



PIZZA SALES REPORT

presented by
Divyanshu Wakodikar





INTRODUCTION

Welcome to our pizza sales report presentation. In this report, we will analyze the pizza sales dataset and analyze few questions by using mySQL



[Github Link](#)

02

RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

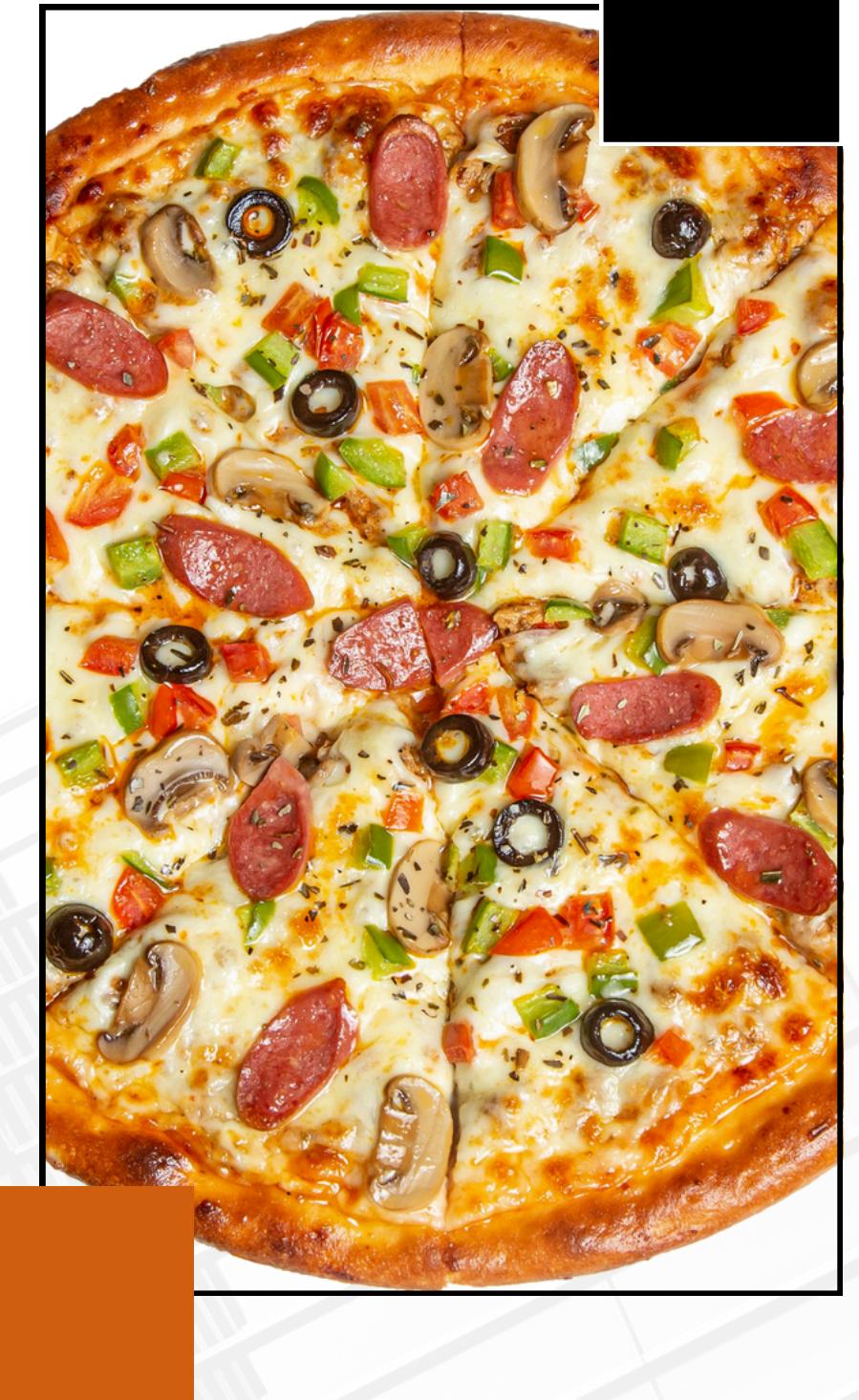
Query 1 × SQL File 1* SQL File 2* SQL File 3* SQL File 4* SQL File 5* SQL

1 -- Retrieve the total number of orders placed.
2 • select count(order_id) as total_ordes from orders;
3
4

OUTPUT-

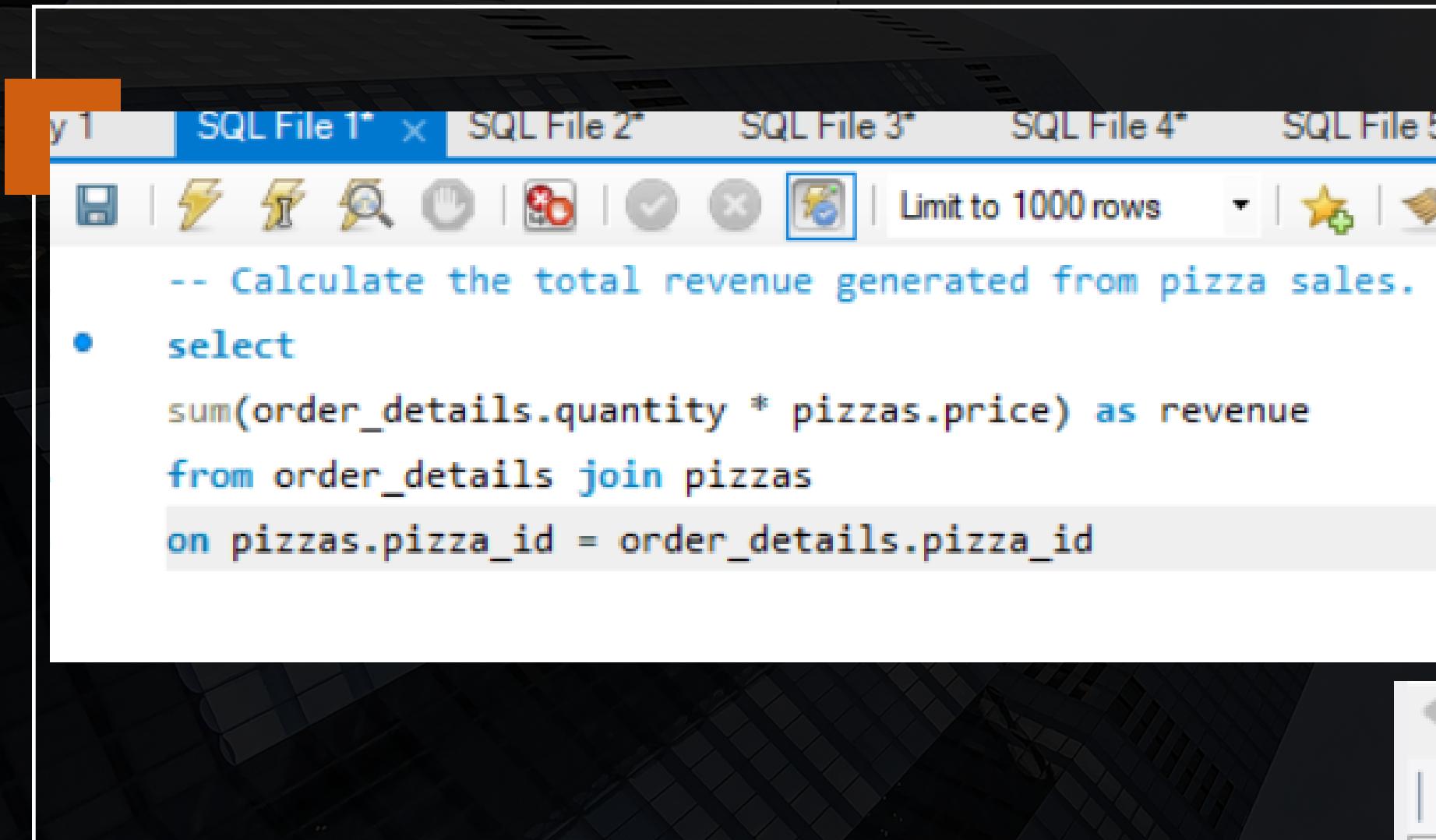
Result Grid | Filter Rows:

total_ordes
21350



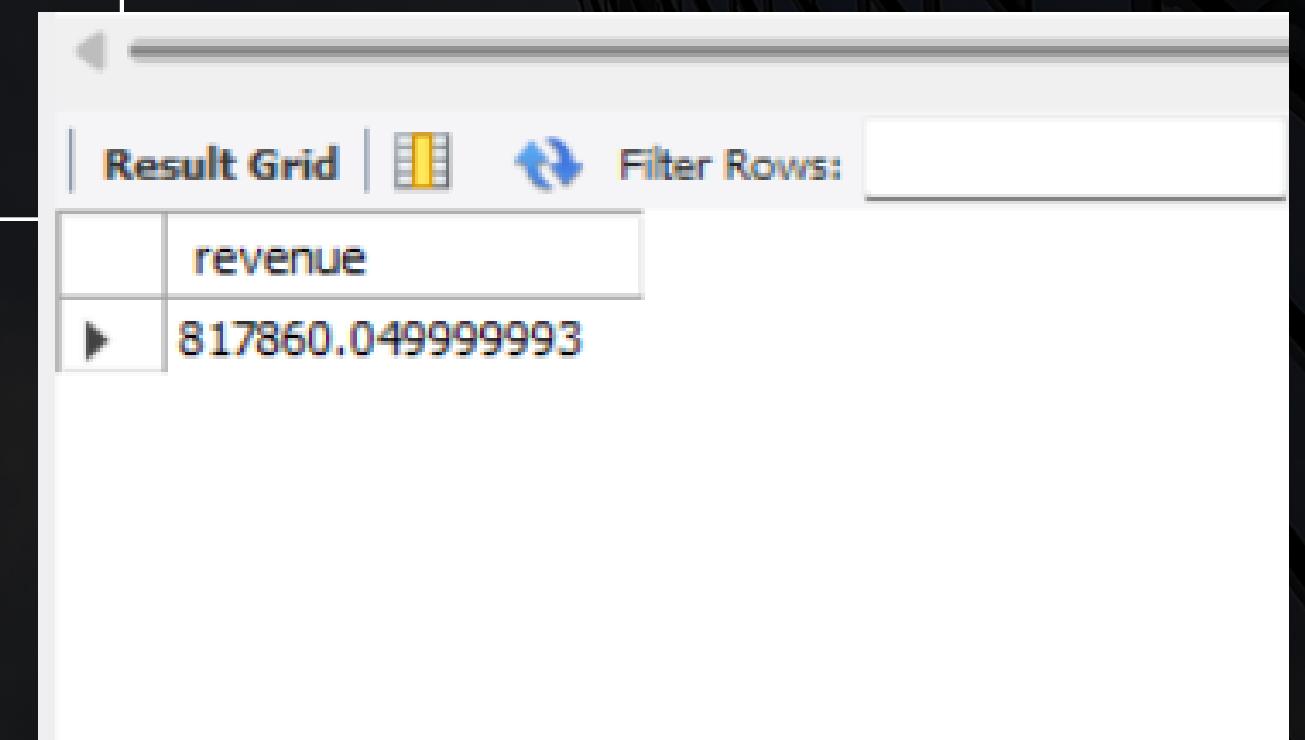
03

CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.



The screenshot shows a MySQL Workbench interface with multiple tabs at the top: SQL File 1*, SQL File 2*, SQL File 3*, SQL File 4*, and SQL File 5*. Below the tabs is a toolbar with various icons. A dropdown menu shows "Limit to 1000 rows". The main area contains a SQL query:

```
-- Calculate the total revenue generated from pizza sales.  
• select  
    sum(order_details.quantity * pizzas.price) as revenue  
  from order_details join pizzas  
    on pizzas.pizza_id = order_details.pizza_id
```



The screenshot shows the "Result Grid" tab in MySQL Workbench. The grid has one column labeled "revenue". The value is displayed as 817860.049999993.

revenue
817860.049999993



IDENTIFY THE HIGHEST-PRICED PIZZA.



```
1  -- Identify the highest-priced pizza.  
2 • select pizza_types.name,pizzas.price  
3   from pizza_types join pizzas  
4     on pizza_types.pizza_type_id=pizzas.pizza_type_id  
5   order by pizzas.price desc limit 1;
```

Result Grid | Filter Rows:

	name	price
▶	The Greek Pizza	35.95

IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

```
1    -- Identify the most common pizza size ordered.  
2  
3 • select pizzas.size, count(order_details.order_details_id) as order_count  
4   from pizzas join order_details  
5     on pizzas.pizza_id=order_details.pizza_id  
6   group by pizzas.size order by order_count desc;
```

Result Grid | Filter Rows:

	size	order_count
▶	L	18526
	M	15385
	S	14137
	XL	544
	XXL	28

LIST THE TOP 5 MOST ORDERED PIZZA TYPES ALONG WITH THEIR QUANTITIES.

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

```
1 -- List the top 5 most ordered pizza types
2 -- along with their quantities.
3 • select pizza_types.name,
4     sum(order_details.quantity) as quantity
5     from pizza_types join pizzas
6     on pizza_types.pizza_type_id=pizzas.pizza_type_id
7     join order_details
8     on order_details.pizza_id=pizzas.pizza_id
9     group by pizza_types.name order by quantity desc limit 5 ;
10
11
```

The result 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

JOIN THE NECESSARY TABLES TO FIND THE TOTAL QUANTITY OF EACH PIZZA CATEGORY ORDERED.

The screenshot shows a MySQL Workbench interface. The top section contains a toolbar with various icons for database management. Below the toolbar is a text area where a SQL query is written. The query joins three tables: `pizza_types`, `pizzas`, and `order_details` to calculate the total quantity of each pizza category ordered. The result grid at the bottom displays the data, showing four categories: Classic, Veggie, Supreme, and Chicken, with their respective total quantities.

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

	category	total_quantity
▶	Classic	14888
	Veggie	11649
	Supreme	11987
	Chicken	11050

DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

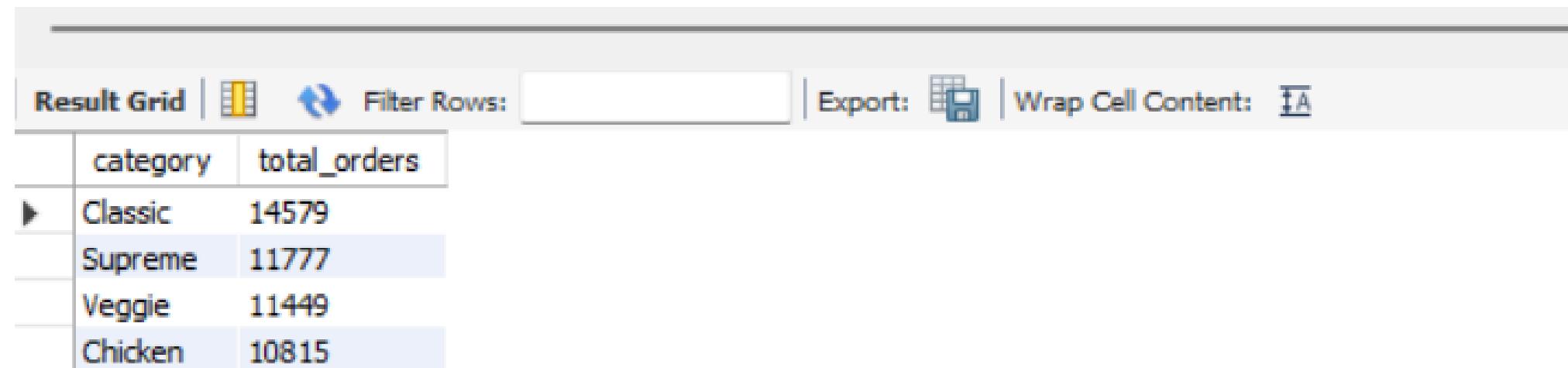
```
1  -- Determine the distribution of orders by hour of the day.  
2 • SELECT EXTRACT(HOUR FROM order_time) AS order_hour,  
3   COUNT(*) AS total_orders  
4   FROM orders  
5   GROUP BY order_hour  
6   ORDER BY order_hour;  
7
```

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

order_hour	total_orders
9	1
10	8
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336
18	2399
19	2009
20	1642

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

```
1  -- Join relevant tables to find the category-wise distribution of pizzas.  
2 • SELECT pizza_types.category, COUNT(*) AS total_orders  
3   FROM order_details  
4   JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id  
5   JOIN pizza_types ON pizzas.pizza_type_id = pizza_types.pizza_type_id  
6   GROUP BY pizza_types.category  
7   ORDER BY total_orders DESC;  
8
```



The screenshot shows a database query results grid. At the top, there are navigation buttons for 'Result Grid' (highlighted), 'Filter Rows:', 'Export:', and 'Wrap Cell Content:'. The table has two columns: 'category' and 'total_orders'. The data rows are: Classic (14579), Supreme (11777), Veggie (11449), and Chicken (10815). The 'Supreme' row is currently selected, indicated by a light blue background.

	category	total_orders
▶	Classic	14579
	Supreme	11777
	Veggie	11449
	Chicken	10815

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

The screenshot shows a MySQL Workbench interface. The top half is a query editor window containing the following SQL code:

```
1 -- Group the orders by date and
2 -- calculate the average number of pizzas ordered per day.
3
4 • SELECT order_date, AVG(total_quantity) AS average_pizzas_ordered
5   FROM (
6     SELECT orders.order_date, SUM(order_details.quantity) AS total_quantity
7       FROM orders
8     JOIN order_details ON orders.order_id = order_details.order_id
9     GROUP BY orders.order_date
10    ) AS daily_orders
11   GROUP BY order_date
12   ORDER BY order_date;
13
14
```

The bottom half is a result grid window displaying the output of the query. The table has two columns: 'order_date' and 'average_pizzas_ordered'. The data is as follows:

order_date	average_pizzas_ordered
2015-01-01	162.0000
2015-01-02	165.0000
2015-01-03	158.0000
2015-01-04	106.0000
2015-01-05	125.0000
2015-01-06	147.0000
2015-01-07	138.0000
2015-01-08	173.0000
2015-01-09	127.0000
2015-01-10	146.0000
2015-01-11	116.0000
2015-01-12	119.0000

DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.

```
1      -- Determine the top 3 most ordered pizza types based on revenue.  
2  
3 •  SELECT pizza_types.name, SUM(order_details.quantity * pizzas.price) AS total_revenue  
4   FROM order_details  
5   JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id  
6   JOIN pizza_types ON pizzas.pizza_type_id = pizza_types.pizza_type_id  
7   GROUP BY pizza_types.name  
8   ORDER BY total_revenue DESC  
9   LIMIT 3;  
10
```

The screenshot shows a MySQL query results window. At the top, there are several buttons: 'Result Grid' (selected), 'Filter Rows:', 'Export:', 'Wrap Cell Content:', and 'Fetch rows:'. The main area displays a table with two columns: 'name' and 'total_revenue'. The data is as follows:

	name	total_revenue
▶	The Thai Chicken Pizza	43434.25
	The Barbecue Chicken Pizza	42768
	The California Chicken Pizza	41409.5

CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.

```
1  -- Calculate the percentage contribution of each pizza type to total revenue.  
2  
3  Open a script file in this editor name,  
4  SUM(order_details.quantity * pizzas.price) AS pizza_revenue,  
5  SUM(order_details.quantity * pizzas.price) / (SELECT SUM(order_details.quantity * pizzas.price)  
6  FROM order_details  
7  JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id) * 100 AS percentage_contribution  
8  FROM order_details  
9  JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id  
10 JOIN pizza_types ON pizzas.pizza_type_id = pizza_types.pizza_type_id  
11 GROUP BY pizza_types.name  
12 ORDER BY percentage_contribution DESC;
```

Result Grid			
	name	pizza_revenue	percentage_contribution
▶	The Thai Chicken Pizza	43434.25	5.310719113863108
	The Barbecue Chicken Pizza	42768	5.2292565213327595
	The California Chicken Pizza	41409.5	5.063152308270878
	The Classic Deluxe Pizza	38180.5	4.668341484585331
	The Spicy Italian Pizza	34831.25	4.258827656394306
	The Southwest Chicken Pizza	34705.75	4.243482732773205
	The Italian Supreme Pizza	33476.75	4.093212524563375
	The Hawaiian Pizza	32273.25	3.9460602092008625
	The Four Cheese Pizza	32265.70000000065	3.945137068377521
	The Sicilian Pizza	30940.5	3.7831044565632306
	The Pepperoni Pizza	30161.75	3.687886454412373
	The Greek Pizza	28454.10000000013	3.4790915634038195

ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```
1  -- Analyze the cumulative revenue generated over time.
2 •   SELECT order_date,
3           SUM(daily_revenue) OVER (ORDER BY order_date) AS cumulative_revenue
4   FROM (
5       SELECT orders.order_date,
6               SUM(order_details.quantity * pizzas.price) AS daily_revenue
7       FROM orders
8       JOIN order_details ON orders.order_id = order_details.order_id
9       JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id
10      GROUP BY orders.order_date
11  ) AS revenue_per_day
12  ORDER BY order_date;
13
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	order_date	cumulative_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		
	2015-01-08	19399.05		
	2015-01-09	21526.4		
	2015-01-10	23990.350000000002		
	2015-01-11	25862.65		
	2015-01-12	27781.7		
	2015-01-13	29821.300000000003		

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

```
1  -- Determine the top 3 most ordered pizza types based on revenue for each pizza category.
2 •  SELECT category, name, pizza_revenue
3   FROM (
4     SELECT pizza_types.category,
5        pizza_types.name,
6        SUM(order_details.quantity * pizzas.price) AS pizza_revenue,
7        RANK() OVER (PARTITION BY pizza_types.category
8          ORDER BY SUM(order_details.quantity * pizzas.price) DESC) AS revenue_rank
9        FROM order_details
10       JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id
11       JOIN pizza_types ON pizzas.pizza_type_id = pizza_types.pizza_type_id
12       GROUP BY pizza_types.category, pizza_types.name
13    ) AS ranked_revenue
14   WHERE revenue_rank <= 3
15   ORDER BY category, revenue_rank;
```

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

	category	name	pizza_revenue
▶	Chicken	The Thai Chicken Pizza	43434.25
	Chicken	The Barbecue Chicken Pizza	42768
	Chicken	The California Chicken Pizza	41409.5
	Classic	The Classic Deluxe Pizza	38180.5
	Classic	The Hawaiian Pizza	32273.25
	Classic	The Pepperoni Pizza	30161.75
	Supreme	The Spicy Italian Pizza	34831.25
	Supreme	The Italian Supreme Pizza	33476.75
	Supreme	The Sicilian Pizza	30940.5
	Veggie	The Four Cheese Pizza	32265.70000000065
	Veggie	The Mexicana Pizza	26780.75
	Veggie	The Five Cheese Pizza	26066.5