VIDYA JYOTHI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution)

(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)

B.Tech(CSE-DS) II Year / I Semester (R22)

LAB MANUAL

Name of the Faculty	**KISHORE K**
Department	**CSE(Data Science)**
Year & Semester	**B.Tech-II & I Sem**
Regulation	**R22**
Lab Name	**DATABASE MANAGEMENT SYSTEMS LAB**

DEPARTMENT OF CSE (Data Science)

VIDYA JYOTHI INSTITUTE OF TECHNOLOGY

(Approved by AICTE, New Delhi & Affiliated to JNTUH, Hyderabad)

An Autonomous Institution

AZIZ NAGAR, C B POST, HYDERABAD-500075

2024-2025

INDEX

1. Database Schema for a customer-sale scenario

Customer(**Cust id : integer,** cust_name: string)

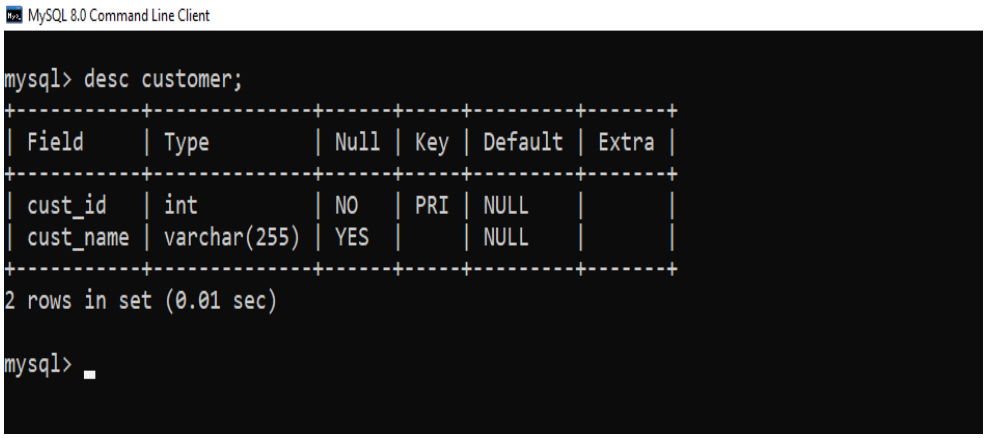
Item(**item_id: integer**, item_name: string, price: integer)

Sale(**bill_no: integer**, bill_data: date, **cust_id: integer, item_id: integer**, qty_sold: integer)

For the above schema, perform the following—

1. **Create Tables**: Write the SQL statements to create the customer, item, and sale tables as described above. Ensure that the sale table has foreign key constraints to reference the customer and item tables.

```
CREATE TABLE customer(
        cust_id INT,
        cust_name VARCHAR(255),
        PRIMARY KEY(cust_id)
);
```




```
CREATE TABLE item(
        item_id INT,
        item_name VARCHAR(255),
        price INT,
        PRIMARY KEY(item_id)
);
```

MySQL 8.0 Command Line Client

```
mysql> desc item;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| item_id   | int          | NO   | PRI | NULL    |       |
| item_name | varchar(255) | YES  |     | NULL    |       |
| price     | int          | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql>
```

CREATE TABLE sale(

bill_no INT PRIMARY KEY,

bill_date DATE,

cust_id INT ,

item_id INT ,

qty_sold INT ,

FOREIGN KEY (cust_id) REFERENCES customer (cust_id),

FOREIGN KEY (item_id) REFERENCES item(item_id));

MySQL 8.0 Command Line Client

```
mysql> desc sale;
+-----------+------+------+-----+---------+-------+
| Field     | Type | Null | Key | Default | Extra |
+-----------+------+------+-----+---------+-------+
| bill_no   | int  | NO   | PRI | NULL    |       |
| bill_date | date | YES  |     | NULL    |       |
| cust_id   | int  | YES  | MUL | NULL    |       |
| item_id   | int  | YES  | MUL | NULL    |       |
| qty_sold  | int  | YES  |     | NULL    |       |
+-----------+------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql>
```

2. Insert Data: Insert 5 records into each of the tables with the given data.

customer:

cust_id	cust_name
1	John Doe
2	Jane Smith
3	Alice Johnson
4	Bob Brown
5	Charlie Davis

item:

item_id	item_name	price
1	Laptop	1000
2	Smartphone	800
3	Tablet	600
4	Headphones	200
5	Camera	100

sale:

bill_no	bill_date	cust_id	item_id	qty_sold
1	2023-07-01	1	1	1
2	2023-07-01	2	2	2
3	2023-07-02	3	3	1
4	2023-07-03	4	4	3
5	2023-07-10	5	5	1

```
INSERT INTO customer VALUES(1,'Jhon Doe');
INSERT INTO customer VALUES(2,'Jane Smith');
INSERT INTO customer VALUES(3,'Alice Jhonson');
INSERT INTO customer VALUES(4,'Bob Brown');
INSERT INTO customer VALUES(5,'Charlie Davis');
```

MySQL 8.0 Command Line Client

```
mysql> select * from customer;
+---------+--------------+
| cust_id | cust_name    |
+---------+--------------+
|       1 | Jhon Doe     |
|       2 | Jane Smith   |
|       3 | Alice Jhonson |
|       4 | Bob Brown    |
|       5 | Charlie Davis |
+---------+--------------+
5 rows in set (0.00 sec)

mysql>
```

```
INSERT INTO item VALUES(1,'Laptop',1000);
INSERT INTO item VALUES(2,'Smartphone',800);
INSERT INTO item VALUES(3,'Tablet',600);
INSERT INTO item VALUES(4,'Headphones',200);
INSERT INTO item VALUES(5,'Camera',100);
```

MySQL 8.0 Command Line Client

```
mysql> select * from item;
+---------+------------+-------+
| item_id | item_name  | price |
+---------+------------+-------+
|       1 | Laptop     |  1000 |
|       2 | Smartphone |   800 |
|       3 | Tablet     |   600 |
|       4 | Headphones |   200 |
|       5 | Camera     |   100 |
+---------+------------+-------+
5 rows in set (0.00 sec)

mysql>
```

INSERT INTO sale VALUES(1,'2023-07-01',1,1,1);

INSERT INTo sale VALUES(2,'2023-07-01',2,2,2);

INSERT INTO sale VALUES(3,'2023-07-02',3,3,1);

INSERT INTO sale VALUES(4,'2023-07-03',4,4,3);

INSERT INTO sale VALUES(5,'2023-07-10',5,5,1);

MySQL 8.0 Command Line Client

```
mysql> select * from sale;
+---------+------------+---------+---------+----------+
| bill_no | bill_date  | cust_id | item_id | qty_sold |
+---------+------------+---------+---------+----------+
|       1 | 2023-07-01 |       1 |       1 |        1 |
|       2 | 2023-07-01 |       2 |       2 |        2 |
|       3 | 2023-07-02 |       3 |       3 |        1 |
|       4 | 2023-07-03 |       4 |       4 |        3 |
|       5 | 2023-07-10 |       5 |       5 |        1 |
+---------+------------+---------+---------+----------+
5 rows in set (0.00 sec)

mysql>
```

3. List all the bill numbers for the date 2023-07-01 along with the customer names and item IDs. Ensure the column names are bill_no, cust_name, and item_id.

```
SELECT s.bill_no, c.cust_name, s.item_id
FROM sale s,customer c
WHERE s.cust_id = c.cust_id AND  s.bill_date = '2023-07-01';
```

```
+---------+------------+---------+
| bill_no | cust_name  | item_id |
+---------+------------+---------+
|       1 | Jhon Doe   |       1 |
|       2 | Jane Smith |       2 |
+---------+------------+---------+
2 rows in set (0.00 sec)

mysql>
```

4. List all the total bill details including bill_no, bill_date, cust_name, item_name, price, qty_sold, and the final amount paid for each item in each transaction (price * qty_sold).

```
SELECT s.bill_no, s.bill_date, c.cust_name, i.item_name, i.price, s.qty_sold, (i.price *
s.qty_sold)
AS amount
FROM sale s,customer c,item i
WHERE s.cust_id = c.cust_id AND  s.item_id = i.item_id;
```

MySQL 8.0 Command Line Client

```
+---------+------------+---------------+------------+-------+----------+--------+
| bill_no | bill_date  | cust_name     | item_name  | price | qty_sold | amount |
+---------+------------+---------------+------------+-------+----------+--------+
|       1 | 2023-07-01 | Jhon Doe      | Laptop     |  1000 |        1 |   1000 |
|       2 | 2023-07-01 | Jane Smith    | Smartphone |   800 |        2 |   1600 |
|       3 | 2023-07-02 | Alice Jhonson | Tablet     |   600 |        1 |    600 |
|       4 | 2023-07-03 | Bob Brown     | Headphones |   200 |        3 |    600 |
|       5 | 2023-07-10 | Charlie Davis | Camera     |   100 |        1 |    100 |
+---------+------------+---------------+------------+-------+----------+--------+
5 rows in set (0.00 sec)

mysql>
```

5. List the details (names) of the customers who have bought a product that has a price greater than 200. Ensure the column name is cust_name.

```
SELECT DISTINCT c.cust_name
FROM sale s, item i, customer c
WHERE  s.item_id = i.item_id AND  s.cust_id = c.cust_id AND  i.price > 200;
```

MySQL 8.0 Command Line Client

```
+---------------+
| cust_name     |
+---------------+
| Jhon Doe      |
| Jane Smith    |
| Alice Jhonson |
+---------------+
3 rows in set (0.00 sec)

mysql>
```

6. Display the total number of products (represented by quantity sold) purchased by each customer, aggregating the quantities for all transactions made by each customer. Ensure the column names are cust_name and total_products_purchased.

```
SELECT c.cust_name, SUM(s.qty_sold)
AS total_products_purchased
FROM sale s,customer c
WHERE s.cust_id = c.cust_id
GROUP BY c.cust_name;
```

MySQL 8.0 Command Line Client

```
+---------------+-------------------------+
| Jhon Doe      |                       1 |
| Jane Smith    |                       2 |
| Alice Jhonson |                       1 |
| Bob Brown     |                       3 |
| Charlie Davis |                       1 |
+---------------+-------------------------+
5 rows in set (0.00 sec)

mysql>
```

7. Give a list of item names bought by the customer with cust_id as 5. Ensure the column name is item_name.

```
SELECT i.item_name
FROM sale s,item i
WHERE  s.item_id = i.item_id AND s.cust_id = 5;
```

MySQL 8.0 Command Line Client

```
+-----------+
| item_name |
+-----------+
| Camera    |
+-----------+
1 row in set (0.00 sec)

mysql>
```

8. List the item names that were sold on the date 2023-07-02. Ensure the column name is item_name.

```
SELECT i.item_name
FROM sale s, item i
WHERE s.item_id = i.item_id AND  s.bill_date = '2023-07-02';
```

MySQL 8.0 Command Line Client

```
+-----------+
| item_name |
+-----------+
| Tablet    |
+-----------+
1 row in set (0.00 sec)

mysql>
```

9. List the bill details including bill_no, bill_date, cust_id, item_id, price, qty_sold, and the amount for each sale transaction.

```
SELECT s.bill_no, s.bill_date, s.cust_id, s.item_id, i.price, s.qty_sold, (i.price *
s.qty_sold)
AS amount
FROM sale s,item i
WHERE s.item_id = i.item_id;
```

MySQL 8.0 Command Line Client

```
+---------+------------+---------+---------+-------+----------+--------+
| bill_no | bill_date  | cust_id | item_id | price | qty_sold | amount |
+---------+------------+---------+---------+-------+----------+--------+
|       1 | 2023-07-01 |       1 |       1 |  1000 |        1 |   1000 |
|       2 | 2023-07-01 |       2 |       2 |   800 |        2 |   1600 |
|       3 | 2023-07-02 |       3 |       3 |   600 |        1 |    600 |
|       4 | 2023-07-03 |       4 |       4 |   200 |        3 |    600 |
|       5 | 2023-07-10 |       5 |       5 |   100 |        1 |    100 |
+---------+------------+---------+---------+-------+----------+--------+
5 rows in set (0.00 sec)

mysql>
```

10. Create a view named bill_details_view that lists detailed bill information including bill_no, bill_date, cust_id, item_id, price, qty_sold, and the total amount as amount for each sale transaction. Ensure the column names are in this exact order.

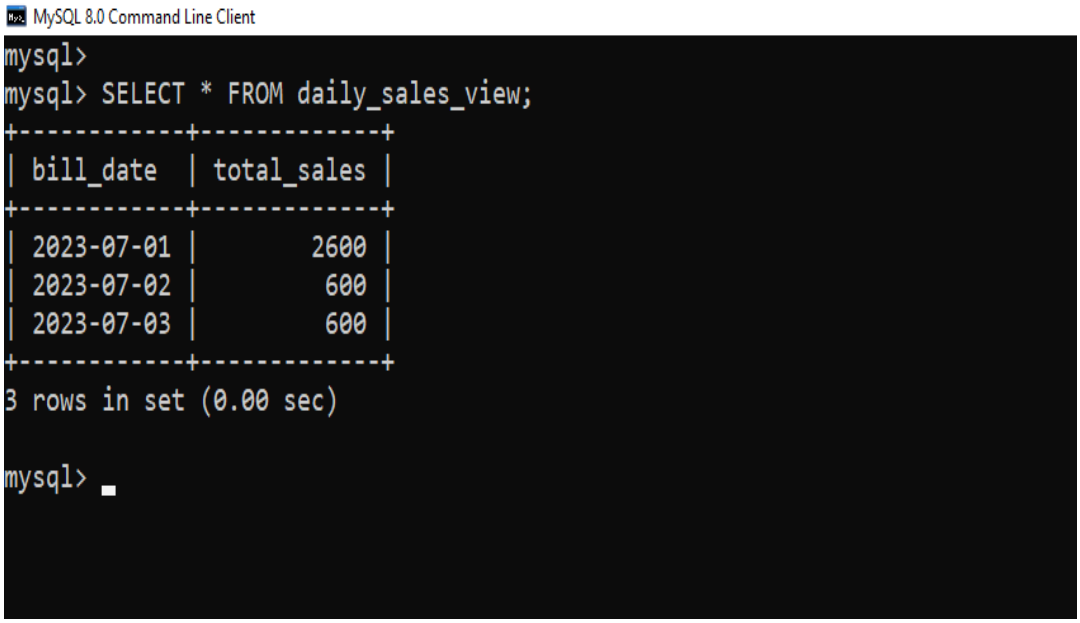
```
CREATE VIEW bill_details_view AS
SELECT s.bill_no, s.bill_date, s.cust_id, s.item_id, i.price, s.qty_sold, (i.price *
s.qty_sold)
AS amount
FROM sale s,item i
WHERE s.item_id = i.item_id;

SELECT * FROM bill_details_view;
```

MySQL 8.0 Command Line Client

```
mysql> SELECT * FROM bill_details_view;
+---------+------------+---------+---------+-------+----------+--------+
| bill_no | bill_date  | cust_id | item_id | price | qty_sold | amount |
+---------+------------+---------+---------+-------+----------+--------+
|       1 | 2023-07-01 |       1 |       1 |  1000 |        1 |   1000 |
|       2 | 2023-07-01 |       2 |       2 |   800 |        2 |   1600 |
|       3 | 2023-07-02 |       3 |       3 |   600 |        1 |    600 |
|       4 | 2023-07-03 |       4 |       4 |   200 |        3 |    600 |
|       5 | 2023-07-10 |       5 |       5 |   100 |        1 |    100 |
+---------+------------+---------+---------+-------+----------+--------+
5 rows in set (0.00 sec)

mysql>
```

11. Create a view named daily_sales_view that lists daily sales (total sales amount) from a start date (2023-07-01) to an end date (2023-07-03), grouped by date. Ensure the column names are bill_date and total_sales.

```
CREATE VIEW daily_sales_view AS
SELECT s.bill_date, SUM(i.price * s.qty_sold) AS total_sales
FROM sale s,item i
WHERE s.item_id = i.item_id and  s.bill_date BETWEEN '2023-07-01' AND '2023-07-03'
GROUP BY s.bill_date;

SELECT * FROM daily_sales_view;
```



2. Database Schema for a Student Library scenario

Student(**Stud_no : integer,** Stud_name: string)

Membership(**Mem_no: integer**, **Stud_no: integer**)

Book(**book_no: integer**, book_name:string, author: string)

Iss_rec(**iss_no:integer**, iss_date: date, **Mem_no: integer**, **book_no: integer**)

For the above schema, perform the following—

1. **Create Tables**: Write the SQL statements to create the student, membership, book, and iss_rec tables as described above.

```
CREATE TABLE student (
        stud_no int PRIMARY KEY,
         stud_name varchar(50)
        );
```

```
mysql> desc student;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| stud_no  | int         | NO   | PRI | NULL    |       |
| stud_name | varchar(50) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
```

```
CREATE TABLE membership(
        mem_no int PRIMARY KEY,
        stud_no int,
        FOREIGN KEY(stud_no)REFERENCES student(stud_no)
        );
```

MySQL 8.0 Command Line Client

```
mysql> desc membership;
+---------+------+------+-----+---------+-------+
| Field   | Type | Null | Key | Default | Extra |
+---------+------+------+-----+---------+-------+
| mem_no  | int  | NO   | PRI | NULL    |       |
| stud_no | int  | YES  | MUL | NULL    |       |
+---------+------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql>
```

```
CREATE TABLE book(
```

```
book_no int PRIMARY KEY,
 book_name varchar(100),
author varchar(50)
);
```

MySQL 8.0 Command Line Client

```
mysql> desc book;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| book_no   | int          | NO   | PRI | NULL    |       |
| book_name | varchar(100) | YES  |     | NULL    |       |
| author    | varchar(50)  | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql>
```

```
CREATE TABLE iss_rec(
iss_no int PRIMARY KEY,
 iss_date date,
 mem_no int,
book_no int ,
FOREIGN KEY(mem_no)REFERENCES membership(mem_no),
FOREIGN KEY(book_no)REFERENCES book(book_no)
);
```

MySQL 8.0 Command Line Client

```
mysql> desc iss_rec;
+----------+------+------+-----+---------+-------+
| Field    | Type | Null | Key | Default | Extra |
+----------+------+------+-----+---------+-------+
| iss_no   | int  | NO   | PRI | NULL    |       |
| iss_date | date | YES  |     | NULL    |       |
| mem_no   | int  | YES  | MUL | NULL    |       |
| book_no  | int  | YES  | MUL | NULL    |       |
+----------+------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql>
```

2. Insert Data: Insert 5 records into each of the tables with the given data.

student:

stud_no	stud_name
1001	John Doe
1002	Jane Smith
1003	Michael Johnson
1004	Emily Brown
1005	David Lee

membership:

mem_no	stud_no
5001	1001
5002	1002
5003	1003
5004	1004
5005	1005

book:

book_no	book_name	author
101	The Great Gatsby	Scott Fitzgerald
102	Moby Dick	Herman Melville
103	Pride and Prejudice	Jane Austen
104	Jane Eyre	Charlotte Bronte
105	Animal Farm	George Orwell

iss_rec:

iss_no	iss_date	mem_no	book_no
1	2024-07-01	5001	101
2	2024-07-02	5002	102
3	2024-07-02	5003	103
4	2024-07-04	5004	104
5	2024-07-05	5005	105

```
INSERT INTO student VALUES(1001,'John Doe');
INSERT INTO student VALUES(1002,'Jane Smith');
INSERT INTO student VALUES(1003,'Michael Johnson');
INSERT INTO student VALUES(1004,'Emily Brown');
INSERT INTO student VALUES(1005,'David Lee');
```

MySQL 8.0 Command Line Client

```
mysql> select * from student;
+---------+----------------+
| stud_no | stud_name      |
+---------+----------------+
|    1001 | John Doe       |
|    1002 | Jane Smith     |
|    1003 | Michael Johnson |
|    1004 | Emily Brown    |
|    1005 | David Lee      |
+---------+----------------+
5 rows in set (0.00 sec)

mysql>
```

```
INSERT INTO membership VALUES(5001,1001);
INSERT INTO membership VALUES(5002,1002);
INSERT INTO membership VALUES(5003,1003);
INSERT INTO membership VALUES(5004,1004);
INSERT INTO membership VALUES(5005,1005);
```

MySQL 8.0 Command Line Client

```
mysql> select * from membership;
+--------+---------+
| mem_no | stud_no |
+--------+---------+
|   5001 |    1001 |
|   5002 |    1002 |
|   5003 |    1003 |
|   5004 |    1004 |
|   5005 |    1005 |
+--------+---------+
5 rows in set (0.00 sec)

mysql>
```

INSERT INTO book VALUES(101,'The Great Gatsby','Scott Fitzgerald');

INSERT INTO book VALUES(102,'Moby Dick','Herman Melville');

INSERT INTO book VALUES(103,'Pride and Prejudice','Jane Austen');

INSERT INTO book VALUES(104,'Jane Eyre','Charlotte Bronte');

INSERT INTO book VALUES(105,'Animal Farm','George Orwell');

MySQL 8.0 Command Line Client

```
mysql> select * from book;
+---------+--------------------+------------------+
| book_no | book_name          | author           |
+---------+--------------------+------------------+
|     101 | The Great Gatsby   | Scott Fitzgerald |
|     102 | Moby Dick          | Herman Melville  |
|     103 | Pride and Prejudice | Jane Austen      |
|     104 | Jane Eyre          | Charlotte Bronte |
|     105 | Animal Farm        | George Orwell    |
+---------+--------------------+------------------+
5 rows in set (0.00 sec)

mysql>
```

INSERT INTO iss_rec VALUES(1,'2024-07-01',5001,101);

INSERT INTO iss_rec VALUES(2,'2024-07-02',5002,102);

INSERT INTO iss_rec VALUES(3,'2024-07-02',5003,103);

INSERT INTO iss_rec VALUES(4,'2024-07-04',5004,104);

INSERT INTO iss_rec VALUES(5,'2024-07-05',5005,105);

```
MySQL 8.0 Command Line Client
mysql> select * from iss_rec;
+--------+------------+--------+---------+
| iss_no | iss_date   | mem_no | book_no |
+--------+------------+--------+---------+
|      1 | 2024-07-01 |   5001 |     101 |
|      2 | 2024-07-02 |   5002 |     102 |
|      3 | 2024-07-02 |   5003 |     103 |
|      4 | 2024-07-04 |   5004 |     104 |
|      5 | 2024-07-05 |   5005 |     105 |
+--------+------------+--------+---------+
5 rows in set (0.00 sec)

mysql>
```

3. List all the student names along with their membership numbers. Ensure the column names are stud_name and mem_no.

```
SELECT stud_name,mem_no
        FROM student s,membership m
        WHERE s.stud_no=m.stud_no;
```

```
MySQL 8.0 Command Line Client
+-----------------+--------+
| stud_name       | mem_no |
+-----------------+--------+
| John Doe        |   5001 |
| Jane Smith      |   5002 |
| Michael Johnson |   5003 |
| Emily Brown     |   5004 |
| David Lee       |   5005 |
+-----------------+--------+
5 rows in set (0.00 sec)

mysql>
```

4. List all the issue numbers for the date 2024-07-02 along with the student names and book names. Ensure the column names are iss_no, stud_name, and book_name.

```
SELECT iss_no,stud_name,book_name
        FROM student s,book b,iss_rec i,membership m
        WHERE b.book_no=i.book_no AND m.mem_no=i.mem_no AND
s.stud_no=m.stud_no AND iss_date='2024-07-02';
```

```
MySQL 8.0 Command Line Client
+--------+-----------------+---------------------+
| iss_no | stud_name       | book_name           |
+--------+-----------------+---------------------+
|      2 | Jane Smith      | Moby Dick           |
|      3 | Michael Johnson | Pride and Prejudice |
+--------+-----------------+---------------------+
2 rows in set (0.00 sec)

mysql>
```

5. List the student number and the name of the student who borrowed a book authored by George Orwell. Ensure the column names are stud_no and stud_name.

```
SELECT s.stud_no,s.stud_name
        FROM student s,iss_rec i,book b,membership m
        WHERE i.book_no=b.book_no AND i.mem_no=m.mem_no AND
m.stud_no=s.stud_no AND b.author='George Orwell';
```

```
MySQL 8.0 Command Line Client
+---------+-----------+
| stud_no | stud_name |
+---------+-----------+
|    1005 | David Lee |
+---------+-----------+
1 row in set (0.00 sec)

mysql>
```

6. Count the number of books each student borrowed. Group the results by the student number and name, and order the results by student number. Ensure the column names are stud_no, stud_name, and books_borrowed.

```
SELECT s.stud_no, s.stud_name, COUNT(i.iss_no)
        AS books_borrowed
        FROM student s,membership m,iss_rec i
        WHERE s.stud_no = m.stud_no AND m.mem_no = i.mem_no
        GROUP BY s.stud_no, s.stud_name
        ORDER BY s.stud_no;
```

```
MySQL 8.0 Command Line Client
+---------+-----------------+----------------+
| stud_no | stud_name       | books_borrowed |
+---------+-----------------+----------------+
|    1001 | John Doe        |              1 |
|    1002 | Jane Smith      |              1 |
|    1003 | Michael Johnson |              1 |
|    1004 | Emily Brown     |              1 |
|    1005 | David Lee       |              1 |
+---------+-----------------+----------------+
5 rows in set (0.00 sec)

mysql>
```

7. List the books borrowed by the student with stud_no as 1005. Ensure the column name is book_name.

```
SELECT b.book_name
        FROM student s,iss_rec i,book b,membership m
        WHERE i.book_no=b.book_no AND i.mem_no=m.mem_no AND
m.stud_no=s.stud_no AND s.stud_no=1005;
```

```
MySQL 8.0 Command Line Client
+-------------+
| book_name   |
+-------------+
| Animal Farm |
+-------------+
1 row in set (0.00 sec)

mysql>
```

8. List the book details (book number, book name, and author) for the books issued on 2024-07-04. Ensure the column names are book_no, book_name, and author.

```
SELECT b.book_no,b.book_name,b.author
        FROM book b,iss_rec i
        WHERE i.book_no=b.book_no AND iss_date='2024-07-04';
```

```
MySQL 8.0 Command Line Client
+---------+-----------+-----------------+
| book_no | book_name | author          |
+---------+-----------+-----------------+
|     104 | Jane Eyre | Charlotte Bronte |
+---------+-----------+-----------------+
1 row in set (0.00 sec)

mysql>
```

9. Create a view named iss_rec_view that lists the issue number, issue date, student name, and book name. Ensure the column names are iss_no, iss_date, stud_name, and book_name.

```
CREATE VIEW iss_rec_view
        AS SELECT i.iss_no,i.iss_date,s.stud_name,b.book_name
        FROM student s,book b,iss_rec i,membership m
        WHERE i.book_no=b.book_no AND i.mem_no=m.mem_no AND
m.stud_no=s.stud_no;
SELECT * FROM iss_rec_view;
```

```
MySQL 8.0 Command Line Client
mysql> SELECT * FROM iss_rec_view;
+--------+------------+----------------+--------------------+
| iss_no | iss_date   | stud_name      | book_name          |
+--------+------------+----------------+--------------------+
|      1 | 2024-07-01 | John Doe       | The Great Gatsby   |
|      2 | 2024-07-02 | Jane Smith     | Moby Dick          |
|      3 | 2024-07-02 | Michael Johnson | Pride and Prejudice |
|      4 | 2024-07-04 | Emily Brown    | Jane Eyre          |
|      5 | 2024-07-05 | David Lee      | Animal Farm        |
+--------+------------+----------------+--------------------+
5 rows in set (0.00 sec)

mysql>
```

10. Create a view named daily_issues_view that lists daily issue details (issue number, issue date, student name, and book name) for the period from 2024-07-02 to 2024-07-04. Ensure the column names are iss_no, iss_date, stud_name, and book_name.

```
CREATE VIEW daily_issues_view
        AS SELECT i.iss_no, i.iss_date, s.stud_name, b.book_name
        FROM iss_rec i,student s,book b, membership m
        WHERE i.mem_no = m.mem_no AND m.stud_no = s.stud_no AND i.book_no
= b.book_no AND i.iss_date BETWEEN '2024-07-02' AND '2024-07-04';
SELECT * FROM daily_issues_view;
```

MySQL 8.0 Command Line Client

```
mysql> SELECT * FROM daily_issues_view;
+--------+------------+----------------+---------------------+
| iss_no | iss_date   | stud_name      | book_name           |
+--------+------------+----------------+---------------------+
|      2 | 2024-07-02 | Jane Smith     | Moby Dick           |
|      3 | 2024-07-02 | Michael Johnson | Pride and Prejudice |
|      4 | 2024-07-04 | Emily Brown    | Jane Eyre           |
+--------+------------+----------------+---------------------+
3 rows in set (0.00 sec)

mysql>
```

Week 3. Database Schema for a Employee-pay scenario

Employee (**emp_id:**integer,emp_name:string)

Department (**dept_id:integer**,dept_name:string)

Pay details (**emp_id : integer**,**dept_id: integer**, basic: integer, deductions: integer, additions: integer, DOJ: date)

Payroll **(emp_id : integer**, pay_date: date)

For the above schema, perform the following—

2. **Create Tables:** Write the SQL statements to create the employee, department, paydetails, and payroll tables as described above. Ensure that the paydetails table references both the employee and department tables, and the payroll table references the employee table. Include appropriate primary keys, foreign keys, and basic integrity constraints.

```
CREATE TABLE employee(
        emp_id int PRIMARY KEY,
        emp_name varchar(50));
```

MySQL 8.0 Command Line Client

```
mysql> desc employee;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| emp_id   | int         | NO   | PRI | NULL    |       |
| emp_name | varchar(50) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
2 rows in set (0.04 sec)

mysql>
```

CREATE TABLE department (

dept_id int PRIMARY KEY,

dept_name varchar(50));

MySQL 8.0 Command Line Client

```
mysql> desc department;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| dept_id   | int         | NO   | PRI | NULL    |       |
| dept_name | varchar(50) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql>
```

CREATE TABLE paydetails(

emp_id int,dept_id int,basic int,deductions int,additions int,doj date,

FOREIGN KEY(emp_id) REFERENCES employee(emp_id),

FOREIGN KEY(dept_id) REFERENCES department (dept_id));

MySQL 8.0 Command Line Client

```
mysql> desc paydetails;
+------------+------+------+-----+---------+-------+
| Field      | Type | Null | Key | Default | Extra |
+------------+------+------+-----+---------+-------+
| emp_id     | int  | YES  | MUL | NULL    |       |
| dept_id    | int  | YES  | MUL | NULL    |       |
| basic      | int  | YES  |     | NULL    |       |
| deductions | int  | YES  |     | NULL    |       |
| additions  | int  | YES  |     | NULL    |       |
| doj        | date | YES  |     | NULL    |       |
+------------+------+------+-----+---------+-------+
6 rows in set (0.00 sec)

mysql>
```

```
CREATE TABLE payroll(
    emp_id int ,pay_date date,
    FOREIGN KEY(emp_id) REFERENCES employee(emp_id));
```

MySQL 8.0 Command Line Client

```
mysql> desc payroll;
+----------+------+------+-----+---------+-------+
| Field    | Type | Null | Key | Default | Extra |
+----------+------+------+-----+---------+-------+
| emp_id   | int  | YES  | MUL | NULL    |       |
| pay_date | date | YES  |     | NULL    |       |
+----------+------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql>
```

3. Insert Data: Insert 5 records into each of the tables with the given data.

employee:

emp_id	emp_name
1	John Doe
2	Jane Smith
3	Michael Johnson
4	Emily Brown
5	David Lee

department:

dept_id	dept_name
101	Human Resources
102	Finance
103	IT
104	Marketing
105	Sales

paydetails:

emp_id	dept_id	basic	deductions	additions	doj
1	101	50000	5000	2000	2022-01-15
2	102	60000	6000	2500	2021-03-22
3	103	55000	4500	3000	2023-07-01
4	104	52000	4000	1500	2020-10-30
5	105	58000	5500	1800	2019-08-14

payroll:

emp_id	pay_date
1	2024-07-01
2	2024-07-02
3	2024-07-02
4	2024-07-03
5	2024-07-05

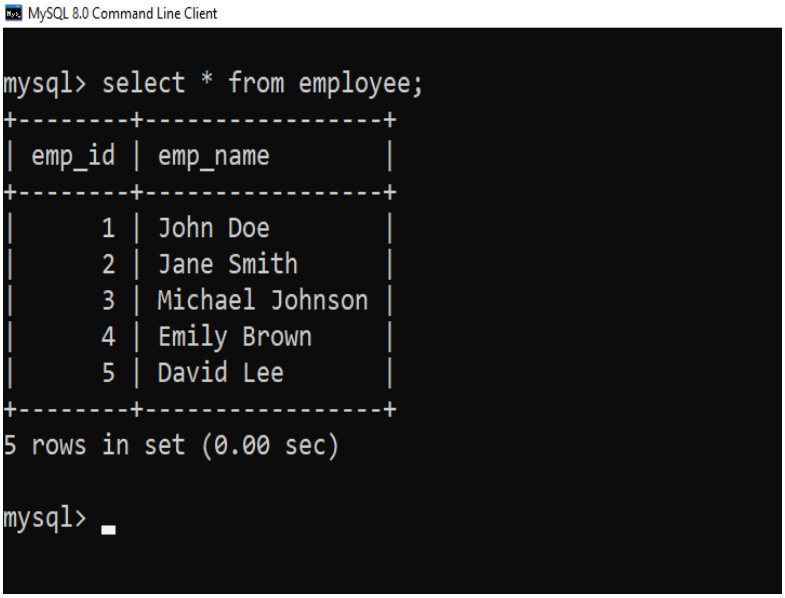
INSERT INTO employee VALUES(1,'John Doe');

INSERT INTO employee VALUES(2,'Jane Smith');

INSERT INTO employee VALUES(3,'Michael Johnson');

INSERT INTO employee VALUES(4,'Emily Brown');

INSERT INTO employee VALUES(5,'David Lee');



INSERT INTO department VALUES(101,'Human Resources');

INSERT INTO department VALUES(102,'Finance');

INSERT INTO department VALUES(103,'IT');

INSERT INTO department VALUES(104,'Marketing');

INSERT INTO department VALUES(105,'Sales');

MySQL 8.0 Command Line Client

```
mysql> select * from department;
+---------+-----------------+
| dept_id | dept_name       |
+---------+-----------------+
|     101 | Human Resources |
|     102 | Finance         |
|     103 | IT              |
|     104 | Marketing       |
|     105 | Sales           |
+---------+-----------------+
5 rows in set (0.00 sec)

mysql>
```

INSERT INTO paydetails VALUES(1,101,50000,5000,2000,'2022-01-15');

INSERT INTO paydetails VALUES(2,102,60000,6000,2500,'2021-03-22');

INSERT INTO paydetails VALUES(3,103,55000,4500,3000,'2023-07-01');

INSERT INTO paydetails VALUES(4,104,52000,4000,1500,'2020-10-30');

INSERT INTO paydetails VALUES(5,105,58000,5500,1800,'2019-08-14');

MySQL 8.0 Command Line Client

```
mysql> select * from paydetails;
+--------+---------+-------+------------+-----------+------------+
| emp_id | dept_id | basic | deductions | additions | doj        |
+--------+---------+-------+------------+-----------+------------+
|      1 |     101 | 50000 |       5000 |      2000 | 2022-01-15 |
|      2 |     102 | 60000 |       6000 |      2500 | 2021-03-22 |
|      3 |     103 | 55000 |       4500 |      3000 | 2023-07-01 |
|      4 |     104 | 52000 |       4000 |      1500 | 2020-10-30 |
|      5 |     105 | 58000 |       5500 |      1800 | 2019-08-14 |
+--------+---------+-------+------------+-----------+------------+
5 rows in set (0.00 sec)

mysql>
```

```
INSERT INTO payroll VALUES(1,'2024-07-01');
INSERT INTO payroll VALUES(2,'2024-07-02');
INSERT INTO payroll VALUES(3,'2024-07-02');
INSERT INTO payroll VALUES(4,'2024-07-03');
INSERT INTO payroll VALUES(5,'2024-07-05');
```

MySQL 8.0 Command Line Client

```
mysql> select * from payroll;
+--------+------------+
| emp_id | pay_date   |
+--------+------------+
|      1 | 2024-07-01 |
|      2 | 2024-07-02 |
|      3 | 2024-07-02 |
|      4 | 2024-07-03 |
|      5 | 2024-07-05 |
+--------+------------+
5 rows in set (0.00 sec)

mysql> _
```

4. Retrieve the emp_id and emp_name of employees, ordered by department name (dept_name).

```
SELECT e.emp_id,e.emp_name
    FROM employee e,department d,paydetails p
    WHERE e.emp_id=p.emp_id AND d.dept_id=p.dept_id
    ORDER BY d.dept_name;
```

MySQL 8.0 Command Line Client

```
+--------+-----------------+
| emp_id | emp_name        |
+--------+-----------------+
|      2 | Jane Smith      |
|      1 | John Doe        |
|      3 | Michael Johnson |
|      4 | Emily Brown     |
|      5 | David Lee       |
+--------+-----------------+
5 rows in set (0.00 sec)

mysql>
```

5. List the names (emp_name) of employees who joined after March 22, 2021.

```
SELECT e.emp_name
        FROM employee e,paydetails p
        WHERE e.emp_id=p.emp_id AND doj>'2021-03-22';
```

MySQL 8.0 Command Line Client

```
+-----------------+
| emp_name        |
+-----------------+
| John Doe        |
| Michael Johnson |
+-----------------+
2 rows in set (0.00 sec)

mysql>
```

6. Display the dept_id, dept_name, and count of employees (emp_id) in each department.

```
SELECT d.dept_id,d.dept_name,count(e.emp_id)
        FROM employee e,department d,paydetails p
        WHERE e.emp_id=p.emp_id AND d.dept_id=p.dept_id
        GROUP BY d.dept_id;
```

MySQL 8.0 Command Line Client

```
+---------+----------------+-----------------+
| dept_id | dept_name      | count(e.emp_id) |
+---------+----------------+-----------------+
|     101 | Human Resources |               1 |
|     102 | Finance        |               1 |
|     103 | IT             |               1 |
|     104 | Marketing      |               1 |
|     105 | Sales          |               1 |
+---------+----------------+-----------------+
5 rows in set (0.00 sec)

mysql>
```

7. Find the names (emp_name) of employees whose net salary (basic salary + additions - deductions) exceeds 50,000.

```
SELECT e.emp_name
        FROM employee e,paydetails p
        WHERE e.emp_id = p.emp_id AND (basic+additions-deductions) >50000;
```

MySQL 8.0 Command Line Client

```
+------------------+
| emp_name         |
+------------------+
| Jane Smith       |
| Michael Johnson  |
| David Lee        |
+------------------+
3 rows in set (0.00 sec)

mysql>
```

8. Fetch the emp_name, dept_id, dept_name, basic, additions, deductions, doj, and pay_date for the employee with emp_id 5.

```
SELECT
e.emp_name,d.dept_id,d.dept_name,p.basic,p.additions,p.deductions,p.doj,r.pay_date
        FROM employee e,department d,paydetails p,payroll r
        WHERE e.emp_id=p.emp_id AND d.dept_id=p.dept_id
        AND e.emp_id=r.emp_id
        AND e.emp_id=5;
```

MySQL 8.0 Command Line Client

emp_name	dept_id	dept_name	basic	additions	deductions	doj	pay_date
David Lee	105	Sales	58000	1800	5500	2019-08-14	2024-07-05

```
1 row in set (0.00 sec)

mysql>
```

9. Create a view named employee_pay_details that includes emp_name, dept_name, basic, deductions, and netsalary (calculated as basic + additions - deductions) for each employee.

```
CREATE VIEW employee_pay_details
        AS SELECT e.emp_name,d.dept_name,basic,deductions,
                        (basic-deductions+additions)
        AS netsalary
        FROM employee e,department d,paydetails p
        WHERE e.emp_id=p.emp_id AND d.dept_id=p.dept_id;

SELECT * FROM employee_pay_details;
```

MySQL 8.0 Command Line Client

```
mysql> SELECT * FROM employee_pay_details;
+----------------+----------------+-------+------------+----------+
| emp_name       | dept_name      | basic | deductions | netsalary |
+----------------+----------------+-------+------------+----------+
| John Doe       | Human Resources | 50000 |       5000 |    47000 |
| Jane Smith     | Finance        | 60000 |       6000 |    56500 |
| Michael Johnson | IT             | 55000 |       4500 |    53500 |
| Emily Brown    | Marketing      | 52000 |       4000 |    49500 |
| David Lee      | Sales          | 58000 |       5500 |    54300 |
+----------------+----------------+-------+------------+----------+
5 rows in set (0.00 sec)

mysql>
```

10. Create another view named employee_net_salary that displays emp_name and netsalary (calculated as basic + additions - deductions) for each employee.

```
CREATE VIEW employee_net_salary
        AS SELECT e.emp_name,(basic+additions-deductions)
        AS netsalary
        FROM employee e,paydetails p
        WHERE e.emp_id=p.emp_id;


SELECT * FROM employee_net_salary;
```

MySQL 8.0 Command Line Client

```
mysql> SELECT * FROM employee_net_salary;
+-----------------+-----------+
| emp_name        | netsalary |
+-----------------+-----------+
| John Doe        |     47000 |
| Jane Smith      |     56500 |
| Michael Johnson |     53500 |
| Emily Brown     |     49500 |
| David Lee       |     54300 |
+-----------------+-----------+
5 rows in set (0.00 sec)

mysql>
```

Week 4. Database Schema for a Video Library scenario

Customer(cust_no: integer,cust_name: string)

Membership(**Mem_no: integer**, **cust_no: integer**)

Cassette(**cass_no:integer**, cass_name:string, Language: String)

Iss_rec(**iss_no: integer**, iss_date: date, **mem_no: integer**, **cass_no: integer**)

For the above schema, perform the following—

1. **Create Tables:** Write the SQL statements to create the employee, department, paydetails, and payroll tables as described above. Ensure that the paydetails table references both the employee and department tables, and the payroll table references the employee table. Include appropriate primary keys, foreign keys, and basic integrity constraints.

```
CREATE TABLE customer(
        cust_no INT PRIMARY KEY,
        cust_name VARCHAR(50));
```

MySQL 8.0 Command Line Client

```
mysql> desc customer;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| cust_no   | int         | NO   | PRI | NULL    |       |
| cust_name | varchar(50) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql>
```

```
CREATE TABLE membership(
        mem_no INT PRIMARY KEY,
        cust_no INT,
        FOREIGN KEY(cust_no)REFERENCES customer(cust_no));
```

MySQL 8.0 Command Line Client

```
mysql> desc membership;
+---------+------+------+-----+---------+-------+
| Field   | Type | Null | Key | Default | Extra |
+---------+------+------+-----+---------+-------+
| mem_no  | int  | NO   | PRI | NULL    |       |
| cust_no | int  | YES  | MUL | NULL    |       |
+---------+------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql>
```

CREATE TABLE cassette(

cass_no INT PRIMARY KEY,

cass_name VARCHAR(100),

language varchar(50));

MySQL 8.0 Command Line Client

```
mysql> desc cassette;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| cass_no   | int          | NO   | PRI | NULL    |       |
| cass_name | varchar(100) | YES  |     | NULL    |       |
| language  | varchar(50)  | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql>
```

CREATE TABLE iss_rec(

iss_no INT PRIMARY KEY,

iss_date DATE,mem_no INT ,

cass_no INT,

FOREIGN KEY(mem_no)REFERENCES membership(mem_no),

FOREIGN KEY(cass_no)REFERENCES cassette(cass_no));

MySQL 8.0 Command Line Client

```
mysql> desc iss_rec;
+----------+------+------+-----+---------+-------+
| Field    | Type | Null | Key | Default | Extra |
+----------+------+------+-----+---------+-------+
| iss_no   | int  | NO   | PRI | NULL    |       |
| iss_date | date | YES  |     | NULL    |       |
| mem_no   | int  | YES  | MUL | NULL    |       |
| cass_no  | int  | YES  | MUL | NULL    |       |
+----------+------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql>
```

2. **Insert Data**: Insert 5 records into each of the tables with the given data.

customer:

cust_no	cust_name
1001	John Doe
1002	Jane Smith
1003	Michael Johnson
1004	Emily Brown
1005	David Lee

membership:

mem_no	cust_no
5001	1001
5002	1002
5003	1003
5004	1004
5005	1005

cassette:

cass_no	cass_name	language
2001	The Great Gatsby	English
2002	Moby Dick	English
2003	Pride and Prejudice	Spanish
2004	Jane Eyre	Spanish
2005	Animal Farm	English

iss_rec:

iss_no	iss_date	mem_no	cass_no
1	2024-07-01	5001	2001
2	2024-07-02	5002	2002
3	2024-07-02	5003	2003
4	2024-07-04	5004	2004
5	2024-07-05	5005	2005

```
INSERT INTO customer VALUES(1001,'John Doe');
INSERT INTO customer VALUES(1002,'Jane Smith');
INSERT INTO customer VALUES(1003,'Michael Johnson');
INSERT INTO customer VALUES(1004,'Emily Brown');
INSERT INTO customer VALUES(1005,'David Lee');
```

MySQL 8.0 Command Line Client

```
mysql> select * from customer;
+---------+-----------------+
| cust_no | cust_name       |
+---------+-----------------+
|    1001 | John Doe        |
|    1002 | Jane Smith      |
|    1003 | Michael Johnson |
|    1004 | Emily Brown     |
|    1005 | David Lee       |
+---------+-----------------+
5 rows in set (0.00 sec)

mysql>
```

```
INSERT INTO membership VALUES(5001,1001);
INSERT INTO membership VALUES(5002,1002);
INSERT INTO membership VALUES(5003,1003);
INSERT INTO membership VALUES(5004,1004);
INSERT INTO membership VALUES(5005,1005);
```

MySQL 8.0 Command Line Client

```
mysql> select * from membership;
+--------+---------+
| mem_no | cust_no |
+--------+---------+
|   5001 |    1001 |
|   5002 |    1002 |
|   5003 |    1003 |
|   5004 |    1004 |
|   5005 |    1005 |
+--------+---------+
5 rows in set (0.00 sec)

mysql>
```

```
INSERT INTO cassette VALUES(2001,'The Great Gatsby','English');
INSERT INTO cassette VALUES(2002,'Moby Dick','English');
INSERT INTO cassette VALUES(2003,'Pride and Prejudice','Spanish');
INSERT INTO cassette VALUES(2004,'Jane Eyre','Spanish');
INSERT INTO cassette VALUES(2005,'Animal Farm','English');
```

MySQL 8.0 Command Line Client

```
mysql> select * from cassette;
+---------+--------------------+----------+
| cass_no | cass_name          | language |
+---------+--------------------+----------+
|    2001 | The Great Gatsby   | English  |
|    2002 | Moby Dick          | English  |
|    2003 | Pride and Prejudice | Spanish  |
|    2004 | Jane Eyre          | Spanish  |
|    2005 | Animal Farm        | English  |
+---------+--------------------+----------+
5 rows in set (0.00 sec)

mysql>
```

```
INSERT INTO iss_rec VALUES(1,'2024-07-01',5001,2001);
INSERT INTO iss_rec VALUES(2,'2024-07-02',5002,2002);
INSERT INTO iss_rec VALUES(3,'2024-07-02',5003,2003);
INSERT INTO iss_rec VALUES(4,'2024-07-04',5004,2004);
INSERT INTO iss_rec VALUES(5,'2024-07-05',5005,2005);
```

MySQL 8.0 Command Line Client

```
mysql> select * from iss_rec;
+--------+------------+--------+---------+
| iss_no | iss_date   | mem_no | cass_no |
+--------+------------+--------+---------+
|      1 | 2024-07-01 |   5001 |    2001 |
|      2 | 2024-07-02 |   5002 |    2002 |
|      3 | 2024-07-02 |   5003 |    2003 |
|      4 | 2024-07-04 |   5004 |    2004 |
|      5 | 2024-07-05 |   5005 |    2005 |
+--------+------------+--------+---------+
5 rows in set (0.00 sec)

mysql>
```

3. List all the customer names along with their membership numbers.

```
SELECT cust_name,mem_no
        FROM customer c,membership m
        WHERE c.cust_no=m.cust_no;
```

MySQL 8.0 Command Line Client

```
+------------------+--------+
| John Doe         |   5001 |
| Jane Smith       |   5002 |
| Michael Johnson  |   5003 |
| Emily Brown      |   5004 |
| David Lee        |   5005 |
+------------------+--------+
5 rows in set (0.00 sec)

mysql>
```

4. List the issue number for 2024-07-04 along with the customer names and cassette names.

```
SELECT iss_no,cust_name,cass_name
        FROM customer c,cassette ca,membership m,iss_rec i
        WHERE ca.cass_no=i.cass_no AND m.mem_no=i.mem_no
        AND c.cust_no=m.cust_no
        AND i.iss_date='2024-07-04';
```

MySQL 8.0 Command Line Client

```
+--------+-------------+-----------+
| iss_no | cust_name   | cass_name |
+--------+-------------+-----------+
|      4 | Emily Brown | Jane Eyre |
+--------+-------------+-----------+
1 row in set (0.00 sec)

mysql>
```

5. List the details of the customer (customer number and customer name) who has borrowed the cassette titled "Moby Dick".

```
SELECT c.cust_no,c.cust_name
        FROM customer c,membership m,cassette ca,iss_rec i
        WHERE i.mem_no=m.mem_no AND i.cass_no=ca.cass_no
         AND c.cust_no=m.cust_no
        AND ca.cass_name='Moby Dick';
```

MySQL 8.0 Command Line Client

```
+---------+------------+
| cust_no | cust_name  |
+---------+------------+
|    1002 | Jane Smith |
+---------+------------+
1 row in set (0.00 sec)

mysql>
```

6. List the cassette names of the customer with membership number 5002 has borrowed.

```
SELECT ca.cass_name
        FROM customer c,membership m,cassette ca,iss_rec i
        WHERE i.mem_no=m.mem_no AND i.cass_no=ca.cass_no
         AND m.cust_no=c.cust_no
        AND m.mem_no=5002;
```

MySQL 8.0 Command Line Client

```
+-----------+
| cass_name |
+-----------+
| Moby Dick |
+-----------+
1 row in set (0.00 sec)

mysql>
```

7. List the cassette details (cassette number, cassette name, and language) for the cassettes issued on 2024-07-05.

```
SELECT ca.cass_no,ca.cass_name,ca.language
        FROM cassette ca,iss_rec i
        WHERE ca.cass_no=i.cass_no AND i.iss_date='2024-07-05';
```

MySQL 8.0 Command Line Client

```
+---------+-------------+----------+
| cass_no | cass_name   | language |
+---------+-------------+----------+
|    2005 | Animal Farm | English  |
+---------+-------------+----------+
1 row in set (0.00 sec)

mysql>
```

8. Create a view named issue_details that lists the issue number (iss_no), issue date (iss_date), customer name (cust_name), and cassette name (cass_name).

```
CREATE VIEW issue_details
        AS SELECT i.iss_no,i.iss_date,c.cust_name,ca.cass_name
        FROM customer c,membership m,cassette ca,iss_rec i
        WHERE m.mem_no=i.mem_no AND i.cass_no=ca.cass_no
         AND c.cust_no=m.cust_no;

SELECT * FROM issue_details;
```

MySQL 8.0 Command Line Client

```
mysql> SELECT * FROM issue_details;
+--------+------------+----------------+---------------------+
| iss_no | iss_date   | cust_name      | cass_name           |
+--------+------------+----------------+---------------------+
|      1 | 2024-07-01 | John Doe       | The Great Gatsby    |
|      2 | 2024-07-02 | Jane Smith     | Moby Dick           |
|      3 | 2024-07-02 | Michael Johnson | Pride and Prejudice |
|      4 | 2024-07-04 | Emily Brown    | Jane Eyre           |
|      5 | 2024-07-05 | David Lee      | Animal Farm         |
+--------+------------+----------------+---------------------+
5 rows in set (0.00 sec)

mysql>
```

9. Create a view named issues_date_wise that lists issue records (issue number, issue date, customer name, and cassette name) within the specific date range from 2024-07-01 to 2024-07-04.

```
CREATE VIEW issues_date_wise
        AS SELECT i.iss_no, i.iss_date,c.cust_name,ca.cass_name
        FROM iss_rec i, customer c,cassette ca, membership m
        WHERE m.mem_no=i.mem_no AND i.cass_no=ca.cass_no AND
c.cust_no=m.cust_no
        AND iss_date BETWEEN '2024-07-01' AND '2024-07-04';
```

SELECT * FROM issues_date_wise;

MySQL 8.0 Command Line Client

```
mysql> SELECT * FROM issues_date_wise;
+--------+------------+----------------+---------------------+
| iss_no | iss_date   | cust_name      | cass_name           |
+--------+------------+----------------+---------------------+
|      1 | 2024-07-01 | John Doe       | The Great Gatsby    |
|      2 | 2024-07-02 | Jane Smith     | Moby Dick           |
|      3 | 2024-07-02 | Michael Johnson | Pride and Prejudice |
|      4 | 2024-07-04 | Emily Brown    | Jane Eyre           |
+--------+------------+----------------+---------------------+
4 rows in set (0.00 sec)

mysql>
```

Week 5. Database Schema for a student-Lab scenario

class:

class(VARCHAR): Unique identifier for the class or grade.
descrip(VARCHAR): Description of the class or grade.

student:

stud_no(INT): Unique identifier for each student.
stud_name(VARCHAR): Name of the student.
class(VARCHAR): Class or grade level of the student, foreign key referencing class.

lab:

mach_no(INT): Unique identifier for each lab machine.
lab_no(INT): Lab number where the machine is located.
description(VARCHAR): Description of the lab machine.

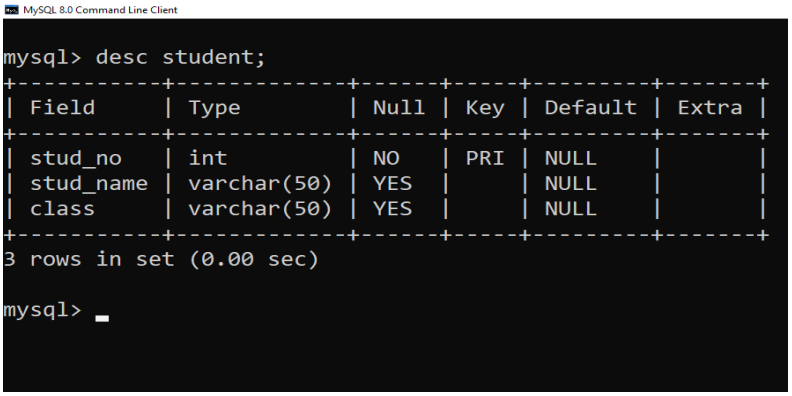
allotment:

stud_no(INT): Student identifier, foreign key referencing student.
mach_no(INT): Machine identifier, foreign key referencing lab.
day_of_week(VARCHAR): Day of the week when the machine is allotted to the student.

For the above schema, perform the following—

1. **Create Tables:** Write the SQL statements to create the student, class, lab, and allotment tables as described above. Include appropriate primary keys, foreign keys, and basic integrity constraints.

```
CREATE TABLE student(
        stud_no INT PRIMARY KEY,
        stud_name VARCHAR(50),
        class VARCHAR(50));
```




```
CREATE TABLE class (
        class VARCHAR(50),
        descrip VARCHAR(50));
```

MySQL 8.0 Command Line Client

```
mysql> desc class;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| class   | varchar(50) | YES  |     | NULL    |       |
| descrip | varchar(50) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql>
```

CREATE TABLE lab(
mach_no INT PRIMARY KEY,
lab_no INT,
description VARCHAR(50));

MySQL 8.0 Command Line Client

```
mysql> desc lab;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| mach_no     | int         | NO   | PRI | NULL    |       |
| lab_no      | int         | YES  |     | NULL    |       |
| description | varchar(50) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql>
```

CREATE TABLE allotment(
stud_no INT REFERENCES student(stud_no),
mach_no INT REFERENCES lab(mach_no),
day_of_week VARCHAR(50));

MySQL 8.0 Command Line Client

```
mysql> desc allotment;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| stud_no     | int         | YES  |     | NULL    |       |
| mach_no     | int         | YES  |     | NULL    |       |
| day_of_week | varchar(50) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql>
```

2. **Insert Data**: Insert 5 records into each of the tables with the given data.

student:

stud_no	stud_name	class
1	Alice Smith	Biology 101
2	Bob Johnson	Chemistry 101
3	Carol Williams	Physics 101
4	David Brown	Mathematics 101
5	Eva Davis	Computer Science 101

class:

class	descrip
Biology 101	Introduction to Biology
Chemistry 101	Basics of Chemistry
Physics 101	Fundamentals of Physics
Mathematics 101	Basic Mathematics
Computer Science 101	Introduction to Computer Science

lab:

mach_no	lab_no	description
1	101	Microscope
2	101	Centrifuge
3	102	Spectrometer
4	103	Oscilloscope
5	104	Computer Workstation

allotment:

stud_no	mach_no	day_of_week
1	1	Monday
2	2	Wednesday
3	3	Tuesday
4	4	Thursday
5	5	Friday

INSERT INTO student VALUES(1,'Alice Smith','Biology 101');

INSERT INTO student VALUES(2,'Bob Johnson','Chemistry 101');

INSERT INTO student VALUES(3,'Carol Williams','Physics 101');

INSERT INTO student VALUES(4,'David Brown','Mathematics 101');

INSERT INTO student VALUES(5,'Eva Davis','Computer Science 101');

MySQL 8.0 Command Line Client

```
mysql> select * from student;
+---------+----------------+----------------------+
| stud_no | stud_name      | class                |
+---------+----------------+----------------------+
|       1 | Alice Smith    | Biology 101          |
|       2 | Bob Johnson    | Chemistry 101        |
|       3 | Carol Williams | Physics 101          |
|       4 | David Brown    | Mathematics 101      |
|       5 | Eva Davis      | Computer Science 101 |
+---------+----------------+----------------------+
5 rows in set (0.00 sec)

mysql>
```

INSERT INTO class VALUES('Biology 101','Introduction to Biology');

INSERT INTO class VALUES('Chemistry 101','Basics of Chemistry');

INSERT INTO class VALUES('Physics 101','Fundamentals of Physics');

INSERT INTO class VALUES('Mathematics 101','Basic Mathematics');

INSERT INTO class VALUES('Computer Science 101','Introduction to Computer Science');

MySQL 8.0 Command Line Client

```
mysql> select * from class;
+----------------------+---------------------------------+
| class                | descrip                         |
+----------------------+---------------------------------+
| Biology 101          | Introduction to Biology         |
| Chemistry 101        | Basics of Chemistry             |
| Physics 101          | Fundamentals of Physics         |
| Mathematics 101      | Basic Mathematics               |
| Computer Science 101 | Introduction to Computer Science |
+----------------------+---------------------------------+
5 rows in set (0.00 sec)

mysql>
```

INSERT INTO lab VALUES(1,101,'Microscope');

INSERT INTO lab VALUES(2,101,'Centrifuge');

INSERT INTO lab VALUES(3,102,'Spectrometer');

INSERT INTO lab VALUES(4,103,'Oscilloscope');

INSERT INTO lab VALUES(5,104,'Computer Workstation');

MySQL 8.0 Command Line Client

```
mysql> select * from lab;
+---------+--------+----------------------+
| mach_no | lab_no | description          |
+---------+--------+----------------------+
|       1 |    101 | Microscope           |
|       2 |    101 | Centrifuge           |
|       3 |    102 | Spectrometer         |
|       4 |    103 | Oscilloscope         |
|       5 |    104 | Computer Workstation |
+---------+--------+----------------------+
5 rows in set (0.00 sec)

mysql>
```

INSERT INTO allotment VALUES(1,1,'Monday');

INSERT INTO allotment VALUES(2,2,'Wednesday');

INSERT INTO allotment VALUES(3,3,'Tuesday');

INSERT INTO allotment VALUES(4,4,'Thursday');

INSERT INTO allotment VALUES(5,5,'Friday');

MySQL 8.0 Command Line Client

```
mysql> select * from allotment;
+---------+---------+-------------+
| stud_no | mach_no | day_of_week |
+---------+---------+-------------+
|       1 |       1 | Monday      |
|       2 |       2 | Wednesday   |
|       3 |       3 | Tuesday     |
|       4 |       4 | Thursday    |
|       5 |       5 | Friday      |
+---------+---------+-------------+
5 rows in set (0.00 sec)

mysql>
```

3. List the student names, machine numbers, and lab numbers for each machine allotment.

```
SELECT s.stud_name,l.mach_no,l.lab_no
        FROM student s,lab l,allotment a
        WHERE a.stud_no=s.stud_no AND l.mach_no=a.mach_no;
```

MySQL 8.0 Command Line Client

```
+----------------+---------+--------+
| stud_name      | mach_no | lab_no |
+----------------+---------+--------+
| Alice Smith    |       1 |    101 |
| Bob Johnson    |       2 |    101 |
| Carol Williams |       3 |    102 |
| David Brown    |       4 |    103 |
| Eva Davis      |       5 |    104 |
+----------------+---------+--------+
5 rows in set (0.00 sec)

mysql>
```

4. List the total number of lab allotments day-wise. Ensure that the output includes each day of the week and the corresponding count of allotments, with the count column labeled as total_allotments. Group the results by day_of_week.

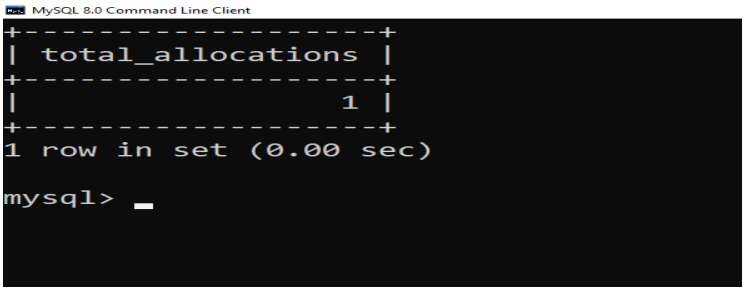
```
SELECT a.day_of_week,
        COUNT(l.lab_no) AS total_allotments
        FROM student s,class c,lab l,allotment a
        WHERE a.stud_no=s.stud_no AND c.class=s.class
         AND l.mach_no=a.mach_no
        GROUP BY a.day_of_week ;
```

MySQL 8.0 Command Line Client

```
+-------------+------------------+
| day_of_week | total_allotments |
+-------------+------------------+
| Monday      |                1 |
| Wednesday   |                1 |
| Tuesday     |                1 |
| Thursday    |                1 |
| Friday      |                1 |
+-------------+------------------+
5 rows in set (0.01 sec)

mysql>
```

5. Count how many machines have been allocated to students in the Biology class. The output should provide the total count of machine allocations, with the count column labeled as total_allocations.

```
SELECT COUNT(a.mach_no)
        AS total_allocations
        FROM student s,class c,lab l,allotment a
        WHERE a.stud_no=s.stud_no AND c.class=s.class
        AND l.mach_no=a.mach_no AND c.class='Biology 101'
        GROUP BY c.class,a.mach_no ;
```



6. Retrieve the machine allotment details for the student with stud_no 5. Your query should include the student's details (name and class), class description, machine details (machine number, lab number, and machine description), and the day of the week the machine is allotted.

```
SELECT
s.stud_no,s.stud_name,c.class,c.descrip,a.mach_no,l.lab_no,l.description,
a.day_of_week
        FROM student s,class c,lab l,allotment a
        WHERE a.stud_no=s.stud_no AND c.class=s.class
        AND l.mach_no=a.mach_no AND a.stud_no=5;
```

```
MySQL 8.0 Command Line Client
+---------+----------+--------------------+-------------------------------+---------+--------+---------------------+-----------
--+
| stud_no | stud_name | class              | descrip                       | mach_no | lab_no | description         | day_of_wee
k |
+---------+----------+--------------------+-------------------------------+---------+--------+---------------------+-----------
--+
|       5 | Eva Davis | Computer Science 101 | Introduction to Computer Science |       5 |    104 | Computer Workstation | Friday
  |
+---------+----------+--------------------+-------------------------------+---------+--------+---------------------+-----------
--+
1 row in set (0.00 sec)

mysql>
```

7. Count the number of students who have been allocated machines, grouped by their class. The count column should be labeled as student_count

```
SELECT c.class,
        COUNT(s.stud_no)
        AS student_count
        FROM student s,class c
        WHERE s.class=c.class
        GROUP BY c.class;
```

```
MySQL 8.0 Command Line Client
+----------------------+---------------+
| class                | student_count |
+----------------------+---------------+
| Biology 101          |             1 |
| Chemistry 101        |             1 |
| Physics 101          |             1 |
| Mathematics 101      |             1 |
| Computer Science 101 |             1 |
+----------------------+---------------+
5 rows in set (0.00 sec)

mysql>
```

8. Create a view named student_machine_allocations that lists out stud_no, stud_name, mach_no, lab_no, and day_of_week

```
CREATE VIEW student_machine_allocations
        AS SELECT s.stud_no,s.stud_name,a.mach_no,l.lab_no,a.day_of_week
        FROM student s,lab l,allotment a
        WHERE a.stud_no=s.stud_no AND l.mach_no=a.mach_no;

SELECT * FROM student_machine_allocations;
```

MySQL 8.0 Command Line Client

```
mysql> SELECT * FROM student_machine_allocations;
+---------+----------------+---------+--------+-------------+
| stud_no | stud_name      | mach_no | lab_no | day_of_week |
+---------+----------------+---------+--------+-------------+
|       1 | Alice Smith    |       1 |    101 | Monday      |
|       2 | Bob Johnson    |       2 |    101 | Wednesday   |
|       3 | Carol Williams |       3 |    102 | Tuesday     |
|       4 | David Brown    |       4 |    103 | Thursday    |
|       5 | Eva Davis      |       5 |    104 | Friday      |
+---------+----------------+---------+--------+-------------+
5 rows in set (0.00 sec)

mysql>
```

9. Create a view named thursday_machine_allocations that lists out stud_no, stud_name, mach_no , lab_no , description, and day_of_week

```
        CREATE VIEW thursday_machine_allocations
        AS SELECT
s.stud_no,s.stud_name,a.mach_no,l.lab_no,l.description,a.day_of_week
        FROM allotment a,student s, lab l
        WHERE s.stud_no=a.stud_no AND l.mach_no=a.mach_no
        AND a.day_of_week='Thursday';

SELECT * FROM thursday_machine_allocations;
```

MySQL 8.0 Command Line Client

```
mysql>
mysql> SELECT * FROM thursday_machine_allocations;
+---------+-------------+---------+--------+--------------+-------------+
| stud_no | stud_name   | mach_no | lab_no | description  | day_of_week |
+---------+-------------+---------+--------+--------------+-------------+
|       4 | David Brown |       4 |    103 | Oscilloscope | Thursday    |
+---------+-------------+---------+--------+--------------+-------------+
1 row in set (0.00 sec)

mysql>
```

6. Create a procedure to find reverse of a given number.

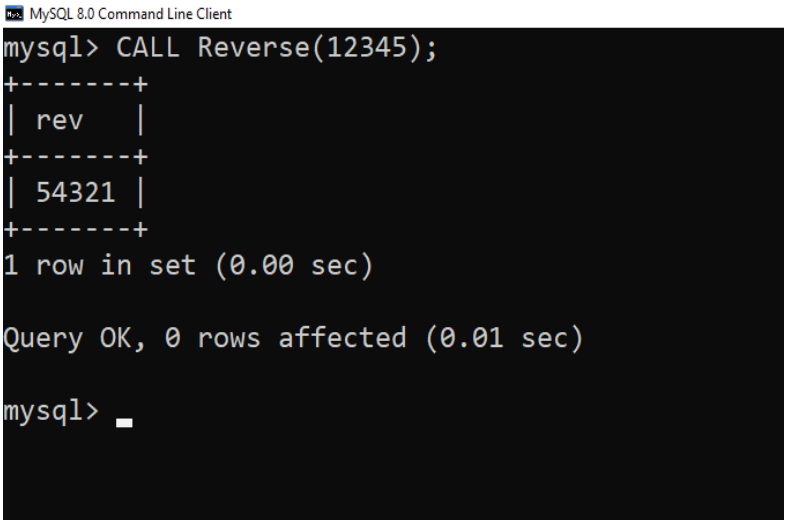
mysql>

```
DELIMITER $$
CREATE PROCEDURE Reverse(n INT)
BEGIN
 DECLARE num,rem,rev INT;
SET rev=0;
SET num:=n;
WHILE num>0 DO
 SET rem:=num MOD 10;
SET rev:=rem+(rev*10);
SET num:=FLOOR(num/10);
 END WHILE;
SELECT rev;
 END  $$
DELIMITER ;
```

mysql>

```
CALL  Reverse(12345);$$
```

OUTPUT:



7. **Create a procedure to update the salaries of all employees as per the given data**

mysql>

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    salary DECIMAL(10, 2)
);
```

mysql>

```
INSERT INTO employees (employee_id, first_name, last_name, salary)
VALUES
(1, 'Rahul', 'Dravid', 44000),
(2, 'Rohith', 'Sharma', 49500),
(3, 'Yuvraj', 'Singh', 55000);
```

mysql>

```
SELECT * FROM employees;
```

mysql>

```
DELIMITER $$

CREATE PROCEDURE update_salaries(
    IN percentage_increase DECIMAL(5, 2)   -- IN parameter: percentage increase
)
BEGIN
    -- Update all employees' salaries based on the percentage
    UPDATE employees
    SET salary = salary + (salary * percentage_increase / 100);
END $$

DELIMITER ;
```

mysql>

```
CALL update_salaries(10);
```

OUTPUT:

MySQL 8.0 Command Line Client

```
mysql> select * from employees;
+-------------+------------+-----------+----------+
| employee_id | first_name | last_name | salary   |
+-------------+------------+-----------+----------+
|           1 | Rahul      | Dravid    | 44000.00 |
|           2 | Rohith     | Sharma    | 49500.00 |
|           3 | Yuvraj     | Singh     | 55000.00 |
+-------------+------------+-----------+----------+
3 rows in set (0.00 sec)

mysql>
```

MySQL 8.0 Command Line Client

```
mysql> CALL update_salaries(10);
Query OK, 3 rows affected (0.01 sec)

mysql>
```

MySQL 8.0 Command Line Client

```
mysql> select * from employees;
+-------------+------------+-----------+----------+
| employee_id | first_name | last_name | salary   |
+-------------+------------+-----------+----------+
|           1 | Rahul      | Dravid    | 48400.00 |
|           2 | Rohith     | Sharma    | 54450.00 |
|           3 | Yuvraj     | Singh     | 60500.00 |
+-------------+------------+-----------+----------+
3 rows in set (0.00 sec)

mysql>
```

8. Create a procedure to demonstrate IN, OUT and INOUT parameters.
The procedure should perform the following operations:
1. Accept three parameters:
a (integer) as an IN parameter.
b (integer) as an IN parameter.
initial_result (integer) as an INOUT parameter.
2. Calculate the sum of the a and b parameters and update initial_result with this sum.
3. Output the final sum through an OUT parameter named final_result.

mysql>

```
DELIMITER $$

CREATE PROCEDURE add_numbers(
   IN num1 INT,          -- IN parameter
   INOUT num2 INT,       -- INOUT parameter
   OUT result INT        -- OUT parameter
)
BEGIN
   -- Perform the addition of num1 and num2
   SET result = num1 + num2;

   -- Modify the INOUT parameter (increment num2 by 1)
   SET num2 = num2 + 1;

   -- Optionally, display values using SELECT for debugging
SELECT CONCAT('Result of addition: ', result) AS result_message;
SELECT CONCAT('Modified num2 (INOUT): ', num2) AS num2_message;
END $$

DELIMITER ;
```

OUTPUT:

MySQL 8.0 Command Line Client

```
mysql> SET @num1 = 5;SET @num2 = 10;SET @result = 0;
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> CALL add_numbers(@num1, @num2, @result);
+------------------------+
| result_message         |
+------------------------+
| Result of addition: 15 |
+------------------------+
1 row in set (0.00 sec)

+---------------------------+
| num2_message              |
+---------------------------+
| Modified num2 (INOUT): 11 |
+---------------------------+
1 row in set (0.00 sec)
```

MySQL 8.0 Command Line Client

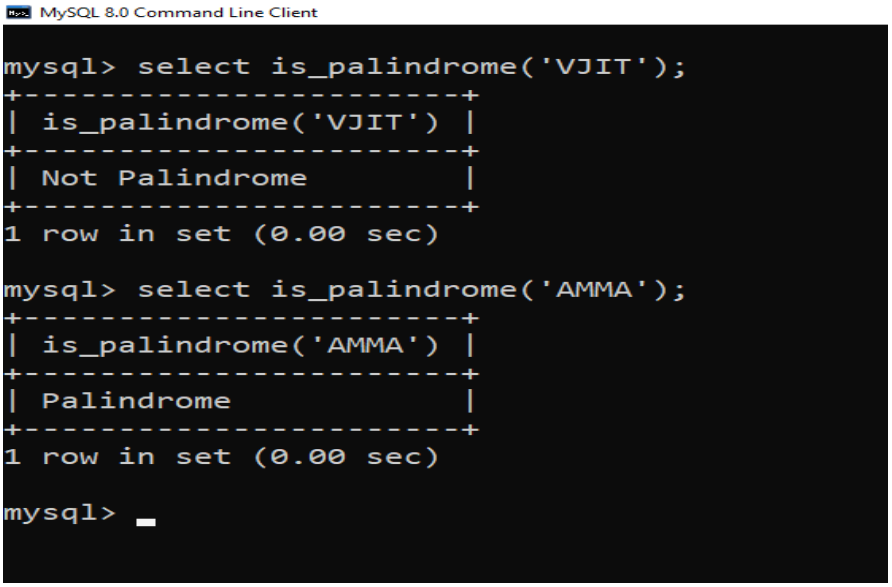
```
mysql> SELECT @result AS final_result;
+--------------+
| final_result |
+--------------+
|           15 |
+--------------+
1 row in set (0.00 sec)

mysql> SELECT @num2 AS final_num2;
+------------+
| final_num2 |
+------------+
|         11 |
+------------+
1 row in set (0.00 sec)

mysql>
```

9. Create a function to check whether given string is palindrome or not.

mysql>

```
DELIMITER $$
CREATE FUNCTION is_palindrome(input_string VARCHAR(255))
RETURNS VARCHAR(255)
DETERMINISTIC
BEGIN
   DECLARE reversed_string VARCHAR(255);
      -- Reverse the input string
   SET reversed_string = REVERSE(input_string);
      -- Compare the original string with the reversed string
   IF input_string = reversed_string THEN
      SET input_string='Palindrome'; -- It's a palindrome
   ELSE
      SET input_string='Not Palindrome'; -- It's not a palindrome
   END IF;
RETURN input_string;
END $$
DELIMITER ;
```

OUTPUT:



10. Create a function to find sum of salaries of all employees working in depart number 10

mysql>

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    salary DECIMAL(10, 2),
    department_id INT
);
```

mysql>

```
INSERT INTO employees (employee_id, first_name, last_name, salary,
department_id)
VALUES
 (1, 'John', 'Doe', 50000, 10),
 (2, 'Jane', 'Smith', 55000, 10),
 (3, 'Jim', 'Brown', 60000, 20),
 (4, 'Jack', 'Davis', 45000, 10);
```

mysql>

```
SELECT * FROM employees;
```

mysql>

```
DELIMITER $$

CREATE FUNCTION sum_salaries_dept()
RETURNS DECIMAL(10, 2)
DETERMINISTIC
BEGIN
    DECLARE total_salary DECIMAL(10, 2);
    -- Sum the salaries of all employees in department 10
    SELECT SUM(salary) INTO total_salary
    FROM employees
    WHERE department_id = 10;
    -- Return the result
    RETURN total_salary;
END $$
DELIMITER ;
```

mysql>

```
SELECT sum_salaries_dept() AS total_salaries_in_dept;
```

OUTPUT:

MySQL 8.0 Command Line Client

```
mysql> select * from employees;
+-------------+------------+-----------+----------+---------------+
| employee_id | first_name | last_name | salary   | department_id |
+-------------+------------+-----------+----------+---------------+
|           1 | John       | Doe       | 50000.00 |            10 |
|           2 | Jane       | Smith     | 55000.00 |            10 |
|           3 | Jim        | Brown     | 60000.00 |            20 |
|           4 | Jack       | Davis     | 45000.00 |            10 |
+-------------+------------+-----------+----------+---------------+
4 rows in set (0.00 sec)

mysql>
```

MySQL 8.0 Command Line Client

```
mysql> SELECT sum_salaries_dept() AS total_salaries_in_dept;
+------------------------+
| total_salaries_in_dept |
+------------------------+
|              150000.00 |
+------------------------+
1 row in set (0.00 sec)

mysql>
```

11. Create a TRIGGER BEFORE/AFTER UPDATE on employee table FOR EACH ROW/statement.

- **BEFORE UPDATE**: The trigger will execute before the update occurs.
- **AFTER UPDATE**: The trigger will execute after the update occurs.
- **FOR EACH ROW**: Ensures that the trigger is executed for each row affected by the UPDATE statement.

BEFORE UPDATE:

mysql>

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    salary DECIMAL(10, 2),
    department_id INT
);
CREATE TABLE salary_changes (
    change_id INT AUTO_INCREMENT PRIMARY KEY,
    employee_id INT,
    old_salary DECIMAL(10, 2),
    new_salary DECIMAL(10, 2),
    change_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

mysql>

```
INSERT INTO employees (employee_id, first_name, last_name, salary,
department_id)
VALUES
 (1, 'John', 'Doe', 50000, 10),
 (2, 'Jane', 'Smith', 55000, 10),
 (3, 'Jim', 'Brown', 60000, 20),
 (4, 'Jack', 'Davis', 45000, 10);
```

mysql>

```
SELECT * FROM employees;
```

mysql>

```
DELIMITER $$
CREATE TRIGGER before_employee_update
BEFORE UPDATE ON employees
```

```
FOR EACH ROW
BEGIN
   -- Check if salary is being changed
   IF OLD.salary != NEW.salary THEN
      -- Insert a record into the salary changes table
      INSERT INTO salary_changes (employee_id, old_salary, new_salary)
      VALUES (OLD.employee_id, OLD.salary, NEW.salary);
   END IF;
END $$
DELIMITER ;
```

mysql>

```
UPDATE employees
SET salary = 55000
WHERE employee_id = 1;
```

mysql>

```
SELECT * FROM employees;
```

mysql>

```
SELECT * FROM salary_changes;
```

OUTPUT:

MySQL 8.0 Command Line Client

```
mysql> select * from employees;
+-------------+------------+-----------+----------+---------------+
| employee_id | first_name | last_name | salary   | department_id |
+-------------+------------+-----------+----------+---------------+
|           1 | John       | Doe       | 50000.00 |            10 |
|           2 | Jane       | Smith     | 55000.00 |            10 |
|           3 | Jim        | Brown     | 60000.00 |            20 |
|           4 | Jack       | Davis     | 45000.00 |            10 |
+-------------+------------+-----------+----------+---------------+
4 rows in set (0.00 sec)

mysql>
```

MySQL 8.0 Command Line Client

```
Empty set (0.00 sec)

mysql> UPDATE employees
    -> SET salary = 55000
    -> WHERE employee_id = 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from employees;
+-------------+------------+-----------+----------+---------------+
| employee_id | first_name | last_name | salary   | department_id |
+-------------+------------+-----------+----------+---------------+
|           1 | John       | Doe       | 55000.00 |            10 |
|           2 | Jane       | Smith     | 55000.00 |            10 |
|           3 | Jim        | Brown     | 60000.00 |            20 |
|           4 | Jack       | Davis     | 45000.00 |            10 |
+-------------+------------+-----------+----------+---------------+
4 rows in set (0.00 sec)

mysql>
```

MySQL 8.0 Command Line Client

```
mysql> select * from employees;
+-------------+------------+-----------+----------+---------------+
| employee_id | first_name | last_name | salary   | department_id |
+-------------+------------+-----------+----------+---------------+
|           1 | John       | Doe       | 55000.00 |            10 |
|           2 | Jane       | Smith     | 55000.00 |            10 |
|           3 | Jim        | Brown     | 60000.00 |            20 |
|           4 | Jack       | Davis     | 45000.00 |            10 |
+-------------+------------+-----------+----------+---------------+
4 rows in set (0.00 sec)

mysql> select * from salary_changes;
+-----------+-------------+------------+------------+---------------------+
| change_id | employee_id | old_salary | new_salary | change_date         |
+-----------+-------------+------------+------------+---------------------+
|         1 |           1 |   50000.00 |   55000.00 | 2024-11-08 01:47:07 |
+-----------+-------------+------------+------------+---------------------+
1 row in set (0.00 sec)

mysql>
```

AFTER UPDATE:

mysql>

```
DELIMITER $$
CREATE TRIGGER after_employee_update
AFTER UPDATE ON employees
FOR EACH ROW
BEGIN
   -- Check if salary is being changed
   IF OLD.salary != NEW.salary THEN
      -- Insert a record into the salary changes table
      INSERT INTO salary_changes (employee_id, old_salary, new_salary)
      VALUES (OLD.employee_id, NEW.salary, NEW.salary);
END IF;
END $$
DELIMITER ;
```

mysql>

```
UPDATE employees
SET salary = 90000
WHERE employee_id = 1;
```

mysql>

```
SELECT * FROM employees;
```

mysql>

```
SELECT * FROM salary_changes;
```

OUTPUT:

MySQL 8.0 Command Line Client

```
mysql> select * from employees;
+-------------+------------+-----------+----------+---------------+
| employee_id | first_name | last_name | salary   | department_id |
+-------------+------------+-----------+----------+---------------+
|           1 | John       | Doe       | 99900.00 |            10 |
|           2 | Jane       | Smith     | 55000.00 |            10 |
|           3 | Jim        | Brown     | 60000.00 |            20 |
|           4 | Jack       | Davis     | 45000.00 |            10 |
+-------------+------------+-----------+----------+---------------+
4 rows in set (0.00 sec)

mysql>
```

MySQL 8.0 Command Line Client

```
mysql> UPDATE employees
    -> SET salary = 90000
    -> WHERE employee_id = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
```

MySQL 8.0 Command Line Client

```
mysql> select * from employees;
+-------------+------------+-----------+----------+---------------+
| employee_id | first_name | last_name | salary   | department_id |
+-------------+------------+-----------+----------+---------------+
|           1 | John       | Doe       | 90000.00 |            10 |
|           2 | Jane       | Smith     | 55000.00 |            10 |
|           3 | Jim        | Brown     | 60000.00 |            20 |
|           4 | Jack       | Davis     | 45000.00 |            10 |
+-------------+------------+-----------+----------+---------------+
4 rows in set (0.00 sec)

mysql>
```

MySQL 8.0 Command Line Client

```
mysql> select * from salary_changes;
+-----------+-------------+------------+------------+---------------------+
| change_id | employee_id | old_salary | new_salary | change_date         |
+-----------+-------------+------------+------------+---------------------+
|         1 |           1 |   99900.00 |   90000.00 | 2024-11-08 03:05:04 |
|         2 |           1 |   90000.00 |   90000.00 | 2024-11-08 03:05:04 |
+-----------+-------------+------------+------------+---------------------+
2 rows in set (0.00 sec)

mysql>
```

12. Create a TRIGGER BEFORE/AFTER DELETE on employee table FOR EACH ROW/statement.

- **BEFORE DELETE**: The trigger will execute before the delete occurs.
- **AFTER DELETE**: The trigger will execute after the delete occurs.
- **FOR EACH ROW**: Ensures that the trigger is executed for each row affected by the DELETE statement.

BEFORE UPDATE:

mysql>

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    salary DECIMAL(10, 2),
    department_id INT
);
CREATE TABLE employee_deletions (
    deletion_id INT AUTO_INCREMENT PRIMARY KEY,
    employee_id INT,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    salary DECIMAL(10, 2),
    department_id INT,
    deletion_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

mysql>

```
INSERT INTO employees (employee_id, first_name, last_name, salary,
department_id)
VALUES
 (1, 'John', 'Doe', 50000, 10),
 (2, 'Jane', 'Smith', 55000, 10),
 (3, 'Jim', 'Brown', 60000, 20),
 (4, 'Jack', 'Davis', 45000, 10);
```

mysql>

```
SELECT * FROM employees;
```

mysql>

```
DELIMITER $$
CREATE TRIGGER before_employee_delete
BEFORE DELETE ON employees
FOR EACH ROW
BEGIN
   -- Log the deletion of the employee in the employee_deletions table
   INSERT INTO employee_deletions (employee_id, first_name, last_name,
salary, department_id)
   VALUES (OLD.employee_id, OLD.first_name, OLD.last_name,
OLD.salary, OLD.department_id);
END $$
DELIMITER ;
```

mysql>

```
DELETE FROM employees WHERE employee_id = 2;
```

mysql>

```
SELECT * FROM employees;
```

mysql>

```
SELECT * FROM employee_deletions;
```

OUTPUT:

MySQL 8.0 Command Line Client

```
mysql> SELECT * FROM employees;
+-------------+------------+-----------+----------+---------------+
| employee_id | first_name | last_name | salary   | department_id |
+-------------+------------+-----------+----------+---------------+
|           1 | John       | Doe       | 90000.00 |            10 |
|           2 | Jane       | Smith     | 55000.00 |            10 |
|           3 | Jim        | Brown     | 60000.00 |            20 |
|           4 | Jack       | Davis     | 45000.00 |            10 |
+-------------+------------+-----------+----------+---------------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM employee_deletions;
Empty set (0.00 sec)

mysql>
```

MySQL 8.0 Command Line Client

```
mysql> DELETE FROM employees WHERE employee_id = 2;
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM employees;
+-------------+------------+-----------+----------+---------------+
| employee_id | first_name | last_name | salary   | department_id |
+-------------+------------+-----------+----------+---------------+
|           1 | John       | Doe       | 90000.00 |            10 |
|           3 | Jim        | Brown     | 60000.00 |            20 |
|           4 | Jack       | Davis     | 45000.00 |            10 |
+-------------+------------+-----------+----------+---------------+
3 rows in set (0.00 sec)

mysql>
```

MySQL 8.0 Command Line Client

```
mysql> SELECT * FROM employee_deletions;
+-------------+-------------+------------+-----------+----------+---------------+---
------------------+
| deletion_id | employee_id | first_name | last_name | salary   | department_id | de
letion_date       |
+-------------+-------------+------------+-----------+----------+---------------+---
------------------+
|           1 |           2 | Jane       | Smith     | 55000.00 |            10 | 20
24-11-08 03:25:53 |
+-------------+-------------+------------+-----------+----------+---------------+---
------------------+
1 row in set (0.00 sec)

mysql>
```

AFTER DELETE:

mysql>

```
DELIMITER $$
CREATE TRIGGER after_employee_delete
AFTER DELETE ON employees
FOR EACH ROW
BEGIN
   -- Log the deletion of the employee in the employee_deletions table
   INSERT INTO employee_deletions (
            employee_id, first_name, last_name, salary, department_id)
   VALUES (OLD.employee_id, OLD.first_name, OLD.last_name,
OLD.salary, OLD.department_id);
END $$
DELIMITER ;
```

mysql>

```
DELETE FROM employees WHERE employee_id = 3;
```

mysql>

```
SELECT * FROM employees;
```

mysql>

```
SELECT * FROM employee_deletions;
```

OUTPUT:

MySQL 8.0 Command Line Client

```
mysql> SELECT * FROM employees;
+-------------+------------+-----------+----------+---------------+
| employee_id | first_name | last_name | salary   | department_id |
+-------------+------------+-----------+----------+---------------+
|           1 | John       | Doe       | 90000.00 |            10 |
|           3 | Jim        | Brown     | 60000.00 |            20 |
|           4 | Jack       | Davis     | 45000.00 |            10 |
+-------------+------------+-----------+----------+---------------+
3 rows in set (0.00 sec)

mysql>
```

MySQL 8.0 Command Line Client

```
mysql> DELETE FROM employees WHERE employee_id = 3;
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM employees;
+-------------+------------+-----------+----------+---------------+
| employee_id | first_name | last_name | salary   | department_id |
+-------------+------------+-----------+----------+---------------+
|           1 | John       | Doe       | 90000.00 |            10 |
|           4 | Jack       | Davis     | 45000.00 |            10 |
+-------------+------------+-----------+----------+---------------+
2 rows in set (0.00 sec)

mysql>
```

MySQL 8.0 Command Line Client

```
mysql> SELECT * FROM employee_deletions;
+-------------+-------------+------------+-----------+----------+---------------+---
------------------+
| deletion_id | employee_id | first_name | last_name | salary   | department_id | de
letion_date       |
+-------------+-------------+------------+-----------+----------+---------------+---
------------------+
|           1 |           2 | Jane       | Smith     | 55000.00 |            10 | 20
24-11-08 03:25:53 |
|           2 |           3 | Jim        | Brown     | 60000.00 |            20 | 20
24-11-08 03:41:33 |
|           3 |           3 | Jim        | Brown     | 60000.00 |            20 | 20
24-11-08 03:41:33 |
+-------------+-------------+------------+-----------+----------+---------------+---
------------------+
3 rows in set (0.00 sec)

mysql>
```

13. Create a TRIGGER BEFORE/AFTER INSERT on employee table FOR EACH ROW/statement.

- **BEFORE INSERT**: The trigger will execute before the insert occurs.
- **AFTER INSERT**: The trigger will execute after the insert occurs.
- **FOR EACH ROW**: Ensures that the trigger is executed for each row affected by the INSERT statement.

BEFORE UPDATE:

mysql>

```
CREATE TABLE employees (
   employee_id INT PRIMARY KEY,
   first_name VARCHAR(100),
   last_name VARCHAR(100),
   salary DECIMAL(10, 2),
   department_id INT
);

CREATE TABLE employee_adding (
    audit_id INT AUTO_INCREMENT PRIMARY KEY, employee_id INT,
    action VARCHAR(100),
    action_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP );
```

mysql>

```
INSERT INTO employees (employee_id, first_name, last_name, salary,
department_id)
VALUES
 (1, 'John', 'Doe', 50000, 10),
 (2, 'Jane', 'Smith', 55000, 10),
 (3, 'Jim', 'Brown', 60000, 20),
 (4, 'Jack', 'Davis', 45000, 10);
```

mysql>

```
SELECT * FROM employees;
```

mysql>

```
DELIMITER $$

CREATE TRIGGER before_employee_insert
BEFORE INSERT ON employees
FOR EACH ROW
BEGIN
   -- Log the insertion into the employee_adding table
   INSERT INTO employee_adding (employee_id, action)
   VALUES (NEW.employee_id, 'Employee Added');
END $$
DELIMITER ;
```

mysql>

```
INSERT INTO employees (employee_id ,first_name, last_name, salary,
department_id)
VALUES (5,'Alice', 'Johnson', 55000, 10);
```

mysql>

```
SELECT * FROM employees;
```

mysql>

```
SELECT * FROM employee_adding;
```

OUTPUT:

MySQL 8.0 Command Line Client

```
mysql> select * from employees;
+-------------+------------+-----------+----------+---------------+
| employee_id | first_name | last_name | salary   | department_id |
+-------------+------------+-----------+----------+---------------+
|           1 | John       | Doe       | 90000.00 |            10 |
|           4 | Jack       | Davis     | 45000.00 |            10 |
+-------------+------------+-----------+----------+---------------+
2 rows in set (0.00 sec)

mysql>
```

MySQL 8.0 Command Line Client

```
mysql> INSERT INTO employees (employee_id ,first_name, last_name, salary, dep
_id) VALUES (5,'Alice', 'Johnson', 55000, 10);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM employees;
+-------------+------------+-----------+----------+---------------+
| employee_id | first_name | last_name | salary   | department_id |
+-------------+------------+-----------+----------+---------------+
|           1 | John       | Doe       | 90000.00 |            10 |
|           4 | Jack       | Davis     | 45000.00 |            10 |
|           5 | Alice      | Johnson   | 55000.00 |            10 |
+-------------+------------+-----------+----------+---------------+
3 rows in set (0.00 sec)

mysql>
```

MySQL 8.0 Command Line Client

```
mysql> SELECT * FROM employee_adding;
+----------+-------------+---------------+---------------------+
| audit_id | employee_id | action        | action_date         |
+----------+-------------+---------------+---------------------+
|        2 |           5 | Employee Added | 2024-11-08 04:04:47 |
+----------+-------------+---------------+---------------------+
1 row in set (0.00 sec)

mysql>
```

AFTER INSERT:

mysql>

```
DELIMITER $$

CREATE TRIGGER after_employee_insert
AFTER INSERT ON employees
FOR EACH ROW
BEGIN
   -- Log the insertion into the employee_adding  table
   INSERT INTO employee_adding (employee_id, action)
   VALUES (NEW.employee_id, 'Employee Added');
END $$
DELIMITER ;
```

mysql>

```
INSERT INTO employees (employee_id ,first_name, last_name, salary,
department_id)
VALUES (2, 'john', 'doe', 50000, 10);
```

mysql>

```
SELECT * FROM employees;
```

mysql>

```
SELECT * FROM employee_adding;
```

OUTPUT:

Select MySQL 8.0 Command Line Client

```
mysql> SELECT * FROM employees;
+-------------+------------+-----------+----------+---------------+
| employee_id | first_name | last_name | salary   | department_id |
+-------------+------------+-----------+----------+---------------+
|           1 | John       | Doe       | 90000.00 |            10 |
|           4 | Jack       | Davis     | 45000.00 |            10 |
|           5 | Alice      | Johnson   | 55000.00 |            10 |
+-------------+------------+-----------+----------+---------------+
3 rows in set (0.00 sec)

mysql> INSERT INTO employees (employee_id ,first_name, last_name, salary, department
_id)
    -> VALUES (2, 'john', 'doe', 50000, 10);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM employees;
+-------------+------------+-----------+----------+---------------+
| employee_id | first_name | last_name | salary   | department_id |
+-------------+------------+-----------+----------+---------------+
|           1 | John       | Doe       | 90000.00 |            10 |
|           2 | john       | doe       | 50000.00 |            10 |
|           4 | Jack       | Davis     | 45000.00 |            10 |
|           5 | Alice      | Johnson   | 55000.00 |            10 |
+-------------+------------+-----------+----------+---------------+
4 rows in set (0.00 sec)
```

MySQL 8.0 Command Line Client

```
mysql> SELECT * FROM employee_adding;
+----------+-------------+----------------+---------------------+
| audit_id | employee_id | action         | action_date         |
+----------+-------------+----------------+---------------------+
|        2 |           5 | Employee Added | 2024-11-08 04:04:47 |
|        3 |           2 | Employee Added | 2024-11-08 04:11:35 |
|        4 |           2 | Employee Added | 2024-11-08 04:11:35 |
+----------+-------------+----------------+---------------------+
3 rows in set (0.00 sec)

mysql>
```