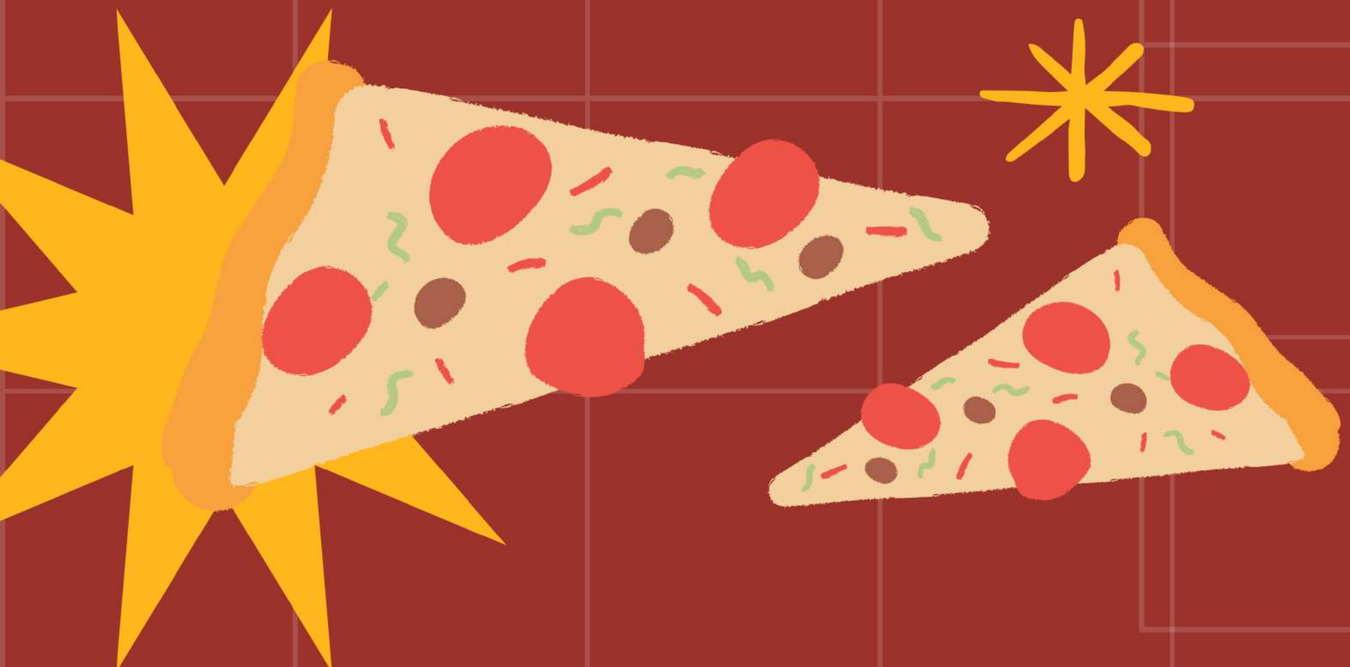
PIZZA SALES ANALYSIS USING SQL



IN THIS PROJECT, WE DIVE INTO THE WORLD OF PIZZA SALES USING SQL TO EXPLORE, ANALYZE, AND DERIVE INSIGHTS FROM A DATASET OF CUSTOMER ORDERS, PIZZA TYPES, AND SALES PERFORMANCE. THROUGH VARIOUS SQL QUERIES, WE'LL UNCOVER PATTERNS, IDENTIFY TOP-SELLING PIZZAS, AND PROVIDE DATA-DRIVEN RECOMMENDATIONS TO OPTIMIZE SALES STRATEGIES. THIS ANALYSIS NOT ONLY DEMONSTRATES THE POWER OF SQL FOR DATA MANIPULATION BUT ALSO HIGHLIGHTS ITS ROLE IN DECISION-MAKING FOR BUSINESS GROWTH.

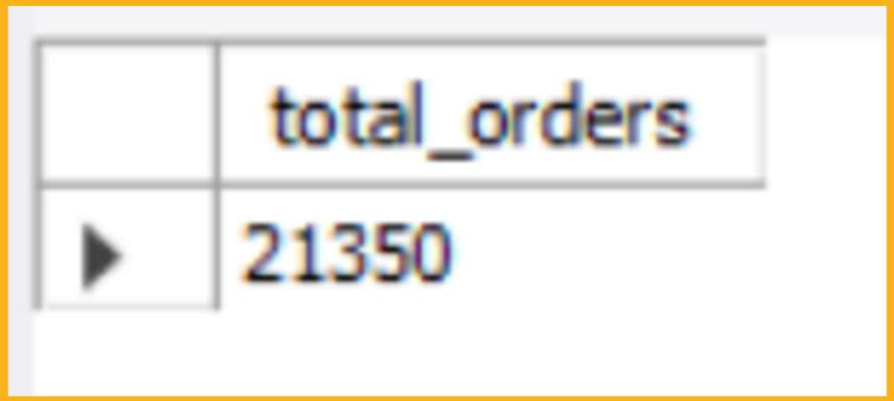
COMPREHENSIVE ANALYSIS OF PIZZA SALES USING SQL

IN THIS SECTION, I'VE TACKLED A RANGE OF QUESTIONS-SPANNING FROM BASIC TO ADVANCED-RELATED TO PIZZA SALES. BASIC QUERIES FOCUSED ON FUNDAMENTAL DATA RETRIEVAL, SUCH AS TOTAL SALES AND CUSTOMER INFORMATION. INTERMEDIATE QUERIES DELVED DEEPER INTO PERFORMANCE METRICS, INCLUDING MOST POPULAR PIZZA TYPES, PEAK ORDER TIMES, AND AVERAGE SALES PER ORDER. FINALLY, ADVANCED QUERIES PROVIDED COMPLEX INSIGHTS, SUCH AS IDENTIFYING SALES TRENDS OVER TIME AND GENERATING FORECASTS FOR FUTURE SALES. EACH QUESTION IS ACCOMPANIED BY THE CORRESPONDING SQL QUERY, SHOWCASING A STEP-BY-STEP APPROACH TO SOLVING REAL-WORLD BUSINESS PROBLEMS USING DATA.

RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

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

SQL QUERY



	total_orders
▶	21350

OUTPUT

CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

```sql
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
```

SQL QUERY

total_sales
817860.05

OUTPUT

IDENTIFY THE HIGHEST-PRICED PIZZA.

```sql
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;
```

SQL QUERY

	name	price
▶	The Greek Pizza	35.95

OUTPUT

IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

```sql
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;
```

SQL QUERY

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

OUTPUT

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

```sql
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;
```

SQL QUERY

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

OUTPUT

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

```sql
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;
```

SQL QUERY

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

OUTPUT

DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

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

SQL QUERY

	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

OUTPUT

JOIN RELEVANT TABLES TO FIND THE CATEGORY-WISE DISTRIBUTION OF PIZZAS.

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

SQL QUERY

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

OUTPUT

GROUP THE ORDERS BY DATE AND CALCULATE THE AVERAGE NUMBER OF PIZZAS ORDERED PER DAY.

```sql
SELECT
    ROUND(AVG(quantity), 0) AS avg_pizzas_ordered_per_day
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;
```

SQL QUERY

avg_pizzas_ordered_per_day
138

OUTPUT

DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE.

```sql
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;
```

SQL QUERY

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

OUTPUT

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

```sql
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;
```

SQL QUERY

category	revenue
Classic	26.91
Supreme	25.46
Chicken	23.96
Veggie	23.68

OUTPUT

ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

```sql
select order_date,
sum(revenue) over(order by order_date) as cum_revenue
from
(select orders.order_date,
sum(order_details.quantity*pizzas.price) as revenue
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;
```

SQL QUERY

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
2015-01-10	23990.350000000002

OUTPUT

DETERMINE THE TOP 3 MOST ORDERED PIZZA TYPES BASED ON REVENUE FOR EACH PIZZA CATEGORY.

```sql
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;
```

SQL QUERY

name	revenue
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
The Sicilian Pizza	30940.5
The Four Cheese Pizza	32265.700000006
The Mexicana Pizza	26780.75
The Five Cheese Pizza	26066.5

OUTPUT

CONCLUSION

IN THIS PROJECT, I DEMONSTRATED HOW SQL CAN BE USED TO EFFECTIVELY ANALYZE PIZZA SALES DATA. BY ADDRESSING BASIC, INTERMEDIATE, AND ADVANCED QUERIES, I WAS ABLE TO UNCOVER KEY INSIGHTS SUCH AS CUSTOMER PREFERENCES, TOP-SELLING PIZZAS, AND TRENDS IN SALES PERFORMANCE. THROUGH THE SQL QUERIES I CREATED, I TURNED RAW DATA INTO MEANINGFUL INFORMATION THAT CAN DRIVE BETTER BUSINESS DECISIONS. THIS PROJECT NOT ONLY SHOWCASES THE CAPABILITIES OF SQL BUT ALSO HIGHLIGHTS HOW DATA ANALYSIS CAN CONTRIBUTE TO OPTIMIZING SALES STRATEGIES AND IMPROVING OVERALL BUSINESS OUTCOMES.

THANK YOU