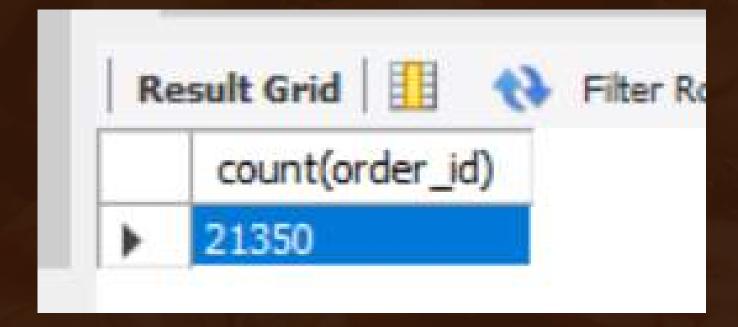
RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

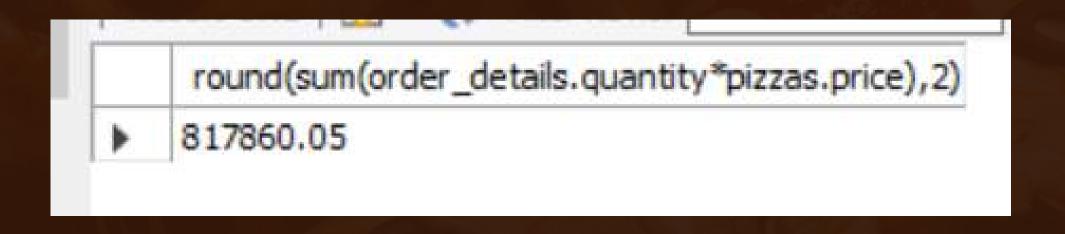
```
select count(order_id) from orders;
```



2) CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.



```
select
round(sum(order_details.quantity*pizzas.price),2)
-- select 1st table
from order_details join pizzas
-- select 2nd table
on pizzas.pizza_id=order_details.pizza_id
```

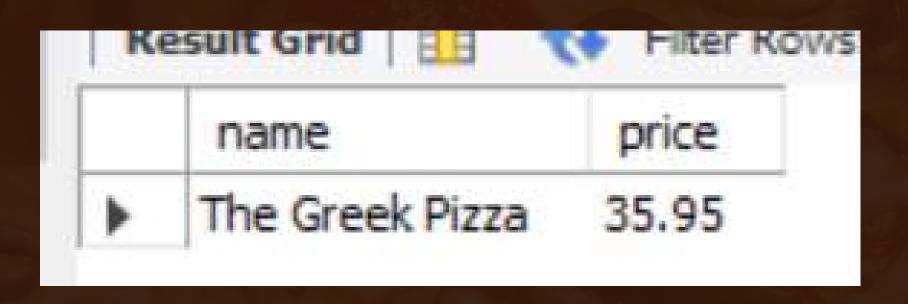


3) IDENTIFY THE HIGHEST-PRICED PIZZA





```
select pizza_types .name , pizzas.price
from pizza_types join pizzas
-- common colum to join-them
on pizza_types.pizza_type_id=pizzas.pizza_type_id
order by pizzas.price desc limit 1;
```



4) IDENTIFY THE MOST COMMON PIZZA SIZE: ORDERED.



```
select pizzas.size,count(order_details.ordrer_details_id) as count_order
FROM pizzas join order_details
on pizzas.pizza_id=order_details.pizza_id
group by pizzas.size order by count_order desc;
-- use have use count so it will generate an error if we dont write group by
```

	size	count_order
•	L	18526
	M	15385
	S	14137
	XL	544
	XXL	28

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





```
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
-- as we use sum in our query so we have to use groupby funtion
-- group kaisa karna bole ti jider be sum use karte us se pahile ka gro
-- order by use karte kaiku bole to top5 hona hai na
group by pizza_types.name order by quantity desc limit 5;
```



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



```
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;
```

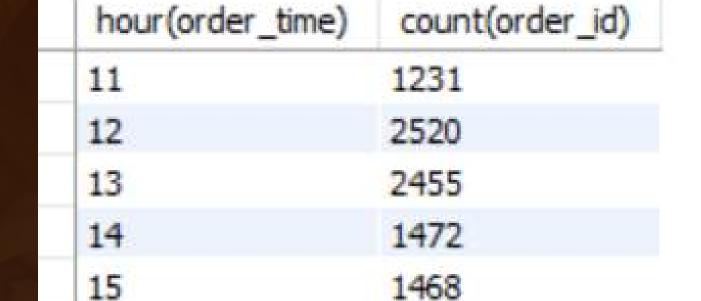
	category	quantity
Þ	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050

7) DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.





```
select hour(order_time) ,count(order_id) from orders
group by hour(order_time);
```



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





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



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





```
select round(avg (quantity),0) from

(select orders.order_date ,sum(order_details.quantity) as quantity
from orders join order_details
on orders.order_id=order_details.order_id
group by orders.order_date) as order_quantity;
```

```
round(avg
(quantity),0)
```

10) DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE



```
select 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.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 PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.





```
select pizza_types.category,
round(sum(order_details.quantity * pizzas.price) /
-- this is the seconf table
select
round(sum(order_details.quantity*pizzas.price),2)
from order_details join pizzas
on pizzas.pizza_id=order_details.pizza_id) * 100,2) 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 order by revenue desc limit 3;
```



•	category	revenue
	Classic	26.91
	Supreme	25.46
	Chicken	23.96

12) ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME



```
select order_date,
sum(revenue) over (order by order_date) as cuma_revenue
from

(select orders.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
on orders.order_id=order_details.order_id
group by orders.order_date )as sales;
```

1 10	result did [25] (4 mer nons.				
	order_date	cuma_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			