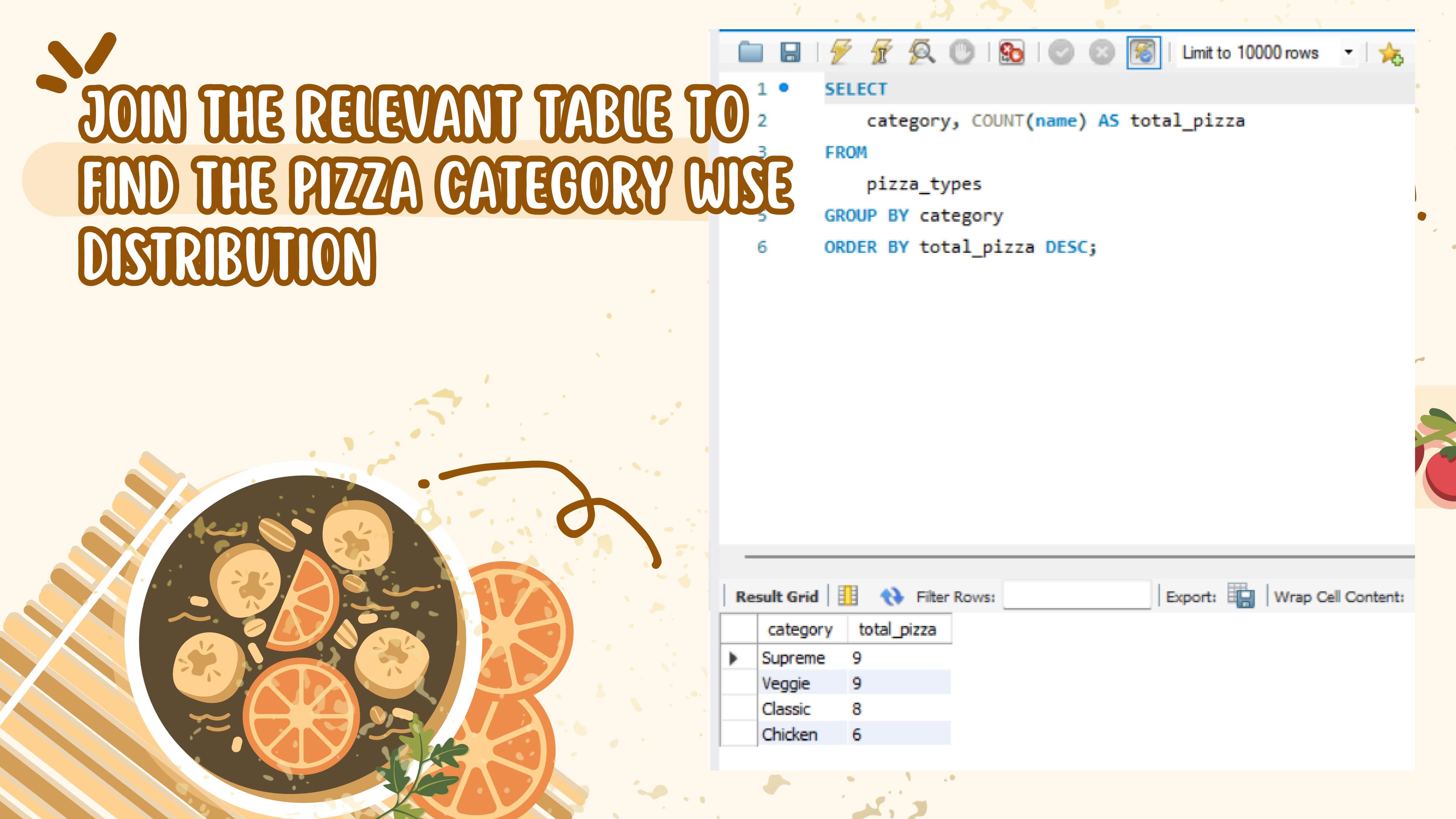
```
1 • SELECT
2     HOUR(order_time) AS hours, COUNT(order_id) AS total_order
3
4     FROM
5     orders
6
7     GROUP BY hours;
```

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

hours	total_order
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336



JOIN THE RELEVANT TABLE TO
FIND THE PIZZA CATEGORY WISE
DISTRIBUTION



```
1 •   SELECT
2      category, COUNT(name) AS total_pizza
3   FROM
4     pizza_types
5 GROUP BY category
6 ORDER BY total_pizza DESC;
```

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

	category	total_pizza
▶	Supreme	9
	Veggie	9
	Classic	8
	Chicken	6

RETRIEVE TOTAL NUMBER OF ORDER PLACED

The screenshot shows a MySQL Workbench interface. At the top, there's a toolbar with various icons. To the right of the toolbar, the text "Limit to 10000 rows" is visible. Below the toolbar, a query editor window displays the following SQL code:

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

At the bottom of the interface, there are several buttons: "Result Grid" (highlighted in blue), "Filter Rows:", "Export:", and "Wrap". A data grid below these buttons shows the results of the query:

total_orders
21350

GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZA ORDERED PER DAY

```
1 •   SELECT
2       ROUND(AVG(number_of_pizza), 2) AS avg_pizza_ordered
3   FROM
4   (SELECT
5       orders.order_date,
6       SUM(orders_detail.quantity) AS number_of_pizza
7   FROM
8       orders
9   INNER JOIN orders_detail ON orders.order_id = orders_detail.order_id
.0   GROUP BY orders.order_date) AS order_quantity;
1
.2
```

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

avg_pizza_ordered
138.47

DETERMINE THE TOP THREE MOST ORDERED PIZZAS TYPE BASE ON THE REVENUE

```
1 •   SELECT
2       pizza_types.name,
3           SUM(pizzas.price * orders_detail.quantity) AS total_revenue
4   FROM
5       pizzas
6           INNER JOIN
7       pizza_types ON pizza_types.pizza_type_id = pizzas.pizza_type_id
8           INNER JOIN
9       orders_detail ON pizzas.pizza_id = orders_detail.pizza_id
10      GROUP BY pizza_types.name
11      ORDER BY total_revenue DESC
12      LIMIT 3;
13
14
```

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

	name	total_revenue
1	The Thai Chicken Pizza	43434.25
2	The Barbecue Chicken Pizza	42768
3	The California Chicken Pizza	41409.5

CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE

```
1 •   SELECT
2     pizza_types.category,
3     ROUND(SUM(orders_detail.quantity * pizzas.price) / (SELECT
4                                         SUM(orders_detail.quantity * pizzas.price)
5                                         FROM
6                                         orders_detail
7                                         INNER JOIN
8                                         pizzas ON pizzas.pizza_id = orders_detail.pizza_id) * 100,
9                                         2) AS total_percent
10    FROM
11    pizza_types
12    INNER JOIN
13    pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
14    INNER JOIN
15    orders_detail ON orders_detail.pizza_id = pizzas.pizza_id
16    GROUP BY pizza_types.category
17    ORDER BY total_percent DESC;
18
```

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

	category	total_percent
1	Classic	26.91
2	Supreme	25.46
3	Chicken	23.96
4	Veggie	23.68

ANALYZE THE CUMMULATIVE REVENUE GENERATED OVER TIME

The screenshot shows a SQL query editor interface with a query window and a results window.

Query Window:

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

Results Window:

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

DETERMINE THE TOP THREE MOST ORDERED PIZZAS TYPE BASED ON REVNEUE FOR EACH PIZZA_TYPE

Query 1 SQL File 3* SQL File 4* SQL File 5* SQL File 6* SQL File 7* SQL File 8* SQL File 9* SQL File 10* X SQL File 11* SQL File 12*

Limit to 10000 rows

```
1 • select name, total_revenue from
2   (select category, name, total_revenue, rank() over( partition by category order by total_revenue desc) as rn
3   from
4   (select pizza_types.category, pizza_types.name, sum(pizzas.price * orders_detail.quantity) as total_revenue
5   from pizza_types inner join pizzas on pizza_types.pizza_type_id = pizzas.pizza_type_id
6   inner join orders_detail on pizzas.pizza_id = orders_detail.pizza_id
7   group by pizza_types.category, pizza_types.name) as b
8   where rn >=3;
```

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

	name	total_revenue
▶	The California Chicken Pizza	41409.5
	The Southwest Chicken Pizza	34705.75
	The Chicken Alfredo Pizza	16900.25
	The Chicken Pesto Pizza	16701.75
	The Pepperoni Pizza	30161.75
	The Greek Pizza	28454.100000000013
	The Italian Capocollo Pizza	25094

Result 7 X

**THANK YOU
FOR YOUR
ATTENTION**

