

SQL PROJECT

PIZZA ANALYSIS

From pizzas to order details, every thing is analyzed! Come and delve in this crusty analysis.



SAHIL ASAD



OBJECTIVE

To analyze pizza sales data to gain insights into customer preferences, sales performance, and business trends.

SCOPE

The project involves creating a relational database, importing data, and performing various SQL queries to extract meaningful insights.

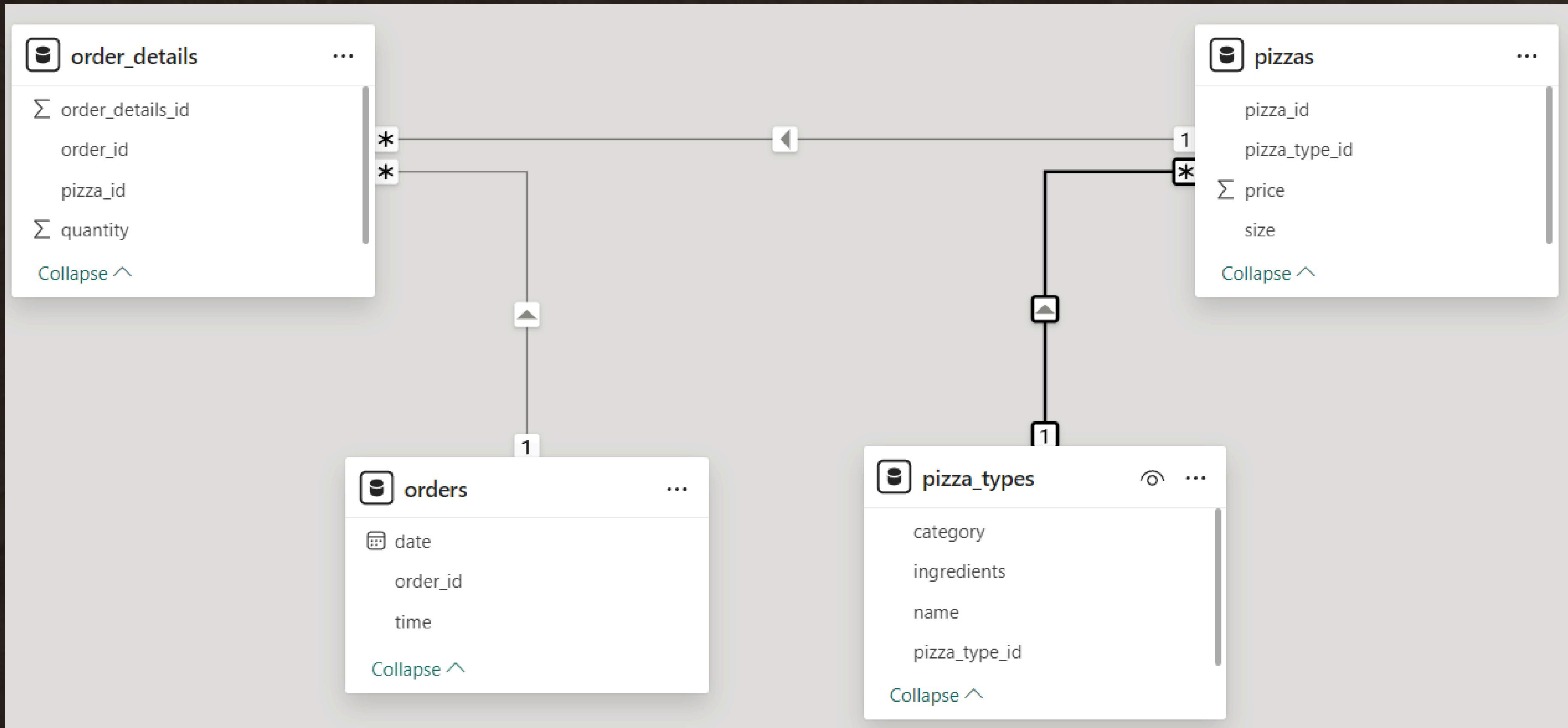


FILES PROVIDED

- **order_details.csv**: Contains details of each order, including order ID, pizza type, quantity, and price.
- **orders.csv**: Contains information about each order, including order ID, date, and customer details.
- **pizza_types.csv**: Contains details about different types of pizzas, including pizza ID, name, and category.
- **pizzas.csv**: Contains details about pizza_id, pizza_type_id, price and size.



MODEL VIEW



RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

The screenshot shows the MySQL Workbench interface. The Navigator pane on the left lists databases: classicmodels, demo, pizzahut, sakila, students, sys, and world. The pizzahut database is selected, showing its tables: order_details, orders, pizza_types, pizzas, Views, Stored Procedures, Functions, and a sub-table sakila. The SQL editor pane contains the following code:

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

The screenshot shows the MySQL Workbench Result Grid. The grid has one row and two columns. The first column is empty, and the second column is labeled "total_orders". The value "21350" is displayed in the cell under "total_orders".

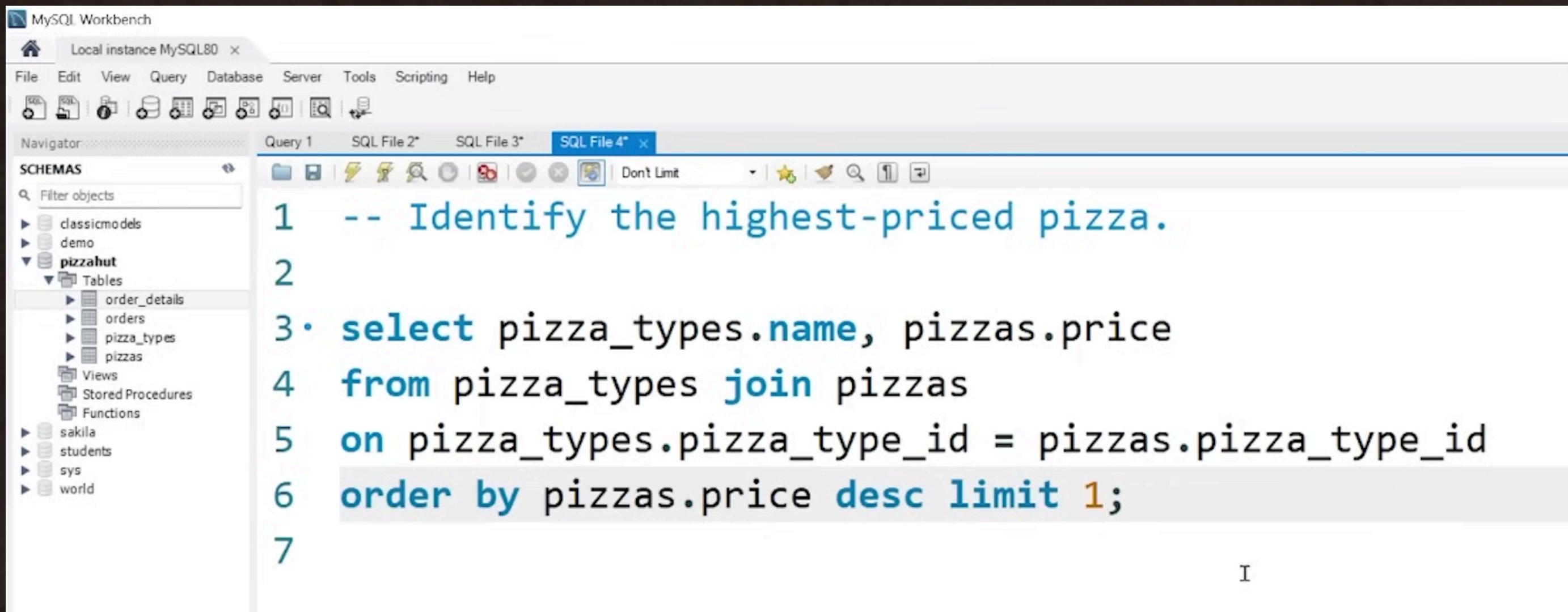
	total_orders
▶	21350

CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

```
1 -- Calculate the total revenue generated from pizza sales.  
2  
3 • select  
4 round(sum(order_details.quantity * pizzas.price),2) as total_sales  
5 from order_details join pizzas  
6 on pizzas.pizza_id = order_details.pizza_id
```

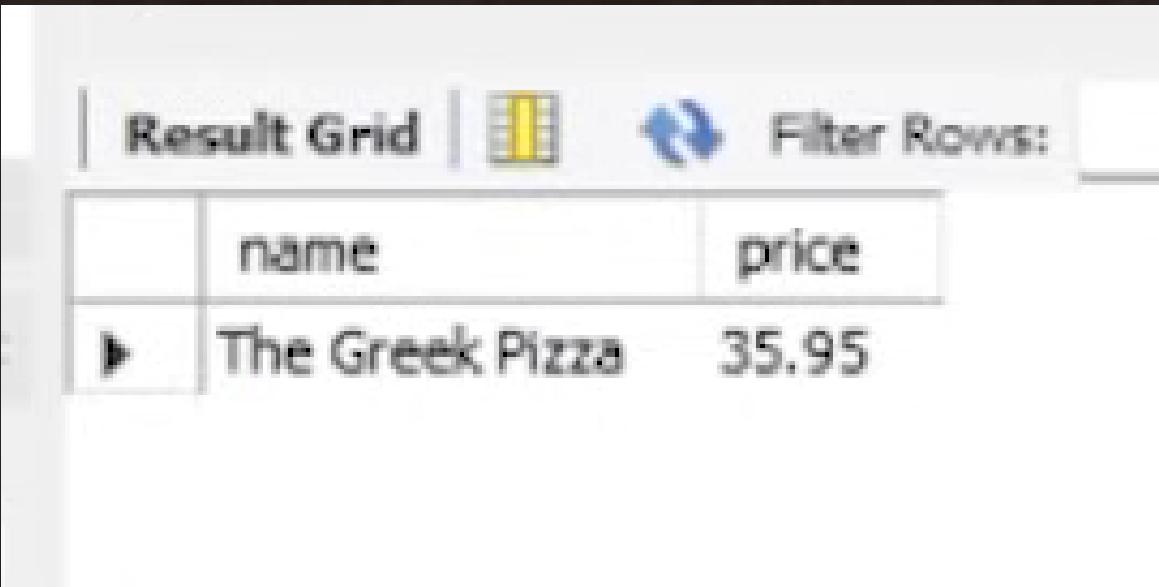
Result Grid	
	total_sales
▶	817860.05

IDENTIFY THE HIGHEST-PRICED PIZZA.



The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema with the 'pizzahut' database selected, showing tables like 'order_details', 'orders', 'pizza_types', and 'pizzas'. The main window contains a SQL query:

```
1 -- Identify the highest-priced pizza.  
2  
3 • select pizza_types.name, pizzas.price  
4 from pizza_types join pizzas  
5 on pizza_types.pizza_type_id = pizzas.pizza_type_id  
6 order by pizzas.price desc limit 1;  
7
```



The result grid shows the output of the executed SQL query:

	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 ;  
7
```

Result Grid | Filter Rows:

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

Result 3 ×

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

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

	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.

```
4• select pizza_types.category,  
5  sum(order_details.quantity) as quantity  
6  from pizza_types join pizzas  
7  on pizza_types.pizza_type_id = pizzas.pizza_type_id  
8  join order_details  
9  on order_details.pizza_id = pizzas.pizza_id  
10 group by pizza_types.category order by quantity desc;
```

Result Grid | Filter Rows:

	category	quantity
▶	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050

DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

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

	hour	order_count
▶	11	1231
	12	2520
	13	2455
	14	1472
	15	1468
	16	1920
	17	2336
	18	2399

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

```
1 -- Join relevant tables to find the
2 -- category-wise distribution of pizzas.
3
4• select category , count(name) from pizza_types
5 group by category;
```

Result Grid | Filter Rows:

	category	count(name)
▶	Chicken	6
	Classic	8
	Supreme	9
	Veggie	9

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

```
1 -- Group the orders by date and calculate the average
2 -- number of pizzas ordered per day.
3
4 select round(avg(quantity),0) from
5 (select orders.order_date, sum(order_details.quantity) as quantity
6 from orders join order_details
7 on orders.order_id = order_details.order_id
8 group by orders.order_date) as order_quantity ;
```

Result Grid	
	round(avg(quantity),0)
▶	138

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,  
4   sum(order_details.quantity * pizzas.price) as revenue  
5   from pizza_types join pizzas  
6   on pizzas.pizza_type_id = pizza_types.pizza_type_id  
7   join order_details  
8   on order_details.pizza_id = pizzas.pizza_id  
9   group by pizza_types.name order by revenue desc limit 3;
```

Result Grid | Filter Rows:

	name	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.

```
4 • select pizza_types.category,  
5   round(sum(order_details.quantity*pizzas.price) / (SELECT  
6     ROUND(SUM(order_details.quantity * pizzas.price),  
7       2) AS total_sales  
8   FROM  
9     order_details  
10    JOIN  
11      pizzas ON pizzas.pizza_id = order_details.pizza_id) *100,2) as revenue  
12   from pizza_types join pizzas  
13   on pizza_types.pizza_type_id = pizzas.pizza_type_id  
14   join order_details  
15   on order_details.pizza_id = pizzas.pizza_id  
16   group by pizza_types.category order by revenue desc;
```

	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68

ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```
3 • select order_date,  
4 sum(revenue) over(order by order_date) as cum_revenue  
5 from  
6 (select orders.order_date,  
7 sum(order_details.quantity * pizzas.price) as revenue  
8 from order_details join pizzas  
9 on order_details.pizza_id = pizzas.pizza_id  
10 join orders  
11 on orders.order_id = order_details.order_id  
12 group by orders.order_date) as sales;
```

	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	11611929.55
	2015-01-06	14358.5

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

```
3 • select name, revenue from
4   (select category, name, revenue,
5    rank() over(partition by category order by revenue desc) as rn
6   from
7   (select pizza_types.category, pizza_types.name,
8    sum(order_details.quantity) * pizzas.price) as revenue
9   from pizza_types join pizzas
10  on pizza_types.pizza_type_id = pizzas.pizza_type_id
11  join order_details
12  on order_details.pizza_id = pizzas.pizza_id
13  group by pizza_types.category, pizza_types.name) as a) as b
14 where rn <= 3;|     I
```

	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