**1.a. Create the tables with the appropriate integrity constraints**

1.create table customer(cust\_id int primary key,customername varchar(100));

2. create table item (item\_id int primary key,itemname varchar(100),price int);

3. create table sale1 (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));

**Insert around 10 records in each of the tables**

**mysql> insert into Customer1 (cust\_id, customername) VALUES (1, 'John Doe'),(2, 'Jane Smith'),(3, 'Alice Johnson'),(4, 'Bob Brown'),(5, 'Charlie White'),(6, 'David Black'),(7, 'Eva Green'),(8, 'Frank Blue'),(9, 'Grace Pink'),(10, 'Hank Gray');**

**mysql> insert into item1 (item\_id, itemname, price) values (1, 'Laptop', 1000),(2, 'Smartphone', 800),(3, 'Tablet', 600),(4, 'Monitor', 300),(5, 'Keyboard', 50),(6, 'Mouse', 30),(7, 'Printer', 150),(8, 'Camera', 500),(9, 'Headphones', 120),(10, 'Speaker', 200);**

**mysql> insert into sale1 (bill\_no, bill\_date, cust\_id, item\_id, qty\_sold) values (1, '2024-08-19', 1, 1, 2),(2, '2024-08-19', 2, 2, 1), (3, '2024-08-18', 3, 3, 3),(4, '2024-08-18', 4, 4, 1),(5, '2024-08-19', 5, 5, 4),(6, '2024-08-17', 6, 6, 2),(7, '2024-08-19', 7, 7, 1),(8, '2024-08-16', 8, 8, 3),(9, '2024-08-15', 9, 9, 2),(10, '2024-08-19', 10, 10, 1);**

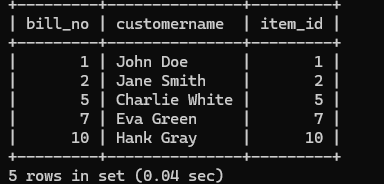
**c. List all the bills for the current date with the customer names and item numbers**

**mysql> select s.bill\_no,c.customername,s.item\_id from sale1 s join customer1 c on s.cust\_id=c.cust\_id where s.bill\_date=2024-08-19;**

**Description:**

**[**This SQL command retrieves the bill number, customer name, and item ID for all sales made on the current date by joining the Sale table (s) with the Customer table (c) based on the matching cust\_id.]

**output:5 rows in set (0.04 sec)**



**d. List the total Bill details with the quantity sold, price of the item, and the final amount**

**mysql>**

**select s.bill\_no,c.customername,i.itemname,s.qty\_sold,i.price,(s.qty\_sold \* i.price) as amount from sale s join customer c on s.cust\_id=c.cust\_id join item i on s.item\_id=i.item\_id;**

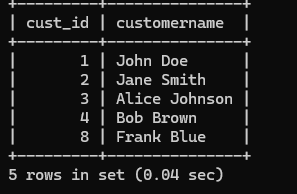
**Description:**

This SQL command retrieves detailed information for each sale, including the bill number, customer name, item name, quantity sold, item price, and the total amount (calculated as quantity sold \* price) by joining the Sale, Customer, and Item tables based on their respective IDs.

**e. List the details of the customer who have bought a product which has a price > 200**

**mysql> select distinct c.cust\_id,c.customername from sale s join customer c on s.cust\_id=c.cust\_id join item i on s.item\_id=i.item\_id where i.price**

**>200;**

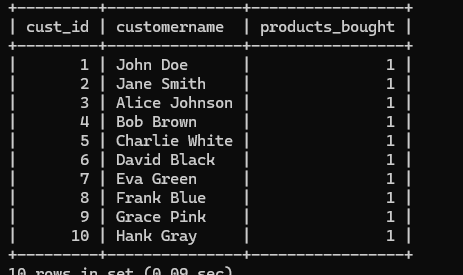


**Description:**

This SQL command retrieves a list of unique customers who have purchased items priced over 200 by joining the Sale, Customer, and Item tables based on their respective IDs.

**f. Give a count of how many products have been bought by each customer**

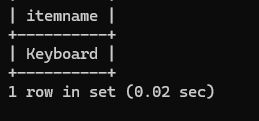
**mysql> select c.cust\_id,c.customername, count(s.item\_id)as products\_bought from sale s join customer c on s.cust\_id=c.cust\_id group by c.cust\_id,c.customername;**



**Description:**

This SQL command retrieves the customer ID, customer name, and the total number of products each customer has bought by joining the Sale and Customer tables, and then grouping the results by customer ID and name.

**g. Give a list of products bought by a customer having cust\_id as 5**



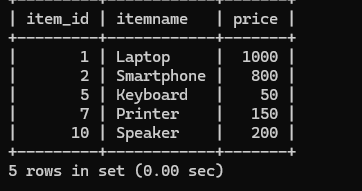
**Description:**

This SQL command retrieves a unique list of item IDs, item names, and prices for items that were sold on the current date by joining the Sale and Item tables.

**h. List the item details which are sold as of today**

**mysql> select distinct i.item\_id,i.itemname,i.price from sale s join item i on s.item\_id=i.item\_id where s.bill\_date=current\_date;**

This SQL command retrieves a unique list of item IDs, item names, and prices for items that were sold today by joining the Sale and Item tables.



**Lab2**

1.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 the tables with the appropriate integrity constraints
  2. Insert around 10 records in each of the tables
  3. List all the student names with their membership numbers
  4. List all the issues for the current date with student and Book names
  5. List the details of students who borrowed book whose author is CJDATE
  6. Give a count of how many books have been bought by each student
  7. Give a list of books taken by student with stud\_no as 5
  8. List the book details which are issued as of today
  9. Create a view which lists out the iss\_no, iss \_date, stud\_name, book name
  10. Create a view which lists the daily issues-date wise for the last one week.

1. a**. Create the Tables with Appropriate Integrity Constraints**

CREATE TABLE Student2 ( Stud\_no INT PRIMARY KEY, Stud\_name VARCHAR(50) NOT NULL );

CREATE TABLE Membership ( Mem\_no INT PRIMARY KEY, Stud\_no INT, FOREIGN KEY (Stud\_no) REFERENCES Student2(Stud\_no) );

CREATE TABLE Book ( book\_no INT PRIMARY KEY, book\_name VARCHAR(100) NOT NULL, author VARCHAR(50) NOT NULL );

CREATE TABLE Iss\_rec ( iss\_no INT PRIMARY KEY, iss\_date DATE NOT NULL, Mem\_no INT, book\_no INT, FOREIGN KEY (Mem\_no) REFERENCES Membership(Mem\_no), FOREIGN KEY (book\_no) REFERENCES Book(book\_no) );

**b. Insert 10 Records into Each Table**

INSERT INTO Student2 (Stud\_no, Stud\_name) VALUES (1, 'John Doe'), (2, 'Jane Smith'), (3, 'Alice Brown'), (4, 'Bob Johnson'), (5, 'Charlie Green'), (6, 'Diana Prince'), (7, 'Eve Adams'), (8, 'Frank Miller'), (9, 'Grace Kelly'), (10, 'Hank Moody');

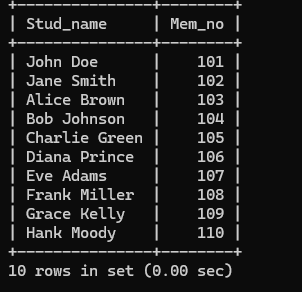
INSERT INTO Membership (Mem\_no, Stud\_no) VALUES (101, 1), (102, 2), (103, 3), (104, 4), (105, 5), (106, 6), (107, 7), (108, 8), (109, 9), (110, 10);

INSERT INTO Book (book\_no, book\_name, author) VALUES (1001, 'Database Systems', 'CJDATE'), (1002, 'Operating Systems', 'Galvin'), (1003, 'Computer Networks', 'Tanenbaum'), (1004, 'Artificial Intelligence', 'Russell'), (1005, 'Data Structures', 'Cormen'), (1006, 'Software Engineering', 'Pressman'), (1007, 'Digital Logic Design', 'Morris Mano'), (1008, 'Algorithm Design', 'Kleinberg'), (1009, 'Web Technologies', 'Pankaj Sharma'), (1010, 'Programming Languages', 'Sebesta');

INSERT INTO Iss\_rec (iss\_no, iss\_date, Mem\_no, book\_no) VALUES (1, '2024-08-31', 101, 1001), (2, '2024-08-31', 102, 1002), (3, '2024-08-30', 103, 1003), (4, '2024-08-29', 104, 1004), (5, '2024-08-28', 105, 1005), (6, '2024-08-31', 106, 1006), (7, '2024-08-30', 107, 1007), (8, '2024-08-31', 108, 1001), (9, '2024-08-27', 109, 1008), (10, '2024-08-31', 110, 1009);

**c. List All the Student Names with Their Membership Numbers**

SELECT s.Stud\_name, m.Mem\_no FROM Student2 s JOIN Membership m ON s.Stud\_no = m.Stud\_no;



**Description:**

 **SELECT s.Stud\_name, m.Mem\_no**: Specifies the columns to retrieve, which are the student names (Stud\_name) from the Student2 table and the membership numbers (Mem\_no) from the Membership table.

 **FROM Student2 s**: Specifies the Student2 table as s (an alias for easier reference).

 **JOIN Membership m**: Joins the Membership table (aliased as m) with Student2.

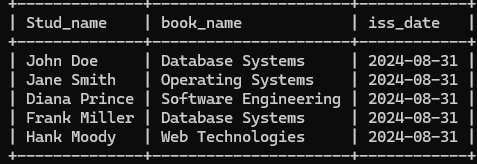
 **ON s.Stud\_no = m.Stud\_no**: Specifies the condition that the Stud\_no column in both tables must match for the rows to be joined.

this command joins the Student2 and Membership tables based on the student number (Stud\_no) and retrieves the matching student names and membership numbers.

**D. List All the Issues for the Current Date with Student and Book Names**

SELECT s.Stud\_name, b.book\_name, i.iss\_date FROM Iss\_rec i JOIN Membership m ON i.Mem\_no = m.Mem\_no JOIN student2 s ON m.Stud\_no = s.Stud\_no JOIN Book b ON i.book\_no = b.book\_no WHERE i.iss\_date = CURDATE();

**Output:**



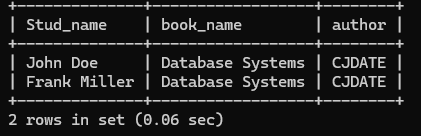
**Description:**

This query provides a list of students, the books they have issued, and the issue date, but only for those records where the books were issued today.

E. List the Details of Students Who Borrowed a Book Whose Author is CJDATE

SELECT s.Stud\_name, b.book\_name, b.author FROM Iss\_rec i JOIN Membership m ON i.Mem\_no = m.Mem\_no JOIN Student2 s ON m.Stud\_no = s.Stud\_no JOIN Book b ON i.book\_no = b.book\_no WHERE b.author = 'CJDATE';

Output:

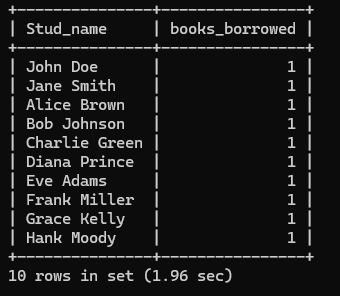


**Description:**

This query lists the names of students, the titles of the books they borrowed, and the author, specifically for books written by "CJDATE."

**F. Give a Count of How Many Books Have Been Borrowed by Each Student**

SELECT s.Stud\_name, COUNT(i.book\_no) AS books\_borrowed FROM Iss\_rec i JOIN Membership m ON i.Mem\_no = m.Mem\_no JOIN Student2 s ON m.Stud\_no = s.Stud\_no GROUP BY s.Stud\_name;

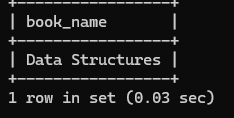


**Description:**

This query provides a list of students along with the total number of books they have borrowed, grouping the results by each student's name.

**G.Give a List of Books Taken by Student with Stud\_no = 5**

SELECT b.book\_name FROM Iss\_rec i JOIN Membership m ON i.Mem\_no = m.Mem\_no JOIN Book b ON i.book\_no = b.book\_no WHERE m.Stud\_no = 5;

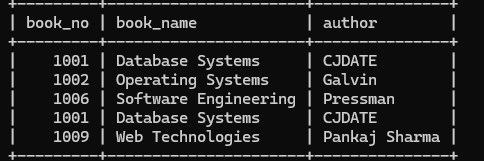


**Description:**

This query lists all the books that have been borrowed by the student whose student number is 5.

H. List the Book Details Which Are Issued as of Today

SELECT b.book\_no, b.book\_name, b.author FROM Iss\_rec i JOIN Book b ON i.book\_no = b.book\_no WHERE i.iss\_date = CURDATE();

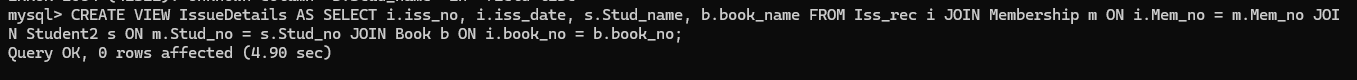


**Description:**

This query lists the book numbers, names, and authors of books that were issued today.

I. Create a View Which Lists Out the iss\_no, iss\_date, Stud\_name, and book\_name.

CREATE VIEW IssueDetails AS SELECT i.iss\_no, i.iss\_date, s.Stud\_name, b.book\_name FROM Iss\_rec i JOIN Membership m ON i.Mem\_no = m.Mem\_no JOIN Student2 s ON m.Stud\_no = s.Stud\_no JOIN Book b ON i.book\_no = b.book\_no;

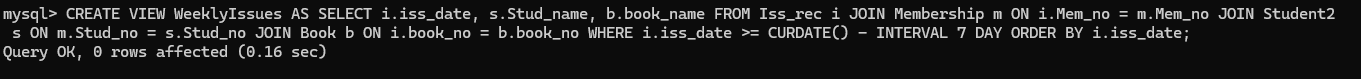


Description

This command creates a view that consolidates information about book issues, displaying the issue number, issue date, student names, and book names, allowing for simplified access to this combined data in future queries.

J.Create a View Which Lists the Daily Issues Date-Wise for the Last One Week

CREATE VIEW WeeklyIssues AS SELECT i.iss\_date, s.Stud\_name, b.book\_name FROM Iss\_rec i JOIN Membership m ON i.Mem\_no = m.Mem\_no JOIN Student2 s ON m.Stud\_no = s.Stud\_no JOIN Book b ON i.book\_no = b.book\_no WHERE i.iss\_date >= CURDATE() - INTERVAL 7 DAY ORDER BY i.iss\_date;



In SQL, tables store and persist data, while views provide a simplified way to access data from one or more tables. **A key difference is that tables physically store data, while views are virtual tables that don't require physical storage.**

**A foreign key is a column or group of columns that links data in two tables, while a join is a way to access data from multiple tables based on logical relationships:**

1. **Database Schema for a Employee-payscenario**

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.

**A.** Create the tables with the appropriate integrity constraints.

**CREATE TABLE Department (**

**dept\_id INT PRIMARY KEY,**

**dept\_name VARCHAR(50) NOT NULL UNIQUE**

**);**

**CREATE TABLE Employee1 (**

**emp\_id INT PRIMARY KEY,**

**emp\_name VARCHAR(100) NOT NULL**

**);**

**CREATE TABLE Pay\_Details (**

**emp\_id INT,**

**dept\_id INT,**

**basic INT CHECK (basic > 0),**

**deductions INT DEFAULT 0,**

**additions INT DEFAULT 0,**

**DOJ DATE,**

**PRIMARY KEY (emp\_id, dept\_id),**

**FOREIGN KEY (emp\_id) REFERENCES Employee1(emp\_id) ON DELETE CASCADE,**

**FOREIGN KEY (dept\_id) REFERENCES Department(dept\_id) ON DELETE CASCADE**

**);**

**CREATE TABLE Payroll (**

**emp\_id INT,**

**pay\_date DATE,**

**PRIMARY KEY (emp\_id, pay\_date),**

**FOREIGN KEY (emp\_id) REFERENCES Employee1(emp\_id) ON DELETE CASCADE**

**);**

**B.** Insert around 10 records in each of the tables.

**INSERT INTO Department VALUES**

**(1, 'HR'),**

**(2, 'Finance'),**

**(3, 'IT'),**

**(4, 'Sales'),**

**(5, 'Marketing');**

**INSERT INTO Employee1 VALUES**

**(1, 'Koushi'),**

**(2, 'Meghana'),**

**(3, 'Srinidhi'),**

**(4, 'Vainavi'),**

**(5, 'Akshaya'),**

**(6, 'Kavya'),**

**(7, 'Snehil'),**

**(8, 'Akshith'),**

**(9, 'Rishith'),**

**(10, 'Pratheesh');**

**INSERT INTO Pay\_Details VALUES**

**(1, 1, 45000, 2000, 5000, '2022-03-10'),**

**(2, 2, 60000, 5000, 10000, '2021-08-15'),**

**(3, 3, 80000, 4000, 8000, '2023-01-05'),**

**(4, 1, 52000, 3000, 7000, '2022-11-20'),**

**(5, 2, 120000, 10000, 15000, '2020-07-01'),**

**(6, 3, 95000, 8000, 5000, '2021-09-12'),**

**(7, 4, 40000, 2000, 3000, '2023-06-18'),**

**(8, 4, 70000, 5000, 4000, '2022-12-22'),**

**(9, 5, 85000, 7000, 6000, '2023-02-10'),**

**(10, 5, 50000, 3000, 2000, '2021-05-25');**

**INSERT INTO Payroll VALUES**

**(1, '2023-12-31'),**

**(2, '2023-12-31'),**

**(3, '2023-12-31'),**

**(4, '2023-12-31'),**

**(5, '2023-12-31'),**

**(6, '2023-12-31'),**

**(7, '2023-12-31'),**

**(8, '2023-12-31'),**

**(9, '2023-12-31'),**

**(10, '2023-12-31');**

**C.** List the employee details department wise.

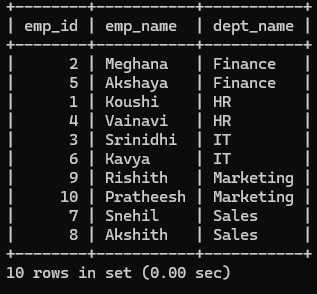
**SELECT e.emp\_id, e.emp\_name, d.dept\_name**

**FROM Employee1 e**

**JOIN Pay\_Details p ON e.emp\_id = p.emp\_id**

**JOIN Department d ON p.dept\_id = d.dept\_id**

**ORDER BY d.dept\_name;**

****

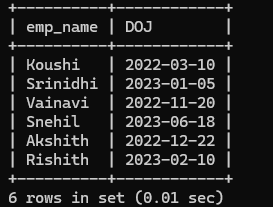
**D.** List all the employee names who joined after particular date.

**SELECT e.emp\_name, p.DOJ**

**FROM Employee1 e**

**JOIN Pay\_Details p ON e.emp\_id = p.emp\_id**

**WHERE p.DOJ > '2022-01-01';**

****

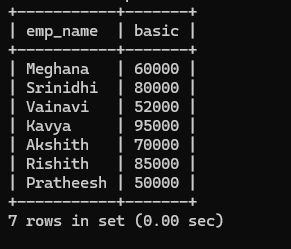
**e.** List the details of employees whose basic salary is between 50,000 and 1,00,000

**SELECT e.emp\_name, p.basic**

**FROM Employee1 e**

**JOIN Pay\_Details p ON e.emp\_id = p.emp\_id**

**WHERE p.basic BETWEEN 50000 AND 100000;**

****

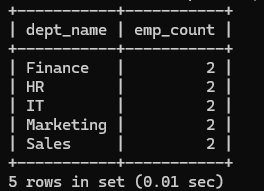
**f.** Give a count of how many employees are working in each department.

**SELECT d.dept\_name, COUNT(\*) AS emp\_count**

**FROM Department d**

**JOIN Pay\_Details p ON d.dept\_id = p.dept\_id**

**GROUP BY d.dept\_name;**

****

**g.** Give a name of the employees whose net salary>1,00,000.

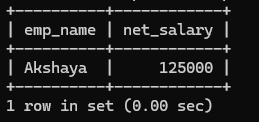
SELECT e.emp\_name,

(p.basic + p.additions - p.deductions) AS net\_salary

FROM Employee1 e

JOIN Pay\_Details p ON e.emp\_id = p.emp\_id

WHERE (p.basic + p.additions - p.deductions) > 100000;



**h.** List the details for an employee\_id=5

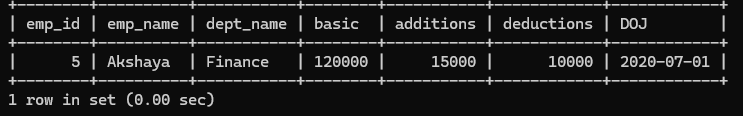
**SELECT e.emp\_id, e.emp\_name, d.dept\_name, p.basic, p.additions, p.deductions, p.DOJ**

**FROM Employee1 e**

**JOIN Pay\_Details p ON e.emp\_id = p.emp\_id**

**JOIN Department d ON p.dept\_id = d.dept\_id**

**WHERE e.emp\_id = 5;**

****

**i.** Create a view which lists out the emp\_name, department, basic, deductions, net salary.

**CREATE VIEW Employee1\_Salary\_View AS**

**SELECT e.emp\_name, d.dept\_name, p.basic, p.deductions,**

**(p.basic + p.additions - p.deductions) AS net\_salary**

**FROM Employee1 e**

**JOIN Pay\_Details p ON e.emp\_id = p.emp\_id**

**JOIN Department d ON p.dept\_id = d.dept\_id;**

**CREATE VIEW Employee\_Salary\_View AS**

**SELECT e.emp\_name, d.dept\_name, p.basic, p.deductions,**

**(p.basic + p.additions - p.deductions) AS net\_salary**

**FROM Employee e**

**JOIN Pay\_Details p ON e.emp\_id = p.emp\_id**

**JOIN Department d ON p.dept\_id = d.dept\_id;**