PIZZA SALES REPORT

YEARLY REPORT

HELLO TEAM!

My name is Paras. In the upcoming slides, I will be retrieving and analyzing data as requested by the manager to derive meaningful insights.

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.

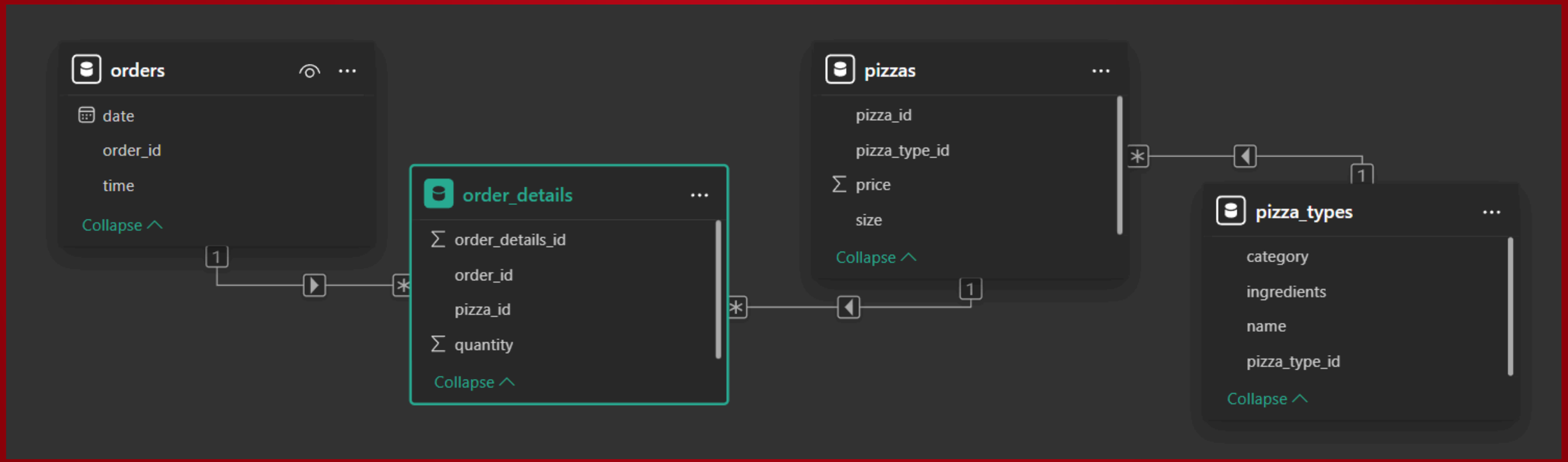
INTERMEDIATE:

- 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.

- The Pizza database consists of four interconnected tables:
- orders and order_details are linked via order_id.
- order_details and pizzas are linked via pizza_id.
- pizzas and pizza_types are linked via pizza_type_id.



RETRIEVE THE TOTAL NUMBER OF ORDERS PLACED.

pizza/postgres@PostgreSQL 17

No limit

Query | Query History

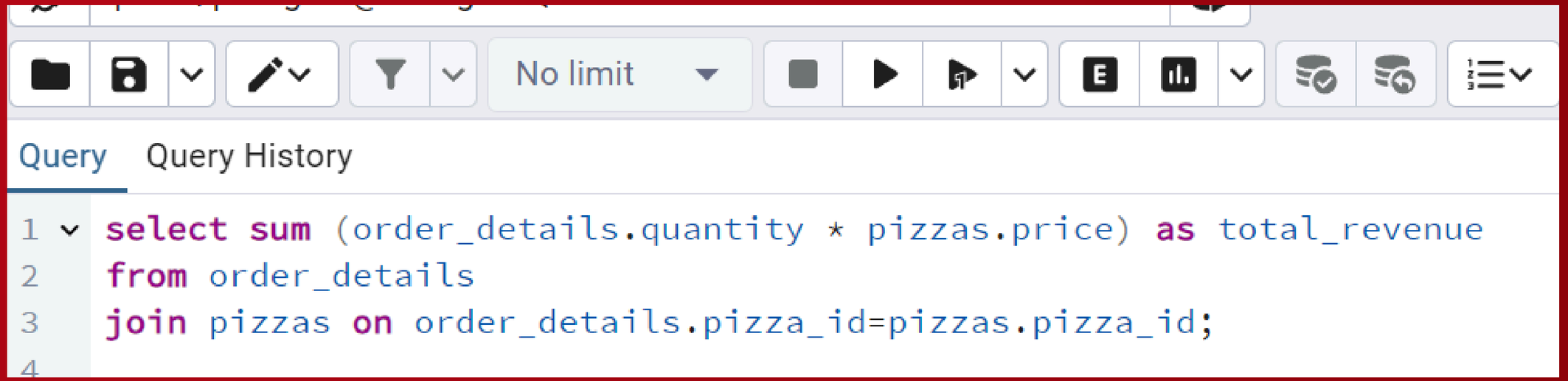
```
1
2  select count(order_details_id) as total_orders from orders;
```

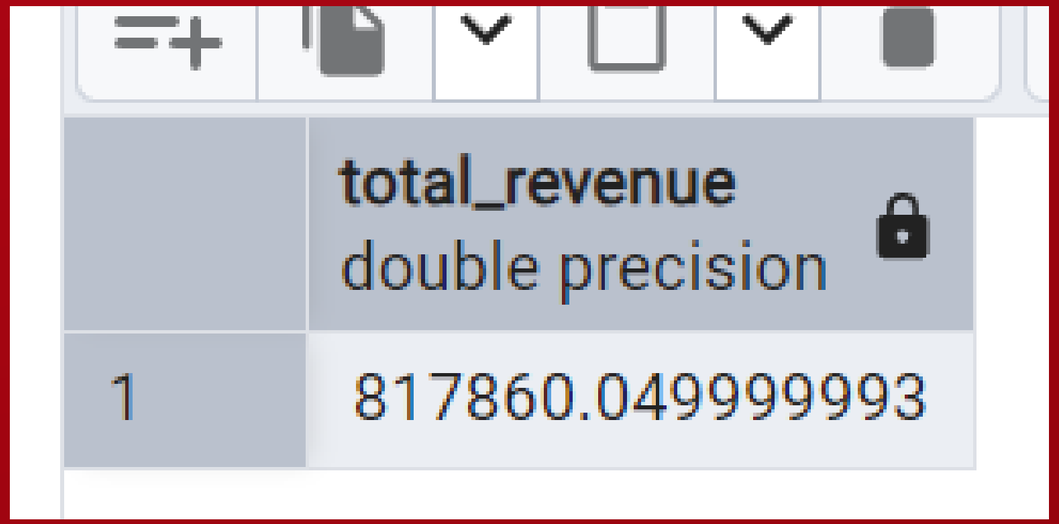
Data Output | Messages | Notifications

SQL

	total_orders bigint
1	21350

CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.





	total_revenue double precision
1	817860.049999993

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;
```

Data Output | Messages | Notifications

	name character varying (150)	price double precision
1	The Greek Pizza	35.95

IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.

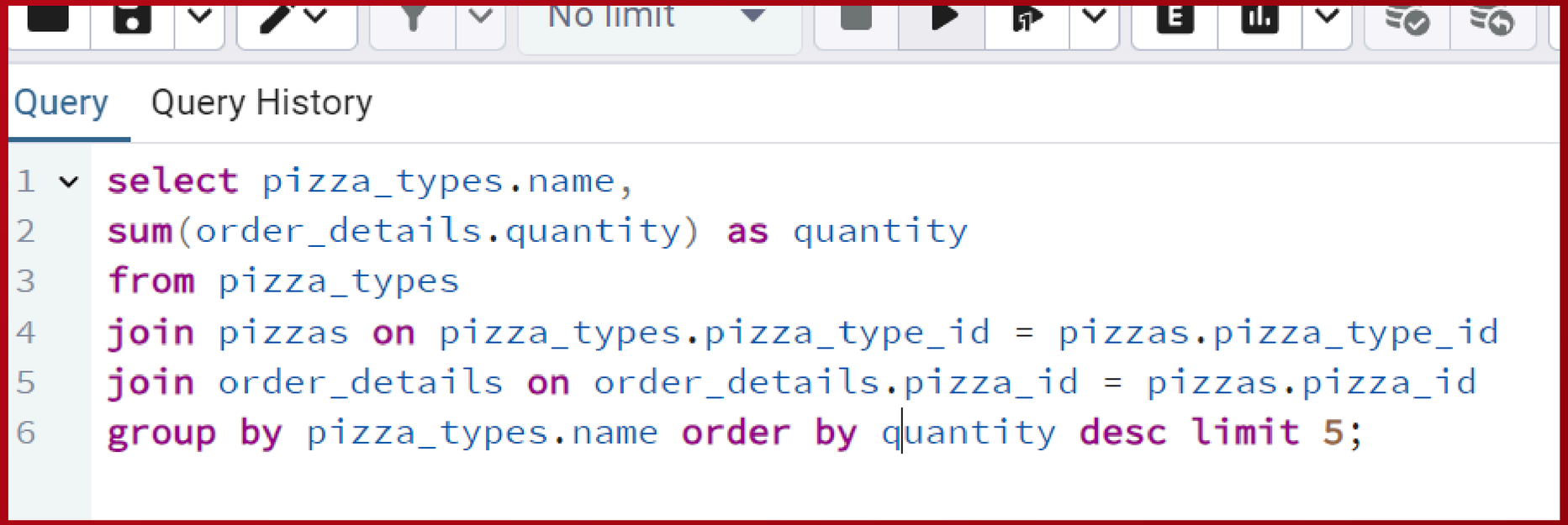
No limit

Query | Query History

```
1  select pizzas.size,
2  count(order_details.order_details_id) as order_count
3  from pizzas
4  join order_details
5  on pizzas.pizza_id = order_details.pizza_id
6  group by pizzas.size order by order_count desc;
```

	size character varying (20)	order_count bigint
1	L	18526
2	M	15385
3	S	14137
4	XL	544
5	XXL	28

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



	name character varying (150)	quantity bigint
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

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

Query | Query History

```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;
```

	category character varying (150)	quantity bigint
1	Classic	14888
2	Supreme	11987
3	Veggie	11649
4	Chicken	11050

DETERMINE THE DISTRIBUTION OF ORDERS BY HOUR OF THE DAY.

Query | Query History

```sql
SELECT EXTRACT(HOUR FROM time) AS hour,
        COUNT(order_details_id) AS order_count
        FROM orders
        GROUP BY hour
        ORDER BY hour;
```

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

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

pizza/postgres@PostgreSQL 17

No limit

Query | Query History

```
1  Select category,
2  count(category) as total_count
3  from pizza_types
4  group by category;
5
```

Data Output | Messages | Notifications

SQL

	category character varying (150)	total_count bigint
1	Veggie	9
2	Chicken	6
3	Supreme	9
4	Classic	8

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

Query | Query History

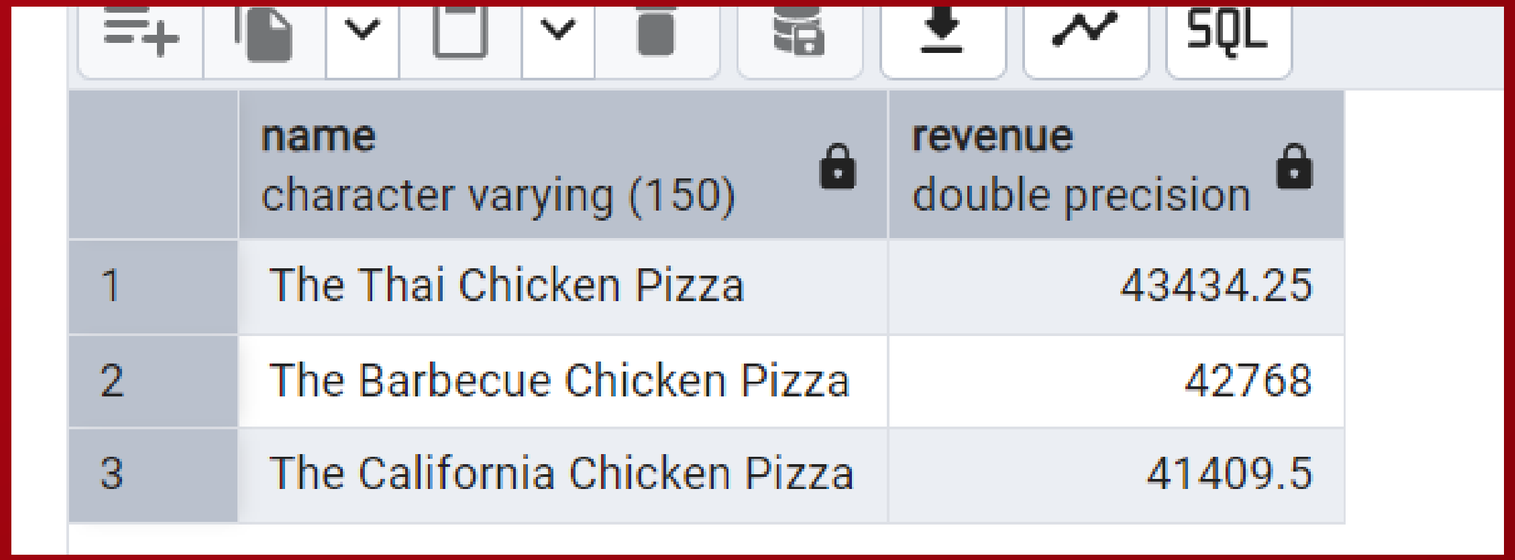
```sql
With CTE as
(select orders.date, sum(order_details.quantity)
                as quantity from orders
                join order_details
                on orders.order_details_id = order_details.order_details_id
                group by orders.date)
Select avg(quantity) from CTE;
```

	avg numeric
1	60.7709497206703911

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

Query | Query History

```sql
select 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 desc limit 3;
```



	name character varying (150)	revenue double precision
1	The Thai Chicken Pizza	43434.25
2	The Barbecue Chicken Pizza	42768
3	The California Chicken Pizza	41409.5

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

Query History

```sql
SELECT pizza_types.category,
       (SUM(order_details.quantity * pizzas.price) /
        (SELECT SUM(order_details.quantity * pizzas.price)
         FROM order_details
         JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id) * 100
       ) AS revenue_percentage
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_percentage DESC;
```

	category character varying	revenue_percentage double precision
1	Classic	26.905960255669903
2	Supreme	25.45631126009884
3	Chicken	23.955137556847493
4	Veggie	23.682590927384783

ANALYZE THE CUMULATIVE REVENUE GENERATED OVER TIME.

Query | Query History

```
SELECT sales.date,
            SUM(revenue) OVER (ORDER BY sales.date) AS cum_revenue
        FROM (SELECT orders.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_details_id = order_details.order_detail
            GROUP BY orders.date) AS sales;
```

	date date	cum_revenue double precision
1	2015-01-01	1171.45
2	2015-01-02	2316.1000000000004
3	2015-01-03	3433.8
4	2015-01-04	4341.8
5	2015-01-05	5247.25
6	2015-01-06	6200.0

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

Query | Query History

```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;
```

	name character varying (150)	revenue double precision
1	The Thai Chicken Pizza	43434.25
2	The Barbecue Chicken Pizza	42768
3	The California Chicken Pizza	41409.5
4	The Classic Deluxe Pizza	38180.5
5	The Hawaiian Pizza	32273.25

THANK YOU!