

```

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;

```

MySQL Workbench screenshot showing the execution of basic SQL queries. The 'SQL File 3*' tab contains the code above. The 'Result Grid' shows a single row with 'total_orders' as 21350. The 'Output' pane shows two log entries:

#	Time	Action	Message	Duration / Fetch
1	00:56:30	USE PROJECTDB	0 row(s) affected	0.000 sec
2	00:56:43	SELECT COUNT(order_id) AS total_orders FROM orders LIMIT 0, 1000	1 row(s) returned	0.031 sec / 0.000 sec

```

-- 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;

```

MySQL Workbench screenshot showing the execution of a query to calculate total revenue. The 'SQL File 3*' tab contains the code above. The 'Result Grid' shows a single row with 'total_revenue' as 817860.05. The 'Output' pane shows two log entries:

#	Time	Action	Message	Duration / Fetch
1	00:56:30	USE PROJECTDB	0 row(s) affected	0.000 sec
2	00:56:43	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;	1 row(s) returned	0.031 sec / 0.000 sec

```
-- 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

Local instance wampmysqld64 ×

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

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

Administration Schemas

Information

No object selected

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

pizza_name	highest_price
The Greek Pizza	35.95

```
-- 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

Local instance wampmysqld64

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

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

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

size	total_ordered
L	18956

No object selected

```
-- 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;
```

The screenshot shows the MySQL Workbench interface. The SQL editor tab contains the query from above. The results are displayed in a 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;
```

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Includes icons for Home, SQL, Database, Schema, Table, View, Procedure, Function, and Utilities.
- Navigator:** Shows the database schema for 'projectdb' with tables: order_details, orders, pizza_types, pizzas.
- SQL Editor:** Contains the SQL code from the previous block. Line 40 is highlighted with a blue background.
- Result Grid:** Displays the results of the query in a tabular format.

	category	total_quantity
1	Classic	14888
2	Supreme	11987
3	Veggie	11649
4	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 with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** SCHEMAS section shows the **projectdb** schema with Tables (order_details, orders, pizza_types, pizzas), Views, Stored Procedures, and Functions. It also lists **sys**.
- SQL Editor:** The query entered 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
```
- Result Grid:** The results of the query are displayed in a grid format:

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
- Bottom Navigation:** Object Info, Session, Vertical Output, 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 ×

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 ordered

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

category	total_pizzas
Veggie	27
Classic	26
Supreme	25
Chicken	18

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 with the following details:

- Title Bar:** MySQL Workbench - Local instance wampmysqld64
- Toolbar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help
- Navigator:** Shows the database schema for 'projectdb' with tables: order_details, orders, pizza_types, pizzas, Views, Stored Procedures, Functions.
- SQL Editor:** Tab titled 'PRATIKSHA_GAWADE*' containing the SQL code for question 9.
- Result Grid:** Shows the output of the query, displaying a single row with 'avg_pizzas_per_day' and its value '138.47'.
- Status Bar:** Administration, Schemas, Information, No object selected.

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;
```

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Includes icons for Home, New, Open, Save, Print, Copy, Paste, Find, Replace, and others.
- Navigator:** Shows the database schema for 'firstdb'. Under 'SCHEMAS', there is a tree view of 'projectdb' containing 'Tables' (order_details, orders, pizza_types, pizzas), 'Views', 'Stored Procedures', and 'Functions'. There is also a 'sys' entry.
- Query Editor:** The tab 'PRATIKSHA_GAWADE*' is active, showing the SQL code for question 10. The code is identical to the one provided above, with line numbers 70 through 79.
- Result Grid:** The result of the query is displayed in a grid. The columns are 'pizza_type' and 'total_revenue'. The data rows are:

pizza_type	total_revenue
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5
- Status Bar:** Shows 'Administration' and 'Schemas' tabs, and a message '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;
```

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Includes icons for SQL, DDL, Scripts, Tables, Views, Procedures, Functions, and other database management tools.
- Navigator:** Shows the schema 'projectdb' containing tables: order_details, orders, pizza_types, pizzas, Views, Stored Procedures, and Functions. A table named 'sys' is also listed.
- SQL Editor:** The tab 'PRATIKSHA_GAWADE*' is active, displaying the SQL query from the code block above. The code is numbered 79 to 90.
- Result Grid:** The results of the query are displayed in a grid format. The columns are 'pizza_type', 'pizza_revenue', and 'percentage_contribution'. The data is as follows:

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;
```

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database schema with the **projectdb** database selected, containing **Tables** (order_details, orders, pizza_types, pizzas), **Views**, **Stored Procedures**, and **Functions**.
- SQL Editor:** The tab is titled "PRATIKSHA_GAWADE*". The query is:

```
91    ORDER BY percentage_contribution DESC;
92
93    -- 12. Analyze the cumulative revenue generated over time.
94 •   SELECT daily_sales.order_date, daily_sales.daily_revenue,
95     ROUND(SUM(daily_sales.daily_revenue))
```
- Result Grid:** A table showing the cumulative revenue analysis from January 1st to 19th, 2015.

	order_date	daily_revenue	cumulative_revenue
1	2015-01-01	2713.85	2713.85
2	2015-01-02	2731.9	5445.75
3	2015-01-03	2662.4	8108.15
4	2015-01-04	1755.45	9863.6
5	2015-01-05	2065.95	11929.55
6	2015-01-06	2428.95	14358.5
7	2015-01-07	2202.2	16560.7
8	2015-01-08	2838.35	19399.05
9	2015-01-09	2127.35	21526.4
10	2015-01-10	2463.95	23990.35
11	2015-01-11	1872.3	25862.65
12	2015-01-12	1919.05	27781.7
13	2015-01-13	2049.6	29831.3
14	2015-01-14	2527.4	32358.7
15	2015-01-15	1984.8	34343.5
16	2015-01-16	2594.15	36937.65
17	2015-01-17	2064.1	39001.75
18	2015-01-18	1976.85	40978.6
19	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. The SQL editor tab is active, displaying the query from above. The results grid below shows the output of the query:

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