Muskan Vyas
Pizza Sales Project

1. retrieve the total numbers of orders placed

```
SELECT
    *
FROM
    orders_pizza;
SELECT
    COUNT(order_id) AS total_orders
FROM
    orders_pizza;
```

	total_orders
▶	21350

2. calculate the total revenue generated from pizza sales.

```
SELECT
    ROUND(SUM(order_details.quantity * pizzas.price),
            2) AS total_sales
FROM
    order_details
        JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id;
```

	total_sales
▶	817860.05

3. identify the highest priced pizza

```
SELECT
    pizza_types.name, pizzas.price
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY pizzas.price DESC
LIMIT 1;
```

	name	price
▶	The Greek Pizza	35.95

4. identify the most common pizza size ordered.

```
SELECT
    pizzas.size,
    COUNT(order_details.order_details_id) AS order_count
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size
ORDER BY order_count DESC;
```

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

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

```sql
SELECT
    pizza_types.name, SUM(order_details.quantity) AS quantity
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.name
ORDER BY quantity DESC
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

6. join necessary table to find the total quantity of each pizza category ordered.

```sql
SELECT
    pizza_types.category,
    SUM(order_details.quantity) AS quantity
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
ORDER BY quantity DESC;
```

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

7. determine the distribution of orders by hour of the day.

```
SELECT
    HOUR(order_time) AS hour, COUNT(order_id) AS order_count
FROM
    orders_pizza
GROUP BY HOUR(order_time);
```

hour	order_count
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336
18	2399
19	2009
20	1642
21	1198
22	663
23	28
10	8
9	1

8. join the relevant tables to category-wise distribution of pizzas

```
select category, count(name) from pizza_types
group by category;
```

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

9. group the orders by date and calculate the avg number of pizzas ordered per day.

```sql
SELECT
    ROUND(AVG(quantity), 0) as avg_pizza_ordered_per_day
FROM
    (SELECT
        orders_pizza.order_date,
            SUM(order_details.quantity) AS quantity
    FROM
        orders_pizza
    JOIN order_details ON orders_pizza.order_id = order_details.order_id
    GROUP BY orders_pizza.order_date) AS order_quantity;
```

	avg_pizza_ordered_per_day
▶	138

10. top 3 most ordered pizzas based on revenue

```sql
SELECT
    pizza_types.name,
    SUM(order_details.quantity * pizzas.price) AS revenue
FROM
    pizza_types
        JOIN
    pizzas ON pizzas.pizza_type_id = pizza_types.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY revenue DESC
LIMIT 3;
```

name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5

11. calculate the perecentage contribution of each pizza type to total revenue

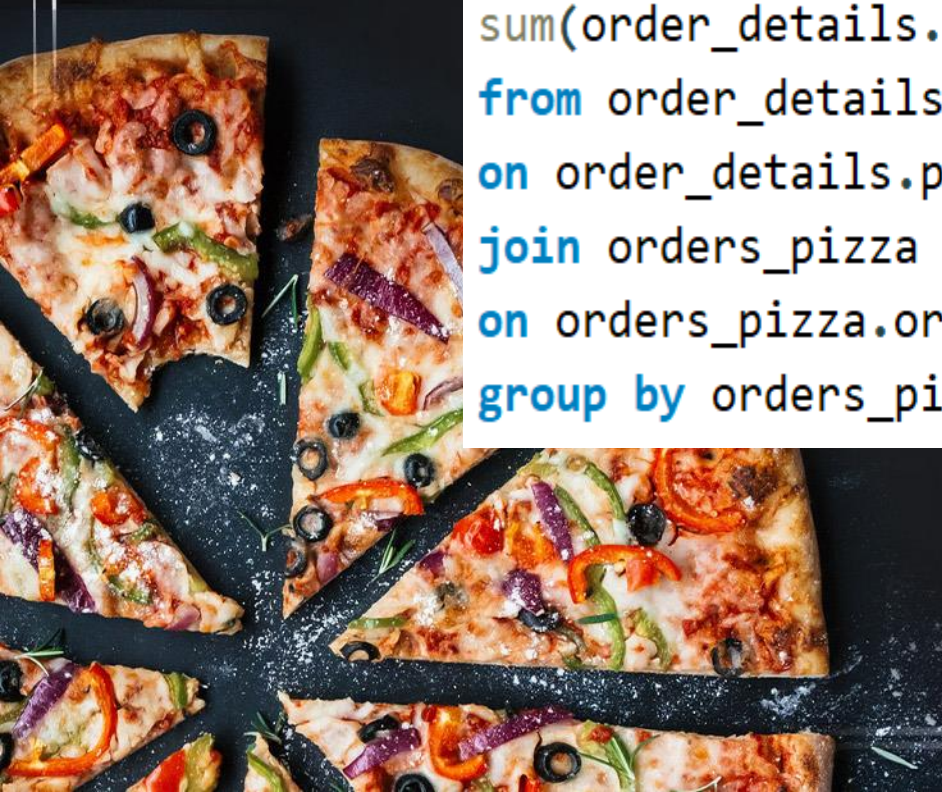
```sql
SELECT
    pizza_types.category,
    ROUND(SUM(order_details.quantity * pizzas.price) / (SELECT
                    ROUND(SUM(order_details.quantity * pizzas.price),
                                2) AS total_sales
                FROM
                    order_details
                        JOIN
                    pizzas ON pizzas.pizza_id = order_details.pizza_id) * 100,
            2) AS revenue
FROM
    pizza_types
        JOIN
        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
ORDER BY revenue DESC;
```

OUTPUT

	category	revenue
▶	Classic	26.91
	Supreme	25.46
	Chicken	23.96
	Veggie	23.68

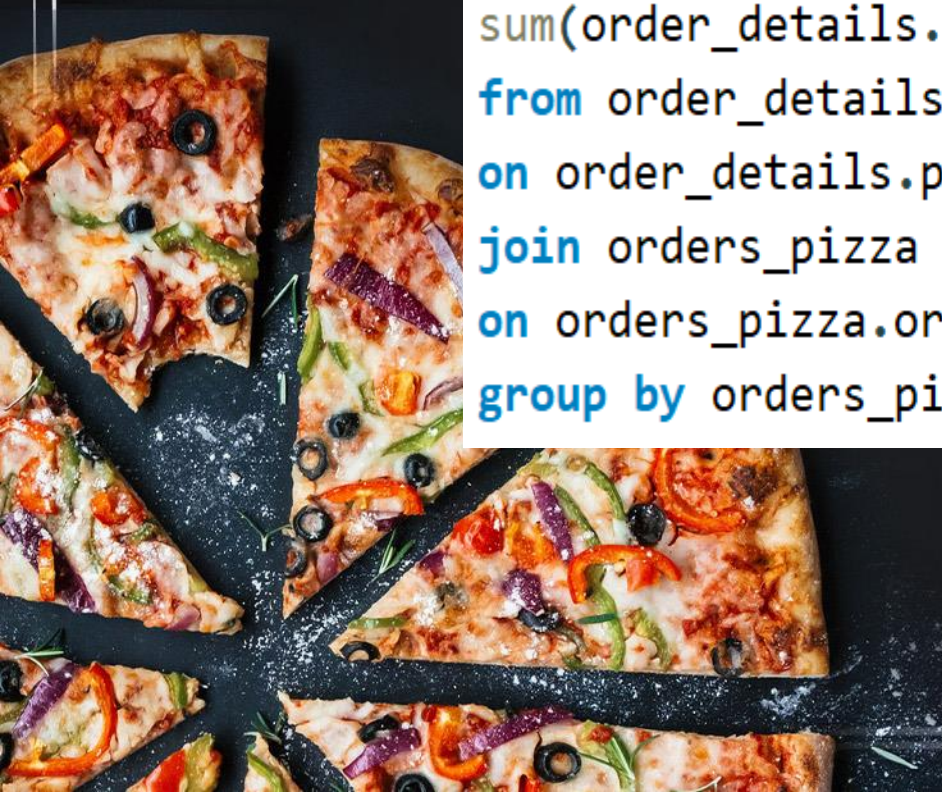
12. Analyze the cummulative revenue generated over time.

```
select order_date,
sum(revenue) over (order by order_date) as cum_revenue
from
(select orders_pizza.order_date,
sum(order_details.quantity * pizzas.price) as revenue
from order_details join pizzas
on order_details.pizza_id = pizzas.pizza_id
join orders_pizza
on orders_pizza.order_id = order_details.order_id
group by orders_pizza.order_date) as sales;
```



12. Analyze the cummulative revenue generated over time.

```
select order_date,
sum(revenue) over (order by order_date) as cum_revenue
from
(select orders_pizza.order_date,
sum(order_details.quantity * pizzas.price) as revenue
from order_details join pizzas
on order_details.pizza_id = pizzas.pizza_id
join orders_pizza
on orders_pizza.order_id = order_details.order_id
group by orders_pizza.order_date) as sales;
```



12. OUTPUT

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
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	29831.300000000003
2015-01-14	32358.700000000004
2015-01-15	34343.50000000001
2015-01-16	36937.65000000001
2015-01-17	39001.75000000001
2015-01-18	40978.600000000006
2015-01-19	43365.75000000001

order_date	cum_revenue
2015-01-20	45763.65000000001
2015-01-21	47804.20000000001
2015-01-22	50300.90000000001
2015-01-23	52724.600000000006
2015-01-24	55013.850000000006
2015-01-25	56631.40000000001
2015-01-26	58515.80000000001
2015-01-27	61043.85000000001
2015-01-28	63059.85000000001
2015-01-29	65105.150000000016
2015-01-30	67375.45000000001
2015-01-31	69793.30000000002
2015-02-01	72982.50000000001
2015-02-02	75311.10000000002
2015-02-03	77925.90000000002
2015-02-04	80159.80000000002
2015-02-05	82375.60000000002
2015-02-06	84885.55000000002
2015-02-07	87123.20000000001

order_date	cum_revenue
2015-02-27	132413.30000000002
2015-02-28	134952.90000000002
2015-03-01	136551.45
2015-03-02	138930.5
2015-03-03	141218.4
2015-03-04	143662.69999999998
2015-03-05	146013.34999999998
2015-03-06	148527.3
2015-03-07	150927.75
2015-03-08	153115.9
2015-03-09	155450.44999999998
2015-03-10	157839.15
2015-03-11	160046.85
2015-03-12	162041.75
2015-03-13	164828.4
2015-03-14	166867.85
2015-03-15	168936.45
2015-03-16	171231.5
2015-03-17	174196.8

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

```sql
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;
```

OUTPUT

	name	revenue
▶	The Thai Chicken Pizza	43434.25
	The Barbecue Chicken Pizza	42768
	The Classic Deluxe Pizza	38180.5
	The Hawaiian Pizza	32273.25
	The Spicy Italian Pizza	34831.25
	The Italian Supreme Pizza	33476.75
	The Four Cheese Pizza	32265.70000000065
	The Mexicana Pizza	26780.75

THANK YOU