

1a. This SQL Statement shows how many customers have received strikes.

```
SELECT COUNT(DISTINCT rental.customer_ID) AS CUSTOMERS from strike
JOIN rental
ON rental.booking_id = strike.rental_id;
```

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

- Query Editor:** Displays the SQL query:

```
1 • SELECT COUNT(DISTINCT rental.customer_ID) AS CUSTOMERS from strike
2   JOIN rental
3     ON rental.booking_id = strike.rental_id;
```
- Result Grid:** Shows the result of the query:

CUSTOMERS
24
- Output Window:** Shows the execution log:

Action Output
Time Action Message Duration / Fetch
102 23:22:26 DEALLOCATE PREPARE stmt OK 0.000 sec
103 23:22:38 SELECT COUNT(DISTINCT rental.customer_ID) AS CUSTOMERS from strike JOIN rental ON rental.booking_id = strike.rental_id LIMIT 0, 1000 1 rows(s) returned 0.000 sec / 0.000 sec
- Session Tab:** Shows the session details: Object Info (selected), Session, 11:23 PM, 4/26/2025.

1b. This SQL Statement shows the percentage of customers that have received strikes.

```

SELECT (COUNT(DISTINCT rental.customer_ID) / (SELECT COUNT(*) FROM abc.customer)) * 100
AS `Strike Percentage`
FROM strike
JOIN abc.rental ON strike.rental_id = rental.booking_id;

```

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

- Query Editor:** Displays the SQL query:

```

1 • SELECT (COUNT(DISTINCT rental.customer_ID) / (SELECT COUNT(*) FROM abc.customer)) * 100
2 AS `Strike Percentage`
3 FROM strike
4 JOIN abc.rental ON strike.rental_id = rental.booking_id;
5

```
- Result Grid:** Shows the output of the query:

Strike Percentage
48.0000
- Session Tab:** Shows the execution history:

Action Output	Time	Action	Message	Duration / Fetch
	103	23:22:38	SELECT COUNT(DISTINCT rental.customer_ID) AS CUSTOMERS from strike JOIN rental ON rental.booking_id = strike.rental_id LIMIT 0, 1000	0.000 sec / 0.000 sec
	104	23:26:01	1 row(s) returned	0.000 sec / 0.000 sec
			SELECT (COUNT(DISTINCT rental.customer_ID) / (SELECT COUNT(*) FROM abc.customer)) * 100 AS `Strike Percentage` FROM abc.strike JOIN abc.rental ON strike.rental_id = rental.booking_id;	1 row(s) returned
- Object Info Tab:** Shows information about the 'strike' table.
- Session Tab:** Shows session details: Read Only, 11:26 PM, 4/26/2025.

1c. This SQL Statement shows the average number of strikes.

```
SELECT
```

```
    AVG(strike_count) AS average
```

```
FROM (
```

```
    SELECT rental.customer_ID, COUNT(*) AS strike_count
```

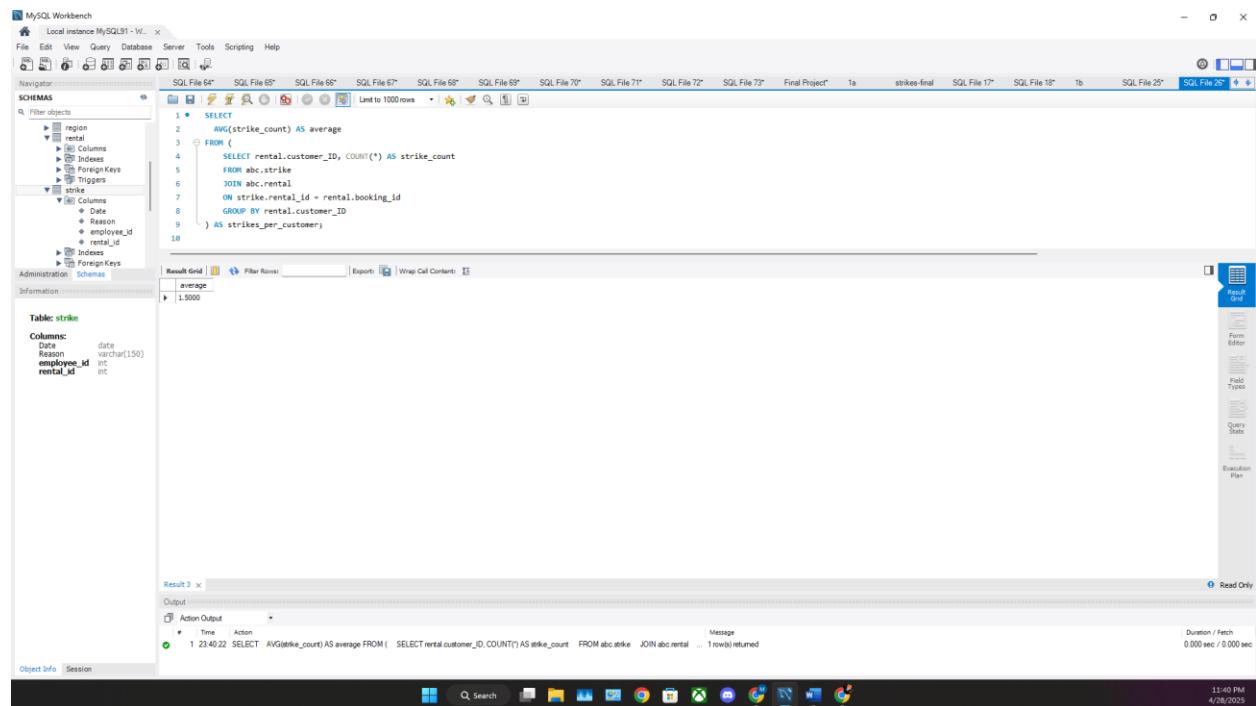
```
        FROM abc.strike
```

```
        JOIN abc.rental
```

```
            ON strike.rental_id = rental.booking_id
```

```
        GROUP BY rental.customer_ID
```

```
) AS strikes_per_customer;
```



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

- Query Editor:** Displays the SQL query for calculating the average number of strikes per customer. The query uses a subquery to count strikes for each customer and then calculates the average.
- Result Grid:** Shows the output of the query, which is a single row with the value 1.5000.
- Object Info:** Shows information about the 'strike' table, including its columns: Date (date), Reason (varchar(150)), employee_id (int), and rental_id (int).
- Session Tab:** Shows the session details, including the time of the query execution (23:40:22) and the duration (0.000 sec / 0.000 sec).

1d. This SQL Statement shows how many customers have 3 or more strikes.

```
SELECT CONCAT(customer.firstName, customer.lastName) AS 'Full Name',
count(rental.customer_ID) AS rcount from strike
```

JOIN rental

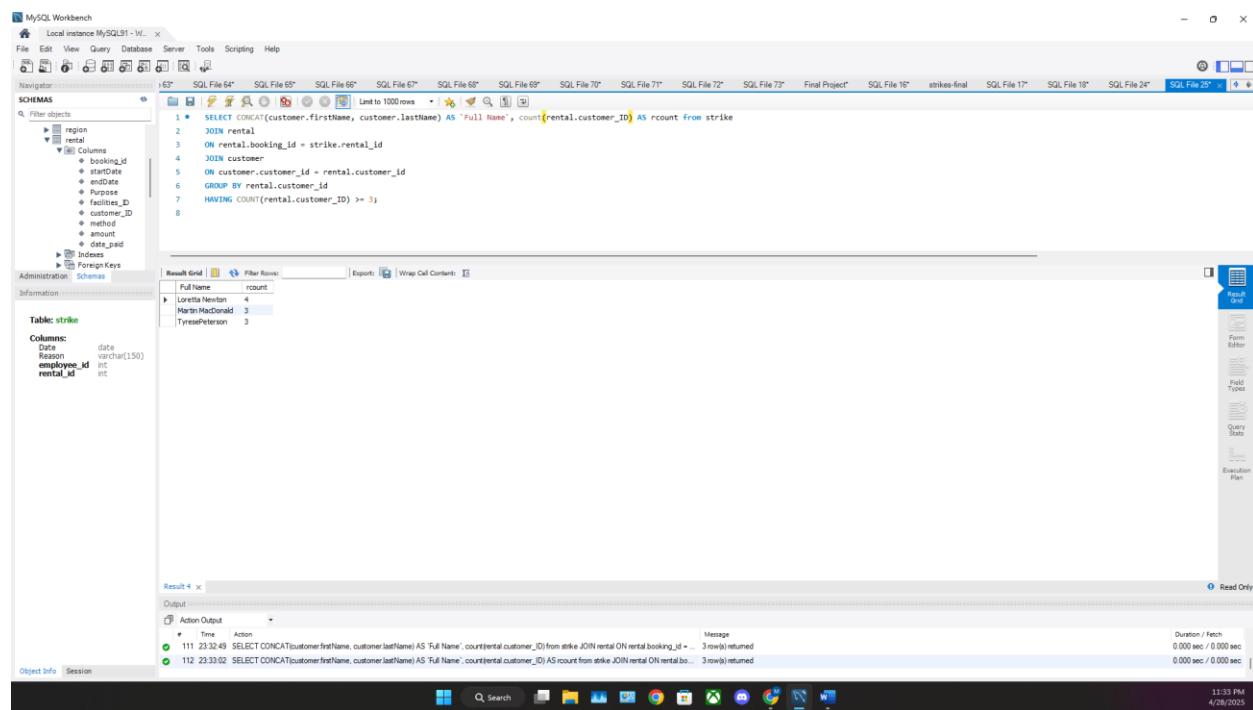
ON rental.booking_id = strike.rental_id

JOIN customer

ON customer.customer_id = rental.customer_id

GROUP BY rental.customer_id

HAVING COUNT(rental.customer_ID) >= 3;



The screenshot shows the MySQL Workbench interface. The SQL editor pane contains the following SQL query:

```
1 • SELECT CONCAT(customer.firstName, customer.lastName) AS 'Full Name',
2   count(rental.customer_ID) AS rcount from strike
3   JOIN rental
4   ON rental.booking_id = strike.rental_id
5   JOIN customer
6   ON customer.customer_id = rental.customer_id
7   GROUP BY rental.customer_id
8   HAVING COUNT(rental.customer_ID) >= 3;
```

The results grid below the editor shows the output of the query:

Full Name	rcount
Loretta Newton	4
Martin MacDonald	3
Tyrese Peterson	3

The status bar at the bottom right indicates the execution time: 11:33 PM 4/26/2015.

1e. This SQL Statement shows the most common reasons for strikes ordered by most common to least.

```
SELECT strike.Reason FROM customer
JOIN rental
ON rental.customer_id = customer.customer_id
JOIN strike
ON strike.rental_id = rental.booking_id
GROUP BY strike.Reason
ORDER BY count(customer.customer_id) DESC;
```

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

- Query Editor:** Displays the SQL query:

```
1 • SELECT strike.Reason FROM customer
2 JOIN rental
3 ON rental.customer_id = customer.customer_id
4 JOIN strike
5 ON strike.rental_id = rental.booking_id
6 GROUP BY strike.Reason
7 ORDER BY count(customer.customer_id) DESC;
```
- Schemas Navigator:** Shows the database schema with tables: region, rental, customer, strike, and booking.
- Table Information:** Shows the structure of the 'strike' table with columns: date, Reason, employee_id, and rental_id.
- Result Grid:** Shows the output of the query, listing reasons for strikes:

Reason
Noise Violation
Violated Max Capacity
Broken Furniture
Started a Fire
Damaged Equipment
Clogged Toilet
Flooded Bathroom
Broken Sink
Broken Window
Broken Front Door
Stained Carpet
- Action Output:** Shows the log of actions taken by the session:

Action	Time	Message
122	23:37:01	SELECT strike.Reason FROM customer JOIN rental ON rental.customer_id = customer.customer_id JOIN strike ON strike.rental_id = rental.booking_id - 2 rows(s) returned
123	23:37:25	SELECT strike.Reason FROM customer JOIN rental ON rental.customer_id = customer.customer_id JOIN strike ON strike.rental_id = rental.booking_id - 11 rows(s) returned

1f. This SQL Statement shows the customer name, facility name, state, strike date, and Employee name where the employee reported Max Capacity, sorted by customer last name

```

SELECT CONCAT(customer.firstName, ',',customer.lastName) AS 'Cust Name',
facility.'Name', facility.state, strike.'date' AS 'strike_date', CONCAT(employee.firstName, ',',
employee.lastName) AS 'Emp Name'

FROM Customer

JOIN rental

ON rental.customer_ID = customer.customer_id

JOIN facility

ON facility.Facilities_ID = rental.facilities_ID

JOIN Strike

ON strike.rental_id = rental.booking_id

JOIN employee

ON employee.employee_id = strike.employee_id

WHERE strike.reason = 'Violated Max Capacity'

ORDER BY customer.lastName;

```

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

- Query Editor:** Displays the complete SQL query from step 1f.
- Result Grid:** Shows the output of the query, listing customers, facilities, strike dates, and employee names. The data includes:

Cust Name	Name	state	strike_date	Emp Name
Cheyenne Connors	Zan Garden	MD	2020-02-11	Morgan Turner
Cheyenne Connors	Blue Roof Inn	NY	2012-09-28	Shawn Coleman
Adam East	Cold Shallows	MD	2021-09-15	Zackery Pratt
Martin McDonald	East Apartments	MD	2016-10-15	Zackery Pratt
Loretta Newton	Pine Cabin	NY	2019-07-23	Bille Simms
Derrick Parker	Queens' Bedroom	NY	2021-02-27	Della Young
Tyrese Peterson	Red Dog Inn	MA	2012-06-02	Kevin Faulkner
Roger Yung	Blue Roof Inn	MA	2012-09-28	Shaun Coleman

- Object Info:** Shows the definition of the 'rental' table.

1g. This SQL statement shows the Customer name, state, facility name, state, rental start date, and the purpose of renting during the year 2022 where the customer and facility state aren't the same and the purpose of renting is either visiting friends or vacationing.

```
SELECT CONCAT(customer.firstName, ' ', customer.lastName) AS 'Cust Name', customer.state,
facility.'Name', facility.state, rental.startDate, rental.purpose
```

FROM Customer

JOIN rental

ON rental.customer_ID = customer.customer_id

JOIN facility

ON facility.Facilities_ID = rental.facilities_ID

WHERE (YEAR(startDate) = 2022) AND

(customer.state != facility.state) AND

(purpose = 'Visiting friends' OR purpose = 'Vacation');

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

- SQL Editor:** Contains the complete SQL query from the previous text.
- Result Grid:** Displays the query results in a tabular format. The columns are Cust Name, state, Name, state, startDate, and purpose. The data rows are:

Cust Name	state	Name	state	startDate	purpose
Cheyenne Connors	VA	Factory #63165	NY	2022-05-18	Visiting Friends
Adam East	PA	West Storage	MA	2022-11-08	Visiting Friends
Cheyenne Connors	VA	Queen's Ballroom	NY	2022-10-30	Vacation
Evangeline Vassiere	FL	Resort International	MA	2022-10-07	Visiting friends
- Object Info:** Shows the definition of the 'customer' table.
- Session:** Displays the current session information.
- System Bar:** Shows the system tray with icons for various applications like Task View, Search, File Explorer, and others.

2a.

This query shows all the maintenance performed at the facilities since the beginning of 2020. It shows which facilities (name) required maintenance, what type of maintenance or repair (description) was performed as well as date when each repair was performed.

SELECT

```
f.Name as name,  
m.mainDate AS date_performed,  
m.description  
FROM maintenance m  
JOIN facility f ON m.facilityID = f.ID  
ORDER BY m.mainDate;
```

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

- File Menu:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Standard MySQL icons for connection, schema selection, and query execution.
- Navigator:** Shows the database schema with the current schema set to **abc_company**.
- Query Editor:** Titled "Query 1", contains the SQL code provided above.
- Result Grid:** Displays the query results in a tabular format. The columns are **name**, **date_performed**, and **description**. The data includes:

name	date_performed	description
Best Place Inn	1993-10-14	Flood Damage
Yoda Resort	1996-02-13	Repair TV
Red Dog Inn	1996-03-16	Expand facility
East Hotel	2001-03-12	Repair walls
Yallah Place	2009-09-21	Fire Damage
Queen's Ballroom	2010-01-28	Clean Facilities
Yats	2010-02-18	Move Furniture
Blue Oasis	2010-08-18	Clean Facilities
Pine Cabin	2011-02-24	Move Furniture
Yards	2011-05-16	Broken Windows
Old Town Inn	2011-11-28	Repair Sink

```
Read Only
```
- Output:** Shows the action log with two entries:# Time Action Message Duration / Fetch
68 18:40:36 SELECT f.Name, m.mainDate AS pe... 50 row(s) returned 0.000 sec / 0.000 sec
69 18:41:03 SELECT f.Name as name, m.mainDate AS... 50 row(s) returned 0.016 sec / 0.000 sec
- Object Info:** Tab for viewing object information.
- Session:** Tab for viewing session details.

2b.

This query shows the top five types of maintenance and their frequency. It identifies the most common types of maintenance or repair that are being performed across all facilities as well as the frequency of the maintenance.

```
SELECT description, COUNT(*) AS frequency
```

```
FROM maintenance
```

```
GROUP BY description
```

```
ORDER BY frequency DESC
```

```
LIMIT 5;
```

The screenshot shows the MySQL Workbench interface. The Navigator pane on the left displays the schema 'abc_company' with tables like department, customer, category, facility, employee, maintenance, rental, region, strike, and views. The Query Editor pane at the top contains the following SQL code:

```
3     m.mainDate AS date_performed,
4     m.description
5   FROM maintenance m
6   JOIN facility f ON m.facilityID = f.ID
7   ORDER BY m.mainDate;
8
9
10 •  SELECT description, COUNT(*) AS frequency
11   FROM maintenance
12   GROUP BY description
13   ORDER BY frequency DESC
14   LIMIT 5;
```

The Result Grid pane below shows the output of the last query:

description	frequency
Clean Facilities	15
Move Furniture	6
Broken Windows	4
Replace Wallpaper	3
Replace Furniture	3

The Status bar at the bottom indicates the execution time and duration for each action.

2 c.

This query shows the employees who have performed more than three maintenance jobs and their salaries. It shows the employees first and last name, salary and the number of times each of them has performed maintenance.

SELECT

e.firstName,

e.lastName,

e.salary,

COUNT(m.ID) AS nbr

FROM maintenance m

JOIN employee e ON m.opsEmpID = e.ID

GROUP BY e.ID, e.firstName, e.lastName, e.salary

HAVING COUNT(m.ID) > 3

ORDER BY nbr DESC;

```

MySQL Workbench - IST 210
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS abc_company
Tables
department
customer
category
facility
employee
maintenance
rental
region
strike
Views
Stored Procedures
Functions
employee
hr
kumar_trees
kumarmoviesfull
moviesbasic
sakila
sakiladb
Information Schemas
Schema: abc_company

Query 1
14    LIMIT 5;
15
16
17 •   SELECT
18     e.firstName,
19     e.lastName,
20     e.salary,
21     COUNT(m.ID) AS nbr
22   FROM maintenance m
23   JOIN employee e ON m.opsEmpID = e.ID
24   GROUP BY e.ID, e.firstName, e.lastName, e.salary
25   HAVING COUNT(m.ID) > 3

Result Grid | Filter Rows: Export: Wrap Cell Content: 
firstName lastName salary nbr
Brandon Chen 45355 7
Shaun Coleman 106109 5
Zackery Pratt 73595 5
Morgan Turner 48676 4
Yvette Wang 84021 4
Celeste Carrillo 99378 4

Action Output
# Time Action Message Duration / Fetch
71 18:46:10 SELECT e.firstName, e.lastName, e.salary, COUNT(m.ID) AS nbr 6 row(s) returned 0.016 sec / 0.000 sec
72 18:46:53 SELECT e.firstName, e.lastName, e.salary, COUNT(m.ID) AS nbr 6 row(s) returned 0.000 sec / 0.000 sec

```

2d.

This query shows the total and average salary of the maintenance employees by year since 2018.

It analyzes the salary expenses related to the maintenance employees by year as the results show the total and average salary.

SELECT

(m.mainDate, 1, 4) AS 'year performed',

SUM(e.salary) AS total,

AVG(e.salary) AS avg

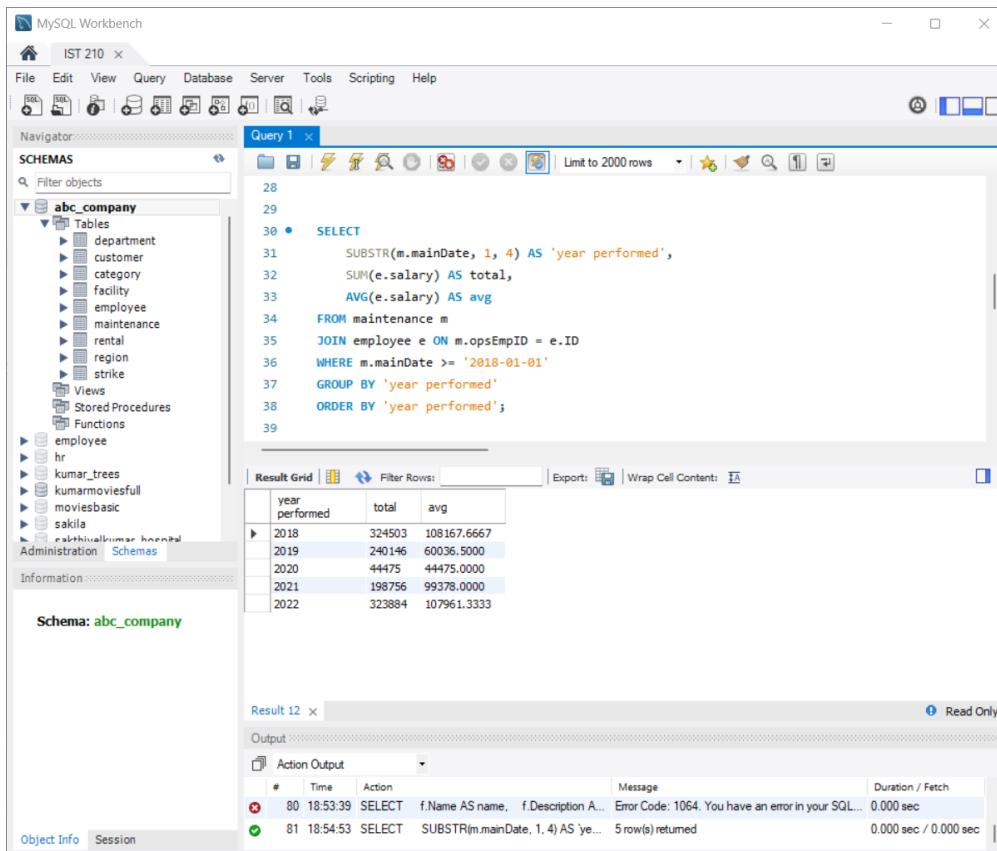
FROM maintenance m

```
JOIN employee e ON m.opsEmpID = e.ID
```

```
WHERE m.mainDate >= '2018-01-01'
```

```
GROUP BY 'year performed'
```

```
ORDER BY 'year performed';
```



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

- Navigator:** Shows the schema `abc_company` with tables: department, customer, category, facility, employee, maintenance, rental, region, strike.
- Query Editor (Query 1):** Contains the following SQL code:

```
28
29
30 •  SELECT
31     SUBSTR(m.mainDate, 1, 4) AS 'year performed',
32     SUM(e.salary) AS total,
33     AVG(e.salary) AS avg
34   FROM maintenance m
35   JOIN employee e ON m.opsEmpID = e.ID
36 WHERE m.mainDate >= '2018-01-01'
37 GROUP BY 'year performed'
38 ORDER BY 'year performed';
```
- Result Grid:** Displays the results of the query:

year performed	total	avg
2018	324503	108167.6667
2019	240146	60036.5000
2020	44475	44475.0000
2021	198756	99378.0000
2022	323884	107961.3333
- Output Window (Result 12):** Shows the log of actions taken:

#	Time	Action	Message	Duration / Fetch
80	18:53:39	SELECT f.Name AS name, f.Description AS...	Error Code: 1064. You have an error in your SQL... 0.000 sec	
81	18:54:53	SELECT SUBSTR(m.mainDate, 1, 4) AS 'ye...	5 row(s) returned	0.000 sec / 0.000 sec

2e.

This query shows the facility type that receives the least amount maintenance. It shows the type of facility and the number of times it receives maintenance ordered by the amount to see the least amount.

```
SELECT  
    c.Name AS name,  
    COUNT(m.ID) AS amount  
FROM category c  
LEFT JOIN facility f ON c.Category_ID = f.`Category ID`  
LEFT JOIN maintenance m ON f.ID = m.facilityID  
GROUP BY c.Name  
ORDER BY amount ASC;
```

```

41
42
43     SELECT
44         c.Name AS name,
45         COUNT(m.ID) AS amount
46     FROM category c
47     LEFT JOIN facility f ON c.Category_ID = f.`Category ID`
48     LEFT JOIN maintenance m ON f.ID = m.facilityID
49     GROUP BY c.Name
50
51     ORDER BY amount ASC;
52

```

name	amount
Home	11
Industrial	11
Other	11
Recreation	17

Result 10 x

Output

#	Time	Action	Message	Duration / Fetch
77	18:50:32	SELECT	c.Name AS category_name, CO... 4 row(s) returned	0.016 sec / 0.000 sec
78	18:51:07	SELECT	c.Name AS name, COUNT(m.ID... 4 row(s) returned	0.000 sec / 0.000 sec

2f.

This query shows a comprehensive overview of each facility. It shows the name of the facility, the type, category and number of bookings.

SELECT

f.Name AS name,

f.Description AS description,

c.Name AS category,

```
COUNT(DISTINCT m.ID) AS log  
  
FROM  
  
facility f  
  
LEFT JOIN  
  
category c ON f.`Category ID` = c.Category_ID  
  
LEFT JOIN  
  
maintenance m ON f.ID = m.facilityID  
  
GROUP BY  
  
f.Name, f.Description, c.Name  
  
ORDER BY  
  
f.Name;
```

```

66     c.Name AS category,
67     COUNT(DISTINCT m.ID) AS log
68   FROM
69     facility f
70   LEFT JOIN
71     category c ON f.`Category ID` = c.Category_ID
72   LEFT JOIN
73     maintenance m ON f.ID = m.facilityID
74   GROUP BY
75     f.Name, f.Description, c.Name
76   ORDER BY
77     f.Name;

```

Result Grid:

name	description	category	log
Best Place Inn	Hotel	Industrial	1
Blue Oasis	Resort	Recreation	3
Blue Roof Inn	Hotel	Industrial	1
Cabin	Home	Home	1
Coastal Hotel	Hotel	Other	2
Cold Shallows	Industrial	Industrial	0
Deerwood Inn	Hotel	Industrial	0
Desert Resort	Resort	Recreation	1
East Apartments	Home	Home	3
East Hotel	Hotel	Other	1
Factorv #65165	Industrial	Industrial	2

Output:

#	Time	Action	Message	Duration / Fetch
82	18:56:34	SELECT	f.Name AS facility_name, f.Descri... 35 row(s) returned	0.000 sec / 0.000 sec
83	18:58:20	SELECT	f.Name AS name, f.Description A... 35 row(s) returned	0.000 sec / 0.000 sec

3a. This query shows the customer that spent the most in rentals, which was Nicole Maynard. Her properties are located in Boston and New York. This was in 1993, 2010, 2014, and 2016. The facility names are Best Place Inn, Queen's Ballroom, High View Apartments, and Pine Cabin. Code: SELECT c.customer_ID AS CUSTID, c.firstName, c.lastName, c.city, c.state, f.Name AS 'Facility Name', f.city AS city, f.State AS state, b.startDate AS begDate, b.endDate FROM payment-final p JOIN booking-final b ON p.booking_id = b.booking_id JOIN customer c ON b.customer_ID = c.customer_id JOIN Facilities-final f ON b.facilities_ID = f.Facilities_ID WHERE c.customer_ID = (SELECT customer_id FROM payment-final GROUP BY customer_id ORDER BY SUM(amount) DESC LIMIT 1) ORDER BY b.startDate;

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

```

8
9
10 •  SELECT c.customer_ID AS CUSTID, c.firstName, c.lastName, c.city, c.state, f.Name AS 'Facility Name', f.city AS city, f.State AS state,
11      b.startDate AS begDate, b.endDate
12  FROM `payment-final` p
13  JOIN `booking-final` b ON p.booking_id = b.booking_id
14  JOIN customer c ON b.customer_ID = c.customer_id
15  JOIN `Facilities-final` f ON b.facilities_ID = f.Facilities_ID
16  WHERE c.customer_ID = (
17      SELECT customer_id
18      FROM `payment-final`
19      GROUP BY customer_id
20      ORDER BY SUM(amount) DESC
21      LIMIT 1
22  )
23  ORDER BY b.startDate;
24

```

The results grid shows three rows of data:

CUSTID	firstName	lastName	city	state	Facility Name	city	state	begDate	endDate
102	Nicole	Maynard	San Francisco	California	Best Place Inn	Boston	MA	1993-10-11	1993-10-12
102	Nicole	Maynard	San Francisco	California	Queen's Ballroom	New York	NY	2010-01-26	2010-01-27
102	Nicole	Maynard	San Francisco	California	High View Apartments	New York	NY	2014-09-18	2018-07-24
102	Nicole	Maynard	San Francisco	California	Pine Cabin	Albany	NY	2016-12-19	2016-12-27

The status bar at the bottom indicates "Query Completed".

3b. The customer who has the most rentals and the amount is Loretta Newton. Code: SELECT c.customer_ID AS CUSTID, c.firstName, c.lastName, c.city, c.state, COUNT(b.booking_id) AS totals, SUM(p.amount) AS Amount FROM payment-final p JOIN booking-final b ON p.booking_id = b.booking_id JOIN customer c ON b.customer_ID = c.customer_id JOIN Facilities-final f ON b.facilities_ID = f.Facilities_ID WHERE c.customer_ID = (SELECT customer_ID FROM abc.booking-final GROUP BY customer_ID ORDER BY COUNT(*) DESC LIMIT 1) GROUP BY c.customer_ID;

The screenshot shows the MySQL Workbench interface. The top menu bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. The title bar says "MySQL Workbench". The main window has tabs for Administration, Schemas, and SQL. The SQL tab contains the following code:

```

25
26
27
28 •   SELECT c.customer_ID AS CUSTID, c.firstName, c.lastName, c.city, c.state, COUNT(b.booking_id) AS totals,
29     SUM(p.amount) AS Amount
30   FROM `payment-final` p
31   JOIN `booking-final` b ON p.booking_id = b.booking_id
32   JOIN customer c ON b.customer_ID = c.customer_id
33   JOIN `Facilities-final` f ON b.facilities_ID = f.Facilities_ID
34   WHERE c.customer_ID =
35     (SELECT customer_ID
36      FROM `abc`.`booking-final`
37      GROUP BY customer_ID
38      ORDER BY COUNT(*) DESC
39      LIMIT 1
40   )
41   GROUP BY c.customer_ID;

```

The results grid shows the following data:

CUSTID	firstName	lastName	city	state	totals	Amount
101	Loretta	Newton	New York	New York	5	2715

Below the results, there is a log table with two entries:

Action Output	Time	Action	Response	Duration / Fetch Time
37	11:47:50	SELECT c.customer_ID AS CUSTID, c.firstName, c.lastName, c.city, c.state, COUNT(b.booking_id) AS totals,...	Error Code: 1140. In aggregated query without GROU...	0.0012 sec
38	11:48:13	SELECT c.customer_ID AS CUSTID, c.firstName, c.lastName, c.city, c.state, COUNT(b.booking_id) AS totals,...	1 row(s) returned	0.0022 sec / 0.00001...

The status bar at the bottom indicates "Read Only".

3c. This shows the top 10 fiscal years total earnings, with 2022 being the highest, and 2011 being the lowest. Code: `SELECT YEAR(r.begDate) AS FiscalYear, SUM(r.amount) AS Amount`

`FROM `rental` r`

`GROUP BY FiscalYear`

`ORDER BY Amount DESC`

LIMIT 10;

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
ORDER BY COUNT(*) DESC
LIMIT 1
)
GROUP BY c.customer_ID;
43
44 •   SELECT YEAR(r.begDate) AS FiscalYear, SUM(r.amount) AS Amount
45   FROM `rental` r
46   GROUP BY FiscalYear
47   ORDER BY Amount DESC
48   LIMIT 10;
```

The result grid displays the following data:

FiscalYear	Amount
2022	12357.00
2014	5995.00
2018	5350.00
2021	5090.00
2001	4740.00
2012	4321.00
2016	3810.00
2015	3625.00
2010	2866.00
2011	2691.00

The status bar at the bottom indicates "Query Completed".

3d. This shows the average payment in those top 10 fiscal years. Code: SELECT

```
YEAR(r.begDate) AS FiscalYear, SUM(r.amount) AS Amount, AVG(r.amount) AS
average_payment FROM rental r JOIN ( SELECT YEAR(r2.begDate) AS top_year FROM rental
r2 GROUP BY YEAR(r2.begDate) ORDER BY SUM(r2.amount) DESC LIMIT 10 ) top_years
ON YEAR(r.begDate) = top_years.top_year GROUP BY FiscalYear ORDER BY Amount
DESC;
```

MySQL Workbench

Local instance 3306

MySQL Workbench

Administration Schemas SQL File 13* SQL File 10* SQL File 14* SQL File 15* SQL File 16* SQL File 17* SQL File 20* SQL File 22* SQL File 24* SQL File 11* Context... Snippets

SCHEMAS Filter objects

department employee... Facilities-fi... Maintenance... payment-fi...

Columns method amount Date bookin... custo... begDate

Indexes Foreign... Triggers Region Rental

Columns id

Object Info Session Table: payment-final

Result Grid Filter Rows: Search Export: Result Grid Form Editor Field Notes

Read Only

Query Completed

```

50
51 • SELECT YEAR(r.begDate) AS FiscalYear, SUM(r.amount) AS Amount, AVG(r.amount) AS average_payment
52 FROM `rental` r
53 JOIN (
54     SELECT
55         YEAR(r2.begDate) AS top_year
56     FROM `rental` r2
57     GROUP BY YEAR(r2.begDate)
58     ORDER BY SUM(r2.amount) DESC
59     LIMIT 10
60 ) top_years ON YEAR(r.begDate) = top_years.top_year
61 GROUP BY FiscalYear
62 ORDER BY Amount DESC;
63

```

FiscalYear	Amount	average_payment
2022	12357.00	882.642857
2014	5995.00	999.166667
2018	5350.00	1783.333333
2021	5450.00	891.666667
2001	4701.00	2350.500000
2012	4321.00	1080.250000
2016	3810.00	1270.000000
2015	3625.00	906.250000
2010	2866.00	716.500000
2011	2691.00	897.000000

Action Output

Action	Time	Action	Response	Duration / Fetch Time
152	14:58:41	SELECT YEAR(r.begDate) AS FiscalYear, SUM(r.amount) AS Amount, AVG(r.amount) AS average_payment FRO...	10 row(s) returned	0.0016 sec / 0.00000...
153	14:58:51	SELECT YEAR(r.begDate) AS FiscalYear, SUM(r.amount) AS Amount, AVG(r.amount) AS average_payment FRO...	10 row(s) returned	0.0015 sec / 0.00000...

3e. This shows how much money each customer has spent on bookings. Code: SELECT c.firstName, c.lastName, SUM(p.amount) AS TotalAmt FROM customer c JOIN booking-final b ON c.customer_id = b.customer_ID JOIN payment-final p ON b.booking_id = p.booking_id GROUP BY c.customer_id, c.firstName, c.lastName ORDER BY c.lastName;

MySQL Workbench

Local instance 3306

MySQL Workbench

Administration Schemas SQL File 13* SQL File 10* SQL File 14* SQL File 15* SQL File 16* SQL File 17* SQL File 20* SQL File 22* SQL File 24* SQL File 11* Context... Snippets

Filter objects

SCHEMAS

Date bookin... custo... Indexes Foreign... Triggers Region Rental Columns Id begDate endDate Purpose faciliti... custo... method amount date_p... Indexes

Object Info Session Table: payment-final

Columns:

method	varchar(45)
amount	int
Date	date
booking_id	int
customer_id	int

Action Output

Time Action Response Duration / Fetch Time

161 15:10:45 SELECT c.firstName, c.lastName, SUM(p.amount) AS TotalAmt FROM customer c JOIN `booking-final` b ON c.c... 40 row(s) returned 0.0026 sec / 0.00001...

162 15:10:53 SELECT c.firstName, c.lastName, SUM(p.amount) AS TotalAmt FROM customer c JOIN `booking-final` b ON c.c... 40 row(s) returned 0.0016 sec / 0.00001...

Result Grid Filter Rows: Search Export: Result Grid Form Editor Field Tools Read Only

Query Completed

100% 21:70

Result Grid Filter Rows: Search Export: Result Grid Form Editor Field Tools Read Only

firstName lastName TotalAmt

Samuel	Adamson	520
Melissa	Anderson	2752
Garrett	Ashley	1200
Sylvia	Baldred	940
Susan	Blake	540
Robert	Bloom	2400
Connor	Bridgerton	962
Rene	Cain	125
Cheyenne	Connors	875
Janelle	Davis	350

3f. This shows the total rental amount of each facility category in each region, with the industrial facility in MA being the highest. Code: `SELECT reg.HQ_State AS hq_state, cat.Name AS name, SUM(r.amount) AS TotalProfits FROM rental r JOIN Facilities-final f ON r.facilities_ID = f.Facilities_ID JOIN Category cat ON f.CategoryID = cat.Category_ID JOIN Region reg ON f.Region = reg.RegionID GROUP BY cat.Name, reg.HQ_State ORDER BY hq_state, name;`

```

    JOIN `payment-final` p ON b.booking_id = p.booking_id
    GROUP BY c.customer_id, c.firstName, c.lastName
    ORDER BY c.lastName;

68   JOIN `payment-final` p ON b.booking_id = p.booking_id
69   GROUP BY c.customer_id, c.firstName, c.lastName
70   ORDER BY c.lastName;

71
72
73
74 •   SELECT reg.HQ_State AS hq_state, cat.Name AS name, SUM(r.amount) AS TotalProfits
75   FROM rental r
76   JOIN `Facilities-final` f ON r.facilities_ID = f.Facilities_ID
77   JOIN Category cat ON f.CategoryID = cat.Category_ID
78   JOIN Region reg ON f.Region = reg.RegionID
79   GROUP BY cat.Name, reg.HQ_State
80   ORDER BY hq_state, name;

```

Result Grid

hq_state	name	TotalProfits
MA	Industrial	12079.00
MA	Other	5881.00
MA	Recreation	7912.00
MD	Home	1750.00
MD	Industrial	807.00
MD	Other	5415.00
MD	Recreation	8264.00
NY	Home	6778.00
NY	Industrial	4696.00
NY	Other	7730.00

Object Info Session

Table: payment-final

Columns:

- method varchar(45)
- amount int
- Date date
- booking_id int
- customer_id int

Action Output

Action	Time	Response	Duration / Fetch Time
172	15:20:08	SELECT reg.HQ_State AS hq_state, cat.Name AS name, SUM(r.amount) AS TotalProfits FROM rental r JOIN `Facilities-final` f ON r.facilities_ID = f.Facilities_ID JOIN Category cat ON f.CategoryID = cat.Category_ID JOIN Region reg ON f.Region = reg.RegionID WHERE f.State = reg.HQ_State GROUP BY reg.HQ_State, cat.Name ORDER BY TotalProfits DESC;	Error Code: 1054. Unknown column 'category_name' in 'on clause' 0.00075 sec
173	15:20:22	SELECT reg.HQ_State AS hq_state, cat.Name AS name, SUM(r.amount) AS TotalProfits FROM rental r JOIN `Facilities-final` f ON r.facilities_ID = f.Facilities_ID JOIN Category cat ON f.CategoryID = cat.Category_ID JOIN Region reg ON f.Region = reg.RegionID WHERE f.State = reg.HQ_State GROUP BY reg.HQ_State, cat.Name ORDER BY TotalProfits DESC;	11 row(s) returned 0.0022 sec / 0.00000...

Query Completed

3g. This shows the profit of the rental facility where the location is the same as the headquarters.

Code: `SELECT reg.HQ_State AS hq_state, reg.HQ_City AS hq_city, SUM(r.amount) AS Amount FROM rental r JOIN Facilities-final f ON r.facilities_ID = f.Facilities_ID JOIN Region reg ON f.Region = reg.RegionID WHERE f.State = reg.HQ_State GROUP BY reg.HQ_State, reg.HQ_City ORDER BY Amount DESC;`

MySQLWorkbench File Edit View Query Database Server Tools Scripting Help

MySQL Workbench

Local instance 3306

Administration Schemas SQL File 13* SQL File 10* SQL File 14* SQL File 15* SQL File 16* SQL File 17* SQL File 20* SQL File 22* SQL File 24* SQL File 11* Context... Snippets

Schemas

payment-n... Columns method amount Date booking_id custo... Indexes Foreign... Triggers Region Columns Region... HQ_St... HQ_City Indexes Foreign... Triggers Rental

Object Info Session

Table: payment-final

Columns:

method	varchar(45)
amount	int
Date	date
booking_id	int
customer_id	int

JOIN `booking-final` b ON p.booking_id = b.booking_id
JOIN `Facilities-final` f ON b.facilities_ID = f.Facilities_ID
JOIN Region r ON f.Region = r.RegionID
WHERE f.State = r.HQ_State;
SELECT reg.HQ_State AS hq_state, reg.HQ_City AS hq_city, SUM(r.amount) AS Amount
FROM rental r
JOIN `Facilities-final` f ON r.facilities_ID = f.Facilities_ID
JOIN Region reg ON f.Region = reg.RegionID
WHERE f.State = reg.HQ_State
GROUP BY reg.HQ_State, reg.HQ_City
ORDER BY Amount DESC;

Result Grid

hq_state	hq_city	Amount
NY	New York City	30220.00
MA	Boston	25872.00
MD	Baltimore	12229.00

Result 55

Action Output Response Duration / Fetch Time

176	15:32:20	SELECT reg.HQ_State AS hq_state, reg.HQ_City AS hq_city, SUM(r.amount) AS Amount FROM rental r JOIN `Fa... Error Code: 1140. In aggregated query without GROUP BY, expression 'reg.HQ_City' is not contained in either an aggregate function or the GROUP BY clause.	0.00071 sec
177	15:34:41	SELECT reg.HQ_State AS hq_state, reg.HQ_City AS hq_city, SUM(r.amount) AS Amount FROM rental r JOIN `Fa... 3 row(s) returned	0.0018 sec / 0.00000...

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.