

```

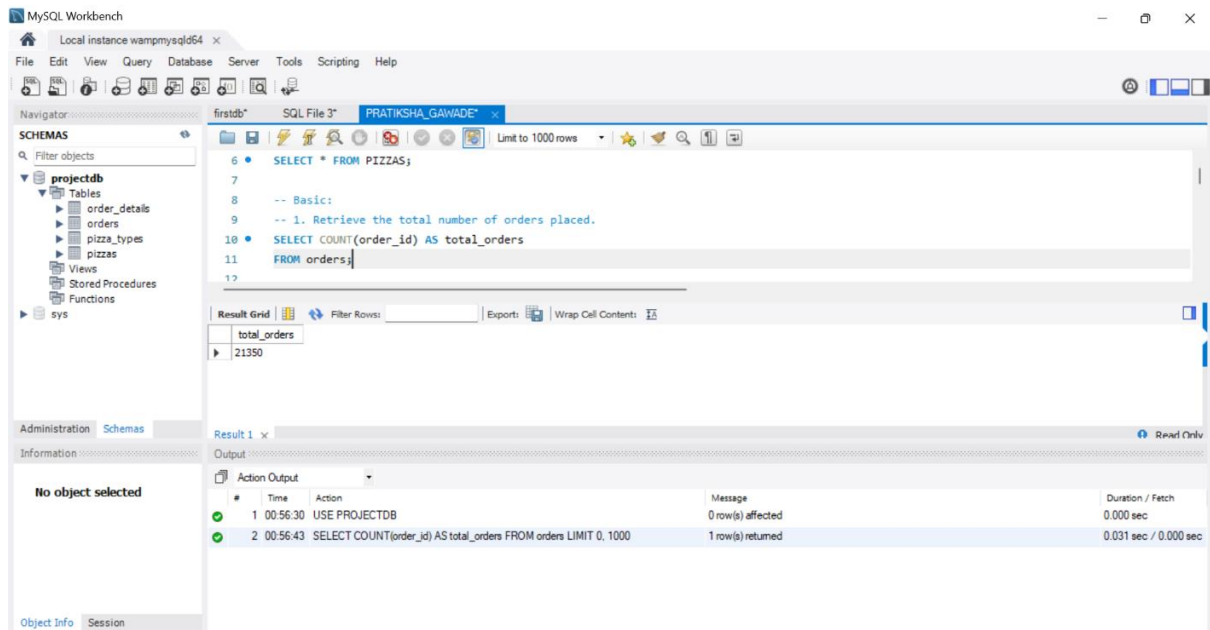
CREATE DATABASE PROJECTDB;
USE PROJECTDB;
SELECT * FROM ORDER_DETAILS;
SELECT * FROM ORDERS;
SELECT * FROM PIZZA_TYPES;
SELECT * FROM PIZZAS;

```

```

-- Basic:
-- 1. Retrieve the total number of orders placed.
SELECT COUNT(order_id) AS total_orders
FROM orders;

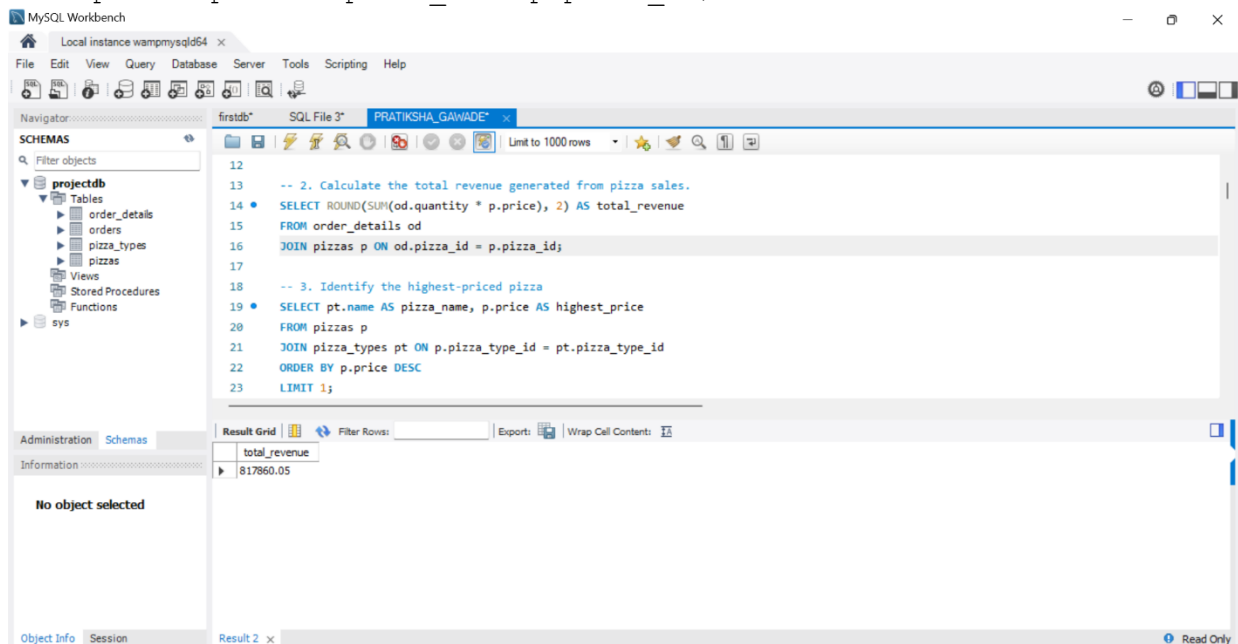
```



```

-- 2. Calculate the total revenue generated from pizza sales.
SELECT ROUND(SUM(od.quantity * p.price), 2) AS total_revenue
FROM order_details od
JOIN pizzas p ON od.pizza_id = p.pizza_id;

```



```
-- 3. Identify the highest-priced pizza
SELECT pt.name AS pizza_name, p.price AS highest_price
FROM pizzas p
JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
ORDER BY p.price DESC
LIMIT 1;
```

MySQL Workbench

The screenshot shows the MySQL Workbench interface. The top toolbar includes icons for File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar shows the 'SCHEMAS' panel with a tree view of the 'projectdb' database, including tables like 'order_details', 'orders', 'pizza_types', and 'pizzas'. The main editor window displays a SQL script with the following queries:

```
16 JOIN pizzas p ON od.pizza_id = p.pizza_id;
17
18 -- 3. Identify the highest-priced pizza
19 • SELECT pt.name AS pizza_name, p.price AS highest_price
20 FROM pizzas p
21 JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
22 ORDER BY p.price DESC
23 LIMIT 1;
24
25 -- 4. Identify the most common pizza size ordered.
26 • SELECT p.size, SUM(od.quantity) AS total_ordered
27 FROM order_details od
```

Below the SQL editor, the 'Result Grid' tab is active, showing the results of the third query. The results are displayed in a table with two columns: 'pizza_name' and 'highest_price'.

pizza_name	highest_price
The Greek Pizza	35.95

The bottom status bar indicates 'No object selected'.

```
-- 4. Identify the most common pizza size ordered.
SELECT p.size, SUM(od.quantity) AS total_ordered
FROM order_details od
JOIN pizzas p ON od.pizza_id = p.pizza_id
GROUP BY p.size
ORDER BY total_ordered DESC
LIMIT 1;
```

MySQL Workbench

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' panel with a tree view of the 'projectdb' database, including tables like 'order_details', 'orders', 'pizza_types', and 'pizzas'. The main editor window shows a SQL query with line numbers 25 to 36. The query is identical to the one in the first block. Below the query editor, the 'Result Grid' tab is active, showing a single row of results with columns 'size' and 'total_ordered'. The row shows 'L' for size and '18956' for total_ordered. The bottom status bar indicates 'No object selected'.

Local instance wampmysqld64 x

File Edit View Query Database Server Tools Scripting Help

Navigator: firstdb* SQL File 3* PRATIKSHA_GAWADE*

Limit to 1000 rows

25 -- 4. Identify the most common pizza size ordered.
 26 • SELECT p.size, SUM(od.quantity) AS total_ordered
 27 FROM order_details od
 28 JOIN pizzas p ON od.pizza_id = p.pizza_id
 29 GROUP BY p.size
 30 ORDER BY total_ordered DESC
 31 LIMIT 1;
 32
 33 -- 5. List the top 5 most ordered pizza types along with their quantities.
 34 • SELECT pt.name AS pizza_type, SUM(od.quantity) AS total_quantity_ordered
 35 FROM order_details od
 36 JOIN pizzas p ON od.pizza_id = p.pizza_id

Administration Schemas

Information

No object selected

size	total_ordered
L	18956

```
-- 5. List the top 5 most ordered pizza types along with their
quantities.
SELECT pt.name AS pizza_type, SUM(od.quantity) AS total_quantity_ordered
FROM order_details od
JOIN pizzas p ON od.pizza_id = p.pizza_id
JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.name
ORDER BY total_quantity_ordered DESC
LIMIT 5;
```

MySQL Workbench

Local instance wampmysqld64 x

File Edit View Query Database Server Tools Scripting Help

Navigator: firstdb* SQL File 3* PRATIKSHA_GAWADE*

Limit to 1000 rows

SCHEMAS

Filter objects

projectdb

- Tables
 - order_details
 - orders
 - pizza_types
 - pizzas
- Views
- Stored Procedures
- Functions
- sys

Administration Schemas

Information: No object selected

```
31 LIMIT 1;
32
33 -- 5. List the top 5 most ordered pizza types along with their quantities.
34 • SELECT pt.name AS pizza_type, SUM(od.quantity) AS total_quantity_ordered
35 FROM order_details od
36 JOIN pizzas p ON od.pizza_id = p.pizza_id
37 JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
38 GROUP BY pt.name
39 ORDER BY total_quantity_ordered DESC
40 LIMIT 5;
41
42 -- Intermediate:
```

Result Grid

pizza_type	total_quantity_ordered
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

```
-- Intermediate:
-- 6. Join the necessary tables to find the total quantity of each pizza
category ordered.
SELECT pt.category, SUM(od.quantity) AS total_quantity
FROM order_details od
JOIN pizzas p ON od.pizza_id = p.pizza_id
JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.category
ORDER BY total_quantity DESC;
```

MySQL Workbench

Local instance wampmysqld64 x

File Edit View Query Database Server Tools Scripting Help

Navigator: firstdb* SQL File 3* PRATIKSHA_GAWADE*

Limit to 1000 rows

40 LIMIT 5;

41

42 -- Intermediate:

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

44 • SELECT pt.category, SUM(od.quantity) AS total_quantity

45 FROM order_details od

46 JOIN pizzas p ON od.pizza_id = p.pizza_id

47 JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id

48 GROUP BY pt.category

49 ORDER BY total_quantity DESC;

50

51 -- 7. Determine the distribution of orders by hour of the day.

Administration Schemas

Information: No object selected

category	total_quantity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

```
-- 7. Determine the distribution of orders by hour of the day.
SELECT HOUR(time) AS order_hour, COUNT(order_id) AS total_orders
FROM orders
GROUP BY HOUR(time)
ORDER BY order_hour;
```

MySQL Workbench

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The left sidebar shows the 'SCHEMAS' panel with a tree view of the 'projectdb' database, including tables like 'order_details', 'orders', 'pizza_types', and 'pizzas'. The main editor window displays a SQL query with line numbers 48 to 56. The query is:


```
48 OUP BY pt.category
49 DER BY total_quantity DESC;
50
51 7. Determine the distribution of orders by hour of the day.
52 • LECT HOUR(time) AS order_hour, COUNT(order_id) AS total_orders
53 OM orders
54 OUP BY HOUR(time)
55 DER BY order_hour;
56
```

 The bottom panel shows the 'Result Grid' with a table of results:

order_hour	total_orders
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

 The bottom status bar shows 'Object Info', 'Session', 'Vertical Output', and 'Result 10'.

```
-- 8. Join relevant tables to find the category-wise distribution of
pizzas.
SELECT pt.category, COUNT(p.pizza_id) AS total_pizzas
FROM pizzas p
JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.category
ORDER BY total_pizzas DESC;
```

MySQL Workbench

Local instance wampmysqld64 x

File Edit View Query Database Server Tools Scripting Help

Navigator: firstdb* SQL File 3* PRATIKSHA_GAWADE*

SCHEMAS

Filter objects

▼ projectdb

- Tables
 - order_details
 - orders
 - pizza_types
 - pizzas
- Views
- Stored Procedures
- Functions

▼ sys

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

58 • SELECT pt.category, COUNT(p.pizza_id) AS total_pizzas

59 FROM pizzas p

60 JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id

61 GROUP BY pt.category

62 ORDER BY total_pizzas DESC;

63

64 -- 9. Group the orders by date and calculate the average number of pizzas order

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [⌕](#)

	category	total_pizzas
▶	Veggie	27
	Classic	26
	Supreme	25
	Chicken	18

Administration Schemas

Information

No object selected

```
-- 9. Group the orders by date and calculate the average number of pizzas
ordered per day.
SELECT ROUND(AVG(daily_pizzas), 2) AS avg_pizzas_per_day
FROM (SELECT o.date, SUM(od.quantity) AS daily_pizzas
FROM orders o
JOIN order_details od ON o.order_id = od.order_id
GROUP BY o.date) AS daily_summary;
```

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view for 'projectdb' containing tables: 'order_details', 'orders', 'pizza_types', and 'pizzas'. The main editor window shows a SQL query with line numbers 64 to 71. The query is the same as the one in the text block above. Below the query editor, the 'Result Grid' is visible, showing a single row with the column 'avg_pizzas_per_day' and the value '138.47'. The bottom status bar indicates 'No object selected'.

```
64 -- 9. Group the orders by date and calculate the average number of pizzas ordered per day.
65 • SELECT ROUND(AVG(daily_pizzas), 2) AS avg_pizzas_per_day
66 FROM (SELECT o.date, SUM(od.quantity) AS daily_pizzas
67 FROM orders o
68 JOIN order_details od ON o.order_id = od.order_id
69 GROUP BY o.date) AS daily_summary;
70
71 -- 10. Determine the top 3 most ordered pizza types based on revenue.
```

avg_pizzas_per_day
138.47

```
-- 10. Determine the top 3 most ordered pizza types based on revenue.
SELECT pt.name AS pizza_type, ROUND(SUM(od.quantity * p.price), 2) AS
total_revenue
FROM order_details od
JOIN pizzas p ON od.pizza_id = p.pizza_id
JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.name
ORDER BY total_revenue DESC
LIMIT 3;
```

MySQL Workbench

Local instance wampmysqld64 x

File Edit View Query Database Server Tools Scripting Help

Navigator firstdb* SQL File 3* PRATIKSHA_GAWADE*

Limit to 1000 rows

70

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

72 • SELECT pt.name AS pizza_type, ROUND(SUM(od.quantity * p.price), 2) AS total_revenue

73 FROM order_details od

74 JOIN pizzas p ON od.pizza_id = p.pizza_id

75 JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id

76 GROUP BY pt.name

77 ORDER BY total_revenue DESC

78 LIMIT 3;

79

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

pizza_type	total_revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5

Administration Schemas

Information

No object selected

```
-- Advanced:
-- 11. Calculate the percentage contribution of each pizza type to total
revenue.
SELECT pt.name AS pizza_type, ROUND(SUM(od.quantity * p.price), 2) AS
pizza_revenue, ROUND((SUM(od.quantity * p.price) /
(SELECT SUM(od2.quantity * p2.price)
FROM order_details od2
JOIN pizzas p2 ON od2.pizza_id = p2.pizza_id)) * 100,2)
AS percentage_contribution
FROM order_details od
JOIN pizzas p ON od.pizza_id = p.pizza_id
JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.name
ORDER BY percentage_contribution DESC;
```

MySQL Workbench

Local instance wampmysqld64 x

File Edit View Query Database Server Tools Scripting Help

Navigator: firstdb* SQL File 3* PRATIKSHA_GAWADE*

SCHEMAS

Filter objects

projectdb

- Tables
 - order_details
 - orders
 - pizza_types
 - pizzas
- Views
- Stored Procedures
- Functions

sys

Administration Schemas

Information

No object selected

79

80 -- Advanced:

81 -- 11. Calculate the percentage contribution of each pizza type to total revenue.

82 SELECT pt.name AS pizza_type, ROUND(SUM(od.quantity * p.price), 2) AS pizza_revenue, ROUND((SUM(od.quantity * p.price) /

83 (SELECT SUM(od2.quantity * p2.price)

84 FROM order_details od2

85 JOIN pizzas p2 ON od2.pizza_id = p2.pizza_id)) * 100,2)

86 AS percentage_contribution

87 FROM order_details od

88

Result Grid

Filter Rows:

Export: Wrap Cell Content: I A

pizza_type	pizza_revenue	percentage_contribution
The Thai Chicken Pizza	43434.25	5.31
The Barbecue Chicken Pizza	42768	5.23
The California Chicken Pizza	41409.5	5.06
The Classic Deluxe Pizza	38180.5	4.67
The Spicy Italian Pizza	34831.25	4.26
The Southwest Chicken Pizza	34705.75	4.24
The Italian Supreme Pizza	33476.75	4.09
The Hawaiian Pizza	32273.25	3.95
The Four Cheese Pizza	32265.7	3.95
The Sicilian Pizza	30940.5	3.78
The Pepperoni Pizza	30161.75	3.69
The Greek Pizza	28454.1	3.48
The Mexicana Pizza	26780.75	3.27
The Five Cheese Pizza	26066.5	3.19
The Pepper Salami Pizza	25529	3.12
The Italian Capocollo Pizza	25094	3.07

```
-- 12. Analyze the cumulative revenue generated over time.
SELECT daily_sales.order_date, daily_sales.daily_revenue,
       ROUND(SUM(daily_sales.daily_revenue)
       OVER (ORDER BY daily_sales.order_date),2)
       AS cumulative_revenue
FROM (SELECT o.date AS order_date,
ROUND(SUM(od.quantity * p.price), 2)
AS daily_revenue
FROM orders o
JOIN order_details od ON o.order_id = od.order_id
JOIN pizzas p ON od.pizza_id = p.pizza_id
GROUP BY o.date
ORDER BY o.date)
AS daily_sales;
```

MySQL Workbench

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' panel with a tree view of the 'projectdb' database, including tables like 'order_details', 'orders', 'pizza_types', and 'pizzas'. The main editor window shows a SQL query with line numbers 91 to 95. The query is a SELECT statement that calculates daily revenue and cumulative revenue. The 'Result Grid' at the bottom displays the output of the query, showing columns for 'order_date', 'daily_revenue', and 'cumulative_revenue' for dates from 2015-01-01 to 2015-01-19.

order_date	daily_revenue	cumulative_revenue
2015-01-01	2713.85	2713.85
2015-01-02	2731.9	5445.75
2015-01-03	2662.4	8108.15
2015-01-04	1755.45	9863.6
2015-01-05	2065.95	11929.55
2015-01-06	2428.95	14358.5
2015-01-07	2202.2	16560.7
2015-01-08	2838.35	19399.05
2015-01-09	2127.35	21526.4
2015-01-10	2463.95	23990.35
2015-01-11	1872.3	25862.65
2015-01-12	1919.05	27781.7
2015-01-13	2049.6	29831.3
2015-01-14	2527.4	32358.7
2015-01-15	1984.8	34343.5
2015-01-16	2594.15	36937.65
2015-01-17	2064.1	39001.75
2015-01-18	1976.85	40978.6
2015-01-19	2387.15	43365.75

```
-- 13. Determine the top 3 most ordered pizza types based on revenue for
each pizza category.
WITH revenue_per_pizza AS (SELECT pt.category, pt.name AS pizza_type,
SUM(od.quantity * p.price) AS total_revenue,
ROW_NUMBER() OVER (PARTITION BY pt.category
ORDER BY SUM(od.quantity * p.price) DESC) AS rn
FROM order_details od
JOIN pizzas p ON od.pizza_id = p.pizza_id
JOIN pizza_types pt ON p.pizza_type_id = pt.pizza_type_id
GROUP BY pt.category, pt.name)
SELECT category, pizza_type,
ROUND(total_revenue, 2) AS revenue
FROM revenue_per_pizza
WHERE rn <= 3
ORDER BY category, revenue DESC;
```

MySQL Workbench

The screenshot shows the MySQL Workbench interface. On the left, the 'Schemas' tab is active, showing a tree view of the 'projectdb' database with tables like 'order_details', 'orders', 'pizza_types', and 'pizzas'. The main editor window displays a SQL query (lines 106-116) that calculates the top 3 most ordered pizza types by revenue for each category. The query uses a CTE named 'revenue_per_pizza' and includes joins for 'pizzas' and 'pizza_types'. Below the query editor, the 'Result Grid' is visible, showing the output of the query. The results are organized by category and pizza type, with revenue values rounded to two decimal places.

category	pizza_type	revenue
Chicken	The Thai Chicken Pizza	43434.25
Chicken	The Barbecue Chicken Pizza	42768
Chicken	The California Chicken Pizza	41409.5
Classic	The Classic Deluxe Pizza	38180.5
Classic	The Hawaiian Pizza	32273.25
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	32265.7
Veggie	The Mexicana Pizza	26780.75
Veggie	The Five Cheese Pizza	26066.5