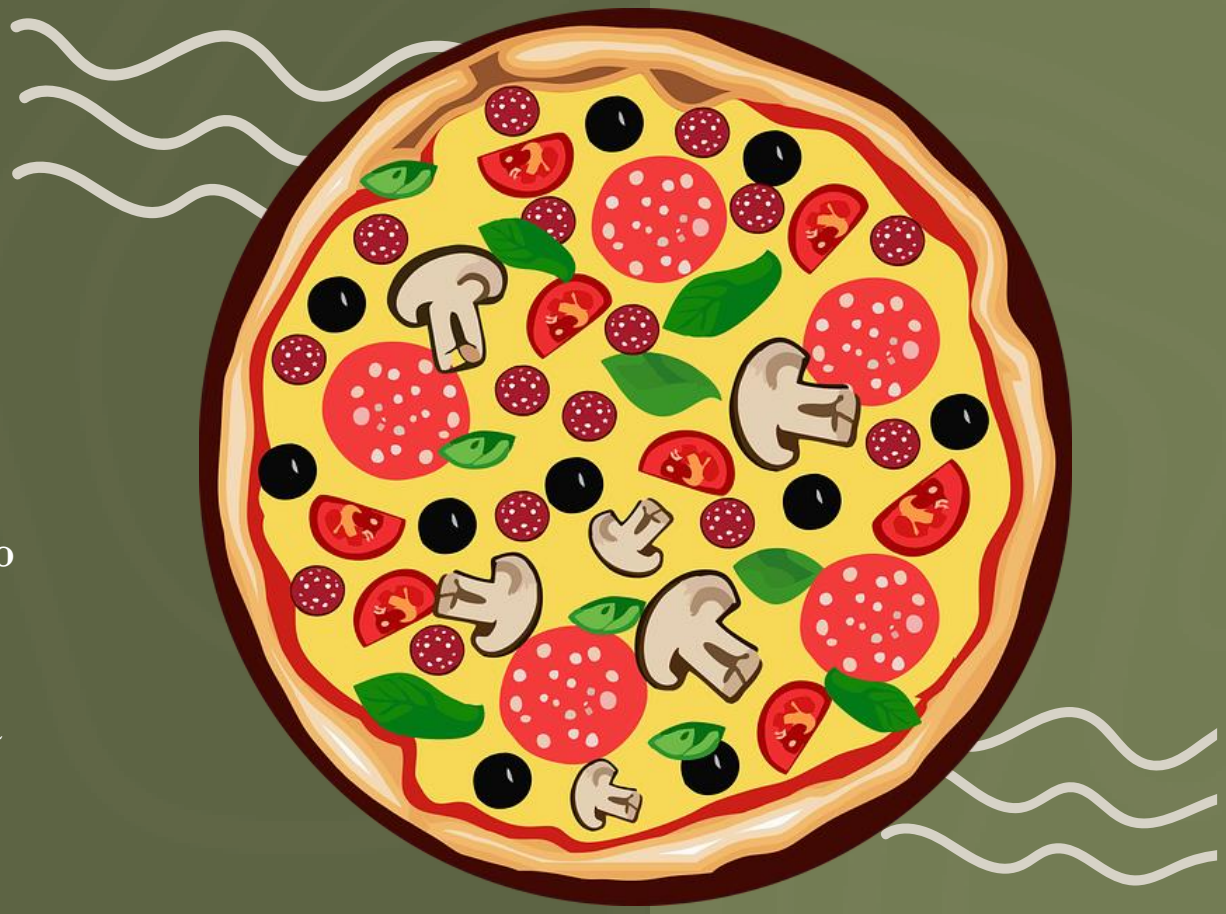
PROJECT ON PIZZA SALES USING SQL

WELCOME TO PIZZASALE PROJECT MANAGEMENT

Hello!

I am Srividya, I have done pizzasales project by using SQL. Developed a comprehensive SQL database to manage pizza sales, including handling queries related to pizzasales.
Happy to share that I have done this project to enhance My sql skills and database management with driven by a strong intrest in data science and a commitment to leveraging data for business insights.



Q. Pizza sale data set with count of rows:30,000 and with different column se

- Category
- Pizza_id
- Order
- Price
- Date
- Name
- time

pizza_type_id	name	category	order_id	date	time	order_detai	order_id	pizza_id	quantity
bbq_ckn	The Barbecue Chicken Pizza	Chicken	1	01-01-2015	11:38:36	1	1	hawaiian_m	1
cali_ckn	The California Chicken Pizza	Chicken	2	01-01-2015	11:57:40	2	2	classic_dlx_m	1
ckn_alfredo	The Chicken Alfredo Pizza	Chicken	3	01-01-2015	12:12:28	3	2	five_cheese_l	1
ckn_pesto	The Chicken Pesto Pizza	Chicken	4	01-01-2015	12:16:31	4	2	ital_supr_l	1
southw_ckn	The Southwest Chicken Pizza	Chicken	5	01-01-2015	12:21:30	5	2	mexicana_m	1
thai_ckn	The Thai Chicken Pizza	Chicken	6	01-01-2015	12:29:36	6	2	thai_ckn_l	1
big_meat	The Big Meat Pizza	Classic	7	01-01-2015	12:50:37	7	3	ital_supr_m	1
classic_dlx	The Classic Deluxe Pizza	Classic	8	01-01-2015	12:51:37	8	3	prsc_argla_l	1
hawaiian	The Hawaiian Pizza	Classic	9	01-01-2015	12:52:01	9	4	ital_supr_m	1
ital_cpcllo	The Italian Capocollo Pizza	Classic	10	01-01-2015	13:00:15	10	5	ital_supr_m	1
napolitana	The Napolitana Pizza	Classic	11	01-01-2015	13:02:59	11	6	bbq_ckn_s	1
pep_msh_pep	The Pepperoni, Mushroom, and Pepper	Classic	12	01-01-2015	13:04:41	12	6	the_greek_s	1
pepperoni	The Pepperoni Pizza	Classic	13	01-01-2015	13:11:55	13	7	spinach_supr_s	1
the_greek	The Greek Pizza	Classic	14	01-01-2015	13:14:19	14	8	spinach_supr_s	1
brie_carre	The Brie Carre Pizza	Supreme	15	01-01-2015	13:33:00	15	9	classic_dlx_s	1
calabrese	The Calabrese Pizza	Supreme	16	01-01-2015	13:34:07	16	9	green_garden_s	1
ital_supr	The Italian Supreme Pizza	Supreme	17	01-01-2015	13:53:00	17	9	ital_cpcllo_l	1
peppr_salami	The Pepper Salami Pizza	Supreme	18	01-01-2015	13:57:08	18	9	ital_supr_l	1
prsc_argla	The Prosciutto and Arugula Pizza	Supreme	19	01-01-2015	13:59:09	19	9	ital_supr_s	1
sicilian	The Sicilian Pizza	Supreme	20	01-01-2015	14:03:08	20	9	mexicana_s	1
soppressata	The Soppressata Pizza	Supreme	21	01-01-2015	14:14:29	21	9	spicy_ital_l	1
spicy_ital	The Spicy Italian Pizza	Supreme	22	01-01-2015	14:16:26	22	9	spin_pesto_l	1
spinach_supr	The Spinach Supreme Pizza	Supreme	23	01-01-2015	14:19:03	23	9	veggie_veg_s	1
five_cheese	The Five Cheese Pizza	Veggie	24	01-01-2015	14:23:01	24	10	mexicana_l	1
four_cheese	The Four Cheese Pizza	Veggie	25	01-01-2015	14:44:44	25	10	southw_ckn_l	1
#NAME?	The Five Cheese Pizza	Veggie	26	01-01-2015	14:54:26	26	11	bbq_ckn_l	1
four_cheese	The Four Cheese Pizza	Veggie	27	01-01-2015	15:11:17	27	11	cali_ckn_l	1
green_garden	The Green Garden Pizza	Veggie	28	01-01-2015	15:35:46	28	11	cali_ckn_m	1
ital_veggie	The Italian Vegetables Pizza	Veggie	29	01-01-2015	15:41:01	29	11	pepperoni_l	1
mediterraneo	The Mediterranean Pizza	Veggie	30	01-01-2015	15:41:25	30	12	cali_ckn_l	1
mexicana	The Mexicana Pizza	Veggie	31	01-01-2015	15:50:18	31	12	cali_ckn_s	1
spin_pesto	The Spinach Pesto Pizza	Veggie	32	01-01-2015	15:53:18	32	12	ckn_pesto_l	1

pizzas

1.Retrieve the total number of orders placed.

```
# Retrieve the total number of orders placed

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

Result Grid

	total_orders
▶	21350

2.Calculate the total revenue generated from pizza sales.

```
#calculate total revenue generated from pizza sales.
SELECT
    ROUND(SUM(d.quantity * pizzas.price), 2) AS total_sales
FROM
    order_details d
        JOIN
    pizzas ON pizzas.pizza_id = d.pizza_id;
```

Result Grid

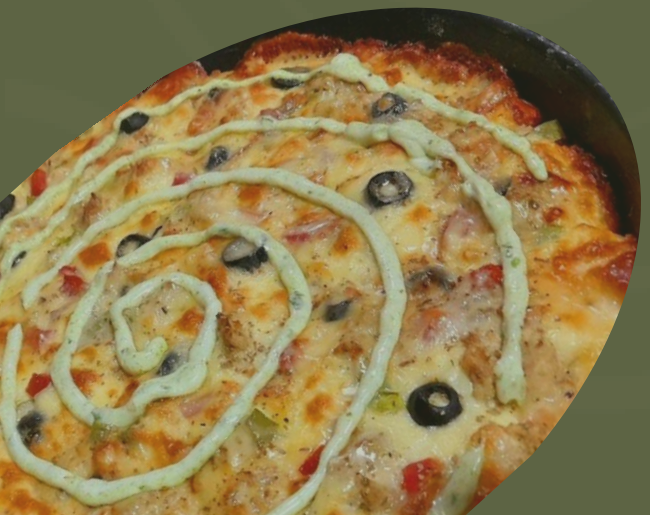
	total_sales
▶	817860.05

3. Identify the highest-priced pizza.

```
#identify the highest-priced pizza.
select p.name,s.price from pizza_details p
join pizzas s on s.pizza_type_id=p.pizza_type_id
order by s.price desc limit 1;
```

Result Grid | Filter Ro

	name	price
▶	The Greek Pizza	35.95



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

```
#List the top 5 most  pizz types with the quantities
SELECT
    s.name, SUM(r.quantity) AS quantity
FROM
    pizzas p
        JOIN
    order_details r ON r.pizza_id = p.pizza_id
        JOIN
    pizza_details s ON s.pizza_type_id = p.pizza_type_id
GROUP BY s.name
ORDER BY quantity DESC
LIMIT 5;
```

Result Grid Filter Rows:

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

5.Join the necessary tables to find the total quantity of each pizza category ordered.

```
#Join the necessary tables to find the total quantity of each pizza category ordered.
SELECT
    p.category, SUM(r.quantity) AS quantity
FROM
    pizza_details p
        JOIN
    pizzas s ON p.pizza_type_id = s.pizza_type_id
        JOIN
    order_details r ON r.pizza_id = s.pizza_id
GROUP BY p.category
ORDER BY quantity DESC;
```

Result Grid | Filter Ro

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

6.Determine the distribution of orders by hour of the day.

```sql
-- Determine the distribution of orders by hour of the day.
SELECT
    HOUR(order_time) AS hour, COUNT(order_id) AS order_count
FROM
    orders r
GROUP BY HOUR(order_time);
```

Result Grid | Filter R

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

7.Join relevant tables to find the category-wise distribution of pizzas.

```sql
-- Join relevant tables to find the category-wise distribution of pizzas.
SELECT
    category, COUNT(name)
FROM
    pizza_details
GROUP BY category;
```

Result Grid | Filter Rows:

category	COUNT(name)
Chicken	6
Classic	8
Supreme	9
Veggie	9

8.Group the orders by date and calculate the average number of pizzas ordered per day.

```sql
-- Group the orders by date and calculate the average number of pizzas ordered per day.
SELECT
    ROUND(AVG(quantity), 0) as avg_order
FROM
    (SELECT
        r.order_date, SUM(p.quantity) AS quantity
    FROM
        orders r
    JOIN order_details p ON p.order_id = r.order_id
    GROUP BY r.order_date) AS order_quantity;
```

Result Grid

	avg_order
▶	138

9.Determine the top 3 most ordered pizza types based on revenue.

```sql
-- Determine the top 3 most ordered pizza types based on revenue.
select p.name,sum(r.quantity* s.price) as revenue
from pizza_details p
join pizzas s on s.pizza_type_id= p.pizza_type_id
join order_details r on r.pizza_id=s.pizza_id
group by p.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

41409.5

10.Calculate the percentage contribution of each pizza type to total revenue.

```
-- Calculate the percentage contribution of each pizza type to total revenue.
select p.category,round(sum(r.quantity*s.price)/ (select
   round(sum(r.quantity * s.price),2)
     as total_sales
from
     order_details r
join pizzas s on s.pizza_id=r.pizza_id)*100,2) as revenue
from pizza_details p
join pizzas s on p.pizza_type_id=s.pizza_type_id
join order_details r on r.pizza_id=s.pizza_id
group by p.category
order by revenue desc ;
```

Result Grid | Filter R

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

```sql
-- Analyze the cumulative revenue generated over time
select order_date, sum(revenue) over (order by order_date) as cum_revenue
from
(select d.order_date,
sum(r.quantity*s.price) as revenue
from order_details r
join pizzas s on r.pizza_id=s.pizza_id
join orders d on d.order_id=r.order_id
group by d.order_date) as sales;
```

Result Grid | Filter Rows:

order_date	cum_revenue
2015-01-01	2713.850000000004
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
2015-01-20	45763.65000000001
2015-01-21	47804.20000000001

11.Analyze the cumulative revenue generated over time.

THANK YOU FOR YOUR ATTENTION

✓ If it useful share and like!