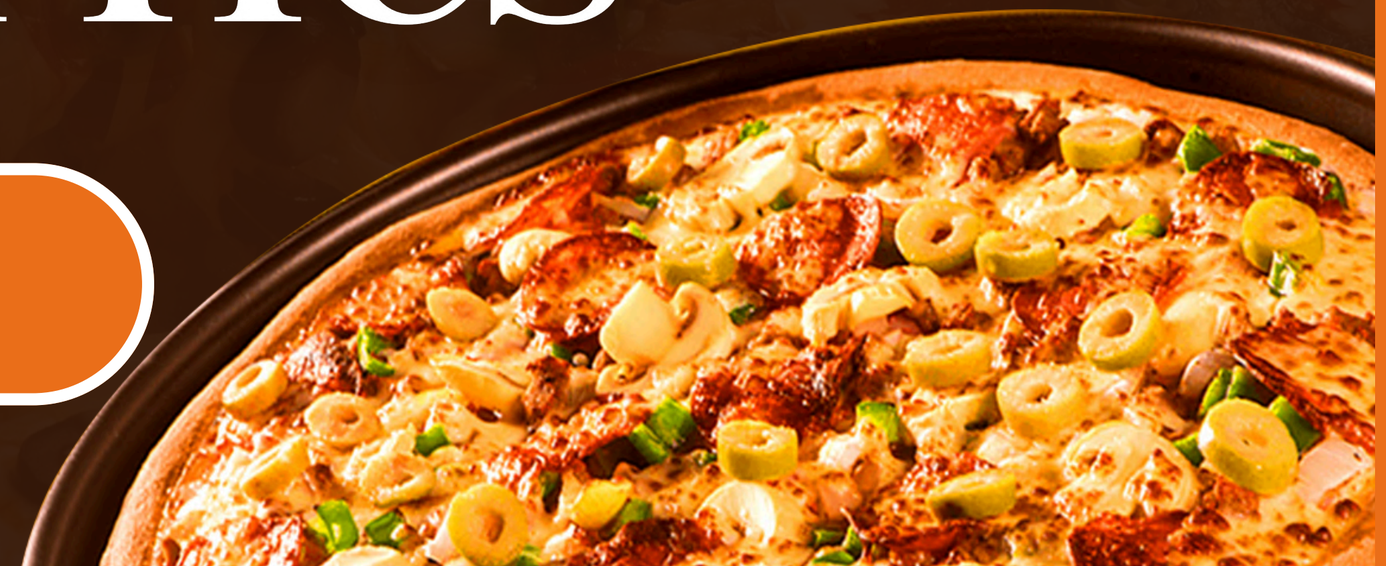
A SQL-Based Data Analysis Project

PIZZA RESTAURANT ANALYTICS

SQL

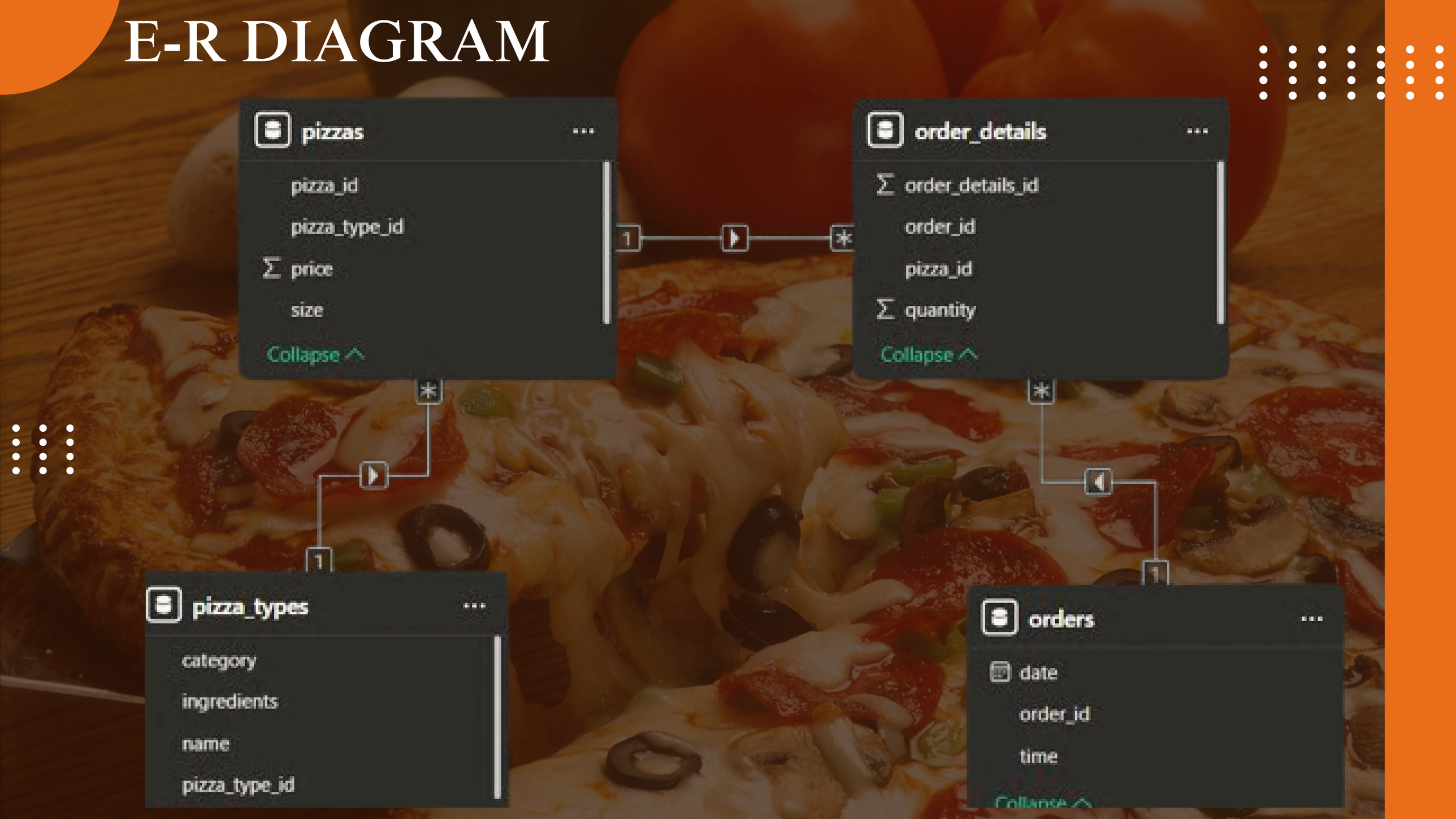


Developed By :
Sagar Sawant

ABSTRACT

This project focuses on analyzing pizza sales data using SQL to uncover key business insights. Through structured queries and views, we explored order trends, revenue patterns, pizza popularity, and ingredient usage. The analysis covered basic metrics like total orders and revenue, as well as advanced insights such as category-wise revenue contribution and cumulative sales. A special focus was given to identifying high-demand ingredients from top-selling pizzas to support smarter inventory decisions.

E-R DIAGRAM


pizzas
pizza_id
pizza_type_id
price
size
Collapse
order_details
order_details_id
order_id
pizza_id
quantity
Collapse
pizza_types
category
ingredients
name
pizza_type_id
orders
date
order_id
time
1
*

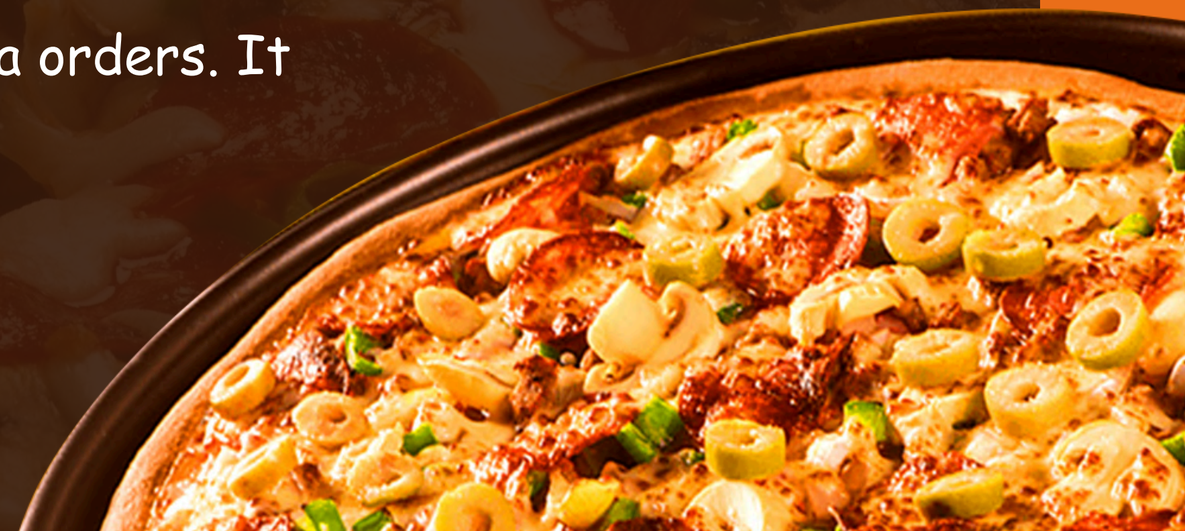
STRUCTURE OF TABLE

ORDERS TABLE

```
describe orders;
```

	Field	Type	Null	Key	Default	Extra
▶	order_id	int	NO	PRI	NULL	
	order_date	date	NO		NULL	
	order_time	time	NO		NULL	

This table contains information about individual pizza orders. It includes fields: order_id,order_date,order_time



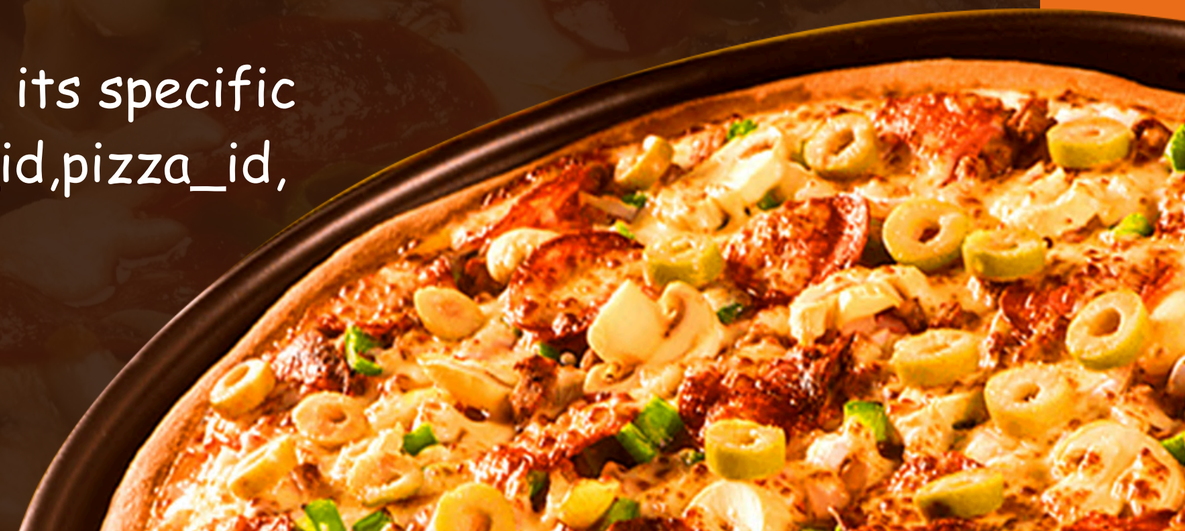
STRUCTURE OF TABLE

ORDERS_DETAILS TABLE

```
describe orders_details;
```

	Field	Type	Null	Key	Default	Extra
▶	order_details_id	int	NO	PRI	NULL	
	order_id	int	NO	MUL	NULL	
	pizza_id	text	NO		NULL	
	quantity	int	NO		NULL	

This table stores the breakdown of each order into its specific pizza items. It has fields: order_details_id,order_id,pizza_id, quantity.



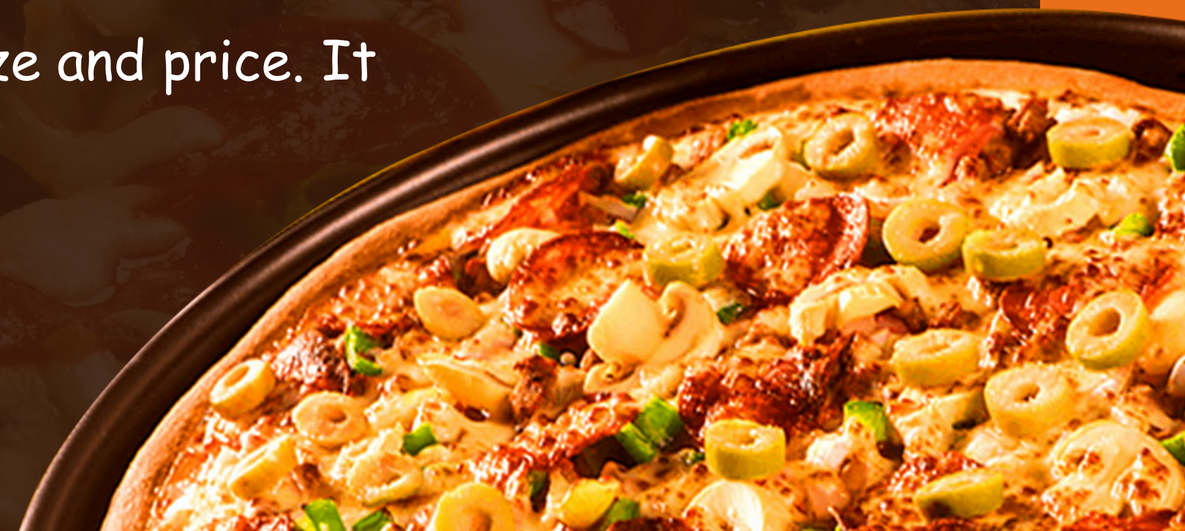
STRUCTURE OF TABLE

PIZZAS TABLE

```
describe pizzas;
```

	Field	Type	Null	Key	Default	Extra
▶	pizza_id	text	YES		NULL	
	pizza_type_id	text	YES		NULL	
	size	text	YES		NULL	
	price	double	YES		NULL	

The pizzas table maps each unique pizza ID to its size and price. It contains fields pizza_id, pizza_type_id, size, price



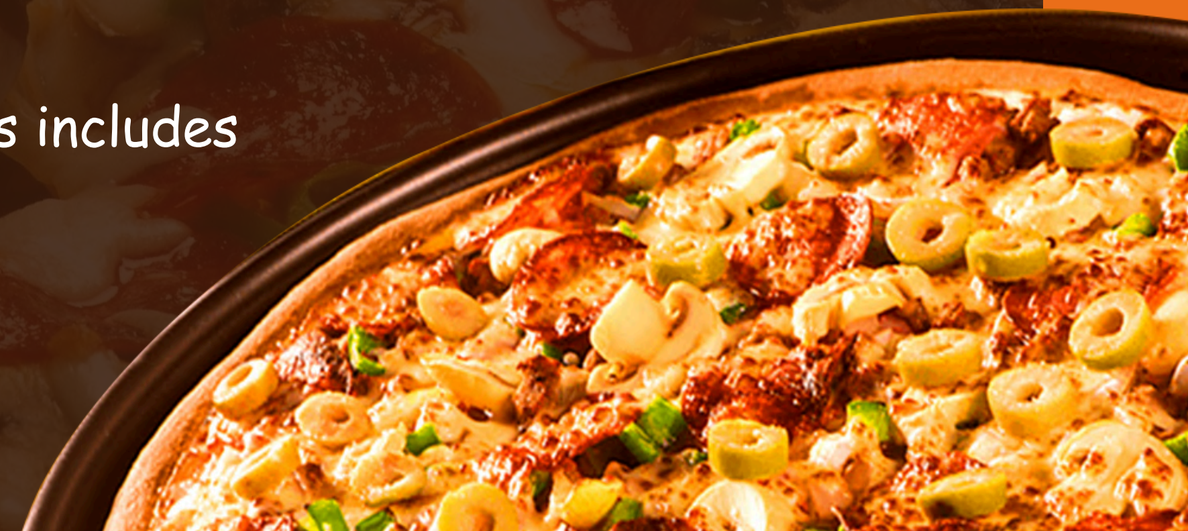
STRUCTURE OF TABLE

PIZZA_TYPES TABLE

```
describe pizza_types;
```

	Field	Type	Null	Key	Default	Extra
▶	pizza_type_id	text	YES		NULL	
	name	text	YES		NULL	
	category	text	YES		NULL	
	ingredients	text	YES		NULL	

This table defines each type of pizza offered,Fields includes pizza_type_id,name,category,ingredients

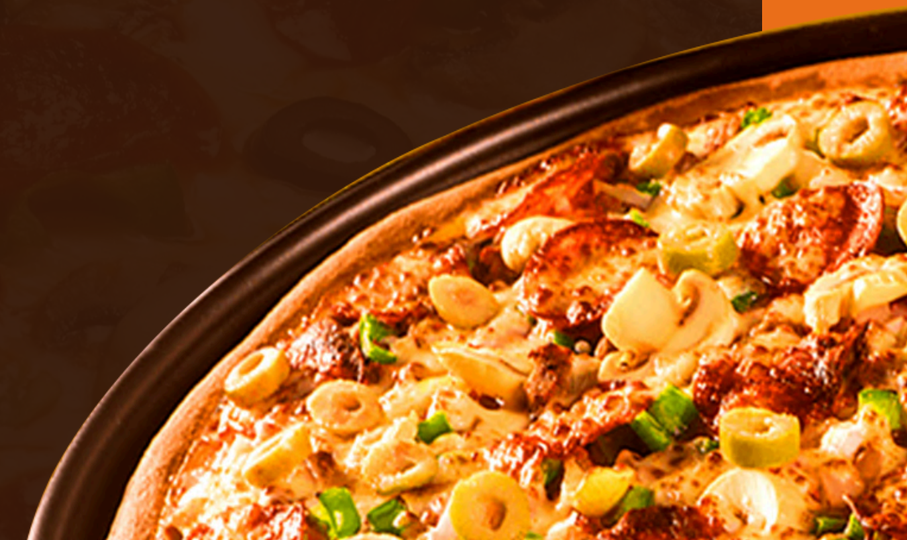


CONTENTS OF TABLE

```
5 •    SELECT * FROM orders ;
```

Result Grid | Filter Rows:

order_id	order_date	order_time
1	2015-01-01	11:38:36
2	2015-01-01	11:57:40
3	2015-01-01	12:12:28
4	2015-01-01	12:16:31
5	2015-01-01	12:21:30
6	2015-01-01	12:29:36
7	2015-01-01	12:50:37
8	2015-01-01	12:51:37
9	2015-01-01	12:52:01
10	2015-01-01	13:00:15
11	2015-01-01	13:02:59
12	2015-01-01	13:04:41
13	2015-01-01	13:11:55
14	2015-01-01	13:14:19
15	2015-01-01	13:33:00

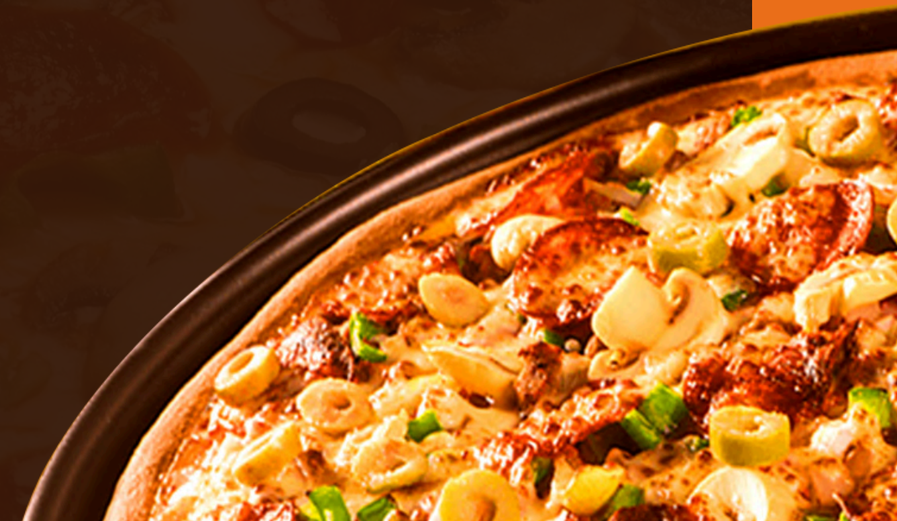


CONTENTS OF TABLE

```
5 •    SELECT * FROM orders_details ;
```

Result Grid | Filter Rows: | Edit:

order_details_id	order_id	pizza_id	quantity
1	1	hawaiian_m	1
2	2	classic_dlx_m	1
3	2	five_cheese_l	1
4	2	ital_supr_l	1
5	2	mexicana_m	1
6	2	thai_ckn_l	1
7	3	ital_supr_m	1
8	3	prsc_argla_l	1
9	4	ital_supr_m	1
10	5	ital_supr_m	1
11	6	bbq_ckn_s	1
12	6	the_greek_s	1
13	7	spinach_supr_s	1
14	8	spinach_supr_s	1
15	9	classic_dlx_s	1



CONTENTS OF TABLE

```
16 •    SELECT * FROM pizza_types;
```

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

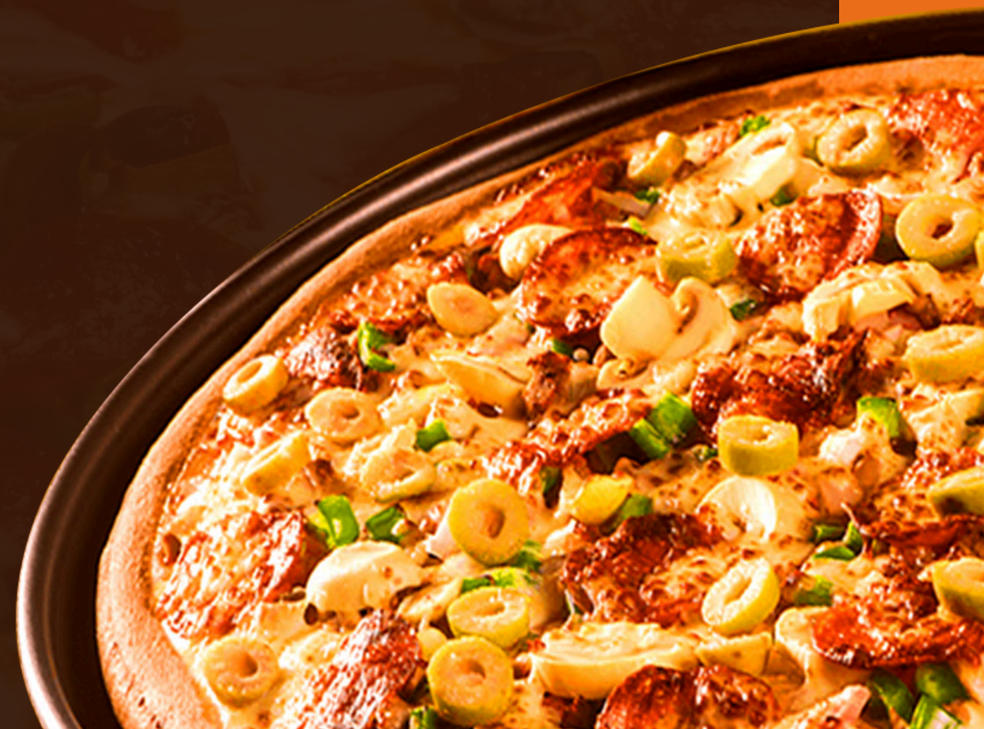
pizza_type_id	name	category	ingredients
bbq_ckn	The Barbecue Chicken Pizza	Chicken	Barbecued Chicken, Red Peppers, Green Peppers, Tomatoes, Red Onions, Barbecue Sauce
cali_ckn	The California Chicken Pizza	Chicken	Chicken, Artichoke, Spinach, Garlic, Jalapeno Peppers, Fontina Cheese, Gouda Cheese
ckn_alfredo	The Chicken Alfredo Pizza	Chicken	Chicken, Red Onions, Red Peppers, Mushrooms, Asiago Cheese, Alfredo Sauce
ckn_pesto	The Chicken Pesto Pizza	Chicken	Chicken, Tomatoes, Red Peppers, Spinach, Garlic, Pesto Sauce
southw_ckn	The Southwest Chicken Pizza	Chicken	Chicken, Tomatoes, Red Peppers, Red Onions, Jalapeno Peppers, Corn, Cilantro, Chipotle ...
thai_ckn	The Thai Chicken Pizza	Chicken	Chicken, Pineapple, Tomatoes, Red Peppers, Thai Sweet Chilli Sauce
big_meat	The Big Meat Pizza	Classic	Bacon, Pepperoni, Italian Sausage, Chorizo Sausage
classic_dlx	The Classic Deluxe Pizza	Classic	Pepperoni, Mushrooms, Red Onions, Red Peppers, Bacon
hawaiian	The Hawaiian Pizza	Classic	Sliced Ham, Pineapple, Mozzarella Cheese
ital_cpclo	The Italian Capocollo Pizza	Classic	Capocollo, Red Peppers, Tomatoes, Goat Cheese, Garlic, Oregano
napolitana	The Napolitana Pizza	Classic	Tomatoes, Anchovies, Green Olives, Red Onions, Garlic
pep_msh_pep	The Pepperoni, Mushroom, ...	Classic	Pepperoni, Mushrooms, Green Peppers
pepperoni	The Pepperoni Pizza	Classic	Mozzarella Cheese, Pepperoni

CONTENTS OF TABLE

```
5 •    SELECT * FROM pizzas ;
```

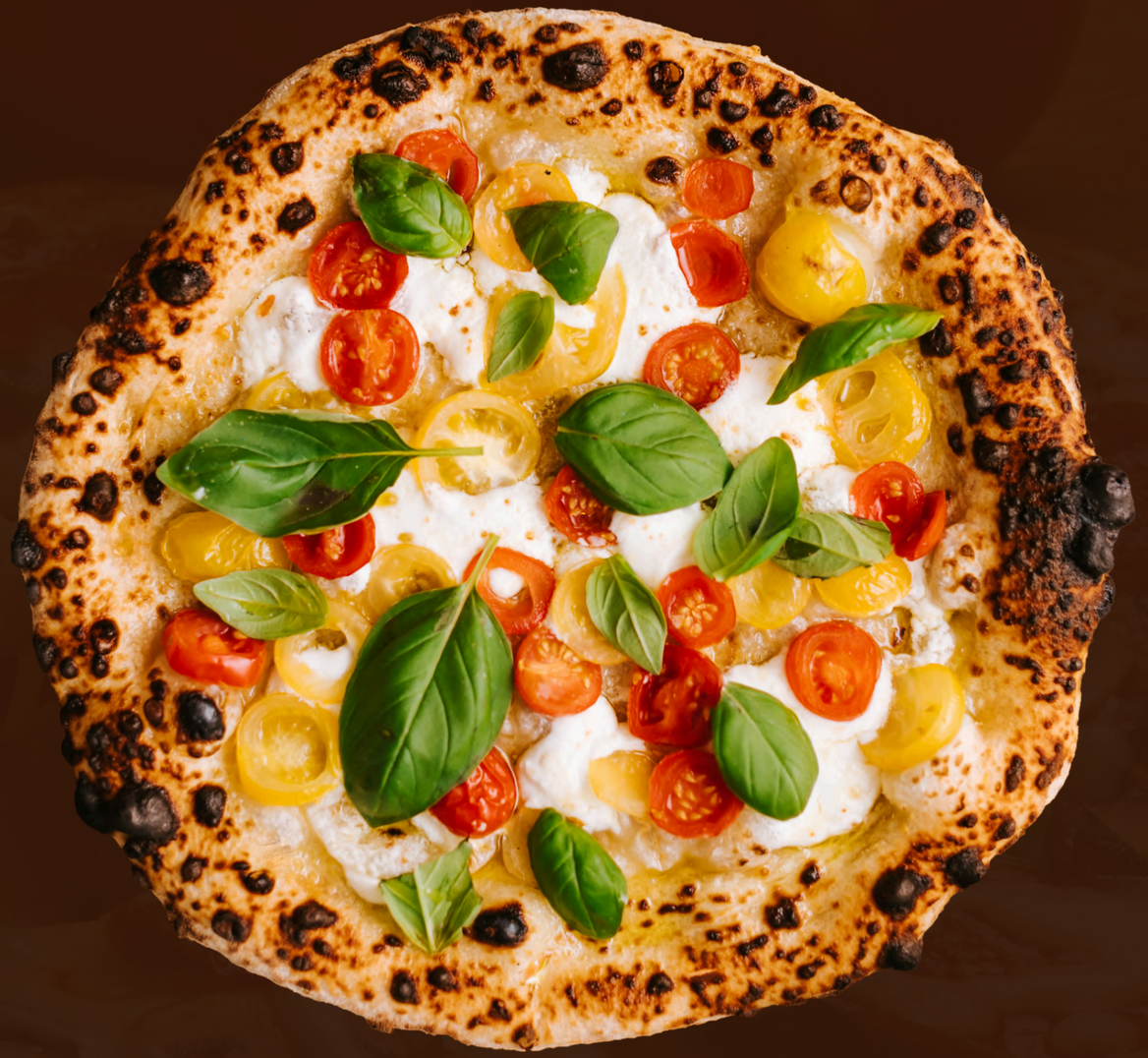
Result Grid | Filter Rows:

pizza_id	pizza_type_id	size	price
bbq_ckn_s	bbq_ckn	S	12.75
bbq_ckn_m	bbq_ckn	M	16.75
bbq_ckn_l	bbq_ckn	L	20.75
cali_ckn_s	cali_ckn	S	12.75
cali_ckn_m	cali_ckn	M	16.75
cali_ckn_l	cali_ckn	L	20.75
ckn_alfredo_s	ckn_alfredo	S	12.75
ckn_alfredo_m	ckn_alfredo	M	16.75
ckn_alfredo_l	ckn_alfredo	L	20.75
ckn_pesto_s	ckn_pesto	S	12.75
ckn_pesto_m	ckn_pesto	M	16.75
ckn_pesto_l	ckn_pesto	L	20.75
southw_ckn_s	southw_ckn	S	12.75
southw_ckn_m	southw_ckn	M	16.75
southw_ckn_l	southw_ckn	L	20.75
thai_ckn_s	thai_ckn	S	12.75



JOINS

A join is used to combine rows from two or more tables based on a related column, allowing you to retrieve data from multiple tables in a single query.

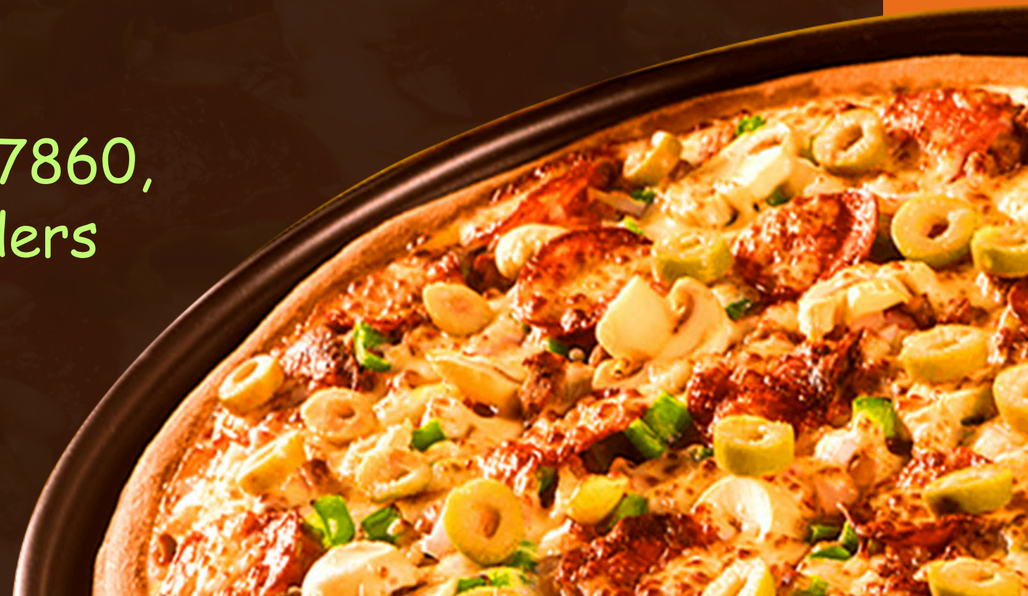


Display Total Revenue Generated From Pizza Sales.

```
select round(sum(od.quantity*p.price)) as total_revenue
from orders_details as od
join
pizzas as p on od.pizza_id=p.pizza_id;
```

	total_revenue
▶	817860

Conclusion : The business earned a total revenue of 817860, indicating its overall sales performance from pizza orders

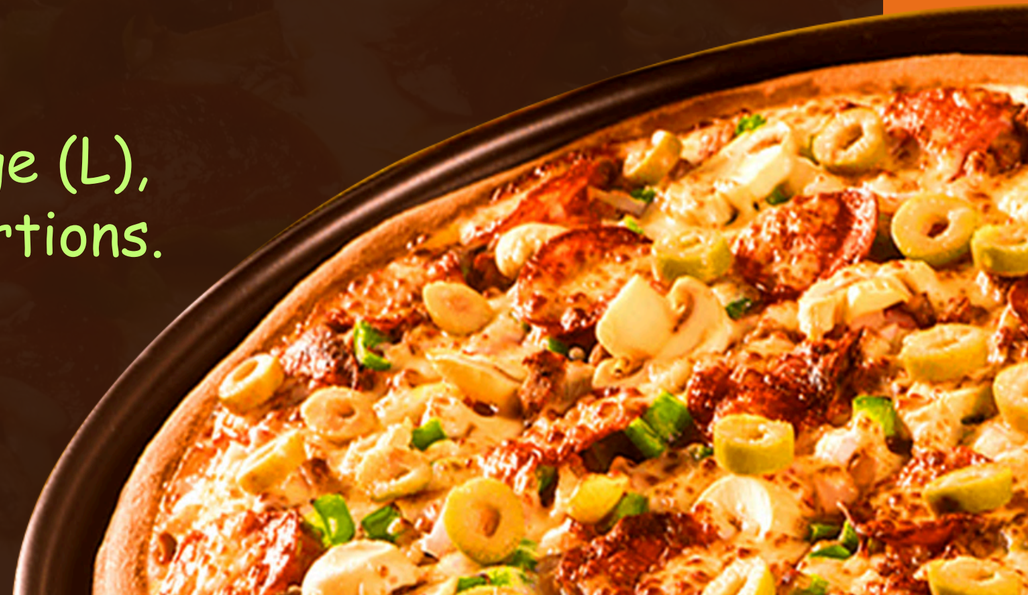


Display The Most Common Pizza Size Ordered

```
select p.size , count(od.order_details_id) as count_orders from pizzas as p
join orders_details as od on p.pizza_id =od.pizza_id
group by p.size order by count_orders desc limit 1;
```

size	count_orders
L	18526

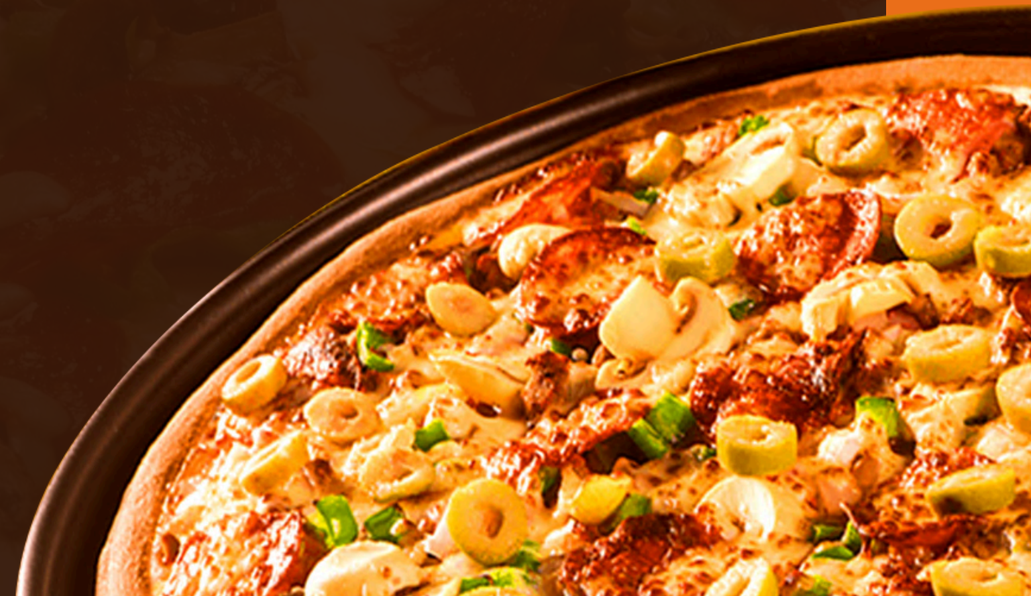
Conclusion : most commonly ordered pizza size is Large (L), indicating a strong customer preference for larger portions.



Display the top 5 most ordered pizza types along with their quantities

```
SELECT pt.name, SUM(od.quantity) AS total_orders_quantity FROM pizza_types AS pt
JOIN
pizzas AS p ON pt.pizza_type_id = p.pizza_type_id
JOIN orders_details AS od ON p.pizza_id = od.pizza_id GROUP BY pt.name
ORDER BY total_orders_quantity DESC LIMIT 5;
```

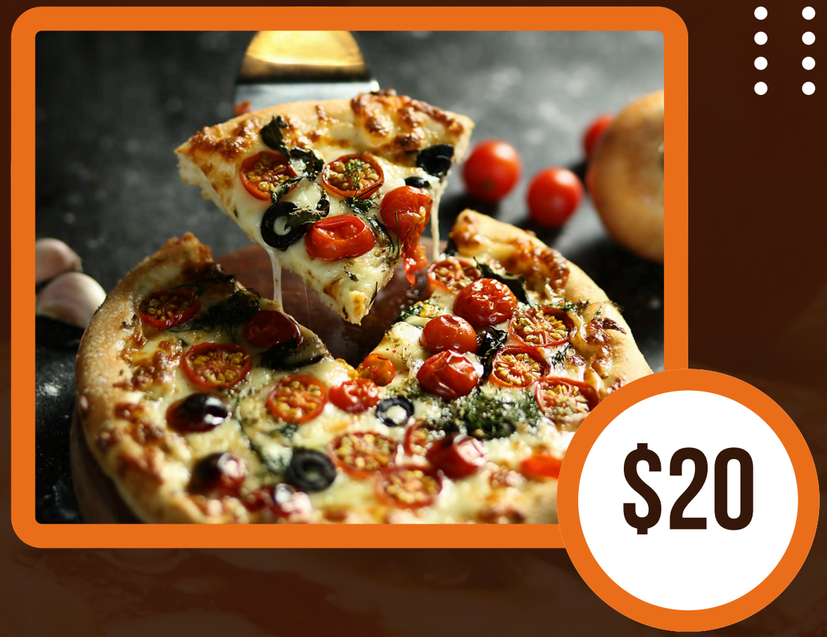
name	total_orders_quantity
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

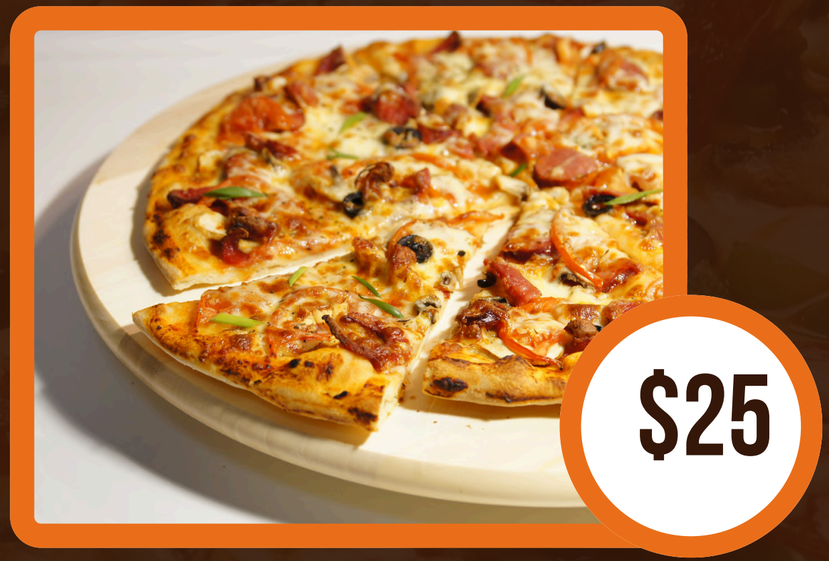


Conclusion :top 5 most ordered pizzas indicate strong customer preference trends that can guide both menu focus and ingredient stocking priorities.

SUBQUERY

A subquery is a query nested inside another query, used to return a result that is then used by the outer query, often in a WHERE, HAVING, or FROM clause.





Display the top 3 most ordered pizza types based on revenue for each pizza category.

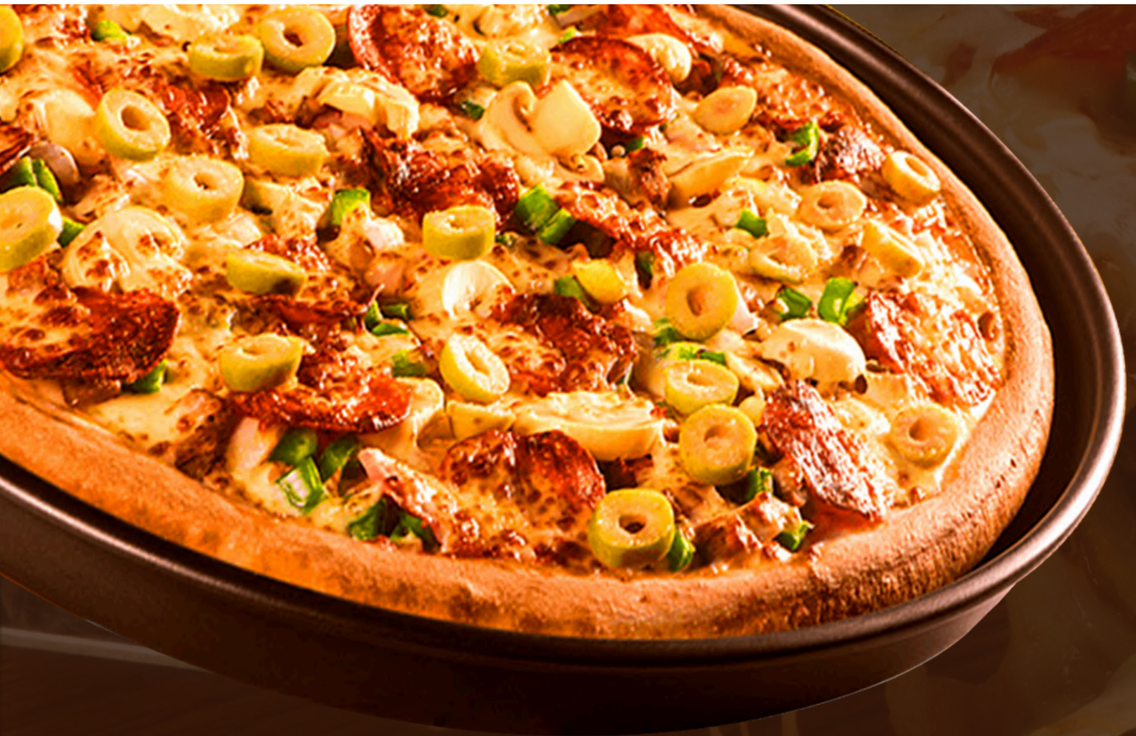
```sql
select name , category, round(revenue) from
(select name,category ,revenue,
rank() over (partition by category order by revenue desc) as ranknum from
(select pt.name,pt.category ,sum(od.quantity * p.price) as revenue from pizzas as p
join orders_details as od on p.pizza_id=od.pizza_id
join pizza_types as pt on pt.pizza_type_id=p.pizza_type_id
group by pt.category,pt.name) as a) as b
where ranknum <= 3;
```

name	category	round(revenue)
The Thai Chicken Pizza	Chicken	43434
The Barbecue Chicken Pizza	Chicken	42768
The California Chicken Pizza	Chicken	41410
The Classic Deluxe Pizza	Classic	38180
The Hawaiian Pizza	Classic	32273
The Pepperoni Pizza	Classic	30162
The Spicy Italian Pizza	Supreme	34831
The Italian Supreme Pizza	Supreme	33477
The Sicilian Pizza	Supreme	30940
The Four Cheese Pizza	Veggie	32266
The Mexicana Pizza	Veggie	26781
The Five Cheese Pizza	Veggie	26066

Conclusion : The top revenue-generating pizzas in each category reveal customer preferences that can guide future marketing and menu planning.

Display the cumulative revenue generated over date.

```sql
select OrderDate,sum(rev) over (order by OrderDate) as Cumulative_Revenue
from
(select o.order_date as OrderDate , round(sum(od.quantity * p.price)) as rev
from orders as o
join orders_details as od on o.order_id=od.order_id
join pizzas as p on p.pizza_id=od.pizza_id
group by o.order_date) as sale ;
```



OrderDate	Cumulative_Revenue
2015-01-01	2714
2015-01-02	5446
2015-01-03	8108
2015-01-04	9863
2015-01-05	11929
2015-01-06	14358
2015-01-07	16560
2015-01-08	19398
2015-01-09	21525
2015-01-10	23989
2015-01-11	25861
2015-01-12	27780
2015-01-13	29830
2015-01-14	32357
2015-01-15	34342
2015-01-16	36936
2015-01-17	39000
2015-01-18	40977
2015-01-19	43364
2015-01-20	45762
2015-01-21	47803
2015-01-22	50300
2015-01-23	52724
2015-01-24	55013
2015-01-25	56631

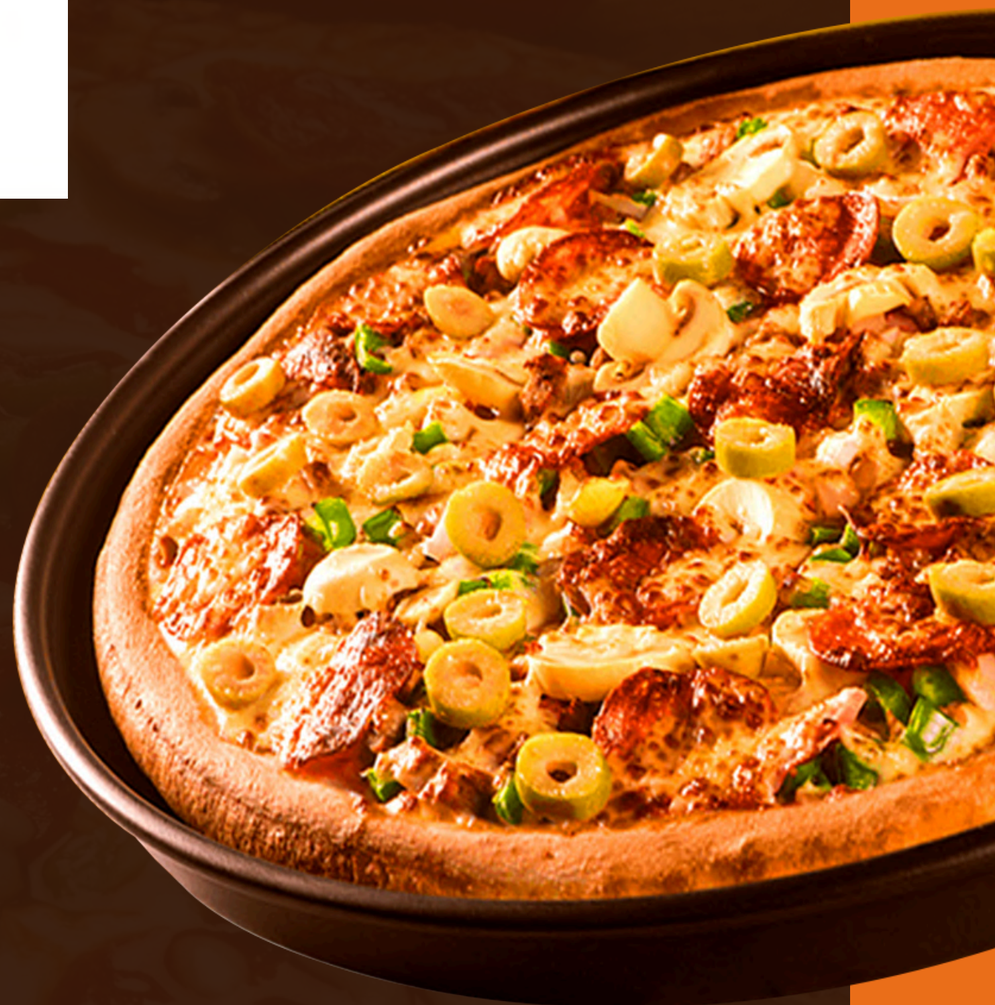
OrderDate	Cumulative_Revenue
2015-12-07	769966
2015-12-08	771822
2015-12-09	774394
2015-12-10	776380
2015-12-11	779014
2015-12-12	780974
2015-12-13	783219
2015-12-14	785392
2015-12-15	787779
2015-12-16	790014
2015-12-17	791895
2015-12-18	794781
2015-12-19	797085
2015-12-20	799190
2015-12-21	801291
2015-12-22	803174
2015-12-23	805418
2015-12-24	807556
2015-12-26	809199
2015-12-27	810618
2015-12-28	812255
2015-12-29	813608
2015-12-30	814946
2015-12-31	817862

Conclusion : The data confirms ongoing customer demand, with total revenue increasing steadily day by day.

Display average number of pizzas ordered per day.

```sql
SELECT ROUND(AVG(quantity)) AS Number_Pizza_order_PerDay
FROM (SELECT o.order_date, SUM(od.quantity) AS quantity
FROM orders AS o
JOIN orders_details AS od ON o.order_id = od.order_id
GROUP BY o.order_date) AS order_pizza;
```

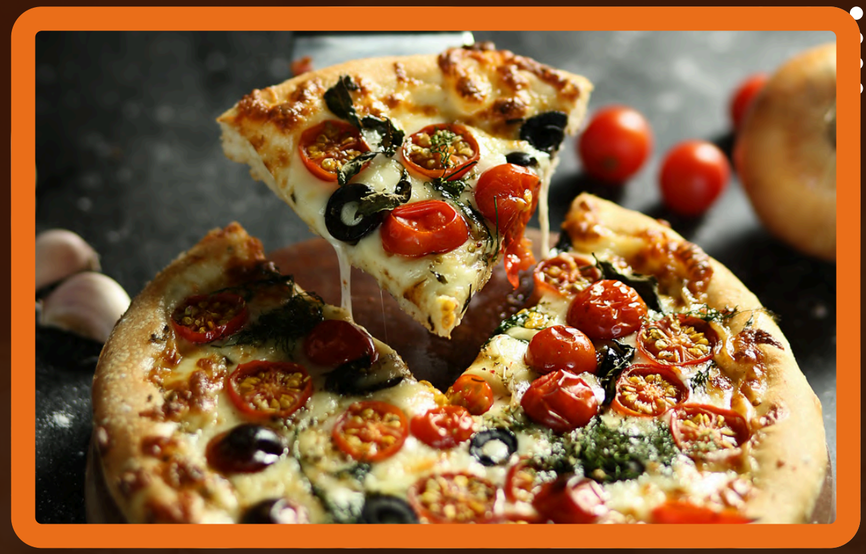
Number_Pizza_order_PerDay
138



Conclusion : This reveals the typical volume the kitchen needs to prepare for daily operations

VIEW

A view is a virtual table created by a SELECT query that can encapsulate complex logic, allowing users to query the view as if it were a real table. It does not store data itself but presents a stored result of a query.



Create a view named bottom 5 low_performing_pizzas that lists pizzas that have sold less than the average quantity sold across all pizzas.

```sql
CREATE VIEW low_performing_pizzas AS
with pizza_sale as
(SELECT pt.name as pizza_name,sum(od.quantity) as sold_qty from pizzas as p
join pizza_types as pt on pt.pizza_type_id=p.pizza_type_id
join orders_details as od on od.pizza_id=p.pizza_id
Group by pt.name) ,
 average_sale as(
select avg(sold_qty) as avg_sold from pizza_sale
)
select ps.pizza_name ,ps.sold_qty as total_quantity_sold
from pizza_sale as ps , average_sale as av
where ps.sold_qty < av.avg_sold order by ps.sold_qty limit 5;
select * from low_performing_pizzas;
```

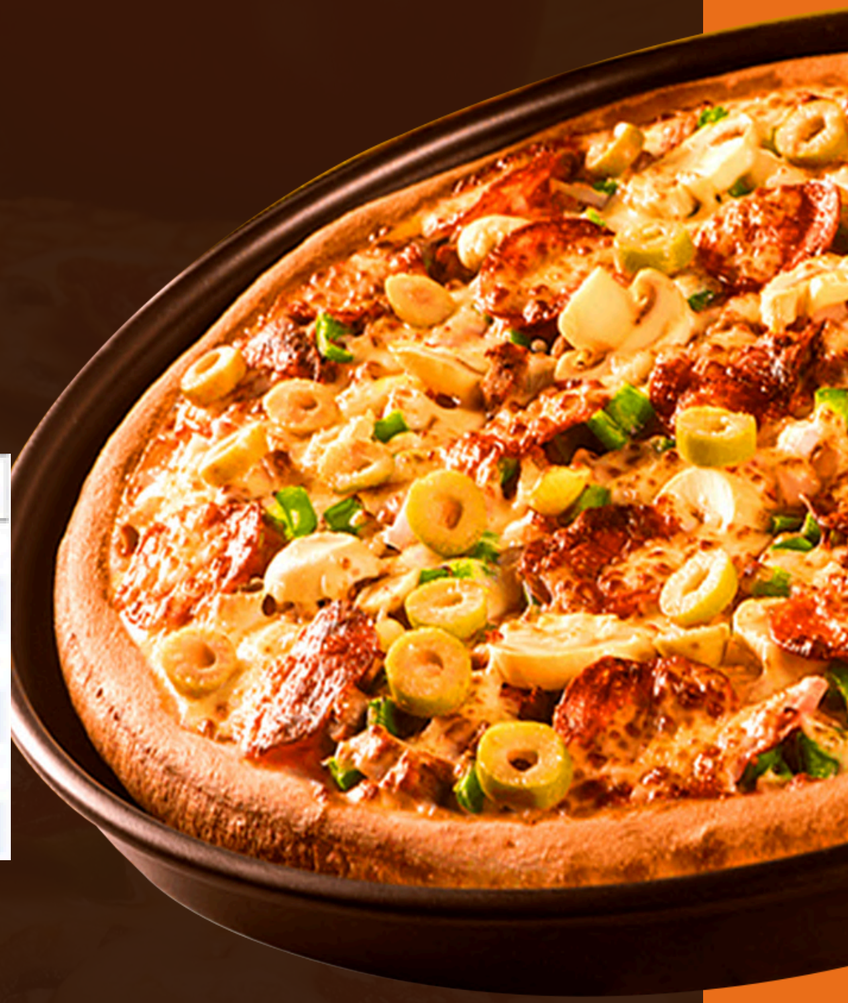
pizza_name	total_quantity_sold
The Brie Carre Pizza	490
The Mediterranean Pizza	934
The Calabrese Pizza	937
The Spinach Supreme Pizza	950
The Soppressata Pizza	961

Conclusion : These pizzas consistently underperform in sales and may need promotion or removal.

create view to identitfy ingredients are most frequently used in the top-selling pizzas

```sql
CREATE VIEW top_pizza_ingredients_inventory AS
SELECT pt.ingredients,sum(od.quantity) as sold_qty from pizzas as p
join pizza_types as pt on pt.pizza_type_id=p.pizza_type_id
join orders_details as od on od.pizza_id=p.pizza_id
Group by  pt.ingredients order by sold_qty desc limit 5;
select * from top_pizza_ingredients_inventory;
```

ingredients	sold_qty
Pepperoni, Mushrooms, Red Onions, Red Peppers, Bacon	2453
Barbecued Chicken, Red Peppers, Green Peppers, Tomatoes, Red Onions, Barbecue Sauce	2432
Sliced Ham, Pineapple, Mozzarella Cheese	2422
Mozzarella Cheese, Pepperoni	2418
Chicken, Pineapple, Tomatoes, Red Peppers, Thai Sweet Chilli Sauce	2371
Chicken, Artichoke, Spinach, Garlic, Jalapeno Peppers, Fontina Cheese, Gouda Cheese	2370



Conclusion :this insight inform inventory purchasing and stocking strategies

create view that find name of pizza which is high price top 5

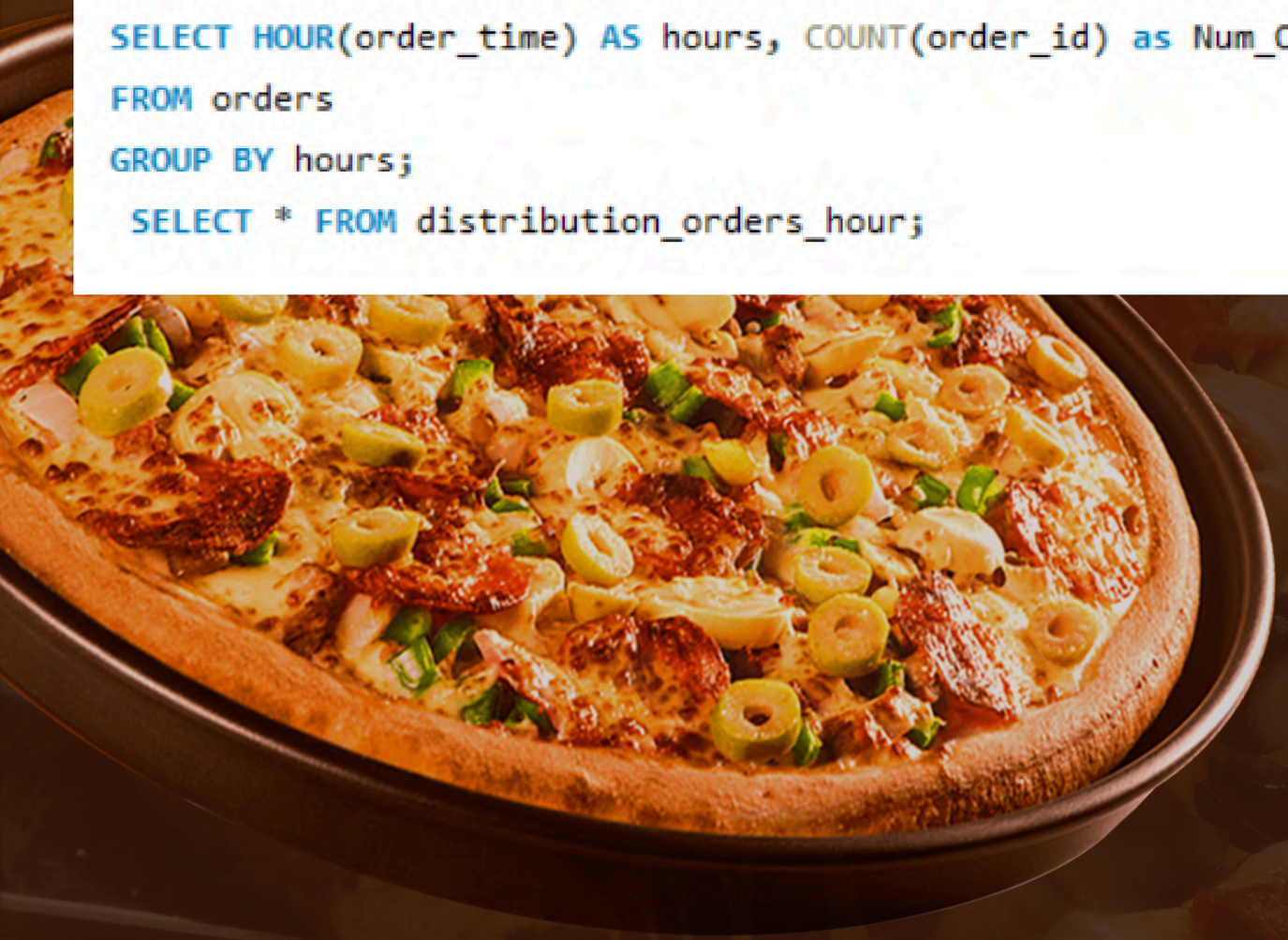
```
CREATE VIEW HIGH_PRICE_PIZZA AS
SELECT pt.name , p.price
FROM pizzas as p
join pizza_types as pt on pt.pizza_type_id=p.pizza_type_id
ORDER by p.price desc limit 5;
select * from high_price_pizza;
```

name	price
The Greek Pizza	35.95
The Greek Pizza	25.5
The Brie Carre Pizza	23.65
The Italian Vegetables Pizza	21
The Barbecue Chicken Pizza	20.75

Conclusion :view provides insights into the five most expensive pizzas, helping businesses focus on premium products with higher price points.

create view that display distribution of orders by hour of the day

```sql
CREATE VIEW distribution_orders_hour AS
SELECT HOUR(order_time) AS hours, COUNT(order_id) as Num_Orders
FROM orders
GROUP BY hours;
 SELECT * FROM distribution_orders_hour;
```



hours	Num_Orders
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

Conclusion :"More staffing is required between 12 PM to 1 PM (lunch rush) and 5 PM to 7 PM (dinner rush), as these hours show the highest customer demand."

Final Conclusion

- Sales analysis revealed consistent daily demand, enabling better forecasting of order volume.
- Most ordered pizza types and sizes helped identify customer preferences and top-selling products.
- High-revenue pizza categories contributed significantly to total earnings, guiding product focus.
- Time-based order trends identified key business hours for efficient staffing and operations planning.
- Ingredient patterns in top-selling pizzas offered actionable insights for smarter inventory management.
- SQL views enabled dynamic, reusable reports for sales, revenue, and ingredient tracking.
- Aligning inventory with popular ingredients helps reduce waste and prevent stockouts.
- Overall, data-driven insights from this project support better business decisions and improved customer satisfaction

Pizza Resto Analytics

THANK YOU

"Behind every slice of pizza sold, there’s a story told by data."