

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.

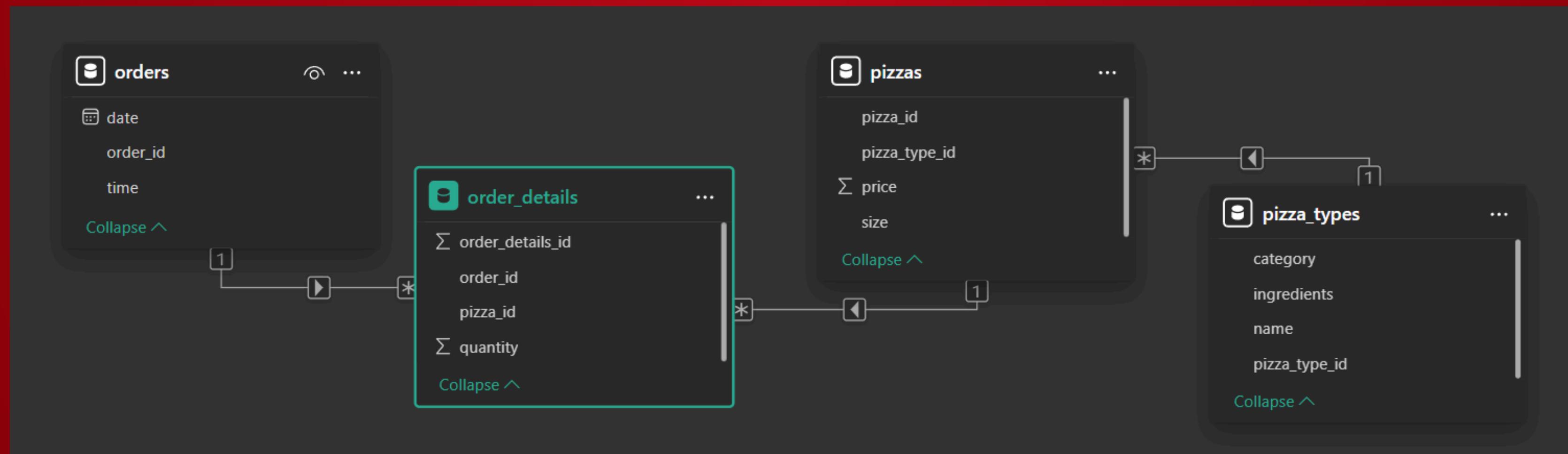
INTERMEDIATE:

- 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.

The screenshot shows a PostgreSQL client interface with the following details:

- Connection:** pizza/postgres@PostgreSQL 17
- Toolbar:** Includes icons for file, edit, search, and various database operations.
- Panel:** Shows "Query History" tab selected.
- Query Editor:** Displays the SQL query:

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

The screenshot shows the "Data Output" panel with the following details:

- Header:** Data Output, Messages, Notifications
- Toolbar:** Includes icons for new query, export, clipboard, and other functions.
- Table:** Shows the results of the query:

	total_orders	
	bigint	lock
1	21350	

CALCULATE THE TOTAL REVENUE GENERATED FROM PIZZA SALES.

The screenshot shows a MySQL Workbench interface. The toolbar at the top includes icons for file, database, edit, filter, and various execution options. Below the toolbar, the tabs 'Query' and 'Query History' are visible, with 'Query' being the active tab. The main area contains the following SQL code:

```
1 select sum (order_details.quantity * pizzas.price) as total_revenue  
2 from order_details  
3 join pizzas on order_details.pizza_id=pizzas.pizza_id;  
4
```

The screenshot shows a results grid in MySQL Workbench. The top row contains column headers: 'total_revenue' and 'double precision'. The bottom row contains a single data point: '1' and '817860.049999993'. A lock icon is present next to the column header.

	total_revenue	double precision
1	817860.049999993	

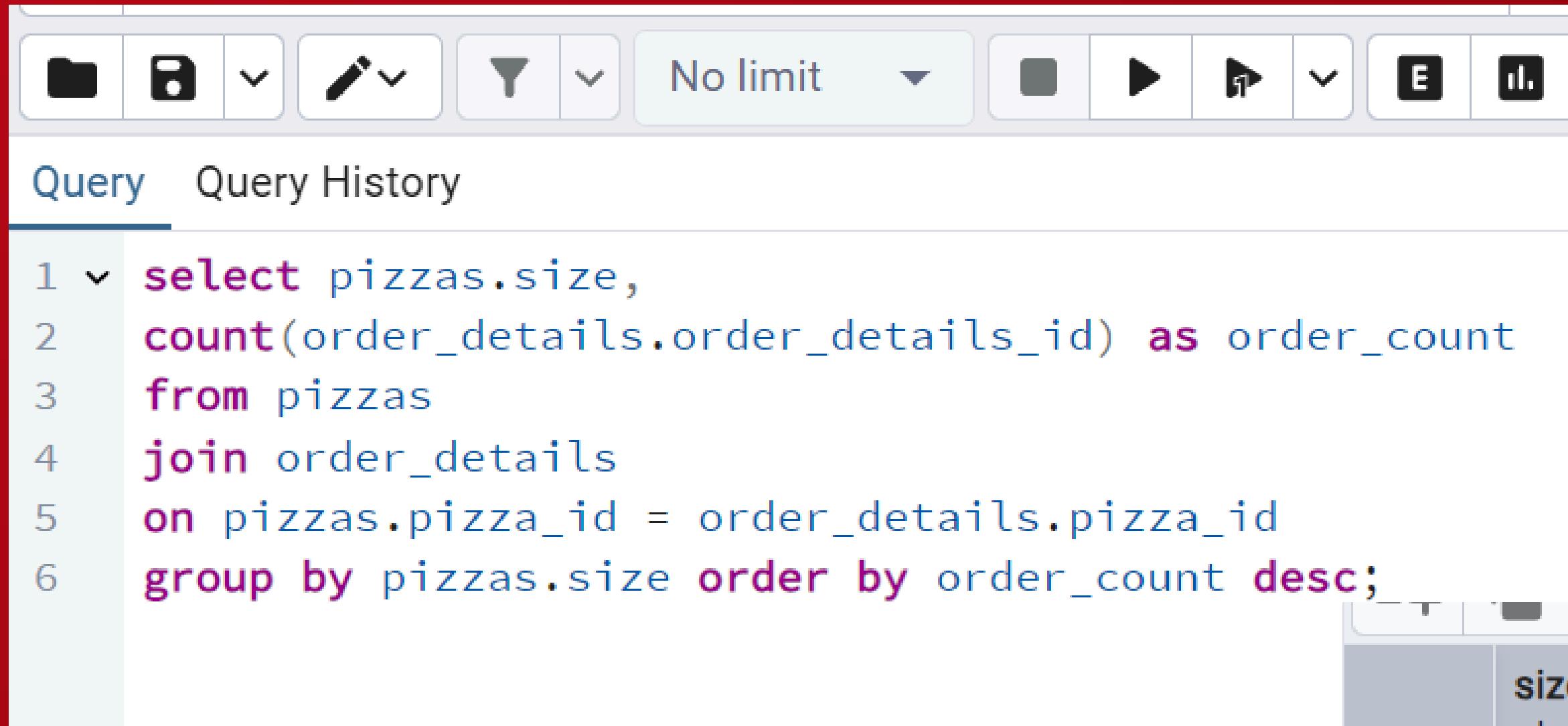
IDENTIFY THE HIGHEST-PRICED PIZZA.

Query Query history

```
1 ✓ select pizza_types.name, pizzas.price
2   from pizza_types
3   join pizzas
4     on pizza_types.pizza_type_id = pizzas.pizza_type_id
5   order by pizzas.price desc
6   limit 1;
```

Data Output	Messages	Notifications

IDENTIFY THE MOST COMMON PIZZA SIZE ORDERED.



The screenshot shows a database query interface with the following components:

- Toolbar:** Includes icons for file operations (New, Open, Save, Print, Copy, Paste, Find, Filter), a search bar, and a dropdown menu set to "No limit".
- Query Tab:** Labeled "Query" with a blue underline, indicating the active tab.
- Query Editor:** Displays the following SQL code:

```
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.

The screenshot shows a database query editor interface with a toolbar at the top and a "Query History" tab selected. Below the toolbar, the query code is displayed:

```
1 select pizza_types.name,
2     sum(order_details.quantity) as quantity
3     from pizza_types
4     join pizzas on pizza_types.pizza_type_id = pizzas.pizza_type_id
5     join order_details on order_details.pizza_id = pizzas.pizza_id
6     group by pizza_types.name order by quantity desc limit 5;
```

	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

```
1 select pizza_types.category, sum(order_details.quantity) as quantity
2   from pizza_types
3   join pizzas on pizza_types.pizza_type_id = pizzas.pizza_type_id
4   join order_details on order_details.pizza_id = pizzas.pizza_id
5 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 History
1  SELECT EXTRACT(HOUR FROM time) AS hour,
2      COUNT(order_details_id) AS order_count
3  FROM orders
4  GROUP BY hour
5  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 History

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

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
1 With CTE as
2   (select orders.date, sum(order_details.quantity)
3     as quantity from orders
4       join order_details
5         on orders.order_details_id = order_details.order_details_id
6           group by orders.date)
7 Select avg(quantity) from CTE;
```

	avg	numeric
1		60.7709497206703911

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

Query Query History

```
1 ✓ select pizza_types.name,  
2   sum(order_details.quantity*pizzas.price) as revenue from pizza_types  
3     join pizzas on pizza_types.pizza_type_id = pizzas.pizza_type_id  
4     join order_details on order_details.pizza_id = pizzas.pizza_id  
5   group by pizza_types.name order by revenue desc limit 3;
```

The screenshot shows a database query interface with a toolbar at the top containing various icons for file operations, a refresh button, and a SQL tab. Below the toolbar is a table displaying the results of the executed SQL query. The table has two columns: 'name' and 'revenue'. The 'name' column is defined as character varying (150) and the 'revenue' column is defined as double precision. The results show three rows, each representing a pizza type and its total revenue:

	name	revenue
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

```
▼ 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

```
1 ▾ SELECT sales.date,  
2           SUM(revenue) OVER (ORDER BY sales.date) AS cum_revenue  
3   FROM (SELECT orders.date,  
4             SUM(order_details.quantity * pizzas.price) AS revenue  
5           FROM order_details  
6             JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id  
7             JOIN orders ON orders.order_details_id = order_details.order_detail  
8           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

```
1 Select name, revenue from
2   (select category, name, revenue, rank() over (partition by category order by revenue desc) as rn
3    from (select pizza_types.category, pizza_types.name, sum(order_details.quantity * pizzas.price) as revenue
4          from pizza_types
5          join pizzas on pizza_types.pizza_type_id = pizzas.pizza_type_id
6          join order_details on order_details.pizza_id = pizzas.pizza_id
7          group by pizza_types.category, pizza_types.name) as a) as b
8  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!

