



PizzaHut Sales Report

using MySQL Workbench

Presented by Somnath Rana






PROJECT OVERVIEW

This project is combination of different queries along with their solutions and outputs from MySql workbench





WHAT I DID

- Imported Sales data as CSV files into MySql workbench.
 - Cleaned and verified data related to Pizzas and Order Details.
 - Run several queries to understand data using SQL.
- 



1) Retrieve the total number of orders placed.

Query:

```
SELECT  
    COUNT(order_id) AS total_orders  
FROM  
    orders;
```

Output:

Result Grid		Filter Rows:	
	total_orders		
▶	21350		





2) Calculate the total revenue generated from pizza sales.

Query:

```
with x as (select p.pizza_id , p.price , o.quantity ,  
              p.price * o.quantity as amount  from pizzas p  
            join order_details o  
            using(pizza_id) )  
  
select round(sum(amount) ,2)  as total_revenue from x ;
```

Output:

Result Grid		Filter Rows:
	total_revenue	
▶	817860.05	





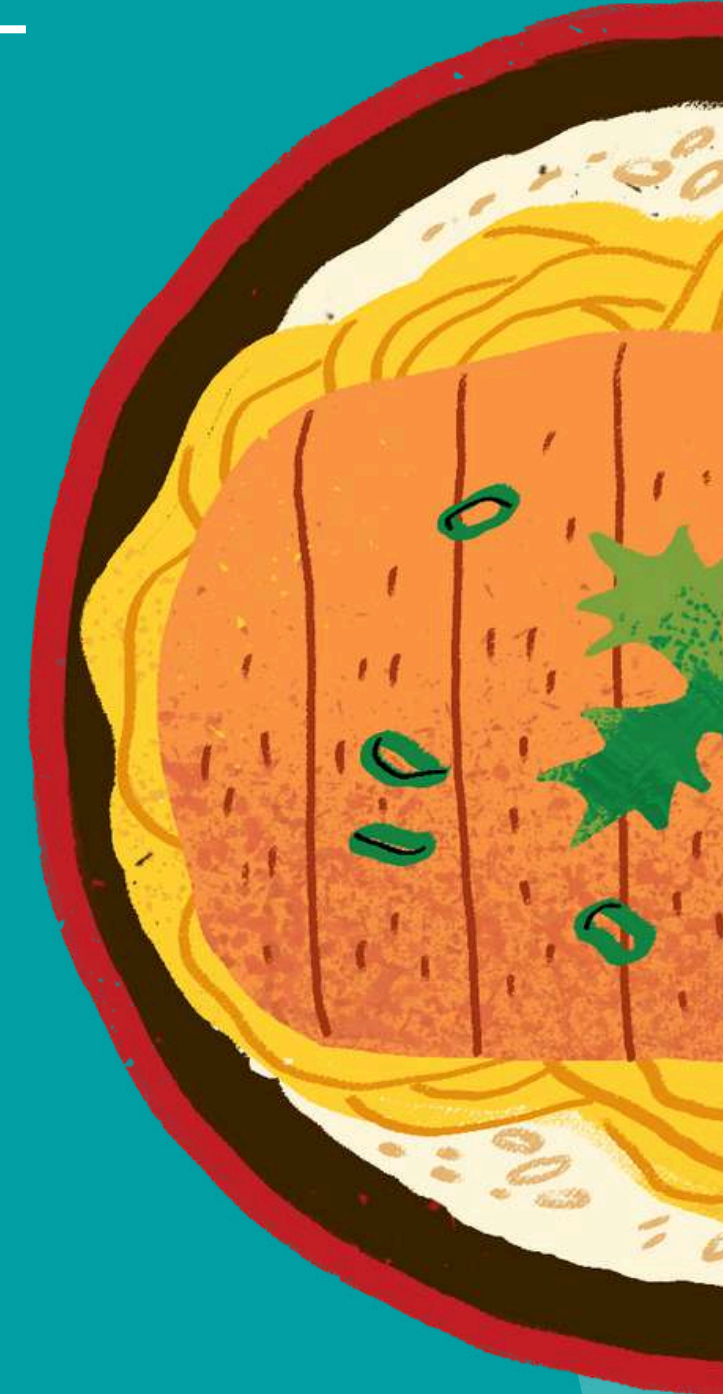
3) Identify the highest-priced pizza.

Query:

```
SELECT
    t.name, p.price
FROM pizzas p
JOIN pizza_types t USING (pizza_type_id)
WHERE p.price = (SELECT MAX(price) FROM pizzas);
```

Output:

Result Grid			Filter Rows:
	name	price	
▶	The Greek Pizza	35.95	





4) Identify the most common pizza size ordered.

Query:

```
SELECT
    size, SUM(quantity) AS count
FROM pizzas p
JOIN order_details o USING (pizza_id)
GROUP BY size
ORDER BY count DESC LIMIT 1;
```

Output:

Result Grid			Filter Rows:
	size	count	
▶	L	18956	





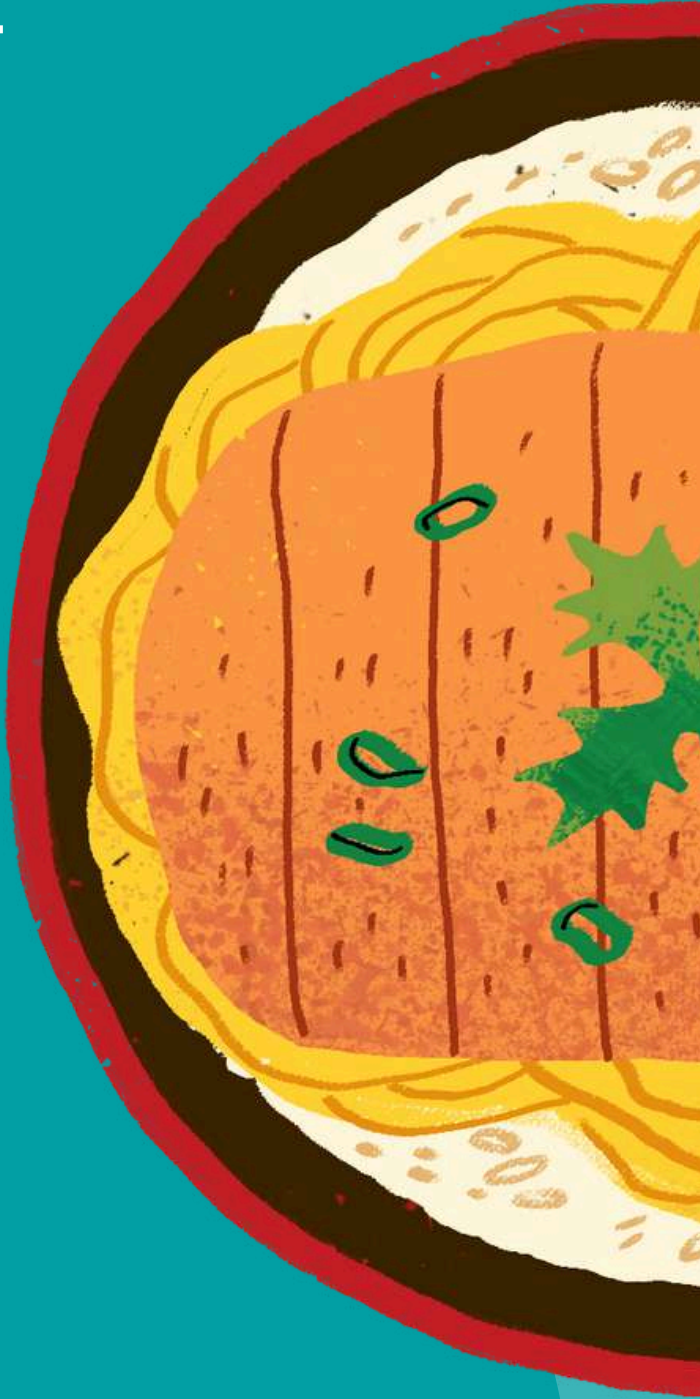
5) List the top 5 most ordered pizza types along with their quantities.

Query:

```
SELECT p.name AS name, SUM(o.quantity) AS total_quantity FROM pizza_types p
JOIN pizzas a
  USING (pizza_type_id)
JOIN order_details o
  USING (pizza_id)
GROUP BY name
ORDER BY total_quantity DESC LIMIT 5;
```

Output:

	name	total_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 the necessary tables to find the total quantity of each pizza category ordered.

Query:

```
SELECT p.category, SUM(d.quantity) AS total_quantity
FROM pizza_types p
JOIN pizzas s
ON p.pizza_type_id = s.pizza_type_id
JOIN order_details d
ON d.pizza_id = s.pizza_id
GROUP BY p.category
ORDER BY total_quantity DESC;
```

Output:

	category	total_quantity
►	Classic	14888
	Supreme	11987
	Veggie	11649
	Chicken	11050





7) Determine the distribution of orders by hour of the day.

Query:

```
SELECT
    HOUR(time) AS hour, COUNT(order_id)
FROM
    orders
GROUP BY HOUR(time)
ORDER BY COUNT(order_id) DESC;
```

Output:

	hour	COUNT(order_id)
▶	12	2520
	13	2455
	18	2399
	17	2336
	19	2009
	16	1920
	20	1642
	14	1472
	15	1468
	11	1231
	21	1198





8) Find the category-wise distribution of pizzas.

Query:

```
SELECT
    category, COUNT(pizza_type_id) AS types
FROM
    pizza_types
GROUP BY category;
```

Output:

	category	types
▶	Chicken	6
	Classic	8
	Supreme	9
	Veggie	9





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

Query:

```
with x as (SELECT
    DATE(o.date) AS date, sum(d.quantity) AS quantity
FROM
    order_details d
    JOIN
    orders o USING (order_id)
GROUP BY o.date
ORDER BY quantity DESC)

select round(avg(quantity) , 0) as avg_quantity from x ;
```

Output:

Result Grid		Filter Rows:
	avg_quantity	
▶	138	





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

Query:

```
SELECT
    p.name,
    SUM(o.quantity * s.price) AS total_revenue
FROM pizza_types p
JOIN pizzas s
    USING (pizza_type_id)
JOIN order_details o
    USING (pizza_id)
GROUP BY p.name
ORDER BY total_revenue DESC
limit 3 ;
```

Output:

	name	total_revenue
►	The Thai Chicken Pizza	43434.25
	The Barbecue Chicken Pizza	42768
	The California Chicken Pizza	41409.5





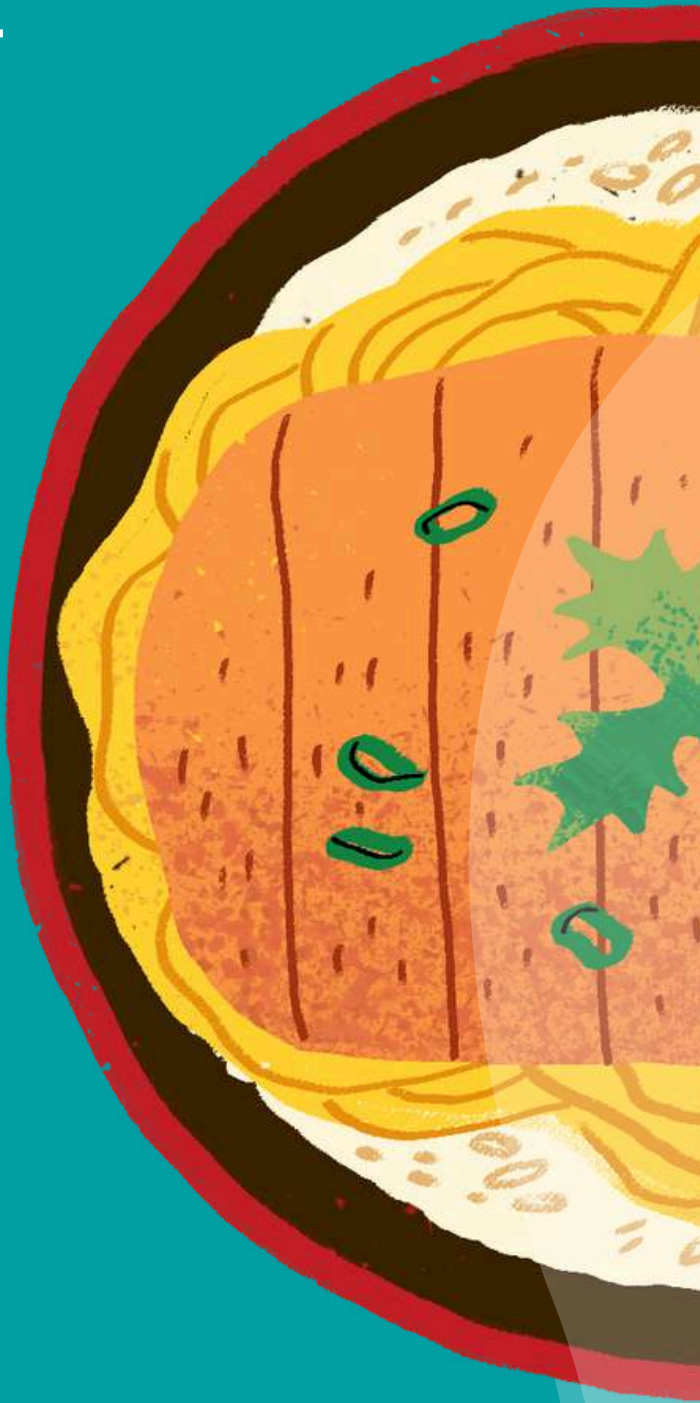
11) Calculate the percentage contribution of each pizza type to total revenue.

Query:

```
SELECT
    p.category,
    ROUND(100.0 * SUM(o.quantity * s.price) / (SELECT SUM(o2.quantity * s2.price)
FROM pizzas s2
JOIN
    order_details o2 USING (pizza_id)), 2) AS pct_contribution
FROM pizza_types p
JOIN pizzas s
    USING (pizza_type_id)
JOIN order_details o
    USING (pizza_id)
GROUP BY p.category;
```

Output:

	category	pct_contribution
▶	Classic	26.91
	Veggie	23.68
	Supreme	25.46
	Chicken	23.96





12) Analyze the cumulative revenue generated over time.

Query:

```
with x as (select date(o.date) as date ,
round(sum(p.price * d.quantity) , 2) as revenue
from orders o
join order_details d
using(order_id)
join pizzas p
using(pizza_id)
group by date)

select date, sum(revenue) over(order by date)
as cummilative_revenue from x ;
```

Output:

	date	cummilative_revenue
▶	2015-01-01	2713.85
	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	21526.399999999998
	2015-01-10	23990.35



13) Determine the top 3 most ordered pizza types based on revenue for each pizza category.

Query:

```
with x as (WITH pizza_revenue AS (SELECT t.name,  
t.category, ROUND(SUM(p.price * o.quantity), 2) AS revenue  
FROM pizza_types t  
JOIN pizzas p  
  USING (pizza_type_id)  
JOIN order_details o  
  USING (pizza_id)  
GROUP BY t.name, t.category  
)  
SELECT name , category,  
RANK() OVER (PARTITION BY category ORDER BY revenue DESC) AS rankings FROM pizza_revenue  
ORDER BY category, rankings)  
  
select * from x  
where rankings <= 3 ;
```

Output:

	name	category	rankings
►	The Thai Chicken Pizza	Chicken	1
	The Barbecue Chicken Pizza	Chicken	2
	The California Chicken Pizza	Chicken	3
	The Classic Deluxe Pizza	Classic	1
	The Hawaiian Pizza	Classic	2
	The Pepperoni Pizza	Classic	3
	The Spicy Italian Pizza	Supreme	1
	The Italian Supreme Pizza	Supreme	2
	The Sicilian Pizza	Supreme	3
	The Four Cheese Pizza	Veggie	1
	The Mexicana Pizza	Veggie	2
	The Five Cheese Pizza	Veggie	3

Thanks for
your time!

