1. Write a MySQL script to create a schema with tables, applying constraints such as PRIMARY KEY, FOREIGN KEY, UNIQUE, and NOT NULL.

CREATE SCHEMA IF NOT EXISTS SchoolDB;

USE SchoolDB;

CREATE TABLE Students (StudentID INT AUTO_INCREMENT PRIMARY KEY, FirstName VARCHAR(50) NOT NULL, LastName VARCHAR(50) NOT NULL, Email VARCHAR(100) UNIQUE NOT NULL, DateOfBirth DATE NOT NULL);

CREATE TABLE Courses (CourseID INT AUTO_INCREMENT PRIMARY KEY, CourseName VARCHAR(100) NOT NULL, CourseCode VARCHAR(10) UNIQUE NOT NULL);

CREATE TABLE Enrollments (EnrollmentID INT AUTO_INCREMENT PRIMARY KEY, StudentID INT NOT NULL, CourseID INT NOT NULL, EnrollmentDate DATE NOT NULL, constraint FOREIGN KEY (StudentID) REFERENCES Students(StudentID) ON DELETE CASCADE, FOREIGN KEY (CourseID) REFERENCES Courses(CourseID) ON DELETE CASCADE);

INSERT INTO Students (FirstName, LastName, Email, DateOfBirth) VALUES ('Alice', 'Johnson', 'alice.johnson@example.com', '2000-05-15'), ('Bob', 'Smith', 'bob.smith@example.com', '1999-07-20');

INSERT INTO Courses (CourseName, CourseCode) VALUES ('Mathematics', 'MATH101'), ('Computer Science', 'CS101');

INSERT INTO Enrollments (StudentID, CourseID, EnrollmentDate) VALUES (1, 1, '2024-01-01'), (2, 2, '2024-01-02');

SELECT * FROM Students;

StudentID	FirstName	LastName	Email	DateOfBirth
1 2	Alice Bob		alice.johnson@example.com bob.smith@example.com	2000-05-15 1999-07-20

SELECT * FROM Courses;

CourseID	CourseName	CourseCode
•	Mathematics Computer Science	MATH101 CS101

SELECT* FROM Enrollments;

EnrollmentID	StudentID	CourseID	EnrollmentDate
1 2	1 2		2024-01-01 2024-01-02

2. Develop a MySQL script to create tables with data types like VRACHAR, INT, DATE, etc.

CREATE TABLE Employees (EmployeeID INT AUTO_INCREMENT PRIMARY KEY, FirstName VARCHAR(50) NOT NULL, LastName VARCHAR(50) NOT NULL, Email VARCHAR(100) UNIQUE NOT NULL, DateOfBirth DATE, HireDate DATE NOT NULL, Salary DECIMAL(10, 2) NOT NULL, IsActive BOOLEAN DEFAULT TRUE);

CREATE TABLE Departments (DepartmentID INT AUTO_INCREMENT PRIMARY KEY, DepartmentName VARCHAR(100) NOT NULL, Location VARCHAR(100), Budget DECIMAL(15, 2));

CREATE TABLE Projects (ProjectID INT AUTO_INCREMENT PRIMARY KEY, ProjectName VARCHAR(100) NOT NULL, StartDate DATE NOT NULL, EndDate DATE, ProjectManagerID INT);

INSERT INTO Employees (FirstName, LastName, Email, DateOfBirth, HireDate, Salary, IsActive) VALUES ('Alice', 'Johnson', 'alice.johnson@example.com', '1985-02-14', '2022-03-01', 55000.00, TRUE), ('Bob', 'Smith', 'bob.smith@example.com', '1990-06-25', '2023-01-15', 60000.00, TRUE);

INSERT INTO Departments (DepartmentName, Location, Budget) VALUES ('Human Resources', 'New York', 120000.00), ('Engineering', 'San Francisco', 300000.00);

INSERT INTO Projects (ProjectName, StartDate, EndDate, ProjectManagerID) VALUES ('Website Redesign', '2024-01-01', '2024-06-30', 1), ('Mobile App Development', '2024-03-01', NULL, 2);

SELECT * FROM Employees;

+ EmployeeID FirstNam	+ :	 Email	DateOfBirth	+ HireDate	 Salary	IsActive
1 Alice 2 Bob		alice.johnson@example.com bob.smith@example.com 	1985-02-14 1990-06-25			1 1

SELECT * FROM Departments;

DepartmentID	DepartmentName	Location	Budget
1	Human Resources	New York	120000.00
2	Engineering	San Francisco	300000.00

SELECT * FROM Projects;

ProjectID	ProjectName	StartDate	EndDate	ProjectManagerID
	Website Redesign Mobile App Development	2024-01-01 2024-03-01		1 2

3. Write a MySQL script to create a new database schema and assign appropriate permission to users (e.g., GRANT, REVOKE).

CREATE TABLE Employees (EmployeeID INT AUTO_INCREMENT PRIMARY KEY, FirstName VARCHAR(50) NOT NULL, LastName VARCHAR(50) NOT NULL, Email VARCHAR(100) UNIQUE NOT NULL, HireDate DATE NOT NULL);

CREATE USER 'manager'@'localhost' IDENTIFIED BY 'ManagerPass123!'; CREATE USER 'developer'@'localhost' IDENTIFIED BY 'DevPass123!';

GRANT ALL PRIVILEGES ON SchoolDB.* TO 'manager'@'localhost';

GRANT SELECT, INSERT, UPDATE ON SchoolDB.* TO 'developer'@'localhost';

FLUSH PRIVILEGES;

SHOW GRANTS FOR 'manager'@'localhost';

```
Grants for manager@localhost

GRANT USAGE ON *.* TO `manager`@`localhost`

GRANT ALL PRIVILEGES ON `schooldb`.* TO `manager`@`localhost`
```

SHOW GRANTS FOR 'developer'@'localhost';

```
| Grants for developer@localhost
|------|
| GRANT USAGE ON *.* TO `developer`@`localhost`
| GRANT SELECT, INSERT, UPDATE ON `schooldb`.* TO `developer`@`localhost`
```

REVOKE UPDATE ON SchoolDB.* FROM 'developer'@'localhost'; FLUSH PRIVILEGES;

4. Write a MySQL program to create a table that reflects different data types and applies constraints such as CHECK and DEFAULT.

CREATE TABLE Employee1 (EmployeeID INT AUTO_INCREMENT PRIMARY KEY, FirstName VARCHAR(50) NOT NULL,LastName VARCHAR(50) NOT NULL, Email VARCHAR(100) UNIQUE NOT NULL, PhoneNumber CHAR(10), Salary DECIMAL(10, 2) NOT NULL DEFAULT 30000, Department ENUM('HR', 'IT', 'Finance', 'Sales') NOT NULL DEFAULT 'HR', DateOfBirth DATE NOT NULL, Gender ENUM('Male', NULL, HireDate TIMESTAMP 'Female', 'Other') NOT CURRENT TIMESTAMP, IsActive BOOLEAN DEFAULT TRUE, CHECK (Salary >= 0), CHECK (Gender IN ('Male', 'Female', 'Other'));

DESC Employee1;

Field	Туре	Null	Key	Default	Extra
EmployeeID FirstName LastName Email PhoneNumber Salary Department DateOfBirth Gender HireDate IsActive	int varchar(50) varchar(50) varchar(100) char(10) decimal(10,2) enum('HR','IT','Finance','Sales') date enum('Male','Female','Other') timestamp tinyint(1)	NO NO NO NO YES NO NO NO YES YES	PRI UNI	NULL NULL NULL NULL NULL 30000.00 HR NULL NULL CURRENT_TIMESTAMP	auto_increment

INSERT INTO Employee1 (FirstName, LastName, Email, PhoneNumber, Salary, Department, DateOfBirth, Gender, HireDate, IsActive) VALUES ('Alice', 'Brown', 'alice.brown@example.com', '1234567890', 45000.00, 'IT', '1990-05-12', 'Female', '2023-06-01', TRUE), ('Bob', 'Smith', 'bob.smith@example.com', '9876543210', 35000.00, 'HR', '1985-09-15', 'Male', '2024-01-15', TRUE);

SELECT * FROM Employee1;

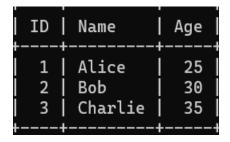
EmployeeID F	irstName	LastName	Email	PhoneNumber	Salary	Department	DateOfBirth	Gender	HireDate	IsActive
1 A 2 B			alice.brown@example.com bob.smith@example.com						2023-06-01 00:00:00 2024-01-15 00:00:00	

5. Write a MySQL script to demonstrate how to drop a table.

CREATE TABLE SampleTable (ID INT AUTO_INCREMENT PRIMARY KEY, Name VARCHAR(50) NOT NULL, Age INT NOT NULL);

INSERT INTO SampleTable (Name, Age) VALUES ('Alice', 25), ('Bob', 30), ('Charlie', 35);

SELECT * FROM SampleTable;



DROP TABLE SampleTable;

SHOW TABLES;

```
MySQL localhost:33060+ ssl employee SQL > show tables; Empty set (0.0037 sec)
```

6. Write a MySQL script to modify an existing table by adding or modifying columns (e.g., ALTER TABLE).

CREATE TABLE Employee1 (EmployeeID INT AUTO_INCREMENT FirstName VARCHAR(50) NOT NULL,LastName PRIMARY KEY. VARCHAR(50) NOT NULL, Email VARCHAR(100) UNIQUE NOT NULL, PhoneNumber CHAR(10), Salary DECIMAL(10, 2) NOT NULL DEFAULT 30000, Department ENUM('HR', 'IT', 'Finance', 'Sales') NOT NULL DEFAULT 'HR', DateOfBirth DATE NOT NULL, Gender ENUM('Male', 'Other') NOT NULL, HireDate TIMESTAMP 'Female', **DEFAULT** CURRENT_TIMESTAMP, IsActive BOOLEAN DEFAULT TRUE, CHECK (Salary >= 0), CHECK (Gender IN ('Male', 'Female', 'Other'));

INSERT INTO Employee1 (FirstName, LastName, Email, PhoneNumber, Salary, Department, DateOfBirth, Gender, HireDate, IsActive) VALUES ('Alice', 'Brown', 'alice.brown@example.com', '1234567890', 45000.00, 'IT', '1990-05-12', 'Female', '2023-06-01', TRUE), ('Bob', 'Smith', 'bob.smith@example.com', '9876543210', 35000.00, 'HR', '1985-09-15', 'Male', '2024-01-15', TRUE);

ALTER TABLE 'table_name' ADD columns 'new_column' DATATYPE [constraints];

ALTER TABLE 'employee1' ADD column 'phoneno' int;

Select * from employee1;

EmployeeID FirstName	LastName	Email	address	DateOfBirth	HireDate	salary	IsActive	phone	phoneno
		alice.johnson@example.com bob.smith@example.com		1985-02-14 1990-06-25				NULL NULL	NULL NULL

Modify an Existing column

ALTER TABLE 'table_name' Modify column 'existing_column_name' DATATYPE [Constraints];

ALTER TABLE 'employee1' Modify column 'salary' decimal(10,2) not null;

Select * from employee1;

EmployeeID FirstNam	e LastName	 Email	address	DateOfBirth	HireDate	salary	IsActive	phone	phoneno
1 Alice 2 Bob		alice.johnson@example.com	NULL NULL	1985-02-14 1990-06-25	:		:	NULL NULL	NULL

Drop a column

ALTER TABLE 'table name' Drop column 'column to drop;

ALTER TABLE 'employee1' Drop column 'phone';

Select * from employee1;

EmployeeID FirstName	LastName	Email	address	DateOfBirth	HireDate	salary	IsActive	phoneno
1 Alice 2 Bob		alice.johnson@example.com bob.smith@example.com		1985-02-14 1990-06-25				NULL NULL

Rename the column (if required)

ALTER TABLE 'table_name' change column 'old_column_name' 'new column name' DATATYPE[constraints];

ALTER TABLE 'employee1' change column 'phoneno' 'contact_number' int;

Select * from employee1;

EmployeeID FirstName	LastName	Email	address	DateOfBirth	HireDate	salary	IsActive	contact_number
1 Alice 2 Bob		alice.johnson@example.com bob.smith@example.com		1985-02-14 1990-06-25			: -	NULL NULL

Rename the table

RENAME 'table name' to 'new_table_name;

RENAME table 'employee1' to 'staff';

Select * from staff;



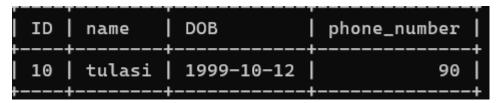
7. Write a MySQL script to drop a specific column from an existing table

ALTER TABLE table_name Drop column column_to_drop;

Create table Student (ID int auto_increment primary key,name varchar(50), DOB date, phone_number int);

Insert into Student (ID,name,DOB,phone_number) values (10,'tulasi',25-10-2003,9008712809);

Select * from Student;



Alter table Student Drop column DOB;

Select * from Student;



8. Write a MySQL script to demonstrate basic SQL queries such as SELECT, DISTINCT, WHERE

CREATE TABLE Product(productID varchar(50) primary key, product_name varchar(50) not null, price decimal (8,2) default 1.00 not null, quantity int default);

INSERT INTO product (productID, product_name, price, quantity) VALUES ('10', 'smart watch', 3000, 3), ('20', 'tab',5000,5), ('30', 'keyboard', 2000, 9), ('40', 'pendrive', 2000, 4);

Select * from product;

productID	 product_name	+	++ quantity
10	smart watch	3000.00	3
20	tab	5000.00	5
30	keyboard	2000.00	9
40	endrive	2000.00	4

Select * from product where productID = '20';

	produce_name	brice	quantity
20	tab	5000.00	5

Select DISTINCT product_name from product;



9. Write a MySQL script to demonstrate set operations like UNION, INTERSECTION, & EXCEPT

CREATE TABLE emp4(empID int, name varchar(50), department varchar(50), email varchar(50));

CREATE TABLE emp5(empID int, name varchar(50), department varchar(50), email varchar(50));

INSERT INTO emp4(empID, name, department, email) values ('1','supri','HR','supi@gmail.com'),('2','lakshmi','CS','lak@gmail.com'),('3','rakshitha','FINANCE','rak@gmail.com'),('4','chintu','pruchase','chin@gmail.com');

INSERT INTO emp5(empID, name, department, email) values ('1','nithin','IT','nithi@gmail.com'),('2','tulasi','pharama','tula@gmail.com'), ('3','ranjitha','HR','ranj@gmail.com'),('4','lokesh','R&D','loki@gmail.com');

Select * from emp4;

+ empID +	 name 	department	 email
2 3			supi@email.com lak@email.com rak@email.com chin@email.com

Select * from emp5;

empID	name	department	email
2 3	tulasi ranjitha		nithi@email.com tula@email.com ranj@email.com loki@email.com

Select * from emp4 union select * from emp5;

empID	name	department	email
1	supri	HR	supi@email.com lak@email.com rak@email.com chin@email.com nithi@email.com tula@email.com loki@email.com
2	lakshmi	CS	
3	rakshitha	FINANCE	
4	chintu	pruchase	
1	nithin	IT	
2	tulasi	pharama	
3	ranjitha	HR	
4	lokesh	R&D	

Select * from emp4 union all select * from emp5;

empID	name	department	email
3 4 1 2	lakshmi	HR CS FINANCE pruchase IT pharama HR R&D	supi@email.com lak@email.com rak@email.com chin@email.com nithi@email.com tula@email.com ranj@email.com loki@email.com

Select emp4.emp1D,emp4.name,emp4.department,emp4.email from emp4 emp4 inner join emp4 emp5 on emp4.emp1D = emp5.emp1D and emp4.name = emp5.name and emp4.department = emp5.department and emp4.email = emp5.email;

empID	name	department	 email
2	supri	HR	supi@email.com
	lakshmi	CS	lak@email.com
	rakshitha	FINANCE	rak@email.com
	chintu	pruchase	chin@email.com

Select empID,name,department,email from emp4 where not exists (select 4 from emp5 where emp4.empID = emp5.empID and emp4.name = emp5.name and emp4.department = emp5.department and emp4.email = emp5.email);

+ empID +	 name	department	
1	supri	HR	supi@email.com
2	lakshmi	CS	lak@email.com
3	rakshitha	FINANCE	rak@email.com
4	chintu	pruchase	chin@email.com

10. Write a MySQL script to demonstrate the use of nested queries (e.g., subqueries in SELECT, WHERE, and FROM).

CREATE TABLE employees2 (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(50), department VARCHAR(50), salary DECIMAL(10, 2), hire_date DATE);

INSERT INTO employees2(name, department, salary, hire_date) VALUES ('rekha', 'HR', 50000.00, '2020-01-10'), ('revathi', 'Finance', 60000.00, '2019-03-15'), ('arun', 'IT', 75000.00, '2021-07-20'), ('lokesh', 'HR', 52000.00, '2020-12-05'), ('bhoomi', 'Finance', 62000.00, '2018-11-30'), ('rakshu', 'IT', 80000.00, '2022-05-10');

Select * from employees2;

id	name	department	salary	hire_date
1 2 3 4 5	rekha revathi arun lokesh bhoomi rakshu	HR Finance IT HR Finance IT	50000.00 60000.00 75000.00 52000.00 62000.00 80000.00	2020-01-10 2019-03-15 2021-07-20 2020-12-05 2018-11-30 2022-05-10

SELECT name, department, salary, (SELECT AVG(salary) FROM employees) AS average_salary FROM employees;

name	department	salary	average_salary
rekha revathi arun lokesh bhoomi rakshu	HR Finance IT HR Finance IT	50000.00 60000.00 75000.00 52000.00 62000.00 80000.00	NULL NULL NULL NULL NULL

SELECT name, department, salary FROM employees WHERE salary > (SELECT AVG(salary) FROM employees);

name	department	salary
arun	IT	75000.00
rakshu	IT	80000.00

SELECT e.name, e.department, e.salary, avg_dept.avg_salary AS department_avg_salary FROM employees e JOIN (SELECT department, AVG(salary) AS avg_salary FROM employees GROUP BY department) avg_dept ON e.department = avg_dept.department;

+ name	department	salary	++ department_avg_salary
rekha	HR	50000.00	51000.000000
revathi	Finance	60000.00	61000.0000000
arun	IT	75000.00	77500.000000
lokesh	HR	52000.00	51000.000000
bhoomi	Finance	62000.00	61000.0000000
rakshu	IT	80000.00	77500.000000

SELECT name, department FROM employees e1 WHERE EXISTS (SELECT 1 FROM employees e2 WHERE e1.department = e2.department GROUP BY e2.department HAVING COUNT(*) > 1);



11. Write a MySQL script that demonstrates the use of the EXISTS function to test the existence of rows in subqueries.

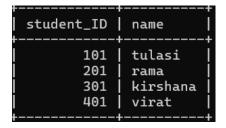
CREATE TABLE students(student_ID int primary key, name varchar(100));

CREATE TABLE grades(grade_ID int primary key,student_ID int, subject varchar(50),score int, foreign key(student_ID)references students (student_ID));

INSERT INTO students(student_ID,name) values (101,'tulasi'), (201,'rama'), (301,'kirshana'),(401,'virat');

INSERT INTO grades(grade_id, student_id, subjecT, score) values(1,101, 'COA', 75), (2,201,'DBMS',71), (3,301,'SE',73), (4,401,'DS',60);

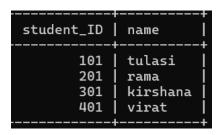
Select * from students;



Select * from grades;

grade_ID	student_ID	subject	score
1 2 3	201 301	COA DBMS SE	75 71 73
4 	401 	DS +	60 ++

Select name from students S where exists (select 1 from grades g where score<80);

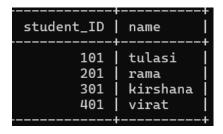


12. Write a MySQL program to handle NULL values, including filtering for NULL in queries

CREATE TABLE students1 (ID INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(100), age INT, grade varchar(50));

INSERT INTO students1 (ID, name, age, grade) VALUES (10,'Suprith', 20,'A'), (20,'lokesh', 21,NULL),(30,'arun', 22,'B'),(40,'chintu',NULL,'C'),(50,'rama', 22,'B');

Select * from students1;



Select * from grades;

grade_ID	student_ID	subject	score
1 2 3 4	201 301	COA DBMS SE DS	75 71 73 60

Select * from students1 where grade is NULL;

	name		_
20	lokesh	21	NULL

Select * from students1 where age is NOT NULL;

ID	name	 age +	grade
10 20 30 50	Suprith lokesh arun rama	20 21 22 22	A NULL B

SELECT ID, name, age, COALESCE(grade, 'no grade') AS grade FROM students1;

ID	name	age	grade	1
10	Suprith	20	A	.
20	lokesh	21	no grade	
30	arun	22	B	
40	chintu	NULL	C	
50	rama	22	B	

Update students1 Set grade = 'F' Where grade is NULL;

ID	name	age	grade
10	Suprith	20	A
20	lokesh	21	F
30	arun	22	B
40	chintu	NULL	C
50	rama	22	B
+	+	+	++

13. Write a MySQL script to demonstrate the use of aggregate functions like COUNT, SUM, AVG, MIN, and MAX.

CREATE TABLE Sales (id INT AUTO_INCREMENT PRIMARY KEY, product_name VARCHAR(50), category VARCHAR(50),quantity_sold INT, price DECIMAL(10,2),sale_date DATE);

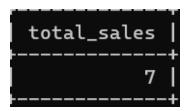
INSERT INTO Sales (product_name, category, quantity_sold, price, sale_date) VALUES ('Laptop', 'Electronics', 5, 1200.00, '2024-03-01'), ('Mouse', 'Electronics', 10, 25.00, '2024-03-02'), ('Keyboard', 'Electronics', 8, 50.00, '2024-03-03'), ('Chair', 'Furniture', 3, 150.00, '2024-03-04'), ('Table', 'Furniture', 2, 300.00, '2024-03-05'), ('Headphones', 'Electronics', 6, 80.00, '2024-03-06'), ('Monitor', 'Electronics', 4, 200.00, '2024-03-07');

Select * from Sales;

1 Laptop Electronics 5 1200.00 2024-03-01 2 Mouse Electronics 10 25.00 2024-03-02 3 Keyboard Electronics 8 50.00 2024-03-03 4 Chair Furniture 3 150.00 2024-03-04	id	product_name	category	quantity_sold	price	
5 Table	4 5 6	Mouse Keyboard Chair Table Headphones	Electronics Electronics Furniture Furniture Electronics	10 8 3 2 6	25.00 50.00 150.00 300.00 80.00	2024-03-02 2024-03-03 2024-03-04 2024-03-05 2024-03-06

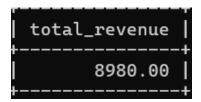
COUNT: Count total number of sales transactions

SELECT COUNT(*) AS total_sales FROM Sales;



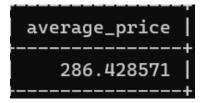
SUM: Calculate total revenue

SELECT SUM(quantity_sold * price) AS total_revenue FROM Sales;



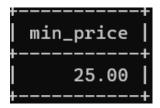
AVG: Calculate the average price of products sold

SELECT AVG(price) AS average_price FROM Sales;



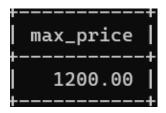
MIN: Find the minimum price of a product

SELECT MIN(price) AS min_price FROM Sales;



MAX: Find the maximum price of a product

SELECT MAX(price) AS max_price FROM Sales;



Aggregate functions with GROUP BY (Total quantity sold per category)

SELECT category, SUM(quantity_sold) AS total_quantity_sold FROM Sales GROUP BY category;

category	 total_quantity_sold
Electronics	33
Furniture	5

14. Write a MySQL script to demonstrate the use of GROUP BY and HAVING for grouping and filtering query results.

CREATE TABLE Sales(id INT AUTO_INCREMENT PRIMARY KEY, salesperson VARCHAR(50), region VARCHAR(50), amount DECIMAL(10, 2), sale_date DATE);

INSERT INTO Sales (salesperson, region, amount, sale_date) VALUES ('Charan', 'North', 500.00, '2024-03-01'), ('Rama', 'South', 300.00, '2024-03-02'), ('Sita', 'North', 700.00, '2024-03-05'), ('Lakshmana', 'East', 450.00, '2024-03-07'),

Select * from Sales1;

id	salesperson	region	amount	sale_date
2 3	Charan Rama Sita Lakshmana	South	300.00 700.00	2024-03-01 2024-03-02 2024-03-05 2024-03-07

SELECT salesperson, region, SUM(amount) AS total_sales FROM Sales GROUP BY salesperson, region HAVING total_sales < 1000;

salesperson	region	total_sales
Charan	North	500.00
Rama	South	300.00
Sita	North	700.00
Lakshmana	East	450.00

15. Write a MySQL script to sort query results using the ORDER BY clause and perform basic arithmetic operations within queries.

CREATE TABLE Products (id INT AUTO_INCREMENT PRIMARY KEY,product_name VARCHAR(100), price DECIMAL(10, 2),quantity INT);

INSERT INTO Products (product_name, price, quantity) VALUES ('Laptop', 800.00, 5), ('Smartphone', 500.00, 10), ('Tablet', 300.00, 8), ('Headphones', 100.00, 15), ('Smartwatch', 250.00, 7);

Select * from products;

id	product_name	price	quantity
	Laptop	800.00	5
	Smartphone	500.00	10
	Tablet	300.00	8
	Headphones	100.00	15
	Smartwatch	250.00	7

SELECT product_name, price, quantity, (price * quantity) AS total_value FROM Products ORDER BY total_value DESC;

+			+ -
product_name +	price	quantity	total_value
Smartphone Laptop Tablet Smartwatch Headphones	500.00 800.00 300.00 250.00	10 5 8 7 15	5000.00 4000.00 2400.00 1750.00 1500.00
+	+		tt,

16. Write a MongoDB script to create a collection and insert documents with various fields, including nested fields.

use CompanyDB:

```
use CompanyDB;
db.Employees.insertMany([
EmpID: 1,
Name: "Athi",
Department: "HR",
Salary: 50000,
Address: {
Street: "123 MG Street", City: "Bangalore", State: "KA", Pin: "560001" },
Projects: [ { ProjectName: "Recruitment Drive", Duration: "3 months" },
{ ProjectName: "Employee Engagement", Duration: "6 months" } ]},
EmpID: 2,
Name: "Bala Swaminathan",
Department: "IT",
Salary: 60000,
Address: { Street: "456 Anand nagar", City: "Bangalore", State: "KA", Pin:
"560100" },
Projects: [ { ProjectName: "Network Security", Duration: "12 months" },
{ ProjectName: "Cloud Migration", Duration: "8 months" } ] },
EmpID: 3,
Name: "Chaithra Kumar",
Department: "Finance",
Salary: 55000,
```

'2': ObjectId('67db889d3d23bbac5db7123b')

db.Employees.find().pretty();

```
_id: ObjectId('67db889d3d23bbac5db71239'),
EmpID: 1,
Name: 'Athi',
 Department: 'HR',
 Salary: 50000,
Address: {
    Street: '123 MG Street',
City: 'Bangalore',
State: 'KA',
Pin: '560001'
Projects: [
    { ProjectName: 'Recruitment Drive', Duration: '3 months' },
    { ProjectName: 'Employee Engagement', Duration: '6 months' }
_id: ObjectId('67db889d3d23bbac5db7123a'),
EmpID: 2,
Name: 'Bala Swaminathan',
Department: 'IT',
Salary: 60000,
Address: {
    Street: '856 }
    dress: {
Street: '456 Anand nagar',
City: 'Bangalore',
State: 'KA',
Pin: '560100'
 Projects: [
     { ProjectName: 'Network Security', Duration: '12 months' },
{ ProjectName: 'Cloud Migration', Duration: '8 months' }
  _id: ObjectId('67db889d3d23bbac5db7123b'),
EmpID: 3,
Name: 'Chaithra Kumar',
  Department: 'Finance',
Salary: 55000,
  Address: {
   Street: '10 Gandhi street',
     City: 'Bangalore',
State: 'KA',
Pin: '60604'
  Projects: [
      { ProjectName: 'Budget Analysis', Duration: '4 months' }, 
{ ProjectName: 'Audit Preparation', Duration: '5 months' }
```

17. Write a MongoDB script to demonstrate CREATE, READ, UPDATE, and DELETE operations on a collection.

```
use CompanyDB;
 db.employees.insertMany([
 EmpID: 1,
 Name: "Athi",
 Department: "HR",
 Salary: 50000,
 Address: {
 Street: "123 MG Street", City: "Bangalore", State: "KA", Pin: "560001" },
 Projects: [ { ProjectName: "Recruitment Drive", Duration: "3 months" },
  { ProjectName: "Employee Engagement", Duration: "6 months" } ]},
  {
 EmpID: 2,
 Name: "Bala Swaminathan",
 Department: "IT",
 Salary: 60000,
 Address: { Street: "456 Anand nagar", City: "Bangalore", State: "KA", Pin:
  "560100" },
 Projects: [ { ProjectName: "Network Security", Duration: "12 months" },
  { ProjectName: "Cloud Migration", Duration: "8 months" } ] },
 EmpID: 3,
 Name: "Chaithra Kumar",
 Department: "Finance",
```

```
Salary: 55000,
Address: { Street: "10 Gandhi street", City: "Bangalore", State: "KA", Pin:
"60604"},
Projects: [ { ProjectName: "Budget Analysis", Duration: "4 months" }, {
ProjectName: "Audit Preparation", Duration: "5 months" } ] }
]);

{
    acknowledged: true,
    insertedIds: {
        '0': ObjectId('67db8d6c3d23bbac5db7123f'),
        '1': ObjectId('67db8d6c3d23bbac5db71240'),
        '2': ObjectId('67db8d6c3d23bbac5db71241')
     }
}
```

db.employees.find().pretty();

Retrieve employees in the "IT" department db.employees.find({ Department: "IT" }).pretty();

```
[
{
    _id: ObjectId('67db8d6c3d23bbac5db71240'),
    EmpID: 2,
    Name: 'Bala Swaminathan',
    Department: 'IT',
    Salary: 60000,
    Address: {
        Street: '456 Anand nagar',
        City: 'Bangalore',
        State: 'KA',
        Pin: '560100'
    },
    Projects: [
        { ProjectName: 'Network Security', Duration: '12 months' },
        { ProjectName: 'Cloud Migration', Duration: '8 months' }
}
```

Increase salary by 10% for employees in the "Finance" department db.employees.updateMany({ Department: "Finance" },

{ \$mul: { Salary: 1.10 } });

```
{
   acknowledged: true,
   insertedId: null,
   matchedCount: 1,
   modifiedCount: 0
}
```

UPDATE: Change the city of "Athi to "Hyderabad" db.employees.updateOne(

```
{ Name: "Athi" },
{ $set: { "Address.city": "Hyderabad" } } );
```

```
{
   acknowledged: true,
   insertedId: null,
   matchedCount: 1,
   modifiedCount: 1,
   upsertedCount: 0
}
```

DELETE: Remove an employee with EmpID = 3 db.employees.deleteOne({ EmpID: 3});

```
{ acknowledged: true, deletedCount: 1 }
```

DELETE: Remove all employees in the "Finance" department db.employees.deleteMany({ Department: "Finance" });

```
{ acknowledged: true, deletedCount: 1 }
```

db.employees.find().pretty();

18. Write a MongoDB script to create indexes on a collection and demonstrate queries that benefit from these indexes.

```
use companyDB;
CREATE:
                                                         "employees"
                                                                         collection
              Insert
                       sample
                                  data
                                          into
                                                  the
db.employees.insertMany([
{ EmpID: 1, Name: "Athi", Department: "HR", Salary: 50000,
Address: { City: "Bangalore", Country: "India" } },
{ EmpID: 2, Name: "Arun", Department: "IT", Salary: 60000,
Address: { City: "Mumbai", Country: "India" } },
{ EmpID: 3, Name: "Bala", Department: "Finance", Salary: 55000,
Address: { City: "Delhi", Country: "India" } },
{ EmpID: 4, Name: "Kavi", Department: "Marketing", Salary: 70000,
Address: { City: "Chennai", Country: "India" } },
{ EmpID: 5, Name: "Shobi", Department: "Sales", Salary: 65000,
Address: { City: "Kolkata", Country: "India" } });
   acknowledged: true,
   insertedIds: {
      '0': ObjectId('67db90783d23bbac5db71242'),
      '1': ObjectId('67db90783d23bbac5db71243'),
'2': ObjectId('67db90783d23bbac5db71244'),
'3': ObjectId('67db90783d23bbac5db71245'),
      '4': ObjectId('67db90783d23bbac5db71246')
```

CREATE INDEX: Single-field index on "Department" db.employees.createIndex({ Department: 1 });

Department_1

CREATE INDEX: Compound index on "Salary" and "Department" db.employees.createIndex({ Salary: -1, Department: 1 });

```
Salary_-1_Department_1
```

CREATE INDEX: Text index on "Name" for text search db.employees.createIndex({ Name: "text" });

Name_text

CREATE INDEX: Index on nested field "Address.City" db.employees.createIndex({ "Address.City": 1 });

Address.City_1

Demonstrating Queries that Utilize Indexes

Query using single-field index (Department)

db.employees.find({ Department: "IT" }).explain("executionStats");

```
{
    explainVersion: '1',
    queryPlanner: {
        namespace: 'companyDB.employees',
        parsedQuery: { Department: { '$eq': 'IT' } },
        indexFilterSet: false,
        queryHash: '9E4738E0',
        planCacheShapeHash: '9E4738E0',
        planCacheKey: 'A85D6A10',
        optimizationTimeMillis: 5,
        maxIndexedOrSolutionsReached: false,
        maxIndexedAndSolutionsReached: false,
        maxScansToExplodeReached: false,
        prunedSimilarIndexes: false,
        winningPlan: {
        isCached: false,
        stage: 'FETCH',
        inputStage: {
            stage: 'IXSCAN',
            keyPattern: { Department: 1 },
            indexName: 'Department: 1',
            isMultiKey: false,
            multiKeyPaths: { Department: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: { Department: [ '["IT", "IT"]' ] }
    }
},
rejectedPlans: []
},
```

```
executionStats: {
    executionSuccess: true,
   nReturned: 1,
executionTimeMillis: 8,
    totalKeysExamined: 1,
   totalDocsExamined: 1,
executionStages: {
       isCached: false,
stage: 'FETCH',
nReturned: 1,
executionTimeMillisEstimate: 0,
       works: 2,
advanced: 1,
needTime: 0,
needYield: 0,
saveState: 0,
restoreState: 0,
      restoreses
isEOF: 1,
docsExamined: 1,
alreadyHasObj: 0,
inputStage: {
   stage: 'IXSCAN',
            nReturned: 1, executionTimeMillisEstimate: 0,
           works: 2,
advanced: 1,
needTime: 0,
needYield: 0,
            saveState: 0, restoreState: 0,
            isEOF: 1,
           keyPattern: { Department: 1 },
indexName: 'Department_1',
isMultiKey: false,
miltikeyPaths: { Department: [] },
            isUnique: false,
isSparse: false,
isPartial: false,
            indexVersion: 2,
direction: 'forward'
            direction: 'forward',
indexBounds: { Department: [ '["IT", "IT"]' ] },
            keysExamined: 1,
            seeks: 1,
            dupsTested: 0,
dupsDropped: 0
```

```
queryShapeHash: 'CF839106DF665F7BB5664F57FC0866320E6850C177265E1943DACD1558E63E9F',
command: {
    find: 'employees',
        filter: { Department: 'IT' },
        'sdb: 'companyDB'
},
serverInfo: {
    host: 'samsung_galaxy',
    port: 27017,
    version: '8.0.5',
        gitVersion: 'cb9e2e5e552ee39dea1e39d7859336456d0c9820'
},
serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalQueryMaxBlockingSortMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryBytes: 104857600,
    internalQueryProhibitBlockingMergeOnMongoS: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalQueryFameworkControl: 'trySbeRestricted',
    internalQueryPrameworkControl: 'trySbeRestricted',
    internalQueryPlannerIgnoreIndexWithCollationForRegex: 1
},
ok: 1
```

Query using compound index (Salary in descending order) db.employees.find().sort({ Salary: -1 }).explain("executionStats");

```
{
    explainVersion: '1',
    queryPlanner: {
        namespace: 'companyDB.employees',
        parsedQuery: {},
        indexFilterSet: false,
        queryHash: 'A2E003D3',
        planCacheShapeHash: 'A2E0D3D3',
        planCacheKey: '4100042F',
        optimizationTimeMillis: 1,
        maxIndexedOrSolutionsReached: false,
        maxIndexedAndSolutionsReached: false,
        maxIndexedAndSolutionsReached: false,
        maxIndexedAndSolutionsReached: false,
        prunedSimilarIndexes: false,
        prunedSimilarIndexes: false,
        isCached: false,
        stage: 'FETCH',
        inputStage: {
            stage: 'IXSCAN',
            keyPattern: { Salary: -1, Department: 1 },
            indexName: 'Salary--1_Department_1',
            isMultikey: false,
            indexName: 'Salary: [], Department: [] },
        isUnique: false,
        isSparse: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: {
            Salary: ['[MaxKey, MinKey]'],
            Department: ['[MinKey, MaxKey]']
        }
    }
    rejectedPlans: []
},
rejectedPlans: []
},
```

```
executionStats: {
        executionSuccess: true,
       nReturned: 5,
executionTimeMillis: 3,
       totalKeysExamined: 5,
       totalDocsExamined: 5, executionStages: {
            isCached: false,
stage: 'FETCH',
            nReturned: 5,
            executionTimeMillisEstimate: 0,
           works: 6, advanced: 5,
           needTime: 0,
needYield: 0,
            saveState: 0,
            restoreState: 0,
           isEOF: 1,
docsExamined: 5,
            alreadyHasObj: 0,
inputStage: {
                 stage: 'IXSCAN',
                nReturned: 5,
                 executionTimeMillisEstimate: 0,
                works: 6, advanced: 5,
                needTime: 0,
needYield: 0,
saveState: 0,
restoreState: 0,
                 isEOF: 1,
                keyPattern: { Salary: -1, Department: 1 },
indexName: 'Salary_-1_Department_1',
isMultiKey: false,
                multiKeyPaths: { Salary: [], Department: [] },
                isUnique: false,
isSparse: false,
isPartial: false,
                 indexVersion: 2,
                direction: 'forward',
                indexBounds: {
   Salary: [ '[MaxKey, MinKey]' ],
   Department: [ '[MinKey, MaxKey]' ]
                keysExamined: 5,
                seeks: 1,
dupsTested: 0,
dupsDropped: 0
            }
},
serverInfo: {
host: 'samsung_galaxy',
port: 27017,
version: '8.0.5',
aitVersion: 'cb9e2e5e552ee39deale39d7859336456d0c9820'
},
serverParameters: {
  internalQueryFacetBufferSizeBytes: 104857600,
  internalQueryFacetMaxOutputDocSizeBytes: 104857600,
  internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
  internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
  internalLookupStageCocumentMaxMamoryBytes: 104857600
    InternalLookupstageIntermediateDocumentMaxSizeBytes: 104857600, internalDocumentSourceGroupMaxMemoryBytes: 104857600, internalQueryMaxBlockingSortMemoryUsageBytes: 104857600, internalQueryProhibitBlockingMergeOnMongoS: 0, internalQueryMaxAddToSetBytes: 104857600, internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600, internalQueryFrameworkControl: 'trySbeRestricted', internalQueryPlannerIgnoreIndexWithCollationForRegex: 1
```

Query using text index (Search employees with "Athi" in Name) db.employees.find({ \$text: { \$search: "Athi" } }).explain("executionStats");

```
explainVersion: '1',
queryPlanner: {
  namespace: 'companyDB.employees',
  parsedQuery: {
      '$text': {
        '$search': 'Athi',
        '$language': 'english',
        '$caseSensitive': false,
'$diacriticSensitive': false
     }
   indexFilterSet: false,
   queryHash: 'CF6F4CEE
  queryHash: 'CF6F4CEE',
planCacheShapeHash: 'CF6F4CEE',
planCacheKey: '08852285',
optimizationTimeMillis: 5,
  maxIndexedOrSolutionsReached: false,
  maxIndexedAndSolutionsReached: false,
  maxScansToExplodeReached: false,
   prunedSimilarIndexes: false,
  winningPlan: {
  isCached: false,
  stage: 'TEXT_MATCH',
     indexPrefix: {},
     indexName: 'Name_text',
     parsedTextQuery: {
        terms: [ 'athi' ],
        negatedTerms: [],
        phrases: [],
        negatedPhrases: []
     textIndexVersion: 3,
     inputStage: {
        stage: 'FETCH'
        inputStage: {
           stage: 'IXSCAN',
keyPattern: { _fts: 'text', _ftsx: 1 },
indexName: 'Name_text',
isMultikey: false,
           isUnique: false,
isSparse: false,
isPartial: false,
           indexVersion: 2,
direction: 'backward',
           indexBounds: {}
```

Query using index on nested field (Find employees in "Delhi") db.employees.find({"Address.City":"Delhi"}).explain("executionStats");

```
executionStats: {
    executionSuccess: true,
    nReturned: 1,
    executionTimeMillis: 0,
    totalKeysExamined: 1,
    executionStages: {
        isCached: false,
        stage: 'FETCH',
             raction. Tatter,
stage: 'FETCH',
nReturned: 1,
executionTimeMillisEstimate: 0,
            works: 2,
advanced: 1,
needTime: 0,
needYield: 0,
saveState: 0,
restoreState: 0,
isFOE: 1
             restorestate: 0,
iscOF: 1,
docsExamined: 1,
alreadyHasObj: 0,
inputStage: {
stage: 'IXSCAN',
nReturned: 1,
executionTimeMillisEstimate: 0,
works: 2
                 executionTimeMillisEstimate: 0,
works: 2,
advanced: 1,
needTime: 0,
needYield: 0,
saveState: 0,
restoreState: 0,
isEOF: 1,
keyPattern: { 'Address.City': 1 },
indexName: 'Address.City_1',
isMultiKey: false,
multiKeyPaths: { 'Address.City': [] },
isUnique: false,
isSparse: false,
isSparse: false,
indexVersion: 2,
direction: 'forward',
indexBounds: { 'Address.City': [ '["Delhi", "Delhi"]' ] },
keysExamined: 1,
seeks: 1,
dupsTested: 0,
 explainVersion: '1',
queryPlanner: {
  namespace: 'companyDB.employees',
  parsedQuery: { 'Address.City': { '$eq': 'Delhi' } },
  indexFilterSet: false,
  queryHash: 'D&F1D7C4'.
        queryHash: 'D6F1D7C4',
planCacheShapeHash: 'D6F1D7C4',
         planCacheKey: '1FA95736', optimizationTimeMillis: 0,
         maxIndexedOrSolutionsReached: false, maxIndexedAndSolutionsReached: false,
          maxScansToExplodeReached: false,
         prunedSimilarIndexes: false,
winningPlan: {
  isCached: false,
               stage: 'FETCH'
inputStage: {
                    nputstage: {
stage: 'IXSCAN',
keyPattern: { 'Address.City': 1 },
indexName: 'Address.City_1',
isMultiKey: false,
multiKeyPaths: { 'Address.City': [] },
                    isUnique: false,
isSparse: false,
isPartial: false,
                      indexVersion: 2,
                     direction: 'forward',
indexBounds: { 'Address.City': [ '["Delhi", "Delhi"]' ] }
          rejectedPlans: []
```

```
}
}
},
queryShapeHash: '258CF6E0C54B5BE4FB74BF93C2F49E01EA76430C3EC835F799E49694F9073014',
command: {
    find: 'employees',
        filter: { 'Address.City': 'Delhi' },
        '$db': 'companyDB'
},
serverInfo: {
    host: 'samsung_galaxy',
    port: 27017,
    version: '8.0.5',
    gitVersion: 'cb9e2e5e552ee39deale39d7859336456d0c9820'
},
serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
    internalQueryMaxAddToSetBytes: 104857600,
    internalQueryMaxAddToSetBytes: 104857600,
    internalQueryMaxAddToSetBytes: 104857600,
    internalQueryFameworkControl: 'trySbeRestricted',
    internalQueryPlannerIgnoreIndexWithCollationForRegex: 1
},
ok: 1
}
```

List all indexes on the "employees" collection

db.employees.getIndexes();

19. Write a MongoDB script to demonstrate the use of the aggregation pipeline, including \$group, \$match, and \$sum.

```
db.sales.insertMany([
{ "product": "Laptop", "category": "Electronics", "price": 1200, "quantity": 5,
"date": new Date("2024-03-01") },
{ "product": "Mouse", "category": "Electronics", "price": 50, "quantity": 20,
"date": new Date("2024-03-02") },
{ "product": "Keyboard", "category": "Electronics", "price": 80, "quantity": 15,
"date": new Date("2024-03-02") },
{ "product": "Desk", "category": "Furniture", "price": 300, "quantity": 10, "date":
new Date("2024-03-03") },
{ "product": "Chair", "category": "Furniture", "price": 150, "quantity": 8, "date":
new Date("2024-03-03") }]);
   acknowledged: true,
   insertedIds: {
     '0': ObjectId('67db96243d23bbac5db71247').
     '1': ObjectId('67db96243d23bbac5db71248'
     '2': ObjectId('67db96243d23bbac5db71249'),
     '3': ObjectId('67db96243d23bbac5db7124a'),
     '4': ObjectId('67db96243d23bbac5db7124b')
db.sales.aggregate([ { $match: { category: "Electronics" } },
{ $group: { _id: "$category", totalRevenue: { $sum: { $multiply: ["$price",
"$quantity"] } } } },
{ $sort: { totalRevenue: -1 } }]).forEach(printjson);
    _id: 'Electronics',
   totalRevenue: 8200
```

printjson ensures the results are structured properly

20. Write a MongoDB script to use change streams to monitor changes in a collection (equivalent to triggers in relational databases).

```
db.sales.insertMany([ {
    "product": "Laptop", "category": "Electronics", "price": 1200, "quantity": 5,
    "date": new Date("2024-03-01") },
    { "product": "Mouse", "category": "Electronics", "price": 50, "quantity": 20,
    "date": new Date("2024-03-02") },
    { "product": "Keyboard", "category": "Electronics", "price": 80, "quantity": 15,
    "date": new Date("2024-03-02") },
    { "product": "Desk", "category": "Furniture", "price": 300, "quantity": 10, "date":
    new Date("2024-03-03") },
    { "product": "Chair", "category": "Furniture", "price": 150, "quantity": 8, "date":
    new Date("2024-03-03") }]);
    {
```

```
{
   acknowledged: true,
   insertedIds: {
     '0': ObjectId('67db96883d23bbac5db7124c'),
     '1': ObjectId('67db96883d23bbac5db7124d'),
     '2': ObjectId('67db96883d23bbac5db7124e'),
     '3': ObjectId('67db96883d23bbac5db7124f'),
     '4': ObjectId('67db96883d23bbac5db71250')
}
```

db.sales.createIndex({ category: 1 });

category_1

db.sales.createIndex({ price: 1, quantity: -1 });

price_1_quantity_-1

db.sales.createIndex({ date: 1 });

date_1

print("Querying sales for Electronics category:");

Querying sales for Electronics category:

db.sales.find({ category: "Electronics" }).forEach(printjson);