

# SQL PORTFOLIO PROJECT

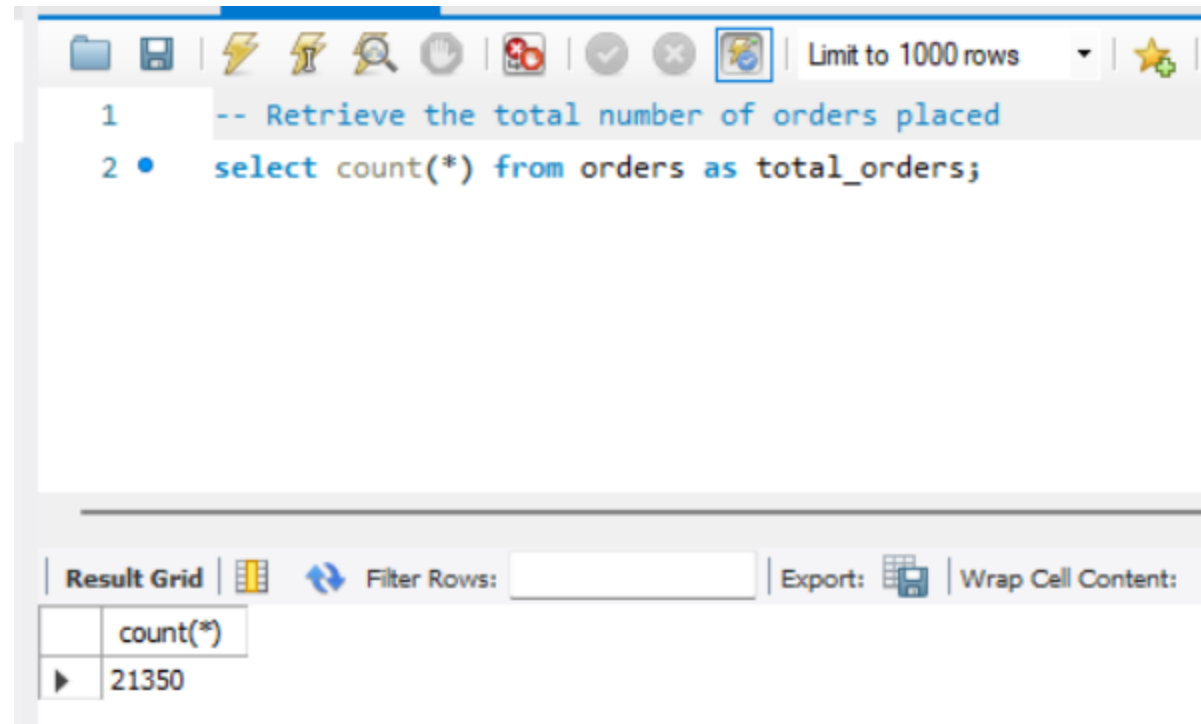
The project includes a series of tasks that involve data extraction, transformation, and analysis using SQL queries.

# Create Database and Tables

The screenshot displays a database management interface with two main panels. The left panel, titled 'Navigator', shows a tree view of the 'pizzhut' database schema. Under 'Tables', the 'orders' table is expanded, showing its columns: 'order\_id', 'order\_date', 'order\_time', and 'primary key(order\_id)'. The 'pizza\_types' table is also expanded, showing its columns: 'pizza\_type\_id', 'name', 'category', and 'ingredients'. The right panel, titled 'Query 1', shows the SQL code used to create the database and tables. The code is as follows:

```
1 • create database pizzhut;  
2 • use pizzahut;  
3 • select * from pizzhut.orders;  
4 • create table orders(  
5     order_id int not null,  
6     order_date date not null,  
7     order_time time not null,  
8     primary key(order_id));  
9  
10 • Create table order_details(  
11     order_details_id int not null,  
12     order_id int not null,  
13     pizza_id text not null,  
14     quantity int not null,  
15     primary key(order_details_id))
```

- Retrieve the total number of orders placed



The screenshot shows a SQL query editor interface. The top toolbar includes icons for file operations, execution, and a dropdown menu set to "Limit to 1000 rows". The query text is as follows:

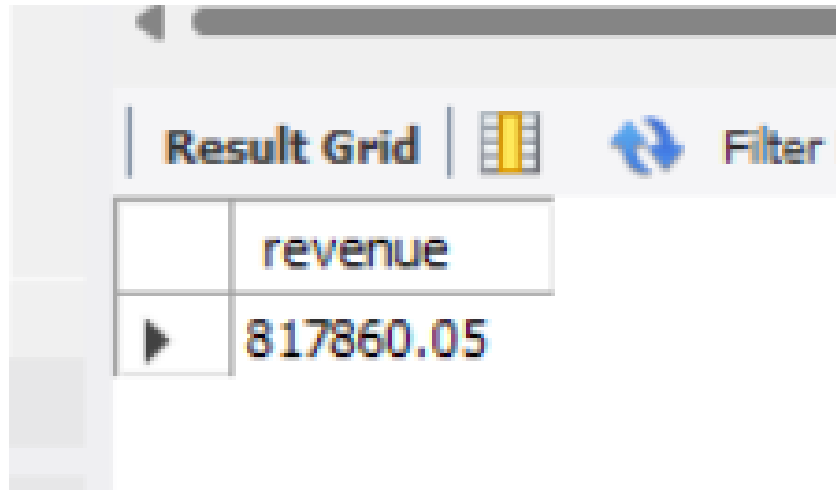
```
1 -- Retrieve the total number of orders placed
2 • select count(*) from orders as total_orders;
```

Below the query editor, the "Result Grid" tab is active. It displays a single row of results:

	count(*)
▶	21350

-- Calculate the total revenue generated from pizza sales.

```
SELECT  
    ROUND(SUM(order_details.quantity * pizzas.price),  
          2) AS revenue  
FROM  
    order_details  
    JOIN  
    pizzas ON pizzas.pizza_id = order_details.pizza_id
```






The screenshot shows a database interface with a 'Result Grid' tab. The grid contains one row with the column name 'revenue' and the value '817860.05'. There are navigation icons and a 'Filter' button above the grid.

	revenue
▶	817860.05

- Identify the highest-priced pizza.

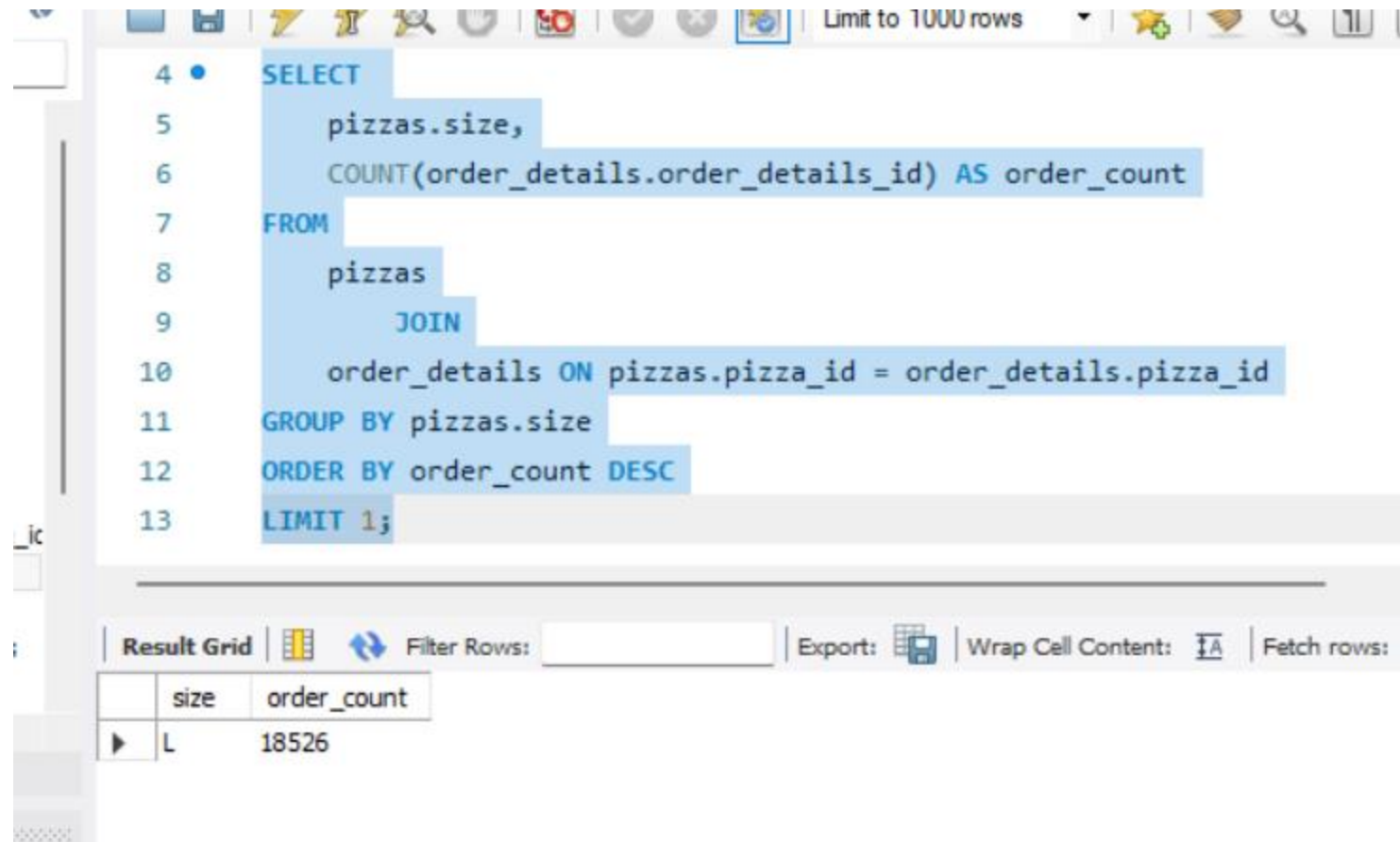
```
4 • SELECT
5     pizza_types.name, pizzas.price
6 FROM
7     pizza_types
8     JOIN
9     pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
10 ORDER BY pizzas.price DESC
11 LIMIT 1;
12
```

ic

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

	name	price
▶	The Greek Pizza	35.95

Identify the most common pizza size ordered.



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```
4 • SELECT
5     pizzas.size,
6     COUNT(order_details.order_details_id) AS order_count
7 FROM
8     pizzas
9     JOIN
10    order_details ON pizzas.pizza_id = order_details.pizza_id
11 GROUP BY pizzas.size
12 ORDER BY order_count DESC
13 LIMIT 1;
```

Below the query editor, there is a toolbar with options: Result Grid, Filter Rows, Export, Wrap Cell Content, and Fetch rows. The Result Grid shows the following data:

	size	order_count
▶	L	18526

List the top 5 most ordered pizza types along with their quantities

```
2 • SELECT
3     pizza_types.name, SUM(order_details.quantity) AS quantity
4 FROM
5     pizzas
6     JOIN
7     order_details ON pizzas.pizza_id = order_details.pizza_id
8     JOIN
9     pizza_types ON pizza_types.pizza_type_id = pizzas.pizza_type_id
10 GROUP BY pizza_types.name
11 ORDER BY quantity DESC
12 LIMIT 5;
```

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

	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

Join the necessary tables to find the total quantity of each pizza category ordered.

```
6 • SELECT
7     pizza_types.category,
8     sum(order_details.quantity) AS quantity
9 FROM
10    pizzas
11    JOIN
12    order_details ON pizzas.pizza_id = order_details.pizza_id
13    JOIN
14    pizza_types ON pizza_types.pizza_type_id = pizzas.pizza_type_id
15 GROUP BY pizza_types.category
16 order by quantity desc;
17
18
```

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

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050



Determine the distribution of orders by hour of the day.

```
2 • SELECT
3     HOUR(order_time) AS hour, COUNT(order_id) AS order_count
4 FROM
5     orders
6 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			
	20	1642			
	21	1198			
	22	663			
	23	28			

Result 4 ×

Output





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

```
3 • SELECT
4     pizza_types.category, COUNT(pizza_types.name)
5 FROM
6     pizza_types
7 GROUP BY pizza_types.category
```

result Grid			 Filter Rows: <input type="text"/>	Export: 	Wrap Cell Content: 
category	count(pizza_types.name)				
Chicken	6				
Classic	8				
Supreme	9				
Veggie	9				

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

```
2 • SELECT
3     ROUND(AVG(quantity), 0) as avg_pizza_order_perday
4 FROM
5     (SELECT
6         orders.order_date, SUM(order_details.quantity) AS quantity
7     FROM
8         orders
9     JOIN order_details ON orders.order_id = order_details.order_id
10    GROUP BY orders.order_date) AS order_quantity;
```

result Grid			Filter Rows: <input type="text"/>	Export: 	Wrap Cell Content: 
avg_pizza_order_perday					
138					

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

```
3 • select pizza_types.name ,sum(order_details.quantity*pizzas.price) as revenue
4   from
5   pizza_types
6   join
7   pizzas
8   on pizza_types.pizza_type_id=pizzas.pizza_type_id
9   join
10  order_details
11  on
12  order_details.pizza_id=pizzas.pizza_id
13  group by pizza_types.name
14  order by revenue desc limit 3;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
	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.

```
WITH TotalRevenue AS (  
    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  
),  
RevenuePerType AS (  
    SELECT  
        pizza_types.category,  
        ROUND(SUM(order_details.quantity * pizzas.price), 2) AS type_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  
)  
  
SELECT  
    r.category,  
    r.type_revenue,  
    t.total_sales,  
    ROUND((r.type_revenue / t.total_sales) * 100, 2) AS revenue_percentage  
FROM  
    RevenuePerType r  
CROSS JOIN  
    TotalRevenue t  
ORDER BY  
    revenue_percentage DESC;
```

Result Grid				
Filter Rows:				
Export:				
Wrap Cell Content:				
	category	type_revenue	total_sales	revenue_percentage
▶	Classic	220053.1	817860.05	26.91
	Supreme	208197	817860.05	25.46
	Chicken	195919.5	817860.05	23.96
	Veggie	193690.45	817860.05	23.68

Analyze the cumulative revenue generated over time.

```
2 • select order_date, sum(revenue) over (order by order_date) as cumm_revenue
3 from(
4   select orders.order_date, sum(order_details.quantity*pizzas.price) as revenue
5   from order_details
6   join
7   pizzas
8   on
9   order_details.pizza_id=pizzas.pizza_id
10  join orders
11  on
12  orders.order_id=order_details.order_id
13  group by orders.order_date)as sales;
```

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

	order_date	cumm_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

Result 1 ▾

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

```
2  with Revenueperpizza as
3  (select
4      pizza_types.category,
5      pizza_types.name,
6      SUM(order_details.quantity * pizzas.price) AS revenue
7  FROM
8      pizza_types
9  JOIN
10     pizzas ON pizza_types.pizza_type_id= pizzas.pizza_type_id
11  JOIN
12     order_details ON order_details.pizza_id= pizzas.pizza_id
13  GROUP BY
14      pizza_types.category,
15      pizza_types.name)
16  ,RankedPizza as
17  (select category,name,revenue,rank()over(partition by category order by revenue desc) as rn
18  from Revenueperpizza )
19
20  select category,name,revenue from
21  RankedPizza
22  where rn<=3
23  order by category,rn;
```

**Result Grid** | Filter Rows: | Exports: | Wrap Cell Contents: |

	category	name	revenue
	Classic	The Pepperoni Pizza	30161.75
	Supreme	The Spicy Italian Pizza	34831.25
	Supreme	The Italian Supreme Pizza	33476.75
	Supreme	The Sicilian Pizza	30940.5
	Veggie	The Four Cheese Pizza	32265.70000000065
	Veggie	The Mexicana Pizza	26780.75
	Veggie	The Five Cheese Pizza	26066.5

Result 5 x