PIZZAHUT SQL PROJECT

QUERY & EXECUTE

PRESENTED BY

REENA KUMARI

THE PIZZAHUT SQL PROJECT INVOLVES CREATING AND MANAGING A DATABASE SYSTEM FOR PIZZA HUT'S OPERATIONS. THIS SYSTEM WOULD LIKELY INCLUDE TABLES FOR STORING INFORMATION SUCH AS CUSTOMER DETAILS, ORDERS, MENU ITEMS, PRICING, INVENTORY, AND DELIVERY INFORMATION. USING SQL QUERIES, THE PROJECT AIMS TO EFFICIENTLY RETRIEVE, UPDATE, AND MANAGE DATA TO SUPPORT PIZZA HUT'S DAY-TO-DAY OPERATIONS, INCLUDING ORDER PROCESSING, INVENTORY MANAGEMENT, AND CUSTOMER SERVICE.

-- Retrieve the total number of orders placed.

```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:

count(order_id)
21350

-- CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

Don't Limit

```sql
-- Calculate the total revenue generated from pizza sales.

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
```

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

total_sales
817860.05

-- IDENTIFY THE HIGHEST-PRICED PIZZA.

Don't Limit

```sql
-- Identify the highest-priced pizza.

SELECT
    pizza_types.name, pizzas.price
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY pizzas.price DESC
LIMIT 1;
```

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

name	price
The Greek Pizza	35.95

-- IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

```sql
-- Identify the most common pizza size ordered.

select quantity, count(order_details_id)
from order_details group by quantity;

SELECT
    pizzas.size,
    COUNT(order_details.order_details_id) AS order_count
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size
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

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

Don't Limit

```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;
```

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

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

Toggle wrapping of cell contents

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

Don't Limit

```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;
```

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

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

Result 1

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

```
-- Determine the distribution of orders by hour of the day.--

SELECT
    HOUR(order_time) AS hour, COUNT(order_id) AS order_count
FROM
    orders
GROUP BY HOUR(order_time);
```

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

hour	order_count
11	1231
12	2520
13	2455
14	1472
15	1468
16	1920
17	2336
18	2399
19	2009

Result 2

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

Don't Limit

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

SELECT
    category, COUNT(name)
FROM
    pizza_types
GROUP BY category;
```

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

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

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

Don't Limit

```sql
-- Group the orders by date and calculate the average number of pizzas ordered per day.


SELECT
    ROUND(AVG(quantity), 0) as avg_pizza_ordered_perday
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;

```

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

round(avg(quantity),0)
138

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

Don't Limit

```sql
-- 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 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 revenue desc limit 3;
```

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

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

-- CALCULATE THE PERCENTAGE CONTRIBUTION OF EACH PIZZA TYPE TO TOTAL REVENUE.

```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 pizzas.pizza_type_id = pizza_types.pizza_type_id
join order_details
on order_details.pizza_id = pizzas.pizza_id
group by pizza_types.category order by revenue desc;
```

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

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

Result 3

-- ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```
1   -- Analyze the cumulative revenue generated over time.
2
3   select order_date,
4   sum(revenue) over(order by order_date) as cum_revenue
5   from
6   (select orders.order_date,
7   sum(order_details.quantity * pizzas.price) as revenue
8   from order_details join pizzas
9   on order_details.pizza_id = pizzas.pizza_id
10  join orders
11  on orders.order_id = order_details.order_id
12  group by orders.order_date) as sales;
```

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

order_date	cum_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
2015-01-08	19399.05
2015-01-09	21526.4

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

Don't Limit

```sql
-- Determine the top 3 most ordered pizza types based on revenue for each pizza category.

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;

```

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

name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5
The Southwest Chicken Pizza	34705.75
The Chicken Alfredo Pizza	16900.25
The Chicken Pesto Pizza	16701.75
The Classic Deluxe Pizza	38180.5

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

Don't Limit

```sql
-- Determine the top 3 most ordered pizza types based on revenue for each pizza category.

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;

```

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

name	revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5
The Southwest Chicken Pizza	34705.75
The Chicken Alfredo Pizza	16900.25
The Chicken Pesto Pizza	16701.75
The Classic Deluxe Pizza	38180.5

FINAL CONCLUSION

IN CONCLUSION, THE PIZZAHUT SQL PROJECT AIMED TO DEVELOP AND IMPLEMENT A ROBUST DATABASE SYSTEM TO SUPPORT PIZZA HUT'S OPERATIONS EFFICIENTLY. BY CREATING A DATABASE SCHEMA ENCOMPASSING ESSENTIAL ASPECTS LIKE CUSTOMER DETAILS, ORDERS, MENU ITEMS, PRICING, AND INVENTORY, THE PROJECT FACILITATED SEAMLESS MANAGEMENT OF PIZZA HUT'S DAY-TO-DAY ACTIVITIES.

THROUGH THE UTILIZATION OF SQL QUERIES, THE PROJECT ENABLED THE RETRIEVAL, UPDATING, AND MANAGEMENT OF DATA, EMPOWERING PIZZA HUT TO STREAMLINE ORDER PROCESSING, INVENTORY CONTROL, AND CUSTOMER SERVICE. DESPITE FACING CHALLENGES ALONG THE WAY, SUCH AS DATABASE OPTIMIZATION AND DATA INTEGRITY ISSUES, THE PROJECT SUCCESSFULLY DELIVERED TANGIBLE OUTCOMES.

MOVING FORWARD, POTENTIAL ENHANCEMENTS COULD INCLUDE REFINING DATABASE PERFORMANCE, IMPLEMENTING ADVANCED ANALYTICS FOR BUSINESS INSIGHTS, AND INTEGRATING ADDITIONAL FEATURES TO ENHANCE CUSTOMER EXPERIENCE AND OPERATIONAL EFFICIENCY.

ULTIMATELY, THE PIZZAHUT SQL PROJECT UNDERSCORED THE SIGNIFICANCE OF EFFECTIVE DATABASE MANAGEMENT IN ENHANCING ORGANIZATIONAL PRODUCTIVITY AND SERVICE DELIVERY WITHIN THE RESTAURANT INDUSTRY.