```
create database employee;
use employee;
CREATE TABLE employee (
    emp_id INT PRIMARY KEY,
    e_name VARCHAR(100),
    salary DECIMAL(10, 2),
    Date_of_Joining DATE,
    Dapt_no INT,
    Designation VARCHAR(50)
);


INSERT INTO employee (emp_id, e_name, salary, Date_of_Joining, Dapt_no,
Designation) VALUES
(1, 'Jaya Bhatt', 50000, '2024-01-15', 1, 'Manager'),
(2, 'Arya shah', 45000, '2022-05-20', 2, 'Marketing'),
(3, 'Vicky Rao', 40000, '2023-07-18', 3, 'Developer'),
(4, 'Sushma Kher', 55000, '2024-02-25', 1, 'Manager'),
(5, 'Ritu Mitra', 38000, '2022-09-10', 4, 'HR'),
(6, 'Manvi Mehta', 47000, '2023-11-05', 2, 'Sales'),
(7, 'Meera Patel', 53000, '2024-06-14', 3, 'Developer'),
(8, 'Radha Sinha', 41000, '2023-08-21', 4, 'Analyst'),
(9, 'Gauksh Sharma', 48000, '2022-10-17', 1, 'Manager'),
(10, 'Viha Raina ', 46000, '2022-03-22', 2, 'Marketing');
select * from employee;

CREATE VIEW emp_v1 AS
SELECT emp_id, e_name, Dapt_no
FROM employee;
select * from emp_v1;

UPDATE emp_v1
SET Dapt_no = 5
WHERE emp_id = 3;

select* from emp_v1;
select*from employee where emp_id=3;

ALTER TABLE employee
ADD CONSTRAINT pk_emp_id PRIMARY KEY (emp_id);

show index from employee;

CREATE INDEX idx_designation ON employee(Designation);
show index from employee;
```

```
create database employeeinfo;
use employeeinfo;

CREATE TABLE employee (
    emp_id INT PRIMARY KEY,
    e_name VARCHAR(100),
    salary DECIMAL(10, 2),
    Date_of_Joining DATE,
    Dept_no INT,
    Designation VARCHAR(50),
    Location VARCHAR(100)
);


INSERT INTO employee (emp_id, e_name, salary, Date_of_Joining, Dept_no,
Designation, Location) VALUES
(1, 'Jaya Bhatt', 50000, '2024-01-15', 1, 'Manager', 'Mumbai'),
(2, 'Arya shah', 45000, '2022-05-20', 2, 'Marketing', 'Delhi'),
(3, 'Vicky Rao', 40000, '2023-07-18', 3, 'Developer', 'Mumbai'),
(4, 'Sushma Kher', 55000, '2024-02-25', 1, 'Manager', 'Bangalore'),
(5, 'Ritu Mitra', 38000, '2022-09-10', 4, 'HR', 'Mumbai'),
(6, 'Manvi Mehta', 47000, '2023-11-05', 2, 'Sales', 'Pune'),
(7, 'Meera Patel', 53000, '2024-06-14', 3, 'Developer', 'Mumbai'),
(8, 'Radha Sinha', 41000, '2023-08-21', 4, 'Analyst', 'Hyderabad'),
(9, 'Gauksh Sharma', 48000, '2022-10-17', 1, 'Manager', 'Mumbai'),
(10, 'Viha Raina ', 46000, '2022-03-22', 2, 'Marketing', 'Delhi');
select* from employee;

SELECT * FROM employee
WHERE e_name LIKE '%e%';

SELECT DISTINCT Designation FROM employee;

SELECT e_name, salary FROM employee
WHERE Location = 'Mumbai';

SELECT e_name, Dept_no FROM employee
WHERE Designation = 'Manager' OR Designation = 'Marketing';

SELECT Dept_no, COUNT(emp_id) AS num_employees FROM employee
GROUP BY Dept_no
HAVING COUNT(emp_id) > 1;

RENAME TABLE employee TO emp1;

ALTER TABLE emp1
ADD COLUMN city VARCHAR(50);
```

```sql
create database employeeinfo;
use employeeinfo;

CREATE TABLE employee (
    emp_id INT PRIMARY KEY,
    e_name VARCHAR(100),
    salary DECIMAL(10, 2),
    Date_of_Joining DATE,
    Dept_no INT,
    Designation VARCHAR(50)
);

INSERT INTO employee (emp_id, e_name, salary, Date_of_Joining, Dept_no,
Designation) VALUES
(1, 'Jaya Bhatt', 50000, '2024-01-15', 1, 'Manager'),
(2, 'Arya shah', 45000, '2022-05-20', 2, 'Marketing'),
(3, 'Vicky Rao', 40000, '2023-07-18', 3, 'Developer'),
(4, 'Sushma Kher', 55000, '2024-02-25', 1, 'Manager'),
(5, 'Ritu Mitra', 38000, '2022-09-10', 4, 'HR'),
(6, 'Manvi Mehta', 47000, '2023-11-05', 2, 'Sales'),
(7, 'Meera Patel', 53000, '2024-06-14', 3, 'Developer'),
(8, 'Radha Sinha', 41000, '2023-08-21', 4, 'Analyst'),
(9, 'Gauksh Sharma', 48000, '2022-10-17', 1, 'Manager'),
(10, 'Viha Raina ', 46000, '2022-03-22', 2, 'Marketing');
select* from employee;

SELECT Dept_no, COUNT(emp_id) AS employee_count
FROM employee
GROUP BY Dept_no
ORDER BY employee_count DESC
LIMIT 1;


SELECT e_name, Designation, Dept_no
FROM employee
WHERE e_name LIKE 'A%' OR e_name LIKE 'P%';


SELECT MAX(salary) AS max_salary_dept2 FROM employee WHERE Dept_no = 2;
SELECT MIN(salary) AS min_salary_dept4 FROM employee WHERE Dept_no = 4;


SELECT * FROM employee
WHERE salary < (SELECT AVG(salary) FROM employee WHERE Dept_no = 3);


SELECT * FROM employee
ORDER BY Date_of_Joining ASC
LIMIT 1;
SELECT * FROM employee
ORDER BY Date_of_Joining DESC
LIMIT 1;


SELECT e_name, Dept_no
FROM employee
WHERE Designation IN ('Manager', 'Market Analyst');


SELECT * FROM employee
```

```
WHERE MONTH(Date_of_Joining) = 8;


SELECT * FROM employee
WHERE Date_of_Joining > '2006-12-31';


SELECT Dept_no, AVG(salary) AS average_annual_salary
FROM employee
GROUP BY Dept_no;
```

```sql
create database customerinfo;
use customerinfo;

CREATE TABLE Customer (
    c_id INT PRIMARY KEY,
    c_name VARCHAR(100),
    email VARCHAR(100),
    city VARCHAR(50),
    pincode INT
);

CREATE TABLE Orders (
    order_id INT PRIMARY KEY,
    date DATE,
    amount DECIMAL(10, 2),
    cust_id INT,
    FOREIGN KEY (cust_id) REFERENCES Customer(c_id)
);


INSERT INTO Customer (c_id, c_name, email, city, pincode) VALUES
(1, 'Amay Kumar', 'amay.kumar@example.com', 'Mumbai', 400001),
(2, 'Riya Varma', 'riya.varma@example.com', 'Delhi', 110001),
(3, 'Ranvir Singh', 'ranvir.singh@example.com', 'Mumbai', 400002),
(4, 'Sunita Das', 'sunita.das@example.com', 'Bangalore', 560001),
(5, 'Rahul Mehta', 'rahul.mehta@example.com', 'Chennai', 600001),
(6, 'Anita Sharma', 'anita.sharma@example.com', 'Pune', 411001),
(7, 'Suresh Jha', 'suresh.jha@example.com', 'Hyderabad', 500001),
(8, 'Piya Patel', 'piya.patel@example.com', 'Ahmedabad', 380001),
(9, 'Raj Khanna', 'raj.khanna@example.com', 'Jaipur', 302001),
(10, 'Kiya Gupta', 'kiya.gupta@example.com', 'Kolkata', 700001);
select * from Customer;

INSERT INTO Orders (order_id, date, amount, cust_id) VALUES
(101, '2023-01-15', 1500.00, 1),
(102, '2023-01-20', 2000.00, 2),
(103, '2023-01-22', 2500.00, 3),
(104, '2023-01-25', 3000.00, 4),
(105, '2023-01-30', 1000.00, 5),
(106, '2023-02-01', 4000.00, 6),
(107, '2023-02-05', 1200.00, 7),
(108, '2023-02-10', 3500.00, 8),
(110, '2023-02-20', 3200.00, 10);
select* from Orders;

SELECT * FROM Orders
WHERE cust_id = 2;


SELECT Customer.*, Orders.order_id, Orders.date, Orders.amount
FROM Customer
JOIN Orders ON Customer.c_id = Orders.cust_id;


SELECT * FROM Customer
WHERE c_id NOT IN (SELECT cust_id FROM Orders);


SELECT Customer.*, Orders.order_id, Orders.date, Orders.amount
FROM Customer
```

```
LEFT JOIN Orders ON Customer.c_id = Orders.cust_id;


SELECT * FROM Customer;


SELECT c1.c_name AS Customer1, c2.c_name AS Customer2, c1.city
FROM Customer c1, Customer c2
WHERE c1.city = c2.city AND c1.c_id < c2.c_id;
```

```
create database library;
use library;

CREATE TABLE Borrower (
    RollNo INT PRIMARY KEY,
    Name VARCHAR(100),
    DateofIssue DATE,
    NameofBook VARCHAR(100),
    Status VARCHAR(10)
);

CREATE TABLE Fine (
    Roll_no INT,
    Date DATE,
    Amt DECIMAL(10, 2),
    PRIMARY KEY (Roll_no, Date)
);


INSERT INTO Borrower (RollNo, Name, DateofIssue, NameofBook, Status)
VALUES
(1, 'Anish Kumar', '2023-01-10', 'Database Systems', 'Issued'),
(2, 'Sujata Singh', '2023-01-15', 'Operating Systems', 'Returned'),
(3, 'Ravina Patel', '2023-01-20', 'Computer Networks', 'Issued'),
(4, 'Priyal Desai', '2023-01-25', 'Algorithms', 'Returned'),
(5, 'Vicky Das', '2023-02-01', 'Data Structures', 'Issued'),
(6, 'Kiya Gupta', '2023-02-05', 'Software Engineering', 'Returned'),
(7, 'Neha Raina', '2023-02-10', 'Discrete Mathematics', 'Issued'),
(8, 'Raj Khanna', '2023-02-25', 'Artificial Intelligence', 'Issued'),
(9, 'Amit Bhatt', '2023-02-20', 'Machine Learning', 'Returned'),
(10, 'Priya Joshi', '2023-02-25', 'Big Data', 'Issued');
select* from Borrower;

INSERT INTO Fine (Roll_no, Date, Amt) VALUES
(1, '2023-01-20', 50),
(2, '2023-01-25', 20),
(3, '2023-02-05', 30),
(4, '2023-02-10', 10),
(5, '2023-02-15', 40),
(6, '2023-02-18', 25),
(7, '2023-02-22', 15),
(8, '2023-02-28', 35),
(9, '2023-03-02', 60),
(10, '2023-03-05', 45);
select* from Fine;

SELECT COUNT(*) AS Issued_Books_Count
FROM Borrower
WHERE Status = 'Issued';


SELECT * FROM Borrower;


SELECT RollNo, DateofIssue
FROM Borrower
WHERE DateofIssue IN (
    SELECT DateofIssue
    FROM Borrower
    GROUP BY DateofIssue
    HAVING COUNT(*) > 1
);
```

```
create database studentinfo;
use studentinfo;


CREATE TABLE student (
    roll_no INT PRIMARY KEY,
    name VARCHAR(100),
    marks INT,
    class VARCHAR(50)
);

INSERT INTO student (roll_no, name, marks, class ) VALUES
(1, 'Anil Sharma', 780, '10th' ),
(2, 'Neha Raina', 850, '12th'),
(3, 'Rakhi Patel', 620, '11th');
select* from student;

ALTER TABLE student
ADD COLUMN age INT;
select* from student;

ALTER TABLE student
CHANGE marks total_marks INT;
select* from student;

UPDATE student
SET total_marks = 900
WHERE roll_no = 2;
select * from student;


DELETE FROM student
WHERE roll_no = 3;
select * from student;

Alter table student
drop column age;
select* from student;
```

```
create database jobinfo;
use jobinfo;

CREATE TABLE jobs (
    job_id INT PRIMARY KEY,
    job_title VARCHAR(100),
    min_sal DECIMAL(10, 2),
    max_sal DECIMAL(10, 2)
);


INSERT INTO jobs (job_id, job_title, min_sal, max_sal) VALUES
(1, 'Software Engineer', 30000, 60000),
(2, 'Data Analyst', 25000, 55000),
(3, 'Project Manager', 50000, 90000),
(4, 'HR Specialist', 20000, 45000);


CREATE TABLE job_history (
    employee_id INT UNIQUE,
    start_date DATE,
    end_date DATE,
    job_id INT,
    department_id INT,
    FOREIGN KEY (job_id) REFERENCES jobs(job_id)
);

INSERT INTO job_history (employee_id, start_date, end_date, job_id,
department_id) VALUES
(101, '2020-01-15', '2022-06-30', 1, 10),
(102, '2019-03-20', '2021-12-01', 2, 20),
(103, '2018-07-10', '2023-08-20', 3, 30),
(104, '2021-09-01', NULL, 4, 40);
select * from jobs;
select * from job_history;
```

```
create database employeeinfo;
use employeeinfo;


CREATE TABLE employee (
    employee_name VARCHAR(100),
    street VARCHAR(100),
    city VARCHAR(50)
);


CREATE TABLE works (
    employee_name VARCHAR(100),
    company_name VARCHAR(100),
    salary DECIMAL(10, 2)
);


CREATE TABLE company (
    company_name VARCHAR(100),
    city VARCHAR(50)
);


CREATE TABLE manages (
    employee_name VARCHAR(100),
    manager_name VARCHAR(100)
);


INSERT INTO employee (employee_name, street, city) VALUES
('Amit Verma', 'MG Road', 'Mumbai'),
('Suman Rai', 'Park Street', 'Kolkata'),
('Rakhi patel', 'Civil Lines', 'Delhi');
select * from employee;

INSERT INTO works (employee_name, company_name, salary) VALUES
('Amit Verma', 'TechCorp', 12000),
('Suman Rai', 'DataSolutions', 13000),
('Rakhi Patel', 'TechCorp', 11000);
select* from works;

INSERT INTO company (company_name, city) VALUES
('TechCorp', 'Mumbai'),
('DataSolutions', 'Kolkata');
select * from company;

INSERT INTO manages (employee_name, manager_name) VALUES
('Amit Verma', 'Suman Rao'),
('Rakhi Patel', 'Amit Sharma');
select* from manages;

SELECT e.employee_name, e.street, e.city
FROM employee e
JOIN works w ON e.employee_name = w.employee_name
WHERE w.salary > 10000;


SELECT e.employee_name
FROM employee e
```

```
JOIN works w ON e.employee_name = w.employee_name
JOIN company c ON w.company_name = c.company_name
WHERE e.city = c.city;


SELECT w.employee_name
FROM works w
JOIN (
    SELECT company_name, AVG(salary) AS avg_salary
    FROM works
    GROUP BY company_name
) AS avg_salaries ON w.company_name = avg_salaries.company_name
WHERE w.salary > avg_salaries.avg_salary;
```

```
create database employeeinfo;
use employeeinfo;

CREATE TABLE employee (
    employee_name VARCHAR(100),
    street VARCHAR(100),
    city VARCHAR(50)
);

CREATE TABLE works (
    employee_name VARCHAR(100),
    company_name VARCHAR(100),
    salary DECIMAL(10, 2)
);

CREATE TABLE company (
    company_name VARCHAR(100),
    city VARCHAR(50)
);

CREATE TABLE manages (
    employee_name VARCHAR(100),
    manager_name VARCHAR(100)
);


INSERT INTO employee (employee_name, street, city) VALUES
('Amit Sharma', 'MG Road', 'Mumbai'),
('Suman Rao', 'MG Road', 'Mumbai'),
('Rakesh Kumar', 'Park Street', 'Delhi'),
('Neha Patel', 'Noida', 'Delhi');
select* from employee;

INSERT INTO works (employee_name, company_name, salary) VALUES
('Amit Sharma', 'TechCorp', 5000),
('Suman Rao', 'TechCorp', 3000),
('Rakesh Kumar', 'DataSolutions', 4000),
('Neha Patel', 'DataSolutions', 4500);
select* from works;

INSERT INTO company (company_name, city) VALUES
('TechCorp', 'Mumbai'),
('DataSolutions', 'Delhi');
select* from company;

INSERT INTO manages (employee_name, manager_name) VALUES
('Amit Sharma', 'Suman Rao'),
('Rakesh Kumar', 'Neha Patel'),
('Suman Rao', 'Amit Sharma');
select* from manages;

SELECT company_name
FROM works
GROUP BY company_name
ORDER BY SUM(salary) ASC
LIMIT 1;


SELECT e1.employee_name AS employee, e2.employee_name AS manager
FROM employee e1
```

```
JOIN manages m ON e1.employee_name = m.employee_name
JOIN employee e2 ON m.manager_name = e2.employee_name
WHERE e1.city = e2.city AND e1.street = e2.street;
```

```
mongosh
show dbs
use mydatabase

db.createCollection("items");

db.items.insertMany([
    { "item_id": 1, "item_quantity": 100, "price": 50, "brand": "BrandA"
},
    { "item_id": 2, "item_quantity": 150, "price": 30, "brand": "BrandB"
},
    { "item_id": 3, "item_quantity": 200, "price": 20, "brand": "BrandA"
},
    { "item_id": 4, "item_quantity": 250, "price": 60, "brand": "BrandC"
},
    { "item_id": 5, "item_quantity": 300, "price": 40, "brand": "BrandB"
}
]);


db.items.insertOne({ "item_id": 6, "item_quantity": 80, "price": 25,
"brand": "BrandD" });


db.items.find().pretty();


db.items.updateOne({ "item_id": 1 }, { $set: { "item_quantity": 120 } });


db.items.deleteOne({ "item_id": 2 });

var mapFunction = function() {
    emit(this.item_id, this.item_quantity);
};

var reduceFunction = function(key, values) {
    return Array.sum(values);
};


db.items.mapReduce(
    mapFunction,
    reduceFunction,
    { out: "item_quantity_count" }
);


db.item_quantity_count.find().pretty();

// db.collection.save(document)
// Create a new item document
// var newItem = { "item_id": 7, "item_quantity": 50, "price": 15,
"brand": "BrandE" };

// Save the new item to the collection
// db.items.save(newItem); // This will insert a new document since
item_id is not present in the collection.

// Later, if we want to update the item with item_id 7
// newItem.item_quantity = 75; // Update the quantity
 //db.items.save(newItem); // This will update the existing document with
item_id 7.
```

```
mongosh
show dbs
use mydatabase2
db.createCollection("items")
db.items.insertOne({ "item_id": 6, "item_quantity": 80, "price": 25,
"brand": "BrandD" });
db.items.insertMany([
    { "item_id": 1, "item_quantity": 100, "price": 50, "brand": "BrandA"
},
    { "item_id": 2, "item_quantity": 150, "price": 30, "brand": "BrandB"
},
    { "item_id": 3, "item_quantity": 200, "price": 20, "brand": "BrandA"
},
    { "item_id": 4, "item_quantity": 250, "price": 60, "brand": "BrandC"
},
    { "item_id": 5, "item_quantity": 300, "price": 40, "brand": "BrandB"
}
]);
db.items.find().pretty();
db.items.find({ item_id: 1 }).pretty();
db.items.updateOne({ "item_id": 1 }, { $set: { "item_quantity": 120 } });
db.items.find().pretty();
db.items.updateMany(
    { brand: "BrandA" },
    { $inc: { item_quantity: 20 } }
);
db.items.find().pretty();
db.items.deleteOne({ "item_id": 2 });
db.items.find().pretty();
db.items.deleteMany({ brand: "BrandB" })
db.items.find().pretty();

var mapFunction = function() {
    emit(this.item_id, this.item_quantity);
};

var reduceFunction = function(key, values) {
    return Array.sum(values);
};


db.items.mapReduce(
    mapFunction,
    reduceFunction,
    { out: "item_quantity_count" }
);
```

```
mongosh
show dbs
use mydatabase2
db.createCollection("items")
db.items.insertOne({ item_id: 101, item_quantity: 40, price: 140 , brand:
"BrandD", discount: 10 });
db.items.insertMany([
    { item_id: 102, item_quantity: 50, price: 150, brand: "BrandA",
discount: 10 },
    { item_id: 103, item_quantity: 30, price: 250, brand: "BrandB",
discount: 15 },
    { item_id: 104, item_quantity: 20, price: 100, brand: "BrandC",
discount: 5 },
    { item_id: 105, item_quantity: 10, price: 300, brand: "BrandA",
discount: 20 },
    { item_id: 106, item_quantity: 40, price: 200, brand: "BrandB",
discount: 25 }
]);
db.items.find().pretty()
db.items.find({ item_id: 101 }).pretty();

db.items.updateOne({ item_id: 101 },{ $set: { price: 160 } })
db.items.find({item_id: 101}).pretty()

db.items.updateMany({ brand: "BrandA" }, { $inc: { item_quantity: 10 } })
db.items.find().pretty()

db.items.aggregate([
    { $group: { _id: "$brand", count: { $sum: 1 } } }
])
db.items.find().sort({ price: 1 }).limit(1).pretty();

db.items.aggregate([
    { $group: { _id: null, maxDiscount: { $max: "$discount" } } }
])

db.items.deleteOne({ item_id: 104 });
db.items.find().pretty()

db.items.deleteMany({ brand: "BrandB" })
db.items.find().pretty()
```

```
mongosh
show dbs
use mydatabase3

db.createCollection("student")

db.student.insertMany([
    { roll_no: 1, name: "Alice", marks: 85, class: "5" },
    { roll_no: 2, name: "Bob", marks: 90, class: "5" },
    { roll_no: 3, name: "Charlie", marks: 75, class: "5" },
    { roll_no: 4, name: "David", marks: 95, class: "5" },
    { roll_no: 5, name: "Eva", marks: 80, class: "5" }
])


var mapFunction = function() {
    emit(this.roll_no, this.marks);
};


var reduceFunction = function(key, values) {
    return Array.sum(values);
};


db.student.mapReduce(
    mapFunction,
    reduceFunction,
    { out: "total_marks" }
)


db.total_marks.find().pretty()
```

```
mongosh
show dbs
use mydatabase
db.createCollection("person")

db.person.insertMany([
    { person_id: 1, name: "Alice", addr: "123 Main St", profession: 
"Engineer" },
    { person_id: 2, name: "Bob", addr: "456 Maple Ave", profession: 
"Doctor" },
    { person_id: 3, name: "Charlie", addr: "789 Oak St", profession: 
"Engineer" },
    { person_id: 4, name: "David", addr: "101 Pine St", profession: 
"Artist" },
    { person_id: 5, name: "Eva", addr: "202 Birch St", profession: 
"Doctor" },
    { person_id: 6, name: "Frank", addr: "303 Cedar St", profession: 
"Artist" },
    { person_id: 7, name: "Grace", addr: "404 Elm St", profession: 
"Engineer" }
])


var mapFunction = function() {
    emit(this.profession, { names: [this.name], count: 1 });
};

var reduceFunction = function(key, values) {
    var reducedValue = { names: [], count: 0 };
    values.forEach(function(value) {
        reducedValue.names = reducedValue.names.concat(value.names);
        reducedValue.count += value.count;
    });
    return reducedValue;
};


db.person.mapReduce(
    mapFunction,
    reduceFunction,
    { out: "professions" }
)

db.professions.find().pretty()
```

```
mongosh

show dbs
use mydatabase

db.createCollection("person")
 db.person.insertOne({
 person_id: 1,
 name: "Amit Sharma",
 addr:"Delhi",
 profession:"Engineer"})

 db.person.insertMany([
 {person_id:2, name:"neha", addr:"Pune", profession:"Doctor"},
 {person_id:3, name:"alok", addr:"Nagpur", profession:"Merchant Navy"},
 {person_id:4, name:"akhilesh", addr:"Nashik", profession:"Doctor"},
 ])

db.person.find()
db.person.find({ person_id: 1 })
db.person.find().pretty()

 db.person.updateOne( { person_id: 2 }, {$set: {profession:"Lawyer"}})
 db.person.updateMany({ profession: "Doctor"}, {$set:
{profession:"Surgeon"}})


db.person.deleteOne({ person_id: 1 })
db.person.deleteMany({ profession: "Lawyer" })
 db.person.deleteMany({})
```

```
mongosh
show dbs
use mydatabase1
db.createCollection("employee")
db.employee.insertMany([
  { emp_id: 1, e_name: "Rahul Sharma", salary: 50000, Date_of_Joining:
new Date("2015-06-10"), Dapt_no: "Sales", Designation: "Sales Executive"
},
  { emp_id: 2, e_name: "Neha Patel", salary: 60000, Date_of_Joining: new
Date("2016-06-15"), Dapt_no: "Sales", Designation: "Sales Manager" },
  { emp_id: 3, e_name: "Amit Verma", salary: 45000, Date_of_Joining: new
Date("2016-06-05"), Dapt_no: "Production", Designation: "Production
Engineer" },
  { emp_id: 4, e_name: "Priya Singh", salary: 70000, Date_of_Joining: new
Date("2018-05-20"), Dapt_no: "Production", Designation: "Production
Manager" },
  { emp_id: 5, e_name: "Suresh Reddy", salary: 55000, Date_of_Joining:
new Date("2019-08-10"), Dapt_no: "HR", Designation: "HR Executive" },
  { emp_id: 6, e_name: "Anjali Mehta", salary: 80000, Date_of_Joining:
new Date("2020-07-01"), Dapt_no: "Sales", Designation: "Sales Lead" }
])

db.employee.aggregate([
  { $group: { _id: "$Dapt_no", count: { $sum: 1 } } }
])

db.employee.aggregate([
  { $match: { Dapt_no: "Sales" } },
  { $group: { _id: "$Dapt_no", avgSalary: { $avg: "$salary" } } }
])

db.employee.aggregate([
  { $match: { Date_of_Joining: { $gte: new Date("2016-06-01"), $lt: new
Date("2016-07-01") } } },
  { $group: { _id: null, minSalary: { $min: "$salary" } } }
])

db.employee.aggregate([
  { $match: { Dapt_no: "Production" } },
  { $group: { _id: "$Dapt_no", maxSalary: { $max: "$salary" } } }
])

db.employee.aggregate([
  { $sort: { Date_of_Joining: 1 } },
  { $group: {
      _id: "$Dapt_no",
      firstEmployee: { $first: "$$ROOT" },
      lastEmployee: { $last: "$$ROOT" }
    }
  }
])
```

```
CREATE TABLE student (
    roll_no INT PRIMARY KEY,
    name VARCHAR(50),
    marks FLOAT,
    class VARCHAR(10)
);

import java.sql.*;
import java.util.Scanner;

public class StudentManagement {

    // Database connection details
    static final String DB_URL =
"jdbc:mysql://localhost:3306/your_database_name";
    static final String USER = "your_username";
    static final String PASS = "your_password";
    static Connection conn = null;

    public static void main(String[] args) {
        try {
            // Establish the connection
            conn = DriverManager.getConnection(DB_URL, USER, PASS);
            System.out.println("Connected to the database.");

            Scanner scanner = new Scanner(System.in);
            int choice;

            // Menu-driven program
            do {
                System.out.println("\n--- Student Management System ---
");
                System.out.println("1. Add Student");
                System.out.println("2. Update Student");
                System.out.println("3. Delete Student");
                System.out.println("4. Exit");
                System.out.print("Enter your choice: ");
                choice = scanner.nextInt();

                switch (choice) {
                    case 1:
                        addStudent();
                        break;
                    case 2:
                        updateStudent();
                        break;
                    case 3:
                        deleteStudent();
                        break;
                    case 4:
                        System.out.println("Exiting...");
                        break;
                    default:
                        System.out.println("Invalid choice. Please try
again.");
                }
            } while (choice != 4);

            scanner.close();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                if (conn != null) conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    // Method to add a student
    public static void addStudent() {
        try (Scanner scanner = new Scanner(System.in)) {
            System.out.print("Enter Roll Number: ");
            int rollNo = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            System.out.print("Enter Name: ");
            String name = scanner.nextLine();
            System.out.print("Enter Marks: ");
            float marks = scanner.nextFloat();
            scanner.nextLine(); // Consume newline
            System.out.print("Enter Class: ");
            String className = scanner.nextLine();

            String sql = "INSERT INTO student (roll_no, name, marks,
class) VALUES (?, ?, ?, ?)";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setInt(1, rollNo);
            stmt.setString(2, name);
            stmt.setFloat(3, marks);
            stmt.setString(4, className);
            stmt.executeUpdate();
            System.out.println("Student added successfully.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    // Method to update student details
    public static void updateStudent() {
        try (Scanner scanner = new Scanner(System.in)) {
            System.out.print("Enter Roll Number of student to update: ");
            int rollNo = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            System.out.print("Enter New Name: ");
            String name = scanner.nextLine();
            System.out.print("Enter New Marks: ");
            float marks = scanner.nextFloat();
            scanner.nextLine(); // Consume newline
            System.out.print("Enter New Class: ");
            String className = scanner.nextLine();

            String sql = "UPDATE student SET name = ?, marks = ?, class =
? WHERE roll_no = ?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setString(1, name);
            stmt.setFloat(2, marks);
            stmt.setString(3, className);
            stmt.setInt(4, rollNo);
            int rowsUpdated = stmt.executeUpdate();
            if (rowsUpdated > 0) {
                System.out.println("Student updated successfully.");
            } else {
                System.out.println("Student not found.");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    // Method to delete a student
    public static void deleteStudent() {
        try (Scanner scanner = new Scanner(System.in)) {
            System.out.print("Enter Roll Number of student to delete: ");
            int rollNo = scanner.nextInt();

            String sql = "DELETE FROM student WHERE roll_no = ?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setInt(1, rollNo);
            int rowsDeleted = stmt.executeUpdate();
            if (rowsDeleted > 0) {
                System.out.println("Student deleted successfully.");
            } else {
                System.out.println("Student not found.");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

```
create database studinfo;
use studinfo;
-- 18.1 Create the Stud_Marks table
CREATE TABLE Stud_Marks (
    name VARCHAR(100),
    total_marks INT
);

-- 18.2 Create the Result table to store student grades with auto-
increment for roll_no
CREATE TABLE Result (
    roll_no INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    grade VARCHAR(20)
);

-- 18.3 Create the stored procedure proc_Grade for grade categorization
DELIMITER //
CREATE PROCEDURE proc_Grade()
BEGIN
    DECLARE student_grade VARCHAR(20);

    DECLARE done INT DEFAULT FALSE;
    DECLARE student_name VARCHAR(100);
    DECLARE student_marks INT;

    DECLARE student_cursor CURSOR FOR SELECT name, total_marks FROM
Stud_Marks;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN student_cursor;

    read_loop: LOOP
        FETCH student_cursor INTO student_name, student_marks;
        IF done THEN
            LEAVE read_loop;
        END IF;

        IF student_marks >= 990 AND student_marks <= 1500 THEN
            SET student_grade = 'Distinction';
        elseif student_marks BETWEEN 900 AND 989 THEN
            SET student_grade = 'First Class';
        elseif student_marks BETWEEN 825 AND 899 THEN
            SET student_grade = 'Higher Second Class';
        ELSE
            SET student_grade = 'Pass';
        END IF;

        -- Insert into Result table with the categorized grade
        INSERT INTO Result (name, grade)
        VALUES (student_name, student_grade);
    END LOOP;

    CLOSE student_cursor;
END //
DELIMITER ;

-- 18.4 Insert sample data into Stud_Marks table
INSERT INTO Stud_Marks (name, total_marks) VALUES
('Amit Kumar', 1200),
('Neha Singh', 950),
('Rajesh Patil', 870),
('Priya Desai', 820),
('Rakesh Sharma', 880);

-- Display table contents before procedure execution
SELECT * FROM Stud_Marks;

-- Expected Output:
-- +-------------+-------------+
-- | name        | total_marks |
-- +-------------+-------------+
-- | Amit Kumar  | 1200        |
-- | Neha Singh  | 950         |
-- | Rajesh Patil| 870         |
-- | Priya Desai | 820         |
-- | Rakesh Sharma | 880       |
-- +-------------+-------------+

-- 18.5 Call the procedure to categorize students
CALL proc_Grade();

-- Display Result table after procedure execution
SELECT * FROM Result;

-- Expected Output:
-- +---------+-------------+---------------------+
-- | roll_no | name        | grade               |
-- +---------+-------------+---------------------+
-- |    1    | Amit Kumar  | Distinction         |
-- |    2    | Neha Singh  | First Class         |
-- |    3    | Rajesh Patil| Higher Second Class |
-- |    4    | Priya Desai | Pass                |
-- |    5    | Rakesh Sharma | Higher Second Class |
-- +---------+-------------+---------------------+

-- Explanation:
-- This procedure processes each studentâ€™s total marks and categorizes
them into grades,
-- storing the results in the `Result` table.
```

```sql
create database custinfo;
use custinfo;

CREATE TABLE customer (
    cust_id INT PRIMARY KEY,
    c_name VARCHAR(100),
    addr VARCHAR(100)
);


CREATE TABLE cust_Audit (
    audit_id INT AUTO_INCREMENT PRIMARY KEY,
    cust_id INT,
    c_name VARCHAR(100),
    addr VARCHAR(100),
    action_type VARCHAR(10), -- 'UPDATE' or 'DELETE'
    action_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

INSERT INTO customer (cust_id, c_name, addr) VALUES
(1, 'Amit Kumar', 'Mumbai'),
(2, 'Neha Singh', 'Delhi'),
(3, 'Rajesh Patil', 'Bangalore');


DELIMITER //


CREATE TRIGGER customer_update_audit
AFTER UPDATE ON customer
FOR EACH ROW
BEGIN
    INSERT INTO cust_Audit (cust_id, c_name, addr, action_type)
    VALUES (OLD.cust_id, OLD.c_name, OLD.addr, 'UPDATE');
END;
//


CREATE TRIGGER customer_delete_audit
AFTER DELETE ON customer
FOR EACH ROW
BEGIN
    INSERT INTO cust_Audit (cust_id, c_name, addr, action_type)
    VALUES (OLD.cust_id, OLD.c_name, OLD.addr, 'DELETE');
END;
//


DELIMITER ;



UPDATE customer
SET addr = 'Hyderabad'
WHERE cust_id = 1;


DELETE FROM customer
WHERE cust_id = 2;
```

```
SELECT * FROM cust_Audit;
```

```
create database clientinfo;
use clientinfo;

CREATE TABLE client_master (
    c_id INT PRIMARY KEY,
    c_name VARCHAR(100),
    acc_no INT
);

CREATE TABLE client_Audit (
    audit_id INT AUTO_INCREMENT PRIMARY KEY,
    c_id INT,
    c_name VARCHAR(100),
    acc_no INT,
    action_type VARCHAR(10), -- 'INSERT' or 'UPDATE'
    action_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

INSERT INTO client_master (c_id, c_name, acc_no) VALUES
(1, 'Amit Kumar', 12345),
(2, 'Neha Singh', 67890),
(3, 'Rajesh Patil', 11223);

DELIMITER //

CREATE TRIGGER client_insert_audit
AFTER INSERT ON client_master
FOR EACH ROW
BEGIN
    INSERT INTO client_Audit (c_id, c_name, acc_no, action_type)
    VALUES (NEW.c_id, NEW.c_name, NEW.acc_no, 'INSERT');
END;
//

CREATE TRIGGER client_update_audit
AFTER UPDATE ON client_master
FOR EACH ROW
BEGIN
    INSERT INTO client_Audit (c_id, c_name, acc_no, action_type)
    VALUES (OLD.c_id, OLD.c_name, OLD.acc_no, 'UPDATE');
END;
//


DELIMITER ;

INSERT INTO client_master (c_id, c_name, acc_no) VALUES (4, 'Pooja
Verma', 33445);

UPDATE client_master
SET acc_no = 55678
WHERE c_id = 1;
```

```
create database roll;
use roll;

CREATE TABLE N_RollCall (
    roll_no INT PRIMARY KEY,
    name VARCHAR(100),
    class VARCHAR(50)
);


CREATE TABLE O_RollCall (
    roll_no INT PRIMARY KEY,
    name VARCHAR(100),
    class VARCHAR(50)
);

INSERT INTO N_RollCall (roll_no, name, class) VALUES
(1, 'Amit Kumar', '10th'),
(2, 'Neha Singh', '12th'),
(3, 'Rajesh Patil', '10th');

INSERT INTO O_RollCall (roll_no, name, class) VALUES
(2, 'Neha Singh', '12th'),
(4, 'Priya Desai', '11th');

SELECT * FROM N_RollCall;
SELECT * FROM O_RollCall;

DELIMITER //

CREATE PROCEDURE merge_rollcall_data()
BEGIN
    DECLARE v_roll_no INT;
    DECLARE v_name VARCHAR(100);
    DECLARE v_class VARCHAR(50);

    DECLARE done INT DEFAULT 0;

    DECLARE n_rollcall_cursor CURSOR FOR
        SELECT roll_no, name, class FROM N_RollCall;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN n_rollcall_cursor;

    read_loop: LOOP
        FETCH n_rollcall_cursor INTO v_roll_no, v_name, v_class;

        IF done THEN
            LEAVE read_loop;
        END IF;

        IF (SELECT COUNT(*) FROM O_RollCall WHERE roll_no = v_roll_no) =
0 THEN
            INSERT INTO O_RollCall (roll_no, name, class)
            VALUES (v_roll_no, v_name, v_class);
        END IF;
    END LOOP;

    CLOSE n_rollcall_cursor;
```

```
END;
//

DELIMITER ;

CALL merge_rollcall_data();

SELECT * FROM O_RollCall;
```

```
CREATE DATABASE library;
USE library;

CREATE TABLE Borrower (
    Rollno INT(4),
    Name VARCHAR(20),
    DateofIssue DATE,
    NameofBook VARCHAR(30),
    Status VARCHAR(10)
);

INSERT INTO Borrower VALUES (14, 'Ram', '2024-10-19', 'Operating System',
'I');
INSERT INTO Borrower VALUES (27, 'Soham', '2024-10-24', 'Object Oriented
Programming', 'I');
INSERT INTO Borrower VALUES (34, 'Mohan', '2024-09-12', 'Microprocessor',
'I');
INSERT INTO Borrower VALUES (48, 'Om', '2024-09-19', 'Mechanics', 'I');

SELECT * FROM Borrower;

CREATE TABLE Fine (
    Rollno INT(4),
    Date DATE,
    Amount INT(10)
);

DELIMITER //

CREATE PROCEDURE calc_Fine(IN r INT(10), IN b VARCHAR(30))
BEGIN
    DECLARE doi DATE;
    DECLARE diff INT(3);
    DECLARE CONTINUE HANDLER FOR NOT FOUND
        BEGIN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'No matching Borrower record found for the
given roll number and book name';
        END;

    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
        BEGIN
            ROLLBACK;
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'An error occurred while calculating
fine.';
        END;

    START TRANSACTION;

    SELECT DateofIssue INTO doi FROM Borrower WHERE Rollno = r AND
NameofBook = b;

    SELECT DATEDIFF(CURDATE(), doi) INTO diff;

    IF diff >= 15 AND diff <= 30 THEN
        INSERT INTO Fine VALUES (r, CURDATE(), diff * 5);
    ELSEIF diff > 30 THEN
        INSERT INTO Fine VALUES (r, CURDATE(), diff * 50);
    END IF;
    COMMIT;
END//

CREATE PROCEDURE submit(IN r INT(2))
BEGIN
    DECLARE CONTINUE HANDLER FOR NOT FOUND
        BEGIN
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'No matching Borrower record found for the
given roll number';
        END;

    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
        BEGIN
            ROLLBACK;
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'An error occurred while submitting the
book.';
        END;

    START TRANSACTION;

    UPDATE Borrower SET Status = 'R' WHERE Rollno = r;
    DELETE FROM Fine WHERE Rollno = r;

    COMMIT;
END//

DELIMITER ;

CALL calc_Fine(14, 'Operating System');
SELECT * FROM Fine;

CALL calc_Fine(27, 'Object Oriented Programming');
CALL calc_Fine(34, 'Microprocessor');
CALL calc_Fine(48, 'Mechanics');
SELECT * FROM Fine;

CALL submit(14);
CALL submit(27);
CALL submit(48);
CALL submit(34);

SELECT * FROM Fine;
SELECT * FROM Borrower;
```

```
create database employee;
use employee;

CREATE TABLE employee (
    emp_id INT PRIMARY KEY,
    e_name VARCHAR(100),
    salary DECIMAL(10, 2),
    date_of_joining DATE,
    dapt_no INT,
    designation VARCHAR(50)
);

INSERT INTO employee (emp_id, e_name, salary, date_of_joining, dapt_no,
designation) VALUES
(1, 'Amit Kumar', 50000, '2020-01-15', 1, 'Manager'),
(2, 'Neha Singh', 45000, '2019-03-20', 2, 'Marketing'),
(3, 'Rajesh Patil', 40000, '2021-06-18', 3, 'Developer'),
(4, 'Priya Desai', 47000, '2018-09-10', 2, 'Sales'),
(5, 'Rakesh Sharma', 52000, '2017-11-05', 1, 'Manager');

SELECT * FROM employee;

CREATE VIEW emp_vl AS
SELECT emp_id, e_name, dapt_no
FROM employee;

SELECT * FROM emp_vl;

SELECT e_name, dapt_no
FROM employee
WHERE designation = 'Manager' OR designation = 'Marketing';


SELECT * FROM employee
ORDER BY date_of_joining ASC
LIMIT 1;
SELECT * FROM employee
ORDER BY date_of_joining DESC
LIMIT 1;

SELECT e_name, dapt_no
FROM employee
WHERE designation IN ('Manager', 'Market Analyst');

SELECT * FROM employee
WHERE MONTH(date_of_joining) = 8;

SELECT * FROM employee
WHERE date_of_joining > '2006-12-31';
```

```
CREATE DATABASE IF NOT EXISTS company_db;
USE company_db;

CREATE TABLE employee (
    emp_id INT PRIMARY KEY,
    e_name VARCHAR(100),
    salary DECIMAL(10, 2),
    Date_of_Joining DATE,
    Dapt_no INT,
    Designation VARCHAR(50)
);

CREATE INDEX idx_employee_name ON employee (e_name);
CREATE INDEX idx_department ON employee (Dapt_no);
CREATE INDEX idx_salary ON employee (salary);

CREATE TABLE customer (
    c_id INT PRIMARY KEY,
    c_name VARCHAR(100),
    email VARCHAR(100),
    city VARCHAR(50),
    pincode VARCHAR(10)
);

CREATE TABLE orders (
    order_id INT PRIMARY KEY AUTO_INCREMENT,
    date DATE,
    amount DECIMAL(10, 2),
    cust_id INT,
    FOREIGN KEY (cust_id) REFERENCES customer (c_id)
);

INSERT INTO employee (emp_id, e_name, salary, Date_of_Joining, Dapt_no,
Designation) VALUES
(1, 'Amit Sharma', 50000.00, '2022-05-01', 101, 'Manager'),
(2, 'Neha Verma', 45000.00, '2021-07-15', 102, 'Engineer'),
(3, 'Rajesh Kumar', 60000.00, '2020-09-10', 103, 'Senior Engineer'),
(4, 'Priya Singh', 40000.00, '2023-02-20', 104, 'Analyst'),
(5, 'Arjun Gupta', 55000.00, '2019-12-05', 105, 'Lead Engineer');

INSERT INTO customer (c_id, c_name, email, city, pincode) VALUES
(1, 'Sunil Mishra', 'sunil.mishra@example.com', 'Mumbai', '400001'),
(2, 'Anjali Jain', 'anjali.jain@example.com', 'Delhi', '110001'),
(3, 'Vikas Desai', 'vikas.desai@example.com', 'Pune', '411001'),
(4, 'Kavita Rao', 'kavita.rao@example.com', 'Bengaluru', '560001'),
(5, 'Rahul Patil', 'rahul.patil@example.com', 'Chennai', '600001');
(6, 'Rajni Singh', 'rajni.singh@example.com', 'Sawantwadi', '600002');

INSERT INTO orders (date, amount, cust_id) VALUES
('2023-08-12', 1200.00, 1),
('2023-08-13', 1500.00, 2),
('2023-08-15', 1800.00, 3),
('2023-08-17', 1100.00, 1),
('2023-08-20', 1400.00, 4),
('2023-08-22', 2000.00, 2),
('2023-08-25', 1700.00, 5);


SELECT customer.c_id, customer.c_name, customer.email, customer.city,
       orders.order_id, orders.date, orders.amount
FROM customer
JOIN orders ON customer.c_id = orders.cust_id;

SELECT customer.c_id, customer.c_name, customer.email, customer.city,
       orders.order_id, orders.date, orders.amount
FROM customer
LEFT JOIN orders ON customer.c_id = orders.cust_id;

SELECT customer.c_id, customer.c_name, customer.email, customer.city
FROM customer
LEFT JOIN orders ON customer.c_id = orders.cust_id
WHERE orders.order_id IS NULL;

TRUNCATE TABLE employee;
TRUNCATE TABLE customer;
TRUNCATE TABLE orders;
```

```
mongosh
show dbs
use mydatabase1
db.createCollection("orders")
db.orders.insertMany([
    { customer_id: 1, item: "Laptop", quantity: 2, price: 50000 },
    { customer_id: 2, item: "Phone", quantity: 1, price: 30000 },
    { customer_id: 1, item: "Tablet", quantity: 3, price: 20000 },
    { customer_id: 3, item: "Headphones", quantity: 5, price: 5000 },
    { customer_id: 2, item: "Laptop", quantity: 1, price: 50000 },
    { customer_id: 4, item: "Phone", quantity: 2, price: 30000 },
    { customer_id: 3, item: "Tablet", quantity: 1, price: 20000 }
])

db.orders.find()
db.orders.createIndex({ customer_id: 1 })
db.orders.createIndex({ customer_id: 1, item: 1 })
db.orders.createIndex({ item: "text" })

db.orders.aggregate([
    { $group: { _id: "$customer_id", totalOrders: { $sum: 1 } } }
])

db.orders.aggregate([
    { $group: {
        _id: "$item",
        totalQuantity: { $sum: "$quantity" },
        averagePrice: { $avg: "$price" }
    } }
])

db.orders.aggregate([
    { $group: {
        _id: "$customer_id",
        totalSales: { $sum: { $multiply: ["$quantity", "$price"] } }
    } }
])


db.orders.find({ customer_id: 1 }).explain("executionStats")
```