Assignment 3:

```
CREATE TABLE Dept (
  Dept_id INT PRIMARY KEY,
  Dept_name VARCHAR(50),
  Location VARCHAR(50)
);

CREATE TABLE Employee (
  Emp_id INT PRIMARY KEY,
  Dept_id INT,
  Emp_fname VARCHAR(50),
  Emp_lname VARCHAR(50),
  Emp_position VARCHAR(50),
  Emp_salary INT,
  Emp_JoinDate DATE
);

CREATE TABLE Project (
  Proj_id INT PRIMARY KEY,
  Dept_id INT,
  Proj_name VARCHAR(50),
  Proj_Location VARCHAR(50),
  Proj_Cost INT,
  Proj_year DATE
);

INSERT INTO Dept (Dept_id, Dept_name, Location) VALUES
(1, 'COMPUTER', 'New York'),
(2, 'IT', 'San Francisco'),
(3, 'Finance', 'Chicago'),

(4, 'Marketing', 'Los Angeles'),

(5, 'Sales', 'Seattle'),

(6, 'Legal', 'Boston'),

(7, 'R&D', 'Austin'),

(8, 'Operations', 'Denver'),

(9, 'Support', 'Atlanta'),

(10, 'Logistics', 'Houston');


INSERT INTO Employee (Emp_id, Dept_id, Emp_fname, Emp_lname, Emp_position,
Emp_salary, Emp_JoinDate) VALUES

(1, 1, 'John', 'Doe', 'HR Manager', 70000, '2020-05-10'),

(2, 2, 'Jane', 'Smith', 'Software Engineer', 85000, '2019-04-15'),

(3, 3, 'Bob', 'Johnson', 'Financial Analyst', 65000, '2021-01-20'),

(4, 5, 'Alice', 'Brown', 'Marketing Specialist', 60000, '2018-11-05'),

(5, 5, 'Tom', 'White', 'Sales Representative', 55000, '2020-03-12'),

(6, 6, 'Sam', 'Green', 'Legal Advisor', 80000, '2017-09-23'),

(7, 7, 'Chris', 'Black', 'Research Scientist', 95000, '2022-06-30'),

(8, 8, 'Jessica', 'Davis', 'Operations Manager', 90000, '2019-08-19'),

(9, 9, 'David', 'Martinez', 'Support Engineer', 60000, '2021-10-25'),

(10, 10, 'Laura', 'Garcia', 'Logistics Coordinator', 58000, '2020-02-14');


INSERT INTO Project (Proj_id, Dept_id, Proj_name, Proj_Location, Proj_Cost, Proj_year)
VALUES

(1, 1, 'Employee Engagement', 'New York', 200000, '2022-01-01'),

(2, 2, 'App Development', 'San Francisco', 500000, '2021-05-15'),

(3, 3, 'Annual Financial Report', 'Chicago', 150000, '2021-12-01'),

(4, 4, 'Ad Campaign', 'Los Angeles', 300000, '2022-06-10'),

(5, 5, 'Sales Strategy', 'Seattle', 250000, '2021-09-30'),

(6, 6, 'Compliance Audit', 'Boston', 100000, '2022-03-20'),
(7, 7, 'Product Research', 'Austin', 750000, '2023-02-01'),

(8, 8, 'Warehouse Operations', 'Denver', 400000, '2021-07-15'),

(9, 9, 'Customer Support Optimization', 'Atlanta', 350000, '2022-11-05'),

(10, 10, 'Supply Chain Overhaul', 'Houston', 600000, '2023-04-18');


-- Find Employee details and Department details using NATURAL JOIN.

select * from Employee e join Dept d on e.dept_id = d.dept_id;


-- Find the emp_fname,Emp_position,location,Emp_JoinDate who have same Dept id.

select e.Emp_fname, e.Emp_position, d.location, e.Emp_JoinDate from Employee e inner
join Dept d on e.dept_id = d.dept_id where d.dept_id in (select dept_id from Employee
group by dept_id having count(*) > 1);


-- Find the Employee details ,Proj_id,Project cost who does not have Project location as
'Hyderabad'.

select e.Emp_id, e.Emp_fname, e.Emp_lname, e.Emp_position, p.Proj_id, p.Proj_Cost from
Employee e inner join Project p on e.Dept_id = p.Dept_id where p.Proj_name !=
"Hyderabad";


-- Find Department Name ,employee name, Emp_position for which project year is 2021

select d.Dept_name, e.Emp_fname, e.Emp_position from Employee e inner join Dept d on
e.Dept_id = d.Dept_id where d.Dept_id in (select Dept_id from Project where
year(Proj_year) = 2021);


-- Display emp_position,D_name who have Project cost >300000

select e.Emp_position, d.Dept_name from Employee e inner join Dept d on e.Dept_id =
d.Dept_id where d.Dept_id in (select Dept_id from Project where Proj_Cost > 300000);


-- Find the names of all the Projects that started in the year 2015.

select * from Project where year(Proj_year) = 2022;


-- List the Dept_name having no_of_emp=2
```

```sql
select d.Dept_name from Employee e inner join Dept d on e.Dept_id = d.Dept_id group by
e.Dept_id having count(*) = 2;


-- Display the total number of employee who have joined any project before 2022

select count(*) as total_employee_before_2022 from Employee e inner join Project p on
e.Dept_id = p.Dept_id where year(p.Proj_year) < 2022;


-- Create a view showing the employee and Department details.

create view EmployeeDeptDetails as select e.Emp_id, e.Emp_fname, e.Emp_lname,
e.Emp_position, e.Emp_salary, e.Emp_JoinDate, e.Dept_id, d.Dept_name, d.Location from
Employee e inner join Dept d on e.Dept_id = d.Dept_id;

select * from EmployeeDeptDetails;


-- Perform Manipulation on simple view-Insert, update, delete, drop view.

CREATE VIEW SimpleEmployeeView AS

SELECT Emp_id, Emp_fname, Emp_lname, Emp_position, Emp_salary

FROM Employee;


INSERT INTO SimpleEmployeeView (Emp_id, Emp_fname, Emp_lname, Emp_position,
Emp_salary)

VALUES (11, 'Michael', 'Scott', 'Branch Manager', 90000);


DELETE FROM SimpleEmployeeView

WHERE Emp_id = 11;


DROP VIEW SimpleEmployeeView;
```

Assignment 4:

```
USE 31380_db;


CREATE TABLE borrower (
    roll_no INT,
    name VARCHAR(50),
    doi DATE,
    book_name VARCHAR(50),
    status VARCHAR(1) DEFAULT 'I'
);


CREATE TABLE fine (
    roll_no INT,
    date DATE,
    amt INT
);


INSERT INTO borrower VALUES
            (1,'deleniti','2012-07-03','enim',0),
            (2,'harum','1998-02-27','magnam',1),
            (3,'velit','1993-10-19','minus',0),
            (4,'ullam','2002-09-10','incidunt',0),
            (5,'totam','1992-01-23','tempora',0),
            (6,'eos','2023-03-03','dolor',0),
            (7,'ut','1979-10-08','nostrum',1),
            (8,'debitis','1981-08-09','quae',0),
            (9,'harum','2014-11-25','voluptate',1),
            (10,'doloremque','1988-11-30','quo',1)
;
```

```
DELIMITER $$

CREATE PROCEDURE library (IN roll INT, IN book VARCHAR(50), IN dt DATE)

BEGIN

    DECLARE fine INT;

    DECLARE dt2 INT;

    DECLARE EXIT HANDLER FOR 1452 SELECT 'Primary Key Not Found' ErrorMessage;

    -- 1452 is the error code for "ERROR 1452 (23000): Cannot add or update a child row: a
foreign key constraint fails"


    SELECT @idt := doi

        FROM borrower

        WHERE (roll_no = roll AND book_name = book);


    SELECT @stt := status

        FROM borrower

        WHERE (roll_no = roll AND book_name = book);


    -- DATEDIFF is an inbuilt function which returns the number of days between two dates.

    SET dt2 := DATEDIFF(dt, @idt);


            -- if book has not been returned then set fine

    IF @stt = False THEN

        IF dt2 BETWEEN 0 and 14 THEN

            SET fine := 0;


        ELSEIF dt2 BETWEEN 15 AND 30 THEN

            SET fine := dt2 * 5;


        ELSE
```

```
         SET fine := dt2 * 50;
      END IF;


      INSERT INTO fine VALUES (roll, dt, fine);


      UPDATE borrower
         SET status = True
         WHERE (roll_no = roll AND book_name = book);


         -- Book has been returned
          ELSE
                 SELECT "Book has already been returned" AS Message;
          END IF;


END $$
DELIMITER ;


-- Sample call
call library(5, "tempora", '2000-03-20');
```

Assignment 5:

```
use college;


create table stud_marks (
            name varchar(50),
    total_marks int
    );


create table result (
            roll int,
    name varchar(50),
    class varchar(100) );



delimiter //


create function if not exists classify (marks int) returns varchar(100)
deterministic
begin
            if marks between 990 and 1500 then
                        return 'Distinction';
            elseif marks between 900 and 989 then
                        return 'First Class';
            ELSEIF marks BETWEEN 825 AND 899 THEN
                        RETURN 'Higher Second Class';
            ELSE
                        RETURN 'Pass';
            END IF;
END //
```

```
delimiter //

create procedure proc_grade(in roll_no int, in stu_name varchar(50))
begin
	declare class varchar(100);
   declare s_marks int;

   select total_marks into s_marks from stud_marks where (stu_name = name);
   set class := classify(s_marks);

   insert into result values(roll_no, stu_name, class);
   select * from result;

end //

delimiter ;

insert into stud_marks
values
("Anuj",1200),
("jui",969);

call proc_grade(1,"jui");
call proc_grade(2,"Anuj");
```

Assignment 6:

```
USE 31380_db;

CREATE TABLE N_EmpID (
        emp_id INT,
        dept_id INT,
        emp_fname VARCHAR(20),
        emp_lname VARCHAR(20),
        emp_position VARCHAR(50),
        emp_salary INT,
        emp_jdate DATE,
        PRIMARY KEY (emp_id)
);

CREATE TABLE O_EmpID (
        emp_id INT,
        dept_id INT,
        emp_fname VARCHAR(20),
        emp_lname VARCHAR(20),
        emp_position VARCHAR(50),
        emp_salary INT,
        emp_jdate DATE,
        PRIMARY KEY (emp_id)
);

INSERT INTO O_EmpID VALUES
        (19,2,'molestiae','nostrum','quisquam',850067,'1990-02-16'),
        (23,2,'aut','quis','harum',313554,'1976-09-16'),
        (32,2,'et','et','architecto',942122,'1972-03-16'),
        (33,3,'voluptatem','at','rerum',731520,'1990-06-28'),
        (34,2,'fugit','harum','omnis',477917,'1985-08-06'),
        (36,1,'provident','voluptatem','at',145306,'1995-07-06'),
        (55,3,'suscipit','et','et',481782,'2011-05-10'),
        (59,1,'ea','temporibus','qui',861072,'1988-07-26'),
        (60,3,'fuga','quia','placeat',393128,'1988-10-12'),
        (63,1,'voluptatem','nihil','reiciendis',856496,'1986-10-27'),
        (67,3,'amet','expedita','similique',720147,'2000-11-24'),
        (69,1,'dolore','magni','eius',414471,'1972-11-21'),
        (70,2,'quos','veritatis','ex',938067,'1997-04-13'),
        (81,2,'sequi','aut','voluptatibus',120184,'1987-12-17'),
        (84,3,'ipsum','sunt','nemo',446727,'1977-03-24'),
        (91,1,'tempore','qui','quia',338584,'1992-07-24'),
        (110,1,'consequatur','ipsam','est',576805,'2001-04-22'),
        (111,2,'eveniet','assumenda','rerum',526825,'2006-04-23'),
        (123,3,'optio','esse','est',730407,'1989-06-18'),
        (138,2,'nihil','rerum','eum',311204,'2009-03-02');


INSERT INTO N_EmpID VALUES
        (19,2,'molestiae','nostrum','quisquam',850067,'1990-02-16'),
        (23,2,'aut','quis','harum',313554,'1976-09-16'),
        (32,2,'et','et','architecto',942122,'1972-03-16'),
        (33,3,'voluptatem','at','rerum',731520,'1990-06-28'),
        (34,2,'fugit','harum','omnis',477917,'1985-08-06'),
        (36,1,'provident','voluptatem','at',145306,'1995-07-06'),
        (55,3,'suscipit','et','et',481782,'2011-05-10'),
        (165,1,'molestiae','itaque','dolorem',948514,'1975-08-09'),
        (168,2,'id','et','ut',771819,'2003-08-17'),
	(172,1,'iure','omnis','ut',486159,'2006-09-16'),
	(173,1,'perferendis','omnis','veniam',845958,'1990-07-14'),
	(174,2,'alias','debitis','aut',679066,'2014-04-10'),
	(179,2,'doloremque','voluptate','voluptate',148700,'1989-03-13'),
	(184,1,'sed','error','reiciendis',993787,'1971-11-13'),
	(186,1,'voluptas','eum','et',940556,'1983-09-08'),
	(195,2,'nesciunt','non','autem',215182,'2011-04-22'),
	(198,1,'pariatur','autem','deserunt',974698,'1972-04-24');

DELIMITER $$
CREATE PROCEDURE mergeEMP ()
BEGIN
	DECLARE done INT DEFAULT 0;
	DECLARE eno INT;
	DECLARE old_cursor CURSOR FOR SELECT emp_id FROM O_EmpID;
	DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
	OPEN old_cursor;

	getEmpID: LOOP
		IF done = TRUE THEN
			LEAVE getEmpID;
		END IF;
		FETCH old_cursor INTO eno;

		IF NOT EXISTS (SELECT 1 FROM N_EmpID where emp_id = eno) THEN
			INSERT INTO N_EmpID
			SELECT * FROM O_EmpID
			WHERE O_EmpID.emp_id = eno;
		END IF;
```

```
        END LOOP;

        CLOSE old_cursor;
END $$
DELIMITER ;

CALL mergeEMP();



DELIMITER $$
CREATE PROCEDURE mergeEMPwithinBounds (IN lb INT, IN ub INT)
BEGIN
        DECLARE done INT DEFAULT 0;
        DECLARE eno INT;
        DECLARE old_cursor1 CURSOR FOR SELECT emp_id FROM O_EmpID WHERE emp_id
BETWEEN lb AND ub;
        DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
        OPEN old_cursor1;

        getEmpID: LOOP
                IF done = TRUE THEN
                        LEAVE getEmpID;
                END IF;
                FETCH old_cursor1 INTO eno;

                IF NOT EXISTS (SELECT 1 FROM N_EmpID where emp_id = eno) THEN
                        INSERT INTO N_EmpID
                        SELECT * FROM O_EmpID
                        WHERE O_EmpID.emp_id = eno;
                END IF;
        END LOOP;


        CLOSE old_cursor1;
END $$
DELIMITER ;
```

Assignment 7:

```
USE 31380_db;

CREATE TABLE Library (
	book_id INT UNIQUE PRIMARY KEY NOT NULL AUTO_INCREMENT,
	book_name VARCHAR(100) DEFAULT '',
	isbn INT UNIQUE NOT NULL,
	page_count INT,
	author VARCHAR(100),
	year DATE,
	copies_sold INT
);

CREATE TABLE Library_Audit (
	update_id INT UNIQUE PRIMARY KEY NOT NULL AUTO_INCREMENT,
	book_id INT,
	old_book_name VARCHAR(100) DEFAULT '',
	old_isbn INT UNIQUE NOT NULL,
	old_page_count INT,
	old_author VARCHAR(100),
	old_year DATE,
	old_copies_sold INT
);

INSERT INTO Library VALUES
	(1,'totam',14448882,425,'expedita','2023-03-19',43849117),
	(2,'voluptates',48822856,333,'sit','1974-08-27',11021007),
	(3,'quae',10721240,165,'voluptates','2011-05-06',72946253),
	(4,'impedit',77143556,133,'quam','1995-01-13',20492263),
	(5,'cum',57158171,667,'ipsa','2000-01-26',29278743),
	(6,'officia',23698255,804,'temporibus','2007-11-06',77006951),
	(7,'rerum',52692996,871,'numquam','2012-08-23',23321626),
	(8,'perspiciatis',80040097,697,'sunt','2006-05-02',80292082),
	(9,'voluptate',78591954,642,'voluptatem','1990-07-11',23638319),
	(10,'iure',37495366,644,'non','2020-03-08',59228850),
	(11,'repellendus',29416436,841,'in','1982-03-28',63717696),
	(12,'earum',14016995,778,'officia','1971-09-02',43033034),
	(13,'quasi',46762666,855,'magni','1973-04-12',22185173),
	(14,'nihil',12714950,583,'aspernatur','1991-07-24',83182555),
	(15,'perspiciatis',86876793,590,'cum','1979-03-02',89074976),
	(16,'qui',71512407,331,'maiores','2009-09-11',56650470),
	(17,'neque',31651442,511,'perspiciatis','1980-11-07',18051664),
	(18,'aut',77667078,386,'officia','1990-09-02',13554780),
	(19,'qui',67772169,114,'expedita','2006-01-31',35066713),
	(20,'quae',78427689,222,'quaerat','1992-07-22',54087575)
;

DELIMITER $$
CREATE TRIGGER update_trig
BEFORE UPDATE ON Library
FOR EACH ROW
BEGIN

	INSERT INTO Library_Audit (book_id, old_book_name, old_isbn, old_page_count,
old_author, old_year,old_copies_sold) VALUES
	(OLD.book_id, OLD.book_name, OLD.isbn, OLD.page_count, OLD.author, OLD.year,
OLD.copies_sold);

END $$
```

```
DELIMITER ;

-- Testing the trigger

UPDATE Library
SET book_name = 'haha totam' WHERE book_id = 1;

DELIMITER $$
CREATE TRIGGER error_trig
BEFORE UPDATE ON Library
FOR EACH ROW
BEGIN
	IF NEW.book_name = OLD.book_name THEN
	SIGNAL SQLSTATE '45000'
	SET message_text = 'Same value of updated book';
	END IF;
END $$
DELIMITER ;

UPDATE Library
SET book_name = 'quae' WHERE book_id = 3;
```

Assignment 9:

```
/*
use:

load('script.js');
*/

db = connect("mongodb://localhost/31380_db");

// Creating a movies collection
db.createCollection("movies");

/*
true
*/

// Inserting data into collection
db.movies.insertMany([
   {
      title: "The Shawshank Redemption",
      director: "Frank Darabont",
      releaseYear: 1994,
      genre: ["Drama", "Crime"],
      rating: 9.3,
   },
   {
      title: "The Godfather",
      director: "Francis Ford Coppola",
      releaseYear: 1972,
        genre: ["Crime", "Drama"],
        rating: 9.2,
    },
    {
        title: "Pulp Fiction",
        director: "Quentin Tarantino",
        releaseYear: 1994,
        genre: ["Crime", "Drama"],
        rating: 8.9,
    },
    {
        title: "The Dark Knight",
        director: "Christopher Nolan",
        releaseYear: 2008,
        genre: ["Action", "Crime", "Drama"],
        rating: 9.0,
    },
    {
        title: "Inception",
        director: "Christopher Nolan",
        releaseYear: 2010,
        genre: ["Action", "Adventure", "Sci-Fi"],
        rating: 8.8,
    },
]);

// Selecting all movies
db.movies.find().pretty();
```

```
/*
[
  {
    _id: ObjectId("6549072b4f850d45854084bb"),
    title: 'The Shawshank Redemption',
    director: 'Frank Darabont',
    releaseYear: 1994,
    genre: [ 'Drama', 'Crime' ],
    rating: 9.3
  },
  {
    _id: ObjectId("6549072b4f850d45854084bc"),
    title: 'The Godfather',
    director: 'Francis Ford Coppola',
    releaseYear: 1972,
    genre: [ 'Crime', 'Drama' ],
    rating: 9.2
  },
  {
    _id: ObjectId("6549072b4f850d45854084bd"),
    title: 'Pulp Fiction',
    director: 'Quentin Tarantino',
    releaseYear: 1994,
    genre: [ 'Crime', 'Drama' ],
    rating: 8.9
  },
  {
    _id: ObjectId("6549072b4f850d45854084be"),
    title: 'The Dark Knight',
    director: 'Christopher Nolan',
    releaseYear: 2008,
    genre: [ 'Action', 'Crime', 'Drama' ],
    rating: 9
  },
  {
    _id: ObjectId("6549072b4f850d45854084bf"),
    title: 'Inception',
    director: 'Christopher Nolan',
    releaseYear: 2010,
    genre: [ 'Action', 'Adventure', 'Sci-Fi' ],
    rating: 8.8
  }
]
*/

// Finding by name
db.movies.find({ title: "The Godfather" });

/*
[
  {
    _id: ObjectId("6549072b4f850d45854084bc"),
    title: 'The Godfather',
    director: 'Francis Ford Coppola',
    releaseYear: 1972,
    genre: [ 'Crime', 'Drama' ],
    rating: 9.2
  }
```

```
]
*/

// Finding by rating
db.movies.find({ rating: { $gt: 9 } });

/*
[
  {
    _id: ObjectId("6549072b4f850d45854084bb"),
    title: 'The Shawshank Redemption',
    director: 'Frank Darabont',
    releaseYear: 1994,
    genre: [ 'Drama', 'Crime' ],
    rating: 9.3
  },
  {
    _id: ObjectId("6549072b4f850d45854084bc"),
    title: 'The Godfather',
    director: 'Francis Ford Coppola',
    releaseYear: 1972,
    genre: [ 'Crime', 'Drama' ],
    rating: 9.2
  }
]
*/

// Updating a document
db.movies.updateOne({ title: "The Godfather" }, { $set: { rating: 9.9 } });
```

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

// Deprecated, use updateOne instead
db.movies.save(
   { _id: ObjectId("6549072b4f850d45854084bb") },
   {
       title: "The Shawshank Redemption",
       director: "Frank Darabont",
       releaseYear: 1994,
       genre: ["Drama", "Crime", "Thriller"],
       rating: 9.5,
   }
);

// Deleting a document
db.movies.deleteOne({ title: "The Shawshank Redemption" });

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

```
// AND operator
db.movies.find({
   title: "The Dark Knight",
   director: "Christopher Nolan",
});

/*
[
  {
   _id: ObjectId("6549072b4f850d45854084be"),
   title: 'The Dark Knight',
   director: 'Christopher Nolan',
   releaseYear: 2008,
   genre: [ 'Action', 'Crime', 'Drama' ],
   rating: 9
  }
]
*/

// OR operator
db.movies.find({
   $or: [{ director: "Christopher Nolan" }, { genre: "Action" }],
});

/*
[
  {
   _id: ObjectId("6549072b4f850d45854084be"),
   title: 'The Dark Knight',
   director: 'Christopher Nolan',
   releaseYear: 2008,
   genre: [ 'Action', 'Crime', 'Drama' ],
   rating: 9
  },
  {
   _id: ObjectId("6549072b4f850d45854084bf"),
   title: 'Inception',
   director: 'Christopher Nolan',
   releaseYear: 2010,
   genre: [ 'Action', 'Adventure', 'Sci-Fi' ],
   rating: 8.8
  }
]
*/
```

Assignment 10:

```
db = connect("mongodb://localhost/31380_db");


db.createCollection("Products");


db.Products.insertMany([
  {
     name: "Laptop",
     category: "Electronics",
     price: 80000,
     attribute: { color: "white", weight: "2pounds" },
     features: ["16GB RAM", "Intel core"],
     reviews: [
        { user: "amit", rating: 9 },
        { user: "sumit", rating: 8 },
     ],
  },
  {
     name: "Smartphone",
     category: "Electronics",
     price: 56000,
     attributes: { color: "blue", weight: "200g" },
     features: ["128GB memory", "Google assistance"],
     reviews: [
        { user: "riya", rating: 7 },
        { user: "rhea", rating: 9 },
     ],
  },
  {
    name: "T-shirt",
    category: "Clothing",
    price: 800,
    attributes: { color: "black", weight: "200g" },
    features: ["100% cotton", "slim fit"],
    reviews: [
      { user: "amit", rating: 9 },
      { user: "sumit", rating: 8 },
    ],
  },
  {
    name: "Jeans",
    category: "Clothing",
    price: 2000,
    attributes: { color: "blue", weight: "300g" },
    features: ["100% cotton", "slim fit"],
    reviews: [
      { user: "riya", rating: 7 },
      { user: "rhea", rating: 9 },
    ],
  },
  {
    name: "Shoes",
    category: "Footwear",
    price: 3000,
    attributes: { color: "black", weight: "500g" },
    features: ["100% leather", "slim fit"],
    reviews: [
      { user: "amit", rating: 9 },
        { user: "sumit", rating: 8 },
    ],
},
{
    name: "Sneakers",
    category: "Footwear",
    price: 2000,
    attributes: { color: "white", weight: "300g" },
    features: ["100% leather", "slim fit"],
    reviews: [
        { user: "riya", rating: 7 },
        { user: "rhea", rating: 9 },
    ],
},
{
    name: "Headphones",
    category: "Electronics",
    price: 2000,
    attributes: { color: "black", weight: "300g" },
    features: ["100% leather", "slim fit"],
    reviews: [
        { user: "amit", rating: 9 },
        { user: "sumit", rating: 8 },
    ],
},
{
    name: "Earphones",
    category: "Electronics",
    price: 2000,
        attributes: { color: "white", weight: "300g" },
        features: ["100% leather", "slim fit"],
        reviews: [
            { user: "riya", rating: 7 },
            { user: "rhea", rating: 9 },
        ],
    },
]);

db.Products.insertOne({
    name: "Bookshelf",
    category: "Furniture",
    price: 25000,
    attributes: { color: "black", weight: "5kgs" },
    features: ["12 small shelves", "2 drawers"],
    reviews: [
        { user: "Ruchi", rating: 10 },
        { user: "Aarti", rating: 8 },
    ],
});

/*
sum category wise
*/

db.Products.aggregate([
    { $group: { _id: "$category", sum: { $sum: "$price" } } },
]);
```

```
// Find Products with price less than 25,000
db.Products.aggregate([{ $match: { price: { $lt: 25000 } } }]);

/*to skip the first two documents*/
db.Products.aggregate([{ $skip: 2 }]);

db.Products.aggregate([
  { $group: { _id: "$category", avgprice: { $avg: "$price" } } },
]);

db.Products.aggregate([
  {
    $project: {
      _id: 0,
      name: 1,
      price: 1,
      no_of_reviews: { $size: "$reviews" },
    },
  },
]);

db.Products.aggregate([
  { $group: { _id: "$category", maxPrice: { $max: "$price" } } },
]);

// Index the table on name and price
db.Products.createIndex({ name: 1, price: -1 });

db.Products.find({ name: "Jeans", price: 2000 }).hint("name_1_price_-1");
db.Products.createIndex({ name: 1 });

// --------- ORDERS ---------

db.createCollection("orders");

db.orders.insertMany([
  {
    _id: 0,
    name: "Pepperoni",
    size: "small",
    price: 19,
    quantity: 10,
    date: ISODate("2021-03-13T08:14:30Z"),
  },
  {
    _id: 1,
    name: "Pepperoni",
    size: "medium",
    price: 20,
    quantity: 20,
    date: ISODate("2021-03-13T09:13:24Z"),
  },
  {
    _id: 2,
    name: "Pepperoni",
    size: "large",
    price: 21,
    quantity: 30,
    date: ISODate("2021-03-17T09:22:12Z"),
  },
  {
    _id: 3,
    name: "Cheese",
    size: "small",
    price: 12,
    quantity: 15,
    date: ISODate("2021-03-13T11:21:39.736Z"),
  },
  {
    _id: 4,
    name: "Cheese",
    size: "medium",
    price: 13,
    quantity: 50,
    date: ISODate("2022-01-12T21:23:13.331Z"),
  },
  {
    _id: 5,
    name: "Cheese",
    size: "large",
    price: 14,
    quantity: 10,
    date: ISODate("2022-01-12T05:08:13Z"),
  },
  {
    _id: 6,
         name: "Vegan",
         size: "small",
         price: 17,
         quantity: 10,
         date: ISODate("2021-01-13T05:08:13Z"),
     },
     {
         _id: 7,
         name: "Vegan",
         size: "medium",
         price: 18,
         quantity: 10,
         date: ISODate("2021-01-13T05:10:13Z"),
     },
  ]);

  db.orders.aggregate(
     {
         $match: {
             date: {
                 $gte: new ISODate("2020-01-30"),
                 $lt: new ISODate("2022-01-30"),
             },
         },
     },
     {
         $group: {
             _id: { $dateToString: { format: "%Y-%m-%d", date: "$date" } },
             totalOrderValue: { $sum: { $multiply: ["$price", "$quantity"] } },
             averageOrderQuantity: { $avg: "$quantity" },
         },
     },
     { $sort: { totalOrderValue: -1 } }
  );
```

Assignment 11:

```
var mapFunction = function () {
   emit(this.category, this.price); // {category: price}
};

var reduceFunction = function (key, values) { // key: category, values: price
   return Array.avg(values);
};

db.Products.mapReduce(mapFunction, reduceFunction, {
   out: "average_prices",
   query: { category: "Electronics" },
});

// { category: "Electronics", price: [80000, 56000, 2000]]}

db.Products.mapReduce(mapFunction, reduceFucntion, { out: "average_prices" });

/*Generalized to get sum of all the products*/
var mapFunction = function () {
   emit("total", this.price);
};
var reduceFunction = function (key, values)  {
   return Array.sum(values);
};

db.Products.mapReduce(mapFunction, reduceFunction, { out: "sum_prices" });
```