

DATABASE ASSIGNMENT

28, May, 2025

- Jahnavi Lakshmi Muttireddy

1. Create the tables below in the database. Use foreign keys and primary keys as required.
 - a. Create a table called as student with the following columns student_id, first_name, last_name ,birthdate ,department_id ,address_id .
 - b. Create Address table with following columns address_id , street_address, city, State, postal_code
 - c. Create department table department_id, department name. Make sure you are using the right data type against all the columns.

The screenshot shows two instances of MySQL Workbench. Both instances have the same interface setup with the 'assignment' schema selected. In the top instance, the SQL editor contains the following SQL code:

```
1 USE assignment;
2
3 • CREATE TABLE address (
4   address_id INT PRIMARY KEY AUTO_INCREMENT,
5   street_address VARCHAR(100) NOT NULL,
6   city VARCHAR(50) NOT NULL,
7   state VARCHAR(50) NOT NULL,
8   postal_code VARCHAR(20) NOT NULL
9 );
10
11 • CREATE TABLE department(
12   department_id INT PRIMARY KEY AUTO_INCREMENT,
13   department_name VARCHAR(50) NOT NULL
14 );
15
16 • CREATE TABLE student(
17   student_id INT PRIMARY KEY AUTO_INCREMENT,
18   first_name VARCHAR(50) NOT NULL,
19   last_name VARCHAR(50) NOT NULL,
20   birthdate DATE NOT NULL,
21   department_id INT,
22   address_id INT,
23   FOREIGN KEY (department_id) REFERENCES department(department_id),
24   FOREIGN KEY (address_id) REFERENCES address(address_id)
25 );
26
```

In the bottom instance, the SQL editor also contains the same SQL code. The 'Output' tab is open, showing the execution results:

#	Time	Action	Message	Duration / Fetch
1	19:14:29	USE assignment	0 row(s) affected	0.000 sec
2	19:14:29	CREATE TABLE address (address_id INT PRIMARY KEY AUTO_INCREMENT,	0 row(s) affected	0.031 sec
3	19:14:29	CREATE TABLE department(department_id INT PRIMARY KEY AUTO_INCRE...	0 row(s) affected	0.016 sec
4	19:14:29	CREATE TABLE student(student_id INT PRIMARY KEY AUTO_INCREMENT, ...	0 row(s) affected	0.047 sec

2. Use Sample data from [sampledata.txt](#) to insert data into the database

The screenshot shows the MySQL Workbench interface with the 'assignment' schema selected. The SQL editor contains the following code:

```
1   INSERT INTO address (street_address, city, state, postal_code)
2   VALUES ('123 Elm St', 'Springfield', 'IL', '62701'),
3   ('456 Oak St', 'Decatur', 'IL', '62521'),
4   ('789 Pine St', 'Champaign', 'IL', '61820'),
5   ('102 Birch Rd', 'Peoria', 'IL', '61602'),
6   ('205 Cedar Ave', 'Chicago', 'IL', '60601'),
7   ('310 Maple Dr', 'Urbana', 'IL', '61801'),
8   ('415 Oak Blvd', 'Champaign', 'IL', '61821'),
9   ('520 Pine Rd', 'Carbondale', 'IL', '62901');

10  •  INSERT INTO department (department_name)
11  VALUES ('Computer Science'),
12  ('Mechanical Engineering'),
13  ('Electrical Engineering'),
14  ('Civil Engineering'),
15  ('Mathematics'),
16  ('Biology');

17
18
19  •  INSERT INTO student (first_name, last_name, birthdate, department_id, address_id)
20  VALUES ('John', 'Doe', '1995-04-15', 1, 1),
21  ('Jane', 'Smith', '1996-07-22', 2, 2),
22  ('Alice', 'Johnson', '1994-11-30', 3, 3),
23  ('Michael', 'Brown', '1997-02-19', 4, 4),
24  ('Sophia', 'Davis', '1998-01-05', 5, 5),
25  ('Daniel', 'Wilson', '1995-06-10', 6, 6),
26  ('Olivia', 'Martinez', '1997-11-25', 1, 7),
```

The screenshot shows the MySQL Workbench interface with the 'assignment' schema selected. The SQL editor contains the same code as the previous screenshot. The output pane displays the results of the executed queries:

#	Time	Action	Message	Duration / Fetch
1	19:36:00	INSERT INTO address (street_address, city, state, postal_code) VALUES ('123 ...	8 row(s) affected Records: 8 Duplicates: 0 Warnings: 0	0.015 sec
2	19:36:01	INSERT INTO department (department_name) VALUES ('Computer Science'), ...	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0	0.016 sec
3	19:36:01	INSERT INTO student (first_name, last_name, birthdate, department_id, address_id)	8 row(s) affected Records: 8 Duplicates: 0 Warnings: 0	0.000 sec

3. Write a query to find the total number of students.

The screenshot shows the MySQL Workbench interface with the 'assignment' schema selected. The SQL editor contains the following query:

```
1   select count(*) as total_students from student;
```

The result grid shows the output of the query:

total_students
8

4. Write a query to find which department john belongs to.

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

- File Bar:** Local instance MySQL80, File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** SCHEMAS, assignment (Tables, Views, Stored Procedures, Functions), practicedb, sakila, sys, world.
- Query Editor:** A query window titled "query" containing the following SQL code:

```
1 • select count(*) as total_students from student;
2
3 • SELECT d.department_name FROM department AS d
4 JOIN student AS s ON s.department_id = d.department_id
5 WHERE s.first_name = 'John';
```
- Result Grid:** Shows the output of the query:

department_name
Computer Science
- SQL Additions:** A note stating "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."
- Bottom Bar:** Result 12, Result 13, Read Only, Context Help, Snippets.

5. List All Departments with Their Number of Students (Including Departments with No Students)

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

- File Bar:** Local instance MySQL80, File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** SCHEMAS, assignment (Tables, Views, Stored Procedures, Functions), practicedb, sakila, sys, world.
- Query Editor:** A query window titled "query" containing the following SQL code:

```
1 • select count(*) as total_students from student;
2
3 • SELECT d.department_name FROM department AS d
4 JOIN student AS s ON s.department_id = d.department_id
5 WHERE s.first_name = 'John';
6
7 • SELECT d.department_name , COUNT(s.student_id) FROM department AS d
8 INNER JOIN student AS s ON s.department_id = d.department_id
9 GROUP BY d.department_name;
```
- Result Grid:** Shows the output of the query:

department_name	COUNT(s.student_id)
Computer Science	2
Mechanical Engineering	2
Electrical Engineering	1
Civil Engineering	1
Mathematics	1
Biology	1
- SQL Additions:** A note stating "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."
- Bottom Bar:** Result 14, Read Only, Context Help, Snippets.

6. Select all students with their department and address.

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

- Navigator:** Shows the schema **assignment** selected, containing tables **student**, **department**, and **address**.
- SQL Editor:** Displays the following SQL query:

```
7 •  SELECT d.department_name , COUNT(s.student_id) FROM department AS d
8   INNER JOIN student AS s ON s.department_id = d.department_id
9   GROUP BY d.department_name;
10
11 •  SELECT s.first_name , s.last_name , d.department_name , a.street_address , a.city , a.state , a.postal_
12   FROM student AS s
13   JOIN department AS d ON s.department_id = d.department_id
14   JOIN address AS a ON s.address_id = a.address_id;
15
```
- Result Grid:** Shows the results of the query, listing students with their first name, last name, department name, street address, city, state, and postal code.

first_name	last_name	department_name	street_address	city	state	postal_code
John	Doe	Computer Science	123 Elm St	Springfield	IL	62701
Olivia	Martinez	Computer Science	415 Oak Blvd	Champaign	IL	61821
Jane	Smith	Mechanical Engineering	456 Oak St	Decatur	IL	62521
Ethan	Miller	Mechanical Engineering	520 Pine Rd	Carbondale	IL	62901
Alice	Johnson	Electrical Engineering	789 Pine St	Champaign	IL	61820
Michael	Brown	Civil Engineering	102 Birch Rd	Peoria	IL	61602
Sophia	Davis	Mathematics	205 Cedar Ave	Chicago	IL	60601
Daniel	Wilson	Biology	310 Maple Dr	Urbana	IL	61801
- Status Bar:** Shows "Result 15" and "Read Only".

7. Find all students who are in the 'Computer Science' department

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

- Navigator:** Shows the schema **assignment** selected, containing tables **student**, **department**, and **address**.
- SQL Editor:** Displays the following SQL query:

```
10
11 •  SELECT s.first_name , s.last_name , d.department_name , a.street_address , a.city , a.state , a.postal_
12   FROM student AS s
13   JOIN department AS d ON s.department_id = d.department_id
14   JOIN address AS a ON s.address_id = a.address_id;
15
16 •  SELECT s.first_name , s.last_name FROM student AS s
17   JOIN department AS d ON s.department_id = d.department_id
18   WHERE department_name = 'Computer Science';
```
- Result Grid:** Shows the results of the query, listing students with their first name and last name.

first_name	last_name
John	Doe
Olivia	Martinez
- Status Bar:** Shows "Result 19" and "Read Only".

8. Update Jane's city name to New York.

The screenshot shows the MySQL Workbench interface. The SQL editor tab is active, displaying the following SQL code:

```
7 •  SELECT d.department_name , COUNT(s.student_id) FROM department AS d
8   INNER JOIN student AS s ON s.department_id = d.department_id
9   GROUP BY d.department_name;
10
11 •  SELECT s.first_name, s.last_name, d.department_name, a.street_address, a.city, a.state, a.postal_
12   FROM student AS s
13   JOIN department AS d ON s.department_id = d.department_id
14   JOIN address AS a ON s.address_id = a.address_id;
15
16 •  SELECT s.first_name, s.last_name FROM student AS s
17   JOIN department AS d ON s.department_id = d.department_id
18   WHERE department_name = 'Computer Science';
19
20 •  UPDATE address AS a
21   JOIN student AS s ON a.address_id = s.address_id
22   SET a.city = 'New York'
23   WHERE s.first_name = 'Jane' AND s.last_name = 'Smith';
```

The Output pane shows two log entries:

#	Time	Action	Message	Duration / Fetch
41	20:09:00	UPDATE address AS a JOIN student AS s ON a.address_id = s.address_id SET a.c...	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0	0.000 sec
42	20:09:55	UPDATE address AS a JOIN student AS s ON a.address_id = s.address_id ...	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0	0.000 sec

9. Delete a student from the student table.

The screenshot shows the MySQL Workbench interface. The SQL editor tab is active, displaying the following SQL code:

```
21   JOIN student AS s ON a.address_id = s.address_id
22   SET a.city = 'New York'
23   WHERE s.first_name = 'Jane' AND s.last_name = 'Smith';
24
25 •  DELETE student FROM student WHERE student_id = 4;
26 •  select * from student;
```

The Result Grid pane displays the student table data:

student_id	first_name	last_name	birthdate	department_id	address_id
1	John	Doe	1995-04-15	1	1
2	Jane	Smith	1996-07-22	2	2
3	Alice	Johnson	1994-11-30	3	3
5	Sophia	Davis	1998-01-05	5	5
6	Daniel	Wilson	1995-06-10	6	6
7	Olivia	Martinez	1997-11-25	1	7
8	Ethan	Miller	1996-03-30	2	8
NULL	NULL	NULL	NULL	NULL	NULL

The Output pane shows two log entries:

#	Time	Action	Message	Duration / Fetch
49	20:12:59	DELETE student FROM student WHERE student_id = 4	1 row(s) affected	0.016 sec
50	20:13:26	select * from student LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec

10. Select all students with their department and address in New York.

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

- Navigator:** Shows the schema **assignment** selected, containing tables like **student**, **department**, and **address**.
- Query Editor:** Displays the following SQL code:

```
22 SET a.city = 'New York'
23 WHERE s.first_name = 'Jane' AND s.last_name = 'Smith';
24
25 • DELETE student FROM student WHERE student_id = 4;
26 • select * from students;
27
28 • SELECT s.first_name, s.last_name, d.department_name, a.street_address, a.city, a.state, a.postal_
29 FROM student AS s
30 JOIN department AS d ON s.department_id = d.department_id
31 JOIN address AS a ON s.address_id = a.address_id
32 WHERE a.city = 'New York';
```
- Result Grid:** Shows the result of the last query, which is a single row:

first_name	last_name	department_name	street_address	city	state	postal_code
Jane	Smith	Mechanical Engineering	456 Oak St	New York	IL	62521
- Information:** Shows the schema **assignment** selected.
- Toolbar:** Includes icons for creating table, inserting into tables, and executing queries.
- Help:** A message states: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

11. Count how many students are in each department

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

- Navigator:** Shows the schema **assignment** selected, containing tables like **student**, **department**, and **address**.
- Query Editor:** Displays the following SQL code:

```
28 • SELECT s.first_name, s.last_name, d.department_name, a.street_address, a.city, a.state, a.postal_
29 FROM student AS s
30 JOIN department AS d ON s.department_id = d.department_id
31 JOIN address AS a ON s.address_id = a.address_id
32 WHERE a.city = 'New York';
33
34 • SELECT d.department_name, COUNT(s.student_id) AS num_students
35 FROM department d
36 LEFT JOIN student s ON s.department_id = d.department_id
37 GROUP BY d.department_name;
```
- Result Grid:** Shows the result of the last query, which is a table of department counts:

department_name	num_students
Computer Science	2
Mechanical Engineering	2
Electrical Engineering	1
Civil Engineering	0
Mathematics	1
Biology	1
- Information:** Shows the schema **assignment** selected.
- Toolbar:** Includes icons for creating table, inserting into tables, and executing queries.
- Help:** A message states: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

12. Find students who live in 'Springfield'

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'File', 'Edit', 'View', 'Query', 'Database', 'Server', 'Tools', 'Scripting', and 'Help' are visible. The 'query' tab is selected. The Navigator pane on the left shows the 'assignment' schema with its tables ('Tables', 'Views', 'Stored Procedures', 'Functions'). The main area displays a SQL query:

```
34 •   SELECT d.department_name, COUNT(s.student_id) AS num_students
35     FROM department d
36     LEFT JOIN student s ON s.department_id = d.department_id
37   GROUP BY d.department_name;
38
39 •   SELECT s.first_name, s.last_name, a.street_address, a.city, a.state, a.postal_code
40     FROM student s
41     JOIN address a ON s.address_id = a.address_id
42     WHERE a.city = 'Springfield';
43
44
```

The results grid below the query shows one row of data:

first_name	last_name	street_address	city	state	postal_code
John	Doe	123 Elm St	Springfield	IL	62701

At the bottom right of the interface, a note says: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

13. Select students whose birthday falls in February

The screenshot shows the MySQL Workbench interface. The 'query' tab is selected. The Navigator pane on the left shows the 'assignment' schema with its tables ('Tables', 'Views', 'Stored Procedures', 'Functions'). The main area displays a SQL query:

```
40   FROM student s
41   JOIN address a ON s.address_id = a.address_id
42   WHERE a.city = 'Springfield';
43
44 •   SELECT first_name, last_name, birthdate
45     FROM student
46   WHERE MONTH(birthdate) = 02;
47
48
49
50
```

The results grid below the query shows one row of data:

first_name	last_name	birthdate
John	Doe	1995-04-15

At the bottom right of the interface, a note says: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

14. Get the department and address details for a specific student, example john

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

```
42 WHERE a.city = 'Springfield';
43
44 • SELECT first_name, last_name, birthdate
45   FROM student
46 WHERE MONTH(birthdate) = 04;
47
48 • SELECT d.department_name, a.street_address, a.city, a.state, a.postal_code
49   FROM student s
50   JOIN department d ON s.department_id = d.department_id
51   JOIN address a ON s.address_id = a.address_id
52 WHERE s.student_id = 2;
53
```

The result grid shows one row of data:

department_name	street_address	city	state	postal_code
Mechanical Engineering	456 Oak St	New York	IL	62521

15. Find all students who are born within 1995 to 1998

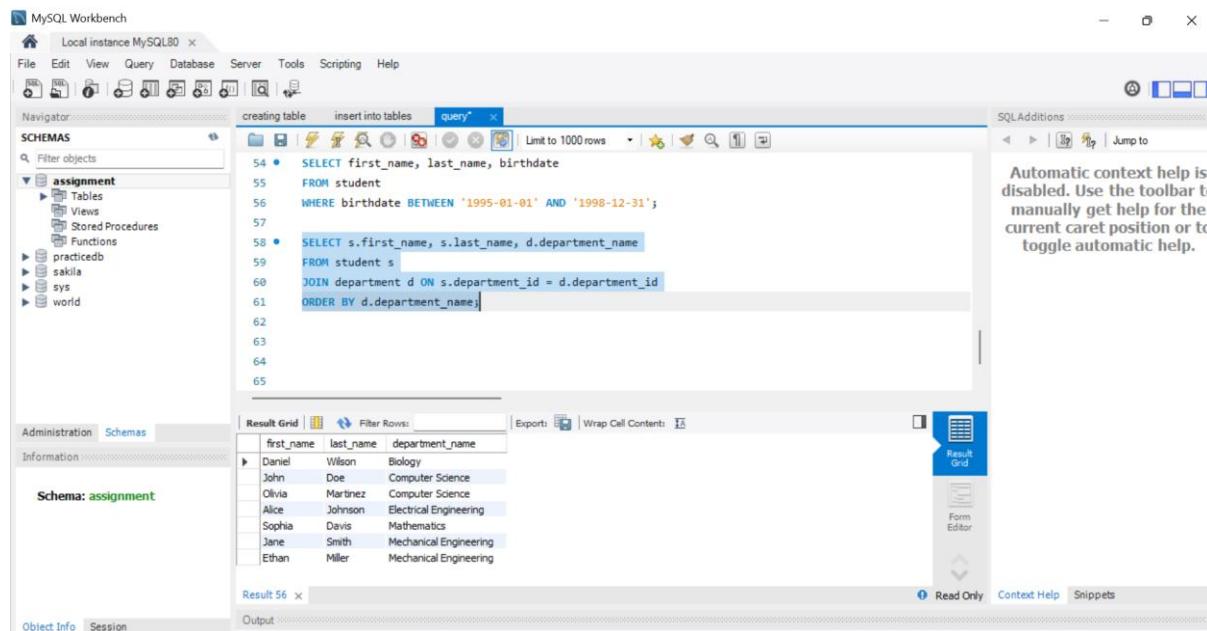
The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```
50 JOIN department d ON s.department_id = d.department_id
51 JOIN address a ON s.address_id = a.address_id
52 WHERE s.student_id = 2;
53
54 • SELECT first_name, last_name, birthdate
55   FROM student
56 WHERE birthdate BETWEEN '1995-01-01' AND '1998-12-31';
57
58
59
60
61
```

The result grid shows multiple rows of data:

first_name	last_name	birthdate
John	Doe	1995-04-15
Jane	Smith	1996-07-22
Sophia	Davis	1998-01-05
Daniel	Wilson	1995-06-10
Olivia	Martinez	1997-11-25
Ethan	Miller	1996-03-30

16. List all students and their corresponding department names, sorted by department



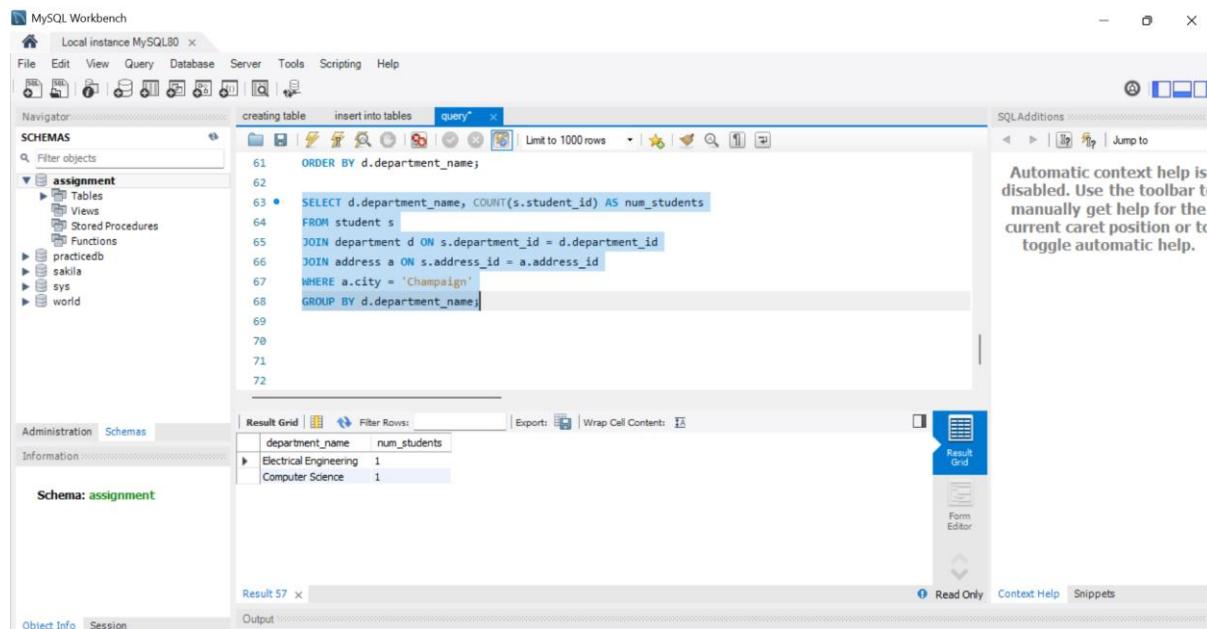
The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema **assignment** selected, containing tables like **student**, **department**, and **address**.
- Query Editor:** Displays the following SQL query:

```
54 • SELECT first_name, last_name, birthdate
  FROM student
  WHERE birthdate BETWEEN '1995-01-01' AND '1998-12-31';
55
56 • SELECT s.first_name, s.last_name, d.department_name
  FROM student s
  JOIN department d ON s.department_id = d.department_id
  ORDER BY d.department_name;
```
- Result Grid:** Shows the results of the query, listing students with their first name, last name, and department name.

first_name	last_name	department_name
Daniel	Wilson	Biology
John	Doe	Computer Science
Olivia	Martinez	Computer Science
Alice	Johnson	Electrical Engineering
Sophia	Davis	Mathematics
Jane	Smith	Mechanical Engineering
Ethan	Miller	Mechanical Engineering

17. Find the number of students in each department who are living in 'Champaign'



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

- Navigator:** Shows the schema **assignment** selected.
- Query Editor:** Displays the following SQL query:

```
61 ORDER BY d.department_name;
62
63 • SELECT d.department_name, COUNT(s.student_id) AS num_students
  FROM student s
  JOIN department d ON s.department_id = d.department_id
  JOIN address a ON s.address_id = a.address_id
  WHERE a.city = 'Champaign'
  GROUP BY d.department_name;
```
- Result Grid:** Shows the results of the query, listing the department name and the count of students.

department_name	num_students
Electrical Engineering	1
Computer Science	1

18. Retrieve the names of students who live on 'Pine' street

The screenshot shows the MySQL Workbench interface with a query editor window. The query is:

```
67 WHERE a.city = 'Champaign'
68 GROUP BY d.department_name;
69
70 • SELECT s.first_name, s.last_name
71 FROM student s
72 JOIN address a ON s.address_id = a.address_id
73 WHERE a.street_address LIKE '%Pine%';
74
75
76
77
78
```

The results grid shows two rows of data:

first_name	last_name
Alice	Johnson
Ethan	Miller

19. Update the department of a student with student_id = 6 to 'Mechanical Engineering'

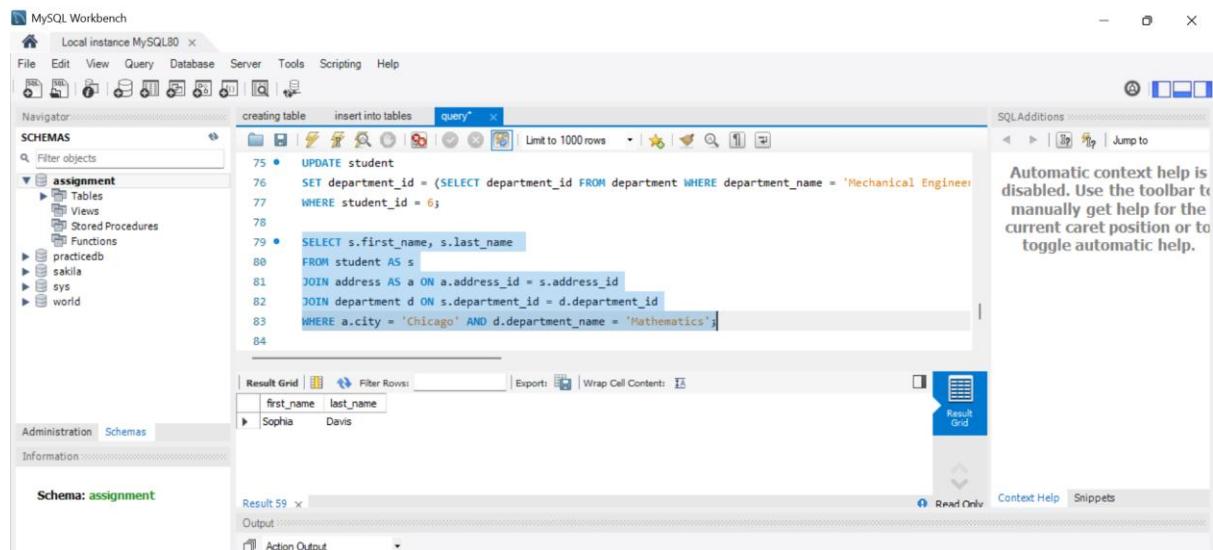
The screenshot shows the MySQL Workbench interface with a query editor window. The query is:

```
64 FROM student s
65 JOIN department d ON s.department_id = d.department_id
66 JOIN address a ON s.address_id = a.address_id
67 WHERE a.city = 'Champaign'
68 GROUP BY d.department_name;
69
70 • SELECT s.first_name, s.last_name
71 FROM student s
72 JOIN address a ON s.address_id = a.address_id
73 WHERE a.street_address LIKE '%Pine%';
74
75 • UPDATE student
76 SET department_id = (SELECT department_id FROM department WHERE department_name = 'Mechanical Engineering')
77 WHERE student_id = 6;
78
79
80
```

The output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
83	20:53:28	UPDATE student SET department_id = (SELECT department_id FROM department WHERE department_name = 'Mechanical Engineering')	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.000 sec
84	20:53:30	UPDATE student SET department_id = (SELECT department_id FROM department WHERE department_name = 'Mechanical Engineering')	0 row(s) affected Rows matched: 1 Changed: 0 Warnings: 0	0.000 sec

20. Find the student(s) who live in the city 'Chicago' and are in the 'Mathematics' department



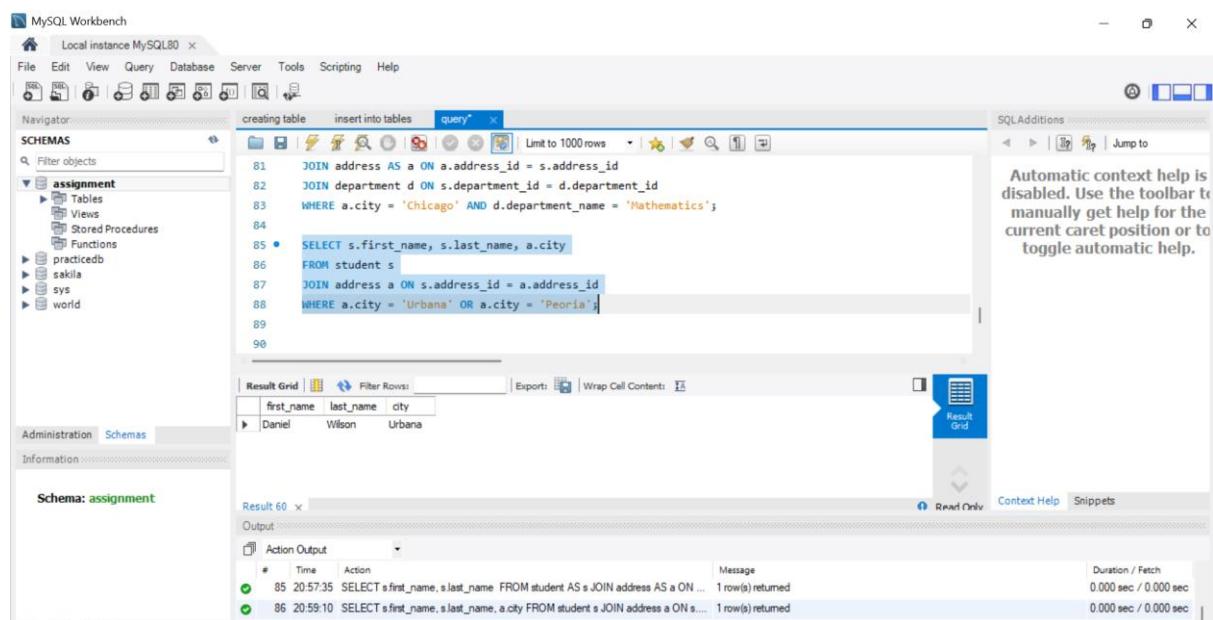
The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
75 • UPDATE student
76   SET department_id = (SELECT department_id FROM department WHERE department_name = 'Mechanical Engineering')
77   WHERE student_id = 6;
78
79 • SELECT s.first_name, s.last_name
80   FROM student AS s
81   JOIN address AS a ON a.address_id = s.address_id
82   JOIN department d ON s.department_id = d.department_id
83   WHERE a.city = 'Chicago' AND d.department_name = 'Mathematics';
84
```

The results grid shows one row of data:

first_name	last_name
Sophia	Davis

21. List all students who have an address in 'Urbana' or 'Peoria'



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

```
81   JOIN address AS a ON a.address_id = s.address_id
82   JOIN department d ON s.department_id = d.department_id
83   WHERE a.city = 'Chicago' AND d.department_name = 'Mathematics';
84
85 • SELECT s.first_name, s.last_name, a.city
86   FROM student s
87   JOIN address a ON s.address_id = a.address_id
88   WHERE a.city = 'Urbana' OR a.city = 'Peoria';
89
90
```

The results grid shows one row of data:

first_name	last_name	city
Daniel	Wilson	Urbana

The action output pane shows two log entries:

#	Time	Action	Message	Duration / Fetch
85	20:57:35	SELECT s.first_name, s.last_name FROM student AS s JOIN address AS a ON ...	1 row(s) returned	0.000 sec / 0.000 sec
86	20:59:10	SELECT s.first_name, s.last_name, a.city FROM student s JOIN address a ON s... ...	1 row(s) returned	0.000 sec / 0.000 sec

22. Find the student with the highest student_id

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, titled 'Navigator', shows the 'SCHEMAS' section with 'assignment' selected, along with other schemas like 'practicedb', 'sakila', 'sys', and 'world'. The main area contains a 'query' tab with the following SQL code:

```
85 •    SELECT s.first_name, s.last_name, a.city
86      FROM student s
87      JOIN address a ON s.address_id = a.address_id
88     WHERE a.city = 'Urband' OR a.city = 'Perlia';
89
90 •    SELECT * FROM student
91      ORDER BY student_id DESC LIMIT 1;
92
93
94
```

Below the code is a 'Result Grid' table with the following data:

student_id	first_name	last_name	birthdate	department_id	address_id
8	Ethan	Miller	1996-03-30	2	8
NULL	NULL	NULL	NULL	NULL	NULL

The bottom pane shows the 'Action Output' log with the following entries:

#	Time	Action	Message	Duration / Fetch
86	20:59:10	SELECT s.first_name, s.last_name, a.city FROM student s JOIN address a ON s...	1 row(s) returned	0.000 sec / 0.000 sec
87	21:00:56	SELECT * FROM student ORDER BY student_id DESC LIMIT 1	1 row(s) returned	0.016 sec / 0.000 sec

23. Find all students who are not in the 'Computer Science' department

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, titled 'Navigator', shows the 'SCHEMAS' section with 'assignment' selected, displaying Tables, Views, Stored Procedures, Functions, practicedb, sakila, sys, and world. The main area has tabs for 'creating table', 'insert into tables', and 'query*'. The 'query*' tab contains the following SQL code:

```
90 • SELECT * FROM student
91 ORDER BY student_id DESC LIMIT 1;
92
93 • SELECT s.first_name, s.last_name, d.department_name
94 FROM student s
95 JOIN department d ON s.department_id = d.department_id
96 WHERE d.department_name != 'Computer Science';
97
98
99
100
101
102
```

Below the code is a 'Result Grid' showing the following data:

first_name	last_name	department_name
Jane	Smith	Mechanical Engineering
Daniel	Wilson	Mechanical Engineering
Ethan	Miller	Mechanical Engineering
Alice	Johnson	Electrical Engineering
Sophia	Davis	Mathematics

The bottom right corner features a vertical toolbar with icons for Result Grid, Form Editor, and Snippets. Status bars at the bottom indicate 'Result 62 x', 'Read Only', 'Context Help', and 'Snippets'.

24. Count the total number of addresses in the 'Champaign' city

The screenshot shows the MySQL Workbench interface. In the query editor, the following SQL code is written:

```
90 •  SELECT * FROM student
91   ORDER BY student_id DESC LIMIT 1;
92
93 •  SELECT s.first_name, s.last_name, d.department_name
94   FROM student s
95   JOIN department d ON s.department_id = d.department_id
96   WHERE d.department_name != 'Computer Science';
97
98 •  SELECT COUNT(*) AS total_addresses
99   FROM address
100  WHERE city = 'Champaign';
101
102
```

The results grid shows a single row with the value '2' under the column 'total_addresses'.

25. Find the name of the student who lives at '520 Pine Rd'

The screenshot shows the MySQL Workbench interface. In the query editor, the following SQL code is written:

```
95   JOIN department d ON s.department_id = d.department_id
96   WHERE d.department_name != 'Computer Science';
97
98 •  SELECT COUNT(*) AS total_addresses
99   FROM address
100  WHERE city = 'Champaign';
101
102 •  SELECT s.first_name, s.last_name
103   FROM student s
104   JOIN address a ON s.address_id = a.address_id
105   WHERE a.street_address = '520 Pine Rd';
106
107
```

The results grid shows a single row with the values 'Ethan' and 'Miller' under the columns 'first_name' and 'last_name' respectively.

26. Get the average age of students in the 'Electrical Engineering' department

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

- Navigator:** Shows the schema **assignment** selected, containing tables like **student**, **department**, etc.
- Query Editor:** Displays the following SQL query:

```
115 LIMIT 1;
116
117 • SELECT AVG(YEAR(CURDATE()) - YEAR(s.birthdate)) AS average_age
118   FROM student s
119   JOIN department d ON s.department_id = d.department_id
120 WHERE d.department_name = 'Electrical Engineering';
121
122
123
124
```
- Result Grid:** Shows the result of the query: **average_age** = **31.000**.
- Output:** Shows the execution log with three entries, all completed in 0.000 sec.

27. List the students, their department, and the city where they live, but only for those in departments starting with 'M'

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

- Navigator:** Shows the schema **assignment** selected, containing tables like **student**, **department**, **address**, etc.
- Query Editor:** Displays the following SQL query:

```
104
105   JOIN address a ON s.address_id = a.address_id
106   WHERE a.street_address = '520 Pine Rd';
107 • SELECT s.first_name, s.last_name, d.department_name, a.city
108   FROM student s
109   JOIN department d ON s.department_id = d.department_id
110   JOIN address a ON s.address_id = a.address_id
111 WHERE d.department_name LIKE 'M%';
112
113
114
115
116
```
- Result Grid:** Shows the result of the query:

first_name	last_name	department_name	city
Jane	Smith	Mechanical Engineering	New York
Daniel	Wilson	Mechanical Engineering	Urbana
Ethan	Miller	Mechanical Engineering	Carbondale
Sophia	Davis	Mathematics	Chicago
- Output:** Shows the execution log with four entries, all completed in 0.000 sec.

28. Delete a student from the 'Mechanical Engineering' department

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'Local instance MySQL80' is selected. The 'Query' tab is active, displaying a multi-line SQL script. The script includes a SELECT query to find students in the 'Mechanical Engineering' department and a corresponding DELETE statement. The 'Output' pane at the bottom shows the execution results for each query, including the time, action, message, and duration.

```
105 WHERE a.street_address = '520 Pine Rd';
106
107 • SELECT s.first_name, s.last_name, d.department_name, a.city
108   FROM student s
109   JOIN department d ON s.department_id = d.department_id
110   JOIN address a ON s.address_id = a.address_id
111 WHERE d.department_name LIKE 'Mechanical%';
112
113 • DELETE FROM student
114 WHERE department_id = (SELECT department_id FROM department WHERE department_name = 'Mechanical Engineering')
115 LIMIT 1;
116
117
118
119
120
121
```

#	Time	Action	Message	Duration / Fetch
90	21:05:27	SELECT s.first_name, s.last_name FROM student s	SELECT s.first_name, s.last_name, d.department_name, a.city	0.000 sec / 0.000 sec
91	21:07:49	SELECT s.first_name, s.last_name, d.department_name, a.city	FROM student s	0.000 sec / 0.000 sec
92	21:09:17	DELETE FROM student WHERE department_id = (SELECT department_id FROM department WHERE department_name = 'Mechanical Engineering')	JOIN department d ON s.department_id = d.department_id	0.015 sec
			JOIN address a ON s.address_id = a.address_id	
			WHERE d.department_name LIKE 'Mechanical%'	
			LIMIT 1;	
			WHENEVER SQLERROR EXIT ROLLBACK;	

Download [order.sql](#)

Open PG Admin and open query tool and select any database of your choice.

Click on “Open file” and select [order.sql](#) from your device and execute it.

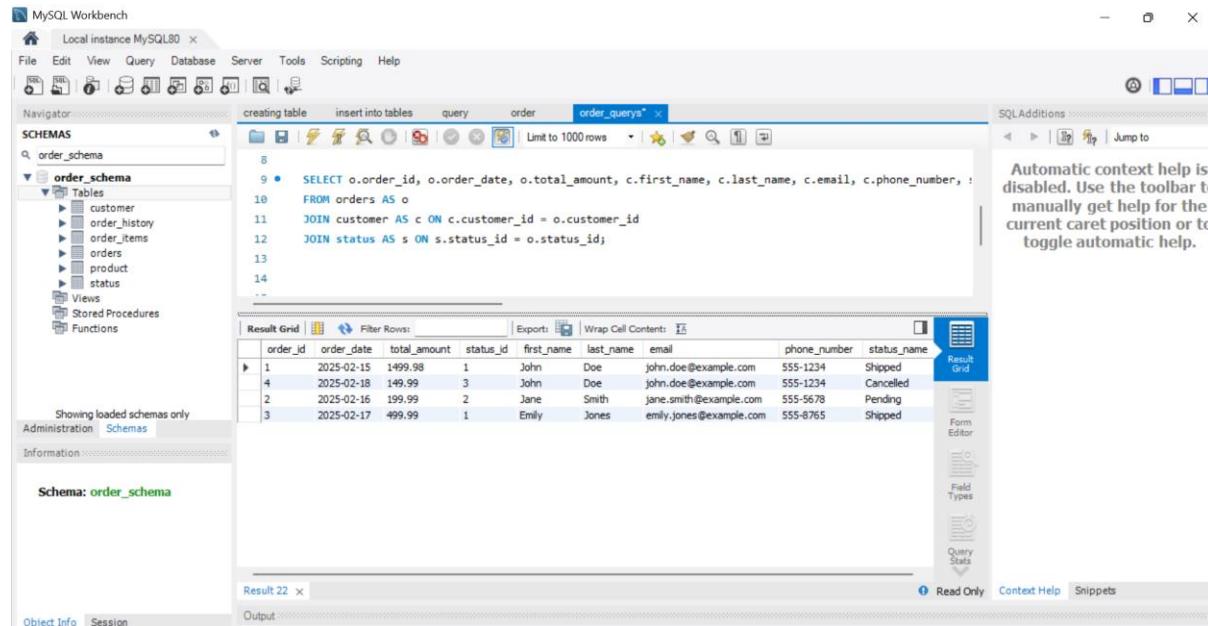
The screenshot shows the MySQL Workbench interface with the 'order' schema selected. The 'Query' tab is active, displaying the contents of the 'order.sql' file. The 'Output' pane at the bottom shows the execution results for each query, including the time, action, message, and duration.

```
-- Create Schema
CREATE SCHEMA order_schema;
-- Create Product Table
CREATE TABLE order_schema.product (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(255),
    price DECIMAL(10, 2),
    stock_quantity INT
);
-- Insert sample data into Product Table
INSERT INTO order_schema.product (product_id, product_name, price, stock_quantity)
VALUES
```

#	Time	Action	Message	Duration / Fetch
96	22:32:07	CREATE SCHEMA order_schema	1 row(s) affected	0.016 sec
97	22:32:07	CREATE TABLE order_schema.product (product_id INT PRIMARY KEY, p...)	0 row(s) affected	0.047 sec
98	22:32:07	INSERT INTO order_schema.product (product_id, product_name, price, stock_q...	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.000 sec
99	22:32:07	CREATE TABLE order_schema.customer (customer_id INT PRIMARY KEY, ...)	0 row(s) affected	0.015 sec
100	22:32:07	INSERT INTO order_schema.customer (customer_id, first_name, last_name, em...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.000 sec
101	22:32:07	CREATE TABLE order_schema.status (status_id INT PRIMARY KEY, stat...	0 row(s) affected	0.031 sec
102	22:32:07	INSERT INTO order_schema.status (status_id, status_name) VALUES (1, 'Sh...	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.000 sec
103	22:32:07	CREATE TABLE order_schema.orders (order_id INT PRIMARY KEY, cust...	0 row(s) affected	0.032 sec
104	22:32:07	INSERT INTO order_schema.orders (order_id, customer_id, order_date, total_am...	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.016 sec
105	22:32:07	CREATE TABLE order_schema.order_items (order_item_id INT PRIMARY KE...	0 row(s) affected	0.016 sec

Questions:

1. Retrieve All Orders with Their Customer Details and Current Status



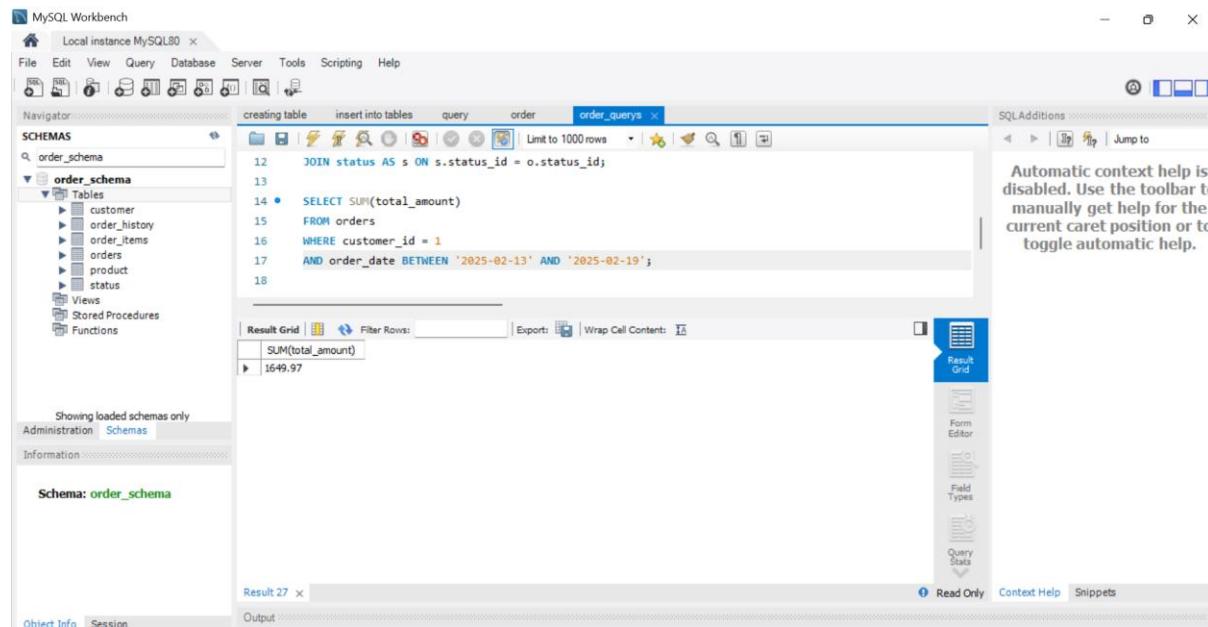
The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Standard MySQL icons for creating table, inserting into tables, querying, etc.
- Navigator:** Shows the schema `order_schema` with tables: customer, order_history, order_items, orders, product, status.
- Query Editor:** A multi-line text area containing the following SQL query:

```
8 •  SELECT o.order_id, o.order_date, o.total_amount, c.first_name, c.last_name, c.email, c.phone_number,
9   FROM orders AS o
10  JOIN customer AS c ON c.customer_id = o.customer_id
11  JOIN status AS s ON s.status_id = o.status_id;
12
13
14
15
```
- Result Grid:** A table displaying the results of the query. The columns are: order_id, order_date, total_amount, status_id, first_name, last_name, email, phone_number, status_name. The data is as follows:

order_id	order_date	total_amount	status_id	first_name	last_name	email	phone_number	status_name
1	2025-02-15	1499.98	1	John	Doe	john.doe@example.com	555-1234	Shipped
4	2025-02-18	149.99	3	John	Doe	john.doe@example.com	555-1234	Cancelled
2	2025-02-16	149.99	2	Jane	Smith	jane.smith@example.com	555-5678	Pending
3	2025-02-17	499.99	1	Emily	Jones	emily.jones@example.com	555-8765	Shipped

2. Get the Total Value of Orders for a Given Customer in a Specific Time Period



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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Standard MySQL icons for creating table, inserting into tables, querying, etc.
- Navigator:** Shows the schema `order_schema` with tables: customer, order_history, order_items, orders, product, status.
- Query Editor:** A multi-line text area containing the following SQL query:

```
12  JOIN status AS s ON s.status_id = o.status_id;
13
14 •  SELECT SUM(total_amount)
15   FROM orders
16  WHERE customer_id = 1
17  AND order_date BETWEEN '2025-02-13' AND '2025-02-19';
18
```
- Result Grid:** A table displaying the results of the query. The column is: SUM(total_amount). The data is as follows:

SUM(total_amount)
1649.97

3. Find the Most Expensive Order by Customer

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (order_schema), Tables (customer, order_history, order_items, orders, product, status).
- SQL Editor:** SQL File 21* contains the following query:

```
35 ORDER BY number_of_orders DESC LIMIT 5;
36
37 • SELECT o.customer_id, c.first_name, c.last_name, o.order_id, o.total_amount
38   FROM orders AS o
39   JOIN customer AS c ON o.customer_id = c.customer_id
40   WHERE o.total_amount = (
41       SELECT MAX(total_amount) FROM orders WHERE customer_id = o.customer_id )
```
- Result Grid:** Displays the results of the query:

customer_id	first_name	last_name	order_id	total_amount
1	John	Doe	1	1499.98
2	Jane	Smith	2	199.99
3	Emily	Jones	3	499.99
- Output:** Shows two actions: SELECT and SELECT, both returning 3 rows and taking 0.016 sec / 0.000 sec.

4. Find the Total Revenue for Each Product Based on Orders

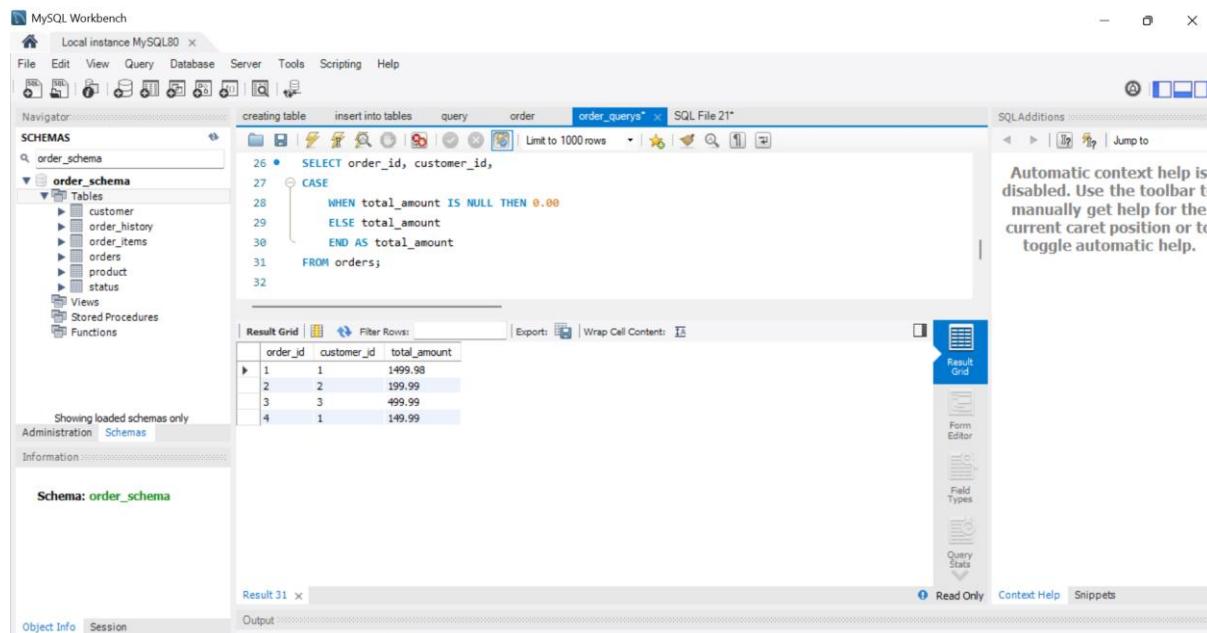
The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (order_schema), Tables (customer, order_history, order_items, orders, product, status).
- SQL Editor:** SQL File 21* contains the following query:

```
21 SELECT p.product_id, p.product_name, SUM(oi.quantity * oi.price) AS total_revenue
22   FROM order_items AS oi
23   JOIN product AS p ON oi.product_id = p.product_id
24   GROUP BY p.product_id, p.product_name;
```
- Result Grid:** Displays the results of the query:

product_id	product_name	total_revenue
1	Laptop	999.99
2	Smartphone	999.98
3	Headphones	449.97
- Output:** Shows two actions: SELECT and SELECT, both returning 3 rows and taking 0.016 sec / 0.000 sec.

5. Write a query to retrieve the order ID, customer ID, and the total amount of each order. If the total amount is null, display '0.00' instead.



```

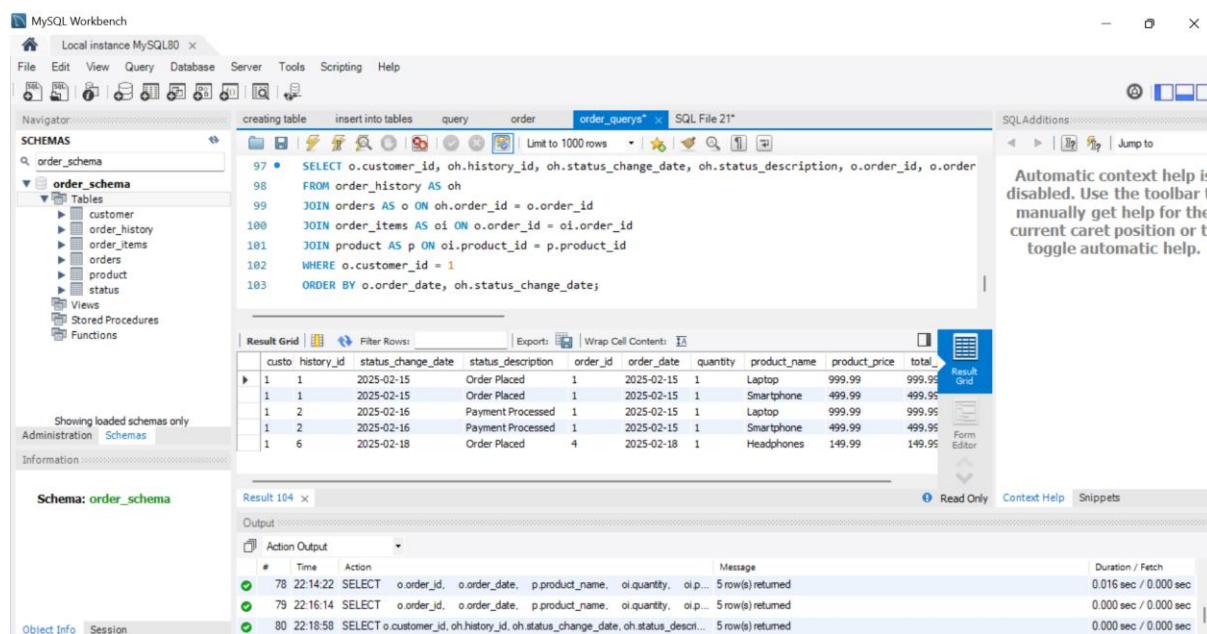
MySQL Workbench - Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator: order_schema
SCHEMAS: order_schema
Tables: customer, order_history, order_items, orders, product, status
Views: 
Stored Procedures: 
Functions: 

SELECT order_id, customer_id,
CASE
    WHEN total_amount IS NULL THEN 0.00
    ELSE total_amount
END AS total_amount
FROM orders;

```

The screenshot shows the MySQL Workbench interface with the SQL editor tab active. The query window contains a SELECT statement that uses a CASE expression to handle NULL values in the total_amount column. The result grid displays four rows of data, showing order IDs 1, 2, 3, and 4, their respective customer IDs, and their total amounts (1499.98, 199.99, 499.99, and 149.99).

6. Retrieve the Order History of a Specific Customer Along with Product Details



```

MySQL Workbench - Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator: order_schema
SCHEMAS: order_schema
Tables: customer, order_history, order_items, orders, product, status
Views: 
Stored Procedures: 
Functions: 

SELECT o.customer_id, oh.history_id, oh.status_change_date, oh.status_description, o.order_id, o.order_history_id, oi.order_id, oi.product_id, p.product_name, p.product_price, total_amount
FROM order_history AS oh
JOIN orders o ON oh.order_id = o.order_id
JOIN order_items AS oi ON o.order_id = oi.order_id
JOIN product p ON oi.product_id = p.product_id
WHERE o.customer_id = 1
ORDER BY o.order_date, oh.status_change_date;

```

The screenshot shows the MySQL Workbench interface with the SQL editor tab active. The query window contains a complex multi-table JOIN query that retrieves order history and product details for customer ID 1. The result grid shows 104 rows of data, mapping customer IDs to their order histories and the products they contain, along with their respective dates and descriptions.

7. Get the Average Order Value Per Customer in the Last 30 Days.

- Zero records as there are no transaction from past 30 days

The screenshot shows the MySQL Workbench interface. In the SQL editor tab, a query is written to calculate the average order value per customer over the last 30 days:

```
48 -- WHERE o.customer_id = 1
49 -- ORDER BY o.order_date, oh.status_change_date;
50
51 • SELECT o.customer_id, c.first_name, c.last_name, AVG(o.total_amount) AS average_order_value
  FROM orders AS o
  JOIN customer AS c ON o.customer_id = c.customer_id
 WHERE o.order_date >= CURRENT_DATE - INTERVAL '30' DAY
 GROUP BY o.customer_id, c.first_name, c.last_name
 ORDER BY average_order_value DESC;
```

The results grid shows the following data:

customer_id	first_name	last_name	average_order_value

In the Output panel, two log entries are displayed:

#	Time	Action	Message	Duration / Fetch
35	20:53:58	SELECT o.customer_id, c.first_name, c.last_name, AVG(o.total_amount) AS ave...	0 row(s) returned	0.000 sec / 0.000 sec
36	20:54:18	SELECT o.customer_id, c.first_name, c.last_name, AVG(o.total_amount) AS ave...	0 row(s) returned	0.000 sec / 0.000 sec

8. Get the Top 5 Products with the Highest Number of Orders.

The screenshot shows the MySQL Workbench interface. In the SQL editor tab, a query is written to find the top 5 products by the number of orders:

```
27     ELSE total_amount
28   END AS total_amount
29
30
31 • SELECT p.product_id, p.product_name, COUNT(oi.order_item_id) AS number_of_orders
  FROM order_items AS oi
  JOIN product AS p ON oi.product_id = p.product_id
 GROUP BY p.product_id, p.product_name
 ORDER BY number_of_orders DESC LIMIT 5;
```

The results grid shows the following data:

product_id	product_name	number_of_orders
2	Smartphone	2
3	Headphones	2
1	Laptop	1

In the Output panel, a message is displayed about the automatic limit clause:

Set limit for number of rows returned by queries.
Workbench will automatically add the LIMIT clause with the configured number of rows to SELECT queries.

9. Get the Customers Who Have Not Placed Any Orders in the Last 60 Days

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

- Navigator:** Shows the schema `order_schema` with tables: customer, order_history, order_items, orders, product, status.
- SQL Editor:** Contains the following SQL query:

```
54 WHERE o.order_date >= CURRENT_DATE - INTERVAL '30' DAY
55 GROUP BY o.customer_id, c.first_name, c.last_name
56 ORDER BY average_order_value DESC;
57
58 • SELECT c.customer_id, c.first_name, c.last_name, c.email, c.phone_number
59 FROM customer AS c
60 LEFT JOIN orders AS o ON c.customer_id = o.customer_id AND o.order_date >= CURRENT_DATE - INTERVAL '30' DAY
61 WHERE o.order_id IS NULL
62 ORDER BY c.last_name, c.first_name;
```
- Result Grid:** Displays the results of the query:

customer_id	first_name	last_name	email	phone_number
1	John	Doe	john.doe@example.com	555-1234
3	Emily	Jones	emily.jones@example.com	555-8765
2	Jane	Smith	jane.smith@example.com	555-5678
- Output:** Shows the execution log:

#	Time	Action	Message	Duration / Fetch
38	20:56:59	SELECT c.customer_id, c.first_name, c.last_name, c.email, c.phone_number	FR... 0 row(s) returned	0.000 sec / 0.000 sec
39	20:57:11	SELECT c.customer_id, c.first_name, c.last_name, c.email, c.phone_number	FR... 3 row(s) returned	0.016 sec / 0.000 sec

10. List the Orders with Products Ordered More Than Once, Sorted by Order Date

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

- Navigator:** Shows the schema `order_schema` with tables: customer, order_history, order_items, orders, product, status.
- SQL Editor:** Contains the following SQL query:

```
61 WHERE o.order_id IS NULL
62 ORDER BY c.last_name, c.first_name;
63
64 • SELECT o.order_id, o.order_date, oi.product_id, p.product_name, oi.quantity, oi.price AS product_price
65 FROM orders o
66 JOIN order_items oi ON o.order_id = oi.order_id
67 JOIN product p ON oi.product_id = p.product_id
68 WHERE oi.order_id IN (SELECT order_id FROM order_items GROUP BY order_id HAVING COUNT(DISTINCT product_id) > 1)
69 ORDER BY o.order_date;
```
- Result Grid:** Displays the results of the query:

order_id	order_date	product_id	product_name	quantity	product_price	total_price
1	2025-02-15	1	Laptop	1	999.99	999.99
1	2025-02-15	2	Smartphone	1	499.99	499.99
- Output:** Shows the execution log:

#	Time	Action	Message	Duration / Fetch
42	21:04:47	SELECT oi.order_id, oi.product_id, p.product_name, COUNT(*) AS pr...	0 row(s) returned	0.000 sec / 0.000 sec
43	21:07:00	SELECT o.order_id, o.order_date, oi.product_id, p.product_name, oi.quantity, oi...	2 row(s) returned	0.032 sec / 0.000 sec

11. Retrieve the Number of Orders and Total Revenue for Each Status

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

- Navigator:** Shows the schema `order_schema` with tables: customer, order_history, order_items, orders, product, status.
- SQL Editor:** Contains the following SQL query:

```
66 JOIN order_items oi ON o.order_id = oi.order_id
67 JOIN product p ON oi.product_id = p.product_id
68 WHERE oi.order_id IN (SELECT order_id FROM order_items GROUP BY order_id HAVING COUNT(DISTINCT product_id) > 1)
69 ORDER BY o.order_date
70
71 • SELECT s.status_name, COUNT(o.order_id) AS number_of_orders, SUM(o.total_amount) AS total_revenue
72 FROM orders AS o
73 JOIN status AS s ON o.status_id = s.status_id
74 GROUP BY s.status_name ORDER BY number_of_orders DESC;
```
- Result Grid:** Displays the results of the query:

status_name	number_of_orders	total_revenue
Shipped	2	1999.97
Pending	1	199.99
Cancelled	1	149.99
- Output:** Shows the execution log:

#	Time	Action	Message	Duration / Fetch
43	21:07:00	SELECT o.order_id, o.order_date, oi.product_id, p.product_name, oi.quantity, o...	2 row(s) returned	0.032 sec / 0.000 sec
44	21:09:41	SELECT s.status_name, COUNT(o.order_id) AS number_of_orders, SUM(o.total...	3 row(s) returned	0.016 sec / 0.000 sec

12. Find Customers Who Have Ordered More Than a Specific Product (e.g., "Laptop")

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

- Navigator:** Shows the schema `order_schema` with tables: customer, order_history, order_items, orders, product, status.
- SQL Editor:** Contains the following SQL query:

```
76 • SELECT c.customer_id, c.first_name, c.last_name, c.email, c.phone_number
77 FROM customer AS c
78 JOIN orders AS o ON c.customer_id = o.customer_id
79 JOIN order_items AS oi ON o.order_id = oi.order_id
80 JOIN product AS p ON oi.product_id = p.product_id
81 WHERE p.product_name = 'Headphones'
82 GROUP BY c.customer_id, c.first_name, c.last_name, c.email, c.phone_number
83 HAVING SUM(oi.quantity) > 1
84 ORDER BY c.last_name, c.first_name;
```
- Result Grid:** Displays the results of the query:

customer_id	first_name	last_name	email	phone_number
2	Jane	Smith	jane.smith@example.com	555-5678
- Output:** Shows the execution log:

#	Time	Action	Message	Duration / Fetch
65	21:16:40	SELECT c.customer_id, c.first_name, c.last_name, c.email, c.phone_number FR...	0 row(s) returned	0.000 sec / 0.000 sec
66	21:19:14	SELECT c.customer_id, c.first_name, c.last_name, c.email, c.phone_number FR...	1 row(s) returned	0.000 sec / 0.000 sec

13. Find the Products That Have Never Been Ordered

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** SCHEMAS (order_schema), TABLES (customer, order_history, order_items, orders, product, status).
- SQL Editor:** SQL File 21* contains the following query:

```
2 GROUP BY c.customer_id, c.first_name, c.last_name, c.email, c.phone_number
3 HAVING SUM(oi.quantity) > 1
4 ORDER BY c.last_name, c.first_name;
5
6 • SELECT p.product_id, p.product_name, p.price, p.stock_quantity
7 FROM product AS p
8 LEFT JOIN order_items AS oi ON p.product_id = oi.product_id
9 WHERE oi.order_item_id IS NULL;
```
- Result Grid:** Shows a single row:

product_id	product_name	price	stock_quantity
4	Monitor	199.99	75
- Action Output:** Displays two log entries:

#	Time	Action	Message	Duration / Fetch
66	21:19:14	SELECT c.customer_id, c.first_name, c.last_name, c.email, c.phone_number FR...	1 row(s) returned	0.000 sec / 0.000 sec
67	21:23:03	SELECT p.product_id, p.product_name, p.price, p.stock_quantity FROM produc...	1 row(s) returned	0.000 sec / 0.000 sec

14. Get the Total Quantity of Products Ordered in the Last 7 Days

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** SCHEMAS (order_schema), TABLES (customer, order_history, order_items, orders, product, status).
- SQL Editor:** SQL File 21* contains the following query:

```
88 LEFT JOIN order_items AS oi ON p.product_id = oi.product_id
89 WHERE oi.order_item_id IS NULL;
90
91 • SELECT p.product_name, SUM(oi.quantity) AS total_quantity_ordered
92 FROM orders AS o
93 JOIN order_items AS oi ON o.order_id = oi.order_id
94 JOIN product AS p ON oi.product_id = p.product_id
95 WHERE o.order_date >= CURDATE() - INTERVAL 7 DAY
96 GROUP BY p.product_name ORDER BY total_quantity_ordered DESC;
```
- Result Grid:** Shows a single row:

product_name	total_quantity_ordered
- Action Output:** Displays two log entries:

#	Time	Action	Message	Duration / Fetch
69	21:27:38	SELECT p.product_name, SUM(oi.quantity) AS total_quantity_ordered FROM or...	3 row(s) returned	0.000 sec / 0.000 sec
70	21:28:20	SELECT p.product_name, SUM(oi.quantity) AS total_quantity_ordered FROM or...	0 row(s) returned	0.016 sec / 0.000 sec

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

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** SCHEMAS (order_schema), TABLES (customer, order_history, order_items, orders, product, status).
- SQL Editor:** SQL File 21* contains the following query:

```
91 • SELECT p.product_name, SUM(oi.quantity) AS total_quantity_ordered
92 FROM orders AS o
93 JOIN order_items AS oi ON o.order_id = oi.order_id
94 JOIN product AS p ON oi.product_id = p.product_id
95 WHERE o.order_date BETWEEN '2025-02-11' AND '2025-02-18'
96 GROUP BY p.product_name ORDER BY total_quantity_ordered DESC;
```
- Result Grid:** Shows a single row:

product_name	total_quantity_ordered
Headphones	3
Smartphone	2
Laptop	1
- Action Output:** Displays two log entries:

#	Time	Action	Message	Duration / Fetch
68	21:26:20	SELECT p.product_name, SUM(oi.quantity) AS total_quantity_ordered FROM or...	0 row(s) returned	0.000 sec / 0.000 sec
69	21:27:38	SELECT p.product_name, SUM(oi.quantity) AS total_quantity_ordered FROM or...	3 row(s) returned	0.000 sec / 0.000 sec

15. Create a view named product_details that includes all columns from the product table.

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

- Navigator:** Shows the schema `order_schema` with its tables: customer, order_history, order_items, orders, product, status.
- SQL Editor:** Contains the following SQL code:

```
94 JOIN product AS p ON oi.product_id = p.product_id
95 WHERE o.order_date >= CURDATE() - INTERVAL 7 DAY
96 GROUP BY p.product_name ORDER BY total_quantity_ordered DESC;
97
98 • CREATE VIEW product_details AS
99     SELECT product_id, product_name, price, stock_quantity
100    FROM product;
101
102 • SELECT * FROM product_details;
```
- Result Grid:** Displays the results of the query:

product_id	product_name	price	stock_quantity
1	Laptop	999.99	50
2	Smartphone	499.99	100
3	Headphones	149.99	200
4	Monitor	199.99	75
- Action Output:** Shows the execution log:

#	Time	Action	Message	Duration / Fetch
73	21:35:40	DROP VIEW IF EXISTS product_details	0 row(s) affected	0.031 sec
74	21:36:00	CREATE VIEW product_details AS SELECT product_id, product_name, price, st...	0 row(s) affected	0.046 sec
75	21:36:20	SELECT * FROM product_details LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec

16. Create a view named order_summary that includes the order_id, customer_id, order_date, total_amount, and status_name (from the status table) for each order.

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

- Navigator:** Shows the schema `order_schema` with its tables: customer, order_history, order_items, orders, product, status.
- SQL Editor:** Contains the following SQL code:

```
94 JOIN product AS p ON oi.product_id = p.product_id
95 WHERE o.order_date >= CURDATE() - INTERVAL 7 DAY
96 GROUP BY p.product_name ORDER BY total_quantity_ordered DESC;
97
98 • CREATE VIEW product_details AS
99     SELECT product_id, product_name, price, stock_quantity
100    FROM product;
101
102 • SELECT * FROM product_details;
```
- Result Grid:** Displays the results of the query:

product_id	product_name	price	stock_quantity
1	Laptop	999.99	50
2	Smartphone	499.99	100
3	Headphones	149.99	200
4	Monitor	199.99	75
- Action Output:** Shows the execution log:

#	Time	Action	Message	Duration / Fetch
73	21:35:40	DROP VIEW IF EXISTS product_details	0 row(s) affected	0.031 sec
74	21:36:00	CREATE VIEW product_details AS SELECT product_id, product_name, price, st...	0 row(s) affected	0.046 sec
75	21:36:20	SELECT * FROM product_details LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec