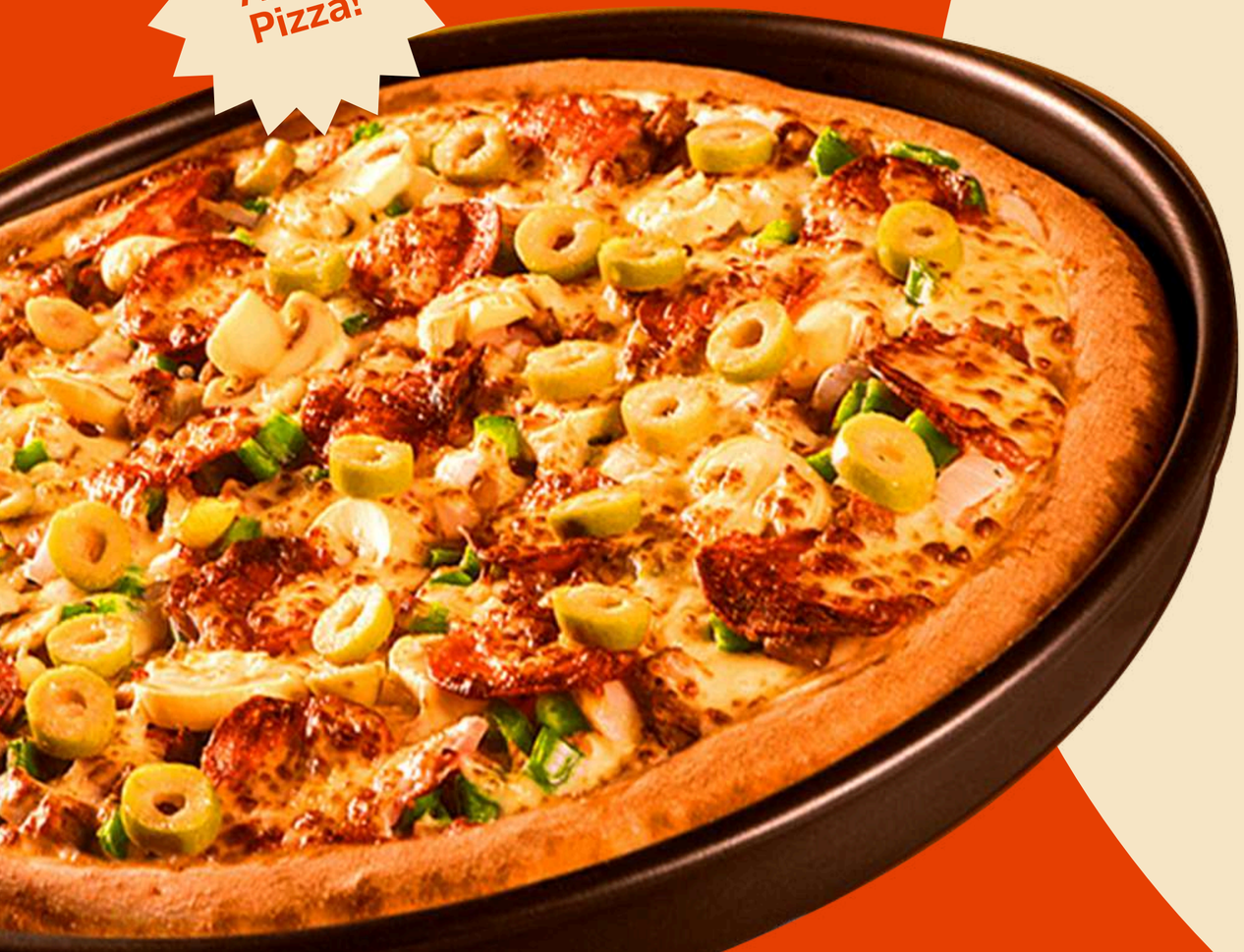
Liceria & Co.

Home About Us Content Contact Us

All About Pizza!



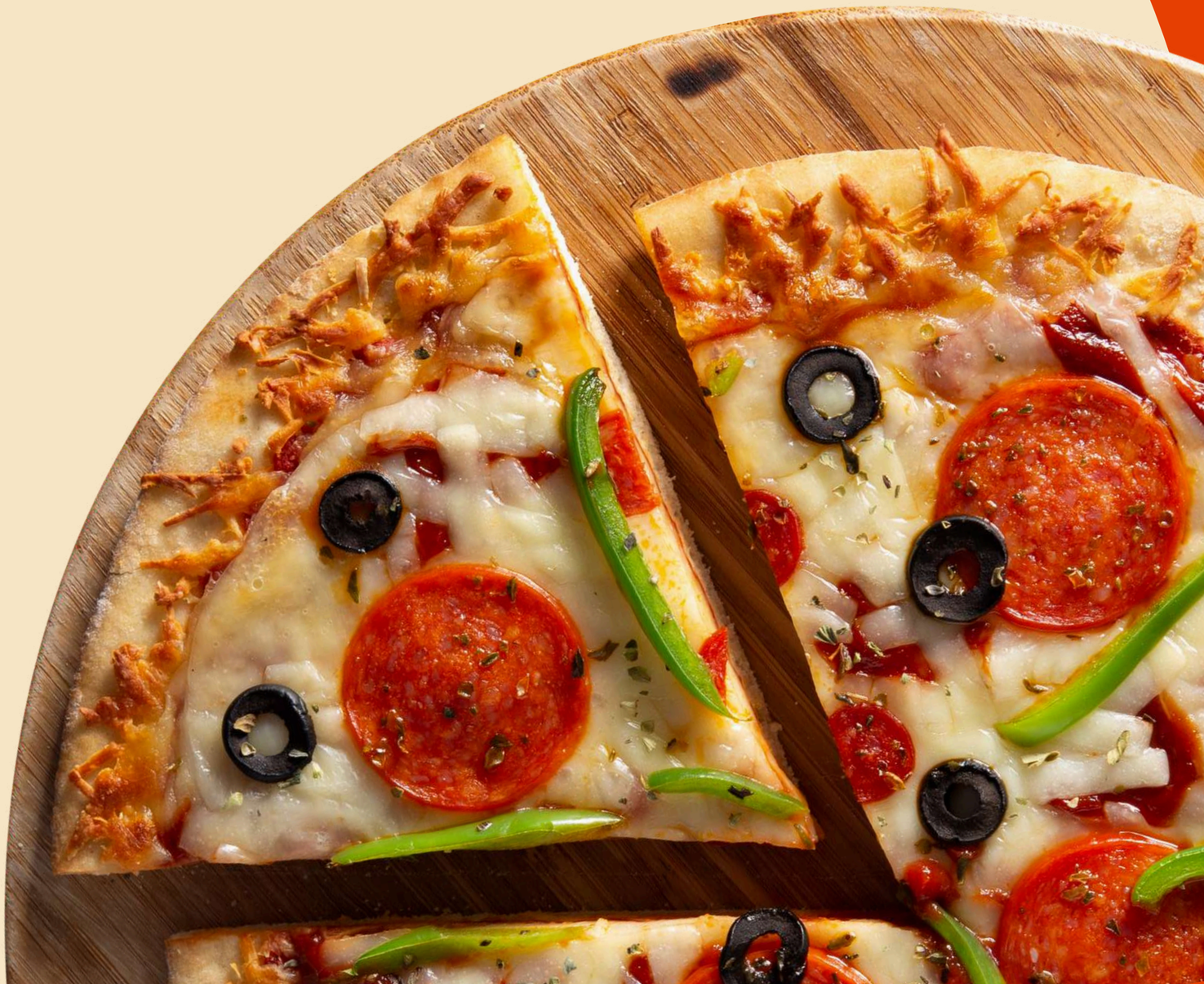
PIZZA SALES SQL PROJECT

Analyzing pizza sales data to extract insights on revenue, orders, and popular pizzas.

```
Basic:
Retrieve the total number of orders placed.
Calculate the total revenue generated from pizza sales.
Identify the highest-priced pizza.
Identify the most common pizza size ordered.
List the top 5 most ordered pizza types along with their quantities.


Intermediate:
Join the necessary tables to find the total quantity of each pizza category ordered.
Determine the distribution of orders by hour of the day.
Join relevant tables to find the category-wise distribution of pizzas.
Group the orders by date and calculate the average number of pizzas ordered per day.
Determine the top 3 most ordered pizza types based on revenue.

Advanced:
Calculate the percentage contribution of each pizza type to total revenue.
Analyze the cumulative revenue generated over time.
Determine the top 3 most ordered pizza types based on revenue for each pizza category.
```

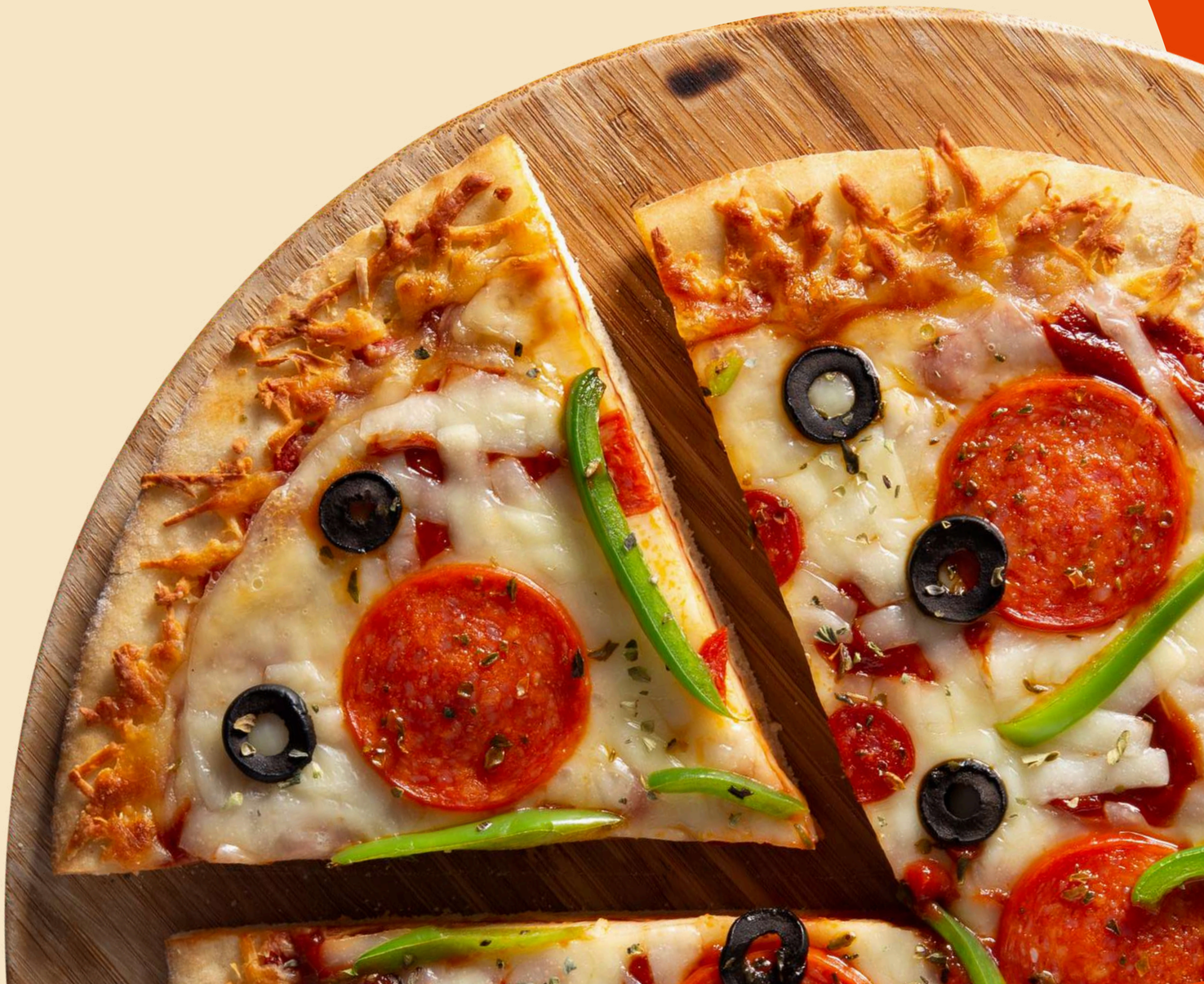


Query 1 | SQL File 1* | SQL File 2* | SQL File 3* | SQL File 4*

Limit to 1000 rows

```sql
create database pizzahut;
use pizzahut;
create table orders (
order_id int not null,
order_date date not null,
order_time time not null,
primary key(order_id));

create table order_details (
order_details_id int not null,
order_id int not null,
pizza_id text not null,
quantity int not null,
primary key(order_details_id));
```



```sql
-- Calculate the total revenue generated from pizza sales.
SELECT
    round(sum(order_details.quantity * pizzas.price),--  selecting quant and price for cal
            2) AS total_sales
FROM
    order_details
        JOIN
    pizzas
    ON pizzas.pizza_id = order_details.pizza_id --  common between 2 tables
```

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

total_sales
817860.05

Server Tools Scripting Help

Query 1 | SQL File 1* | SQL File 2* | SQL File 3* | SQL File 4* | SQL File 5

```sql
-- Retrieve the total number of orders placed.
select count(order_id) as total_orders from  orders;
```

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

total_orders
21350

SQL File 1* SQL File 2* SQL File 3* SQL File 4* SQL File 5* SQL File 6* SQL File 7* SQL File 8* SQL File 9* SQL Fi

Limit to 1000 rows

```sql
-- Identify the highest-priced pizza.
select pizza_types.name,pizzas.price -- selection of two tables name and price
from pizza_types join pizzas
on pizza_types.pizza_type_id=pizzas.pizza_type_id -- common part in both table is pizza id
order by pizzas.price desc limit 1;-- limit 1 → After sorting, LIMIT 1 tells SQL to return on
```

Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

me	price
Greek Pizza	35.95

hut)

abase Server Tools Scripting Help

Query 1 SQL File 1* SQL File 2* SQL File 3* SQL File 4* SQL File 5* SQL File 6* SQL File 7* S

Limit to 1000 rows

```sql
-- Identify the most common pizza size ordered.
SELECT
    pizzas.size,
    COUNT(order_details.order_details_id) AS order_count
FROM
    order_details
        JOIN
    pizzas ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizzas.size -- group bu use when we reqq to group catagories
ORDER BY order_count DESC;
```

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

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

```sql
-- 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
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

```sql
-- 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
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

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

hours	order_count
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336
18	2399
19	2009
20	1642

```sql
-- Join relevant tables to find
-- the 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

```sql
-- 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)
138

```sql
-- Determine the top 3 most ordered pizza types based on revenue.
select pizza_types.name,
sum(order_details.quantity * pizzas.price) as revanue
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 revanue desc limit 3;
```

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

```sql
-- Calculate the percentage contribution of each pizza type to total revenue.
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 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;
```

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

```sql
-- Analyze the cumulative revenue generated over time.
-- Show revenue growing day by day
select order_date,
        -- Add today's revenue + all previous days = running total
        sum(revanue) over (order by order_date) as cum_revanue
from (
        -- Find money made each day
        select orders.order_date,
               sum(order_details.quantity * pizzas.price) as revanue
        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;
```

order_date	cum_revanue
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

```sql
-- Determine the top 3 most ordered pizza types based on revenue

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;
```

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
The Pepperoni Pizza	30161.75
The Spicy Italian Pizza	34831.25
The Italian Supreme Pizza	33476.75

THANK YOU