



DISCOUNT
50%
OFF

FREE
DELIVERY

SOL PROJECT

SQL PROJECT ON PIZZA
SALES

1. Retrieve the total number of orders placed.
2. Calculate the total revenue generated from pizza sales.
3. Identify the highest-priced pizza.
4. Identify the most common pizza size ordered.
5. List the top 5 most ordered pizza types along with their quantities.
6. Join the necessary tables to find the total quantity of each pizza category ordered.
7. Determine the distribution of orders by hour of the day.
8. Join relevant tables to find the category-wise distribution of pizzas.
9. Group the orders by date and calculate the average number of pizzas ordered per day.
10. Determine the top 3 most ordered pizza types based on revenue.

Pizza

time

CREATING TABLES

The screenshot shows a MySQL Workbench interface with the following SQL code:

```
1 • create database dominos;
2
3
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 orders_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) );
```

The code creates a database named 'dominos' and two tables: 'orders' and 'orders_details'. The 'orders' table has columns for order_id (int, not null), order_date (date, not null), and order_time (time, not null). It includes a primary key constraint on order_id. The 'orders_details' table has columns for order_details_id (int, not null), order_id (int, not null), pizza_id (text, not null), and quantity (int, not null). It includes a primary key constraint on order_details_id.

Pizza

time



```
1 -- Join relevant tables to find the category-wise distribution of pizzas.  
2  
3 SELECT  
4     category, COUNT(name)  
5 FROM  
6     pizza_types  
7 GROUP BY category
```

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

	category	count(name)
▶	Chicken	6
	Classic	8
	Supreme	9
	Veggie	9

Pizza

time

The screenshot shows a MySQL Workbench interface with a query editor containing the following SQL code:

```
-- Group the orders by date and calculate the average number of pizzas ordered per day
SELECT
    ROUND(AVG(quantity), 0)
FROM
    (
        SELECT
            orders.order_date, SUM(orders_details.quantity) AS quantity
        FROM
            orders
        JOIN orders_details ON orders.order_id = orders_details.order_id
        GROUP BY orders.order_date) AS order_quantity;
```

The code is designed to calculate the average quantity of pizzas ordered per day. It uses a subquery to group the main table 'orders' by 'order_date' and sum the quantity from the joined table 'orders_details'. The result is then rounded to zero decimal places.

Below the code, the results pane shows the output of the query:

order_quantity
3

Pizza

time

```
1  -- Determine the distribution of orders by hour of the day.  
2  Save the script to a file.  
3 • SELECT  
4      HOUR(order_time), COUNT(order_id) AS order_count  
5  FROM  
6      orders  
7  GROUP BY HOUR(order_time);  
8
```

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

	hour(order_time)	order_count
▶	11	1231
	12	2520
	13	2455
	14	1472
	15	1468
	16	1920
	17	2336
	18	2399

Pizza

time

The screenshot shows a MySQL Workbench interface. At the top is a toolbar with various icons for database management. Below the toolbar is a query editor window containing the following SQL code:

```
1 -- Group the orders by date and calculate the average number of pizzas ordered
2 SELECT
3     ROUND(AVG(quantity), 0)
4 FROM
5     (SELECT
6         orders.order_date, SUM(orders_details.quantity) AS quantity
7     FROM
8         orders
9     JOIN orders_details ON orders.order_id = orders_details.order_id
10    GROUP BY orders.order_date) AS order_quantity;
```

The code uses a subquery to calculate the total quantity for each order date and then applies a round function to the average quantity.

At the bottom of the interface is a results grid titled "Result Grid". It contains one row of data:

round(avg(quantity),0)
138

Pizza

time



Limit to 1000 rows

```
1      -- Join relevant tables to find the category-wise dis
2
3      SELECT
4          category, COUNT(name)
5      FROM
6          pizza_types
7      GROUP BY category
```

Result Grid



Filter Rows:

Export:



Wrap Cell Conte

	category	count(name)
▶	Chicken	6
	Classic	8

Pizza

time

```
4     pizza_types.category,  
5     SUM(orders_details.quantity) AS quantity  
6   FROM  
7     pizza_types  
8       JOIN  
9     pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id  
10      JOIN  
11    orders_details ON orders_details.pizza_id = pizzas.pizza_id  
12  GROUP BY pizza_types.category  
13 ORDER BY quantity DESC;
```

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

category	quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

Pizza

time

```
3     pizza_types.name, SUM(orders_details.quantity) AS quantity
4 FROM
5     pizza_types
6         JOIN
7     pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
8         JOIN
9     orders_details ON orders_details.pizza_id = pizzas.pizza_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

Pizza

time

The screenshot shows a MySQL Workbench interface with a query editor and a results grid.

Query Editor:

```
1 -- Identify the most common pizza size ordered.
2
3 • SELECT
4     pizzas.size,
5     COUNT(orders_details.order_details_id) AS order_count
6 FROM
7     pizzas
8     JOIN
9     orders_details ON pizzas.pizza_id = orders_details.pizza_id
10 GROUP BY pizzas.size
```

Results Grid:

size	order_count
L	18526

Pizza

time

```
1  -- Identify the highest-priced pizza.  
2  
3 • SELECT  
4      pizza_types.name, pizzas.price  
5  FROM  
6      pizza_types  
7      JOIN  
8      pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id  
9  ORDER BY pizzas.price DESC  
10 LIMIT 1;
```

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

name	price
The Greek Pizza	35.95

Pizza

time



Limit to 1000 rows

```
1 -- Calculate the total revenue generated from pizza sales.  
2  
3 • SELECT  
4     ROUND(SUM(orders_details.quantity * pizzas.price),  
5             2) AS total_sales  
6 FROM  
7     orders_details  
8     JOIN  
9     pizzas ON pizzas.pizza_id = orders_details.pizza_id
```

Result Grid |



Filter Rows:

Export:



Wrap Cell Content:



	total_sales
▶	817860.05

Pizza

time

The screenshot shows a MySQL Workbench interface. The top toolbar has various icons for database management. The main area contains a numbered SQL query:

```
1 -- Retrieve the total number of orders placed.  
2  
3 • SELECT  
4     COUNT(order_id) AS total_orders  
5 FROM  
6     orders;  
7
```

The result grid at the bottom shows the output of the query:

total_orders
21350

Below the result grid are buttons for 'Result Grid' (selected), 'Filter Rows', 'Export', and 'Wrap Cell Content'.