

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	Alice	Johnson	alice.johnson@example.com	2000-05-15
2	Bob	Smith	bob.smith@example.com	1999-07-20

SELECT \* FROM Courses;

CourseID	CourseName	CourseCode
1	Mathematics	MATH101
2	Computer Science	CS101

SELECT\* FROM Enrollments;

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

2. Develop a MySQL script to create tables with data types like VARCHAR, 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	FirstName	LastName	Email	DateOfBirth	HireDate	Salary	IsActive
1	Alice	Johnson	alice.johnson@example.com	1985-02-14	2022-03-01	55000.00	1
2	Bob	Smith	bob.smith@example.com	1990-06-25	2023-01-15	60000.00	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
1	Website Redesign	2024-01-01	2024-06-30	1
2	Mobile App Development	2024-03-01	NULL	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',
'Female', 'Other') NOT NULL, HireDate TIMESTAMP DEFAULT
CURRENT_TIMESTAMP, IsActive BOOLEAN DEFAULT TRUE, CHECK
(Salary >= 0), CHECK (Gender IN ('Male', 'Female', 'Other')) ;
```

DESC Employee1;

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

```
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	FirstName	LastName	Email	PhoneNumber	Salary	Department	DateOfBirth	Gender	HireDate	IsActive
1	Alice	Brown	alice.brown@example.com	1234567890	45000.00	IT	1990-05-12	Female	2023-06-01 00:00:00	1
2	Bob	Smith	bob.smith@example.com	9876543210	35000.00	HR	1985-09-15	Male	2024-01-15 00:00:00	1

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;
```

ID	Name	Age
1	Alice	25
2	Bob	30
3	Charlie	35

```
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  
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',  
'Female', 'Other') NOT NULL, HireDate TIMESTAMP 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
1	Alice	Johnson	alice.johnson@example.com	NULL	1985-02-14	2022-03-01	55000.00	1	NULL	NULL
2	Bob	Smith	bob.smith@example.com	NULL	1990-06-25	2023-01-15	60000.00	1	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	FirstName	LastName	Email	address	DateOfBirth	HireDate	salary	IsActive	phone	phoneno
1	Alice	Johnson	alice.johnson@example.com	NULL	1985-02-14	2022-03-01	55000.00	1	NULL	NULL
2	Bob	Smith	bob.smith@example.com	NULL	1990-06-25	2023-01-15	60000.00	1	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	Johnson	alice.johnson@example.com	NULL	1985-02-14	2022-03-01	55000.00	1	NULL
2	Bob	Smith	bob.smith@example.com	NULL	1990-06-25	2023-01-15	60000.00	1	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	Johnson	alice.johnson@example.com	NULL	1985-02-14	2022-03-01	55000.00	1	NULL
2	Bob	Smith	bob.smith@example.com	NULL	1990-06-25	2023-01-15	60000.00	1	NULL

Rename the table

RENAME 'table\_name' to 'new\_table\_name';

RENAME table 'employee1' to 'staff';

Select \* from staff;

```
MySQL localhost:33060+ ssl schooldb SQL > select * from staff;
```

EmployeeID	FirstName	LastName	Email	address	DateOfBirth	HireDate	salary	IsActive	contact_number
1	Alice	Johnson	alice.johnson@example.com	NULL	1985-02-14	2022-03-01	55000.00	1	NULL
2	Bob	Smith	bob.smith@example.com	NULL	1990-06-25	2023-01-15	60000.00	1	NULL

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;
```

ID	name	DOB	phone_number
10	tulasi	1999-10-12	90

```
Alter table Student Drop column DOB;
```

```
Select * from Student;
```

ID	name	phone_number
10	tulasi	90

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	price	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';
```

productID	product_name	price	quantity
20	tab	5000.00	5

```
Select DISTINCT product_name from product;
```

product_name
smart watch
tab
keyboard
endrive

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','prurchase','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','lokes','R&D','loki@gmail.com');
```

```
Select * from emp4;
```

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

```
Select * from emp5;
```

empID	name	department	email
1	nithin	IT	nithi@email.com
2	tulasi	pharama	tula@email.com
3	ranjitha	HR	ranj@email.com
4	lokes	R&D	loki@email.com

```
Select * from emp4 union select * from emp5;
```

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

Select \* from emp4 union all select \* from emp5;

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

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

empID	name	department	email
1	supri	HR	supi@email.com
2	lakshmi	CS	lak@email.com
3	rakshitha	FINANCE	rak@email.com
4	chintu	prurchase	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	email
1	supri	HR	supi@email.com
2	lakshmi	CS	lak@email.com
3	rakshitha	FINANCE	rak@email.com
4	chintu	prurchase	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	rekha	HR	50000.00	2020-01-10
2	revathi	Finance	60000.00	2019-03-15
3	arun	IT	75000.00	2021-07-20
4	lokesh	HR	52000.00	2020-12-05
5	bhoomi	Finance	62000.00	2018-11-30
6	rakshu	IT	80000.00	2022-05-10

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

name	department	salary	average_salary
rekha	HR	50000.00	NULL
revathi	Finance	60000.00	NULL
arun	IT	75000.00	NULL
lokesh	HR	52000.00	NULL
bhoomi	Finance	62000.00	NULL
rakshu	IT	80000.00	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	61000.000000
arun	IT	75000.00	77500.000000
lokesh	HR	52000.00	51000.000000
bhoomi	Finance	62000.00	61000.000000
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 );
```

name	department
rekha	HR
revathi	Finance
arun	IT
lokesh	HR
bhoomi	Finance
rakshu	IT

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;
```

student_ID	name
101	tulasi
201	rama
301	kirshana
401	virat

```
Select * from grades;
```

grade_ID	student_ID	subject	score
1	101	COA	75
2	201	DBMS	71
3	301	SE	73
4	401	DS	60

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

student_ID	name
101	tulasi
201	rama
301	kirshana
401	virat



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,'lokes', 21,NULL),(30,'arun', 22,'B'),(40,'chintu',NULL,'C'),(50,'rama',  
22,'B');
```

Select \* from students1;

student_ID	name
101	tulasi
201	rama
301	kirshana
401	virat

Select \* from grades;

grade_ID	student_ID	subject	score
1	101	COA	75
2	201	DBMS	71
3	301	SE	73
4	401	DS	60

Select \* from students1 where grade is NULL;

ID	name	age	grade
20	lokes	21	NULL

Select \* from students1 where age is NOT NULL;

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

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

ID	name	age	grade
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;

id	product_name	category	quantity_sold	price	sale_date
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
5	Table	Furniture	2	300.00	2024-03-05
6	Headphones	Electronics	6	80.00	2024-03-06
7	Monitor	Electronics	4	200.00	2024-03-07

COUNT: Count total number of sales transactions

```
SELECT COUNT(*) AS total_sales FROM Sales;
```

total_sales
7

SUM: Calculate total revenue

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

total_revenue
8980.00

AVG: Calculate the average price of products sold

SELECT AVG(price) AS average\_price FROM Sales;

average_price
286.428571

MIN: Find the minimum price of a product

SELECT MIN(price) AS min\_price FROM Sales;

min_price
25.00

MAX: Find the maximum price of a product

SELECT MAX(price) AS max\_price FROM Sales;

max_price
1200.00

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
1	Charan	North	500.00	2024-03-01
2	Rama	South	300.00	2024-03-02
3	Sita	North	700.00	2024-03-05
4	Lakshmana	East	450.00	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
1	Laptop	800.00	5
2	Smartphone	500.00	10
3	Tablet	300.00	8
4	Headphones	100.00	15
5	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	500.00	10	5000.00
Laptop	800.00	5	4000.00
Tablet	300.00	8	2400.00
Smartwatch	250.00	7	1750.00
Headphones	100.00	15	1500.00

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

```
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('67db889d3d23bbac5db71239'),
    '1': ObjectId('67db889d3d23bbac5db7123a'),
    '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: '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();

```
[
  {
    _id: ObjectId('67db8d6c3d23bbac5db7123f'),
    EmpID: 1,
    Name: 'Aathi',
    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('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' }
    ]
  },
  {
    _id: ObjectId('67db8d6c3d23bbac5db71241'),
    EmpID: 3,
    Name: 'Chaithra Kumar',
    Department: 'Finance',
    Salary: 60500.000000000001,
    Address: {
      Street: '10 Gandhi street',
      City: 'Bangalore',
      State: 'KA',
      Pin: '60604'
    },
    Projects: [
      { ProjectName: 'Budget Analysis', Duration: '4 months' },
      { ProjectName: 'Audit Preparation', Duration: '5 months' }
    ]
  }
]
```

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: 1,
  upsertedCount: 0
}
```

UPDATE: Change the city of "Aathi" to "Hyderabad"

```
db.employees.updateOne(
{ Name: "Aathi" },
{ $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();
```

```
[
  {
    _id: ObjectId('67db8d6c3d23bbac5db7123f'),
    EmpID: 1,
    Name: 'Athi',
    Department: 'HR',
    Salary: 50000,
    Address: {
      Street: '123 MG Street',
      City: 'Bangalore',
      State: 'KA',
      Pin: '560001',
      city: 'Hyderabad'
    },
    Projects: [
      { ProjectName: 'Recruitment Drive', Duration: '3 months' },
      { ProjectName: 'Employee Engagement', Duration: '6 months' }
    ]
  },
  {
    _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' }
    ]
  }
]
```

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

use companyDB;

CREATE: Insert sample data into the "employees" collection  
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,
        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,
          multiKeyPaths: { Department: [ ] },
          isUnique: false,
          isSparse: false,
          isPartial: false,
          indexVersion: 2,
          direction: 'forward',
          indexBounds: { Department: [ '[ "IT", "IT" ] ' ] },
          keysExamined: 1,
          seeks: 1,
          dupsTested: 0,
          dupsDropped: 0
        }
      }
    }
  },
}

```

```

queryShapeHash: 'CF839106DF665F7BB5664F57FC0866320E6850C177265E1943DACD1558E63E9F',
command: {
  find: 'employees',
  filter: { Department: 'IT' },
  '$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,
  internalQueryProhibitBlockingMergeOnMongoS: 0,
  internalQueryMaxAddToSetBytes: 104857600,
  internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
  internalQueryFrameworkControl: '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: 'A2E0D3D3',
    planCacheShapeHash: 'A2E0D3D3',
    planCacheKey: '4100042F',
    optimizationTimeMillis: 1,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        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]' ]
        }
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionTimeMillis: 1,
    totalKeysExamined: 1,
    indexesExamined: 1,
    docsExamined: 1,
    docsReturned: 1,
    docsSkipped: 0,
    indexBoundsUsed: true,
    keyPatternMatch: true,
    queryHashMatch: true,
    planCacheHit: true,
    planCacheKey: '4100042F',
    planCacheShapeHash: 'A2E0D3D3',
    queryHash: 'A2E0D3D3',
    namespace: 'companyDB.employees',
    stage: 'FETCH',
    inputStage: {
      stage: 'IXSCAN',
      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]' ]
      }
    }
  }
}
```

```

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
    }
  }
}

```

```

},
queryShapeHash: 'E96E43A3626E0DFEB360931ACC40FCB75232C93C23FF1E16BA45BDC2C09A8933',
command: {
  find: 'employees',
  filter: {},
  sort: { Salary: -1 },
  '$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,
  internalQueryProhibitBlockingMergeOnMongoS: 0,
  internalQueryMaxAddToSetBytes: 104857600,
  internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
  internalQueryFrameworkControl: 'trySbeRestricted',
  internalQueryPlannerIgnoreIndexWithCollationForRegex: 1
},
ok: 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',
    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,
  totalDocsExamined: 1,
  executionStages: {
    isCached: false,
    stage: 'FETCH',
    nReturned: 1,
    executionTimeMillisEstimate: 0,
    works: 2,
    advanced: 1,
    needTime: 0,
    needYield: 0,
    saveState: 0,
    restoreState: 0,
    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: { '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"] ] },
      keysExamined: 1,
      seeks: 1,
      dupsTested: 0,
    }
  }
}
```

```
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'companyDB.employees',
    parsedQuery: { 'Address.City': { '$eq': 'Delhi' } },
    indexFilterSet: false,
    queryHash: 'D6F1D7C4',
    planCacheShapeHash: 'D6F1D7C4',
    planCacheKey: '1FA95736',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    prunedSimilarIndexes: false,
    winningPlan: {
      isCached: false,
      stage: 'FETCH',
      inputStage: {
        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,
    internalQueryProhibitBlockingMergeOnMongoS: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
    internalQueryFrameworkControl: 'trySbeRestricted',
    internalQueryPlannerIgnoreIndexWithCollationForRegex: 1
  },
  ok: 1
}

```

List all indexes on the "employees" collection

db.employees.getIndexes();

```

[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { Department: 1 }, name: 'Department_1' },
  {
    v: 2,
    key: { Salary: -1, Department: 1 },
    name: 'Salary_-1_Department_1'
  },
  {
    v: 2,
    key: { _fts: 'text', _ftsx: 1 },
    name: 'Name_text',
    weights: { Name: 1 },
    default_language: 'english',
    language_override: 'language',
    textIndexVersion: 3
  },
  { v: 2, key: { 'Address.City': 1 }, name: 'Address.City_1' }
]

```

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);
```

```
print("Querying sales sorted by price and quantity:");
db.sales.find().sort({ price: 1, quantity: -1 }).forEach(printjson);
db.sales.aggregate([ { $match: { category: "Electronics" } },
{ $group: { _id: "$category", totalRevenue: { $sum: { $multiply: ["$price",
"$quantity"] } } } }, { $sort: { totalRevenue: -1 } }]).forEach(printjson);
print("Monitoring changes in the sales collection...");
const changeStream = db.sales.watch();
changeStream.forEach(change => {
print("Detected change:");
printjson(change);
});
```