PizzaHut Sales Analysis Using SQL

Overview:

This project conducts a comprehensive analysis of sales data from Pizza Hut to uncover valuable insights into customer behaviour, product popularity, and revenue trends. The analysis is performed using SQL queries on a relational database containing transactional data from Pizza Hut outlets.

Analysis:

The analysis covers the following aspects:

- **Sales Trends:** Identifying patterns and trends in sales over time.
- **Customer Segmentation:** Analyzing customer demographics and behavior to identify key segments.
- **Product Performance:** Evaluating the popularity and profitability of different products.
- **Category wise Analysis:** Examining sales trends and preferences across different categories.

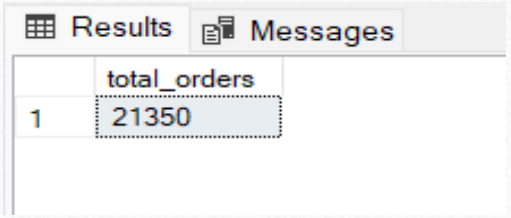
Create the database.
Create the tables by importing the files.

```
Create database PizzaHut;

--import files
select * from order_details;
select * from orders;
select * from pizza_types;
select * from pizzas;
```

Total Orders: Retrieve the total number of orders placed.

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



Total Revenue: Calculate the total revenue generated from pizza sales.

```
select
Round(Sum(order_details.quantity*pizzas.price),2) as Total_Revenue
from order_details
Join
pizzas on order_details.Pizza_id=Pizzas.pizza_id;
```

Results | Messages

	Total_Revenue
1	817860.05

Highest-Priced Pizza: Identify the highest-priced pizza.

```
Select top 1 pizza_types.name, pizzas.price
from pizza_types
Join
pizzas
on pizza_types.pizza_type_id = Pizzas.pizza_type_id
order by price desc;
```

Results Messages

	name	price
1	The Greek Pizza	35.9500007629395

Most Common Pizza Size: Identify the most common pizza size ordered.

```
select pizzas.size, count(order_details.quantity) 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;
```

Results | Messages

	size	order_count
1	L	18526
2	M	15385
3	S	14137
4	XL	544
5	XXL	28

Top 5 Most Ordered Pizza Types: List the top 5 most ordered pizza types along with their quantities.

```
select top 5 pizza_types.name, Sum(order_details.quantity) as Quantities
from pizza_types
Join pizzas on pizza_types.pizza_type_id = pizzas.pizza_type_id
Join order_details on pizzas.pizza_id = order_details.pizza_id
group by pizza_types.name
order by Quantities desc;
```

Results | Messages

	name	Quantities
1	The Classic Deluxe Pizza	2453
2	The Barbecue Chicken Pizza	2432
3	The Hawaiian Pizza	2422
4	The Pepperoni Pizza	2418
5	The Thai Chicken Pizza	2371

Pizza Category Quantity: 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 Pizzas.pizza_type_id = pizza_types.pizza_type_id
join order_details on pizzas.pizza_id = order_details.pizza_id
group by pizza_types.category
order by quantity desc;
```

Results | Messages

	category	quantity
1	Classic	14888
2	Supreme	11987
3	Veggie	11649
4	Chicken	11050

Order Distribution by Hour: Determine the distribution of orders by hour of the day.

```
select datepart(hour, order_time) as order_hour, count(order_id) as order_count
from orders
group by datepart(hour, order_time)
order by order_count;
```

Results | Messages

	order_hour	order_count
1	9	1
2	10	8
3	23	28
4	22	663
5	21	1198
6	11	1231
7	15	1468
8	14	1472
9	20	1642
10	16	1920
11	19	2009
12	17	2336
13	18	2399
14	13	2455
15	12	2520

Category-Wise Pizza Distribution: Join relevant tables to find the category-wise distribution of pizzas.

```
select * from pizza_types;

select category, count(name) as Pizzas_count
from pizza_types
group by category;
```

Results | Messages

	category	Pizzas_count
1	Chicken	6
2	Classic	8
3	Supreme	9
4	Veggie	9

Average Pizzas Ordered per Day: Group the orders by date and calculate the average number of pizzas ordered per day.

```
;with order_quantity as
(select orders.order_date, sum(order_details.quantity) as total_orders
from orders
Join order_details on orders.order_id = order_details.order_id
group by orders.order_date)
select avg(total_orders) as average_pizzas_orderered_per_day from order_quantity;
```

Results Messages

	average_pizzas_orderered_per_day
1	138

Top 3 Pizza Types by Revenue: Determine the top 3 most ordered pizza types based on revenue.

```
select * from pizzas
select * from pizza_types

select top 3 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.name
order by revenue;
```

Results Messages

	name	revenue
1	The Brie Carre Pizza	11588.4998130798
2	The Green Garden Pizza	13955.75
3	The Spinach Supreme Pizza	15277.75

Pizza Type Revenue Contribution: Calculate the percentage contribution of each pizza type to total revenue.

```
select * from pizzas
select * from pizza_types
select * from order_details

select pizza_types.category, Round(sum(order_details.quantity*pizzas.price)/(select sum(order_details.quantity*pizzas.price)
as total_sales from order_details
join Pizzas on order_details.pizza_id=pizzas.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;
```

Results | Messages

	category	revenue
1	Classic	26.91
2	Supreme	25.46
3	Chicken	23.96
4	Veggie	23.68

Cumulative Revenue Over Time: Analyze the cumulative revenue generated over time.

```sql
select * from orders;
select * from order_details;
select * from Pizzas;

;with total_sales_on_each_day as
(
select orders.order_date, sum(order_Details.quantity*pizzas.price) as total_sales
from orders
join order_details on orders.order_id = order_details.order_id
join pizzas on order_details.pizza_id = pizzas.pizza_id
group by orders.order_date
)
select order_date, sum(total_sales) over (order by order_date) as cumulative_revenue from total_sales_on_each_day;
```

Results | Messages

	order_date	total_sales
1	2015-02-14	2319.15000343323
2	2015-01-22	2496.70000076294
3	2015-12-20	2104.90000152588
4	2015-04-01	2176.85000228882
5	2015-03-09	2334.55000305176
6	2015-07-27	2172.00000190735
7	2015-08-02	1910.14999961853
8	2015-08-19	2332.95000267029
9	2015-08-25	1958.89999961853
10	2015-06-11	2650.5000038147
11	2015-11-27	4422.45000267029
12	2015-07-04	3864.20000076294
13	2015-11-04	1966.10000228882
14	2015-05-02	2400.20000267029
15	2015-01-16	2594.15000152588
16	2015-01-28	2016
17	2015-12-26	1643.05000495911
18	2015-06-17	2137.10000038147
19	2015-10-04	2142.20000267029
20	2015-11-10	2042.60000038147
21	2015-08-13	2074.15000343323
22	2015-06-23	2042.75
23	2015-09-28	2035.30000114441
24	2015-05-08	3052.30000686646
25	2015-11-13	2264.55000114441
26	2015-03-20	2461.2500038147
27	2015-12-06	2350.25000190735
28	2015-04-15	2578.85000419617
29	2015-06-03	1907.05000305176
30	2015-02-05	2215.80000114441

Top 3 Pizza Types by Revenue (Category-Wise): Determine the top 3 most ordered pizza types based on revenue for each pizza category.

```
;with revenue_by_Pizza_category as
(
select pizza_types.category, pizza_types.name, Sum(order_details.quantity*pizzas.price) as revenue,
rank() over (partition by category order by Sum(order_details.quantity*pizzas.price) desc) as rank
from pizza_types
join pizzas on pizza_types.pizza_type_id = pizzas.pizza_type_id
join order_details on pizzas.pizza_id = order_details.pizza_id
group by pizza_types.category, pizza_types.name
)
select category, revenue from revenue_by_Pizza_category where rank <= 3
```

Results | Messages

	category	revenue
1	Chicken	43434.25
2	Chicken	42768
3	Chicken	41409.5
4	Classic	38180.5
5	Classic	32273.25
6	Classic	30161.75
7	Supreme	34831.25
8	Supreme	33476.75
9	Supreme	30940.5
10	Veggie	32265.7010040283
11	Veggie	26780.75
12	Veggie	26066.5

Contributions:

Contributions to enhance the analysis or add new features are welcome. Please fork the repository, make your changes, and submit a pull request for review.

Acknowledgments:

- The data used in this project is sourced from Kaggle.

Thank you!!