1.	Consider following Relation **Account(Acc_no, branch_name,balance)** **Branch(branch_name,branch_city,assets)** **Customer(cust_name,cust_street,cust_city)** **Depositor(cust_name,acc_no)** **Loan(loan_no,branch_name,amount)** **Borrower(cust_name,loan_no)** Create above tables with appropriate constraints like primary key, foreign key, not null 1. Find the names of all branches in loan relation. 2. Find all loan numbers for loans made at Akurdi Branch with loan amount > 12000. 3. Find all customers who have a loan from bank. Find their names,loan_no and loan amount. 4. Find all customers who have both account and loan at bank. 5. Find the average account balance at each branch 6. Find no. of depositors at each branch. 7. Calculate total loan amount given by bank. 8. Display distinct cities of branch.
2.	Consider following Relation **Employee(employee_name, street, city)** **Works(employee_name, company_name, salary)** **Company(company_name, city)** **Manages(employee_name, manager_name)** Create above tables with appropriate constraints like primary key, foreign key, not 1. Find the names of all employees who work for 'TCS'. 2. Find the names and company names of all employees sorted in ascending order of company name and descending order of employee names of that company. 3. Change the city of employee working with InfoSys to 'Bangalore' 4. Find the names of all employees who earn more than the average salary of all employees of their company. Assume that all people work for at most one company. 5. Find the names, street address, and cities of residence for all employees who work for 'TechM' and earn more than $10,000. 6. Change name of table Manages to Management. 7. Add Column Asset to Company table.

3.

Consider following Relation

Account(Acc_no, branch_name,balance)
Branch(branch_name,branch_city,assets)
Customer(cust_name,cust_street,cust_city)
Depositor(cust_name,acc_no)
Loan(loan_no,branch_name,amount)
Borrower(cust_name,loan_no)

Execute the following query:

1. Create a View1 to display List all customers in alphabetical order who have loan from Pune_Station branch.
2. Create View2 on branch table by selecting any two columns and perform insert update delete operations.
3. Create View3 on borrower and depositor table by selecting any one column from each table perform insert update delete operations.
4. Create Union of left and right joint for all customers who have an account or loan or both at bank
5. Display content of View1,View2,View3
6. Create Simple and Unique index.
7. Display index Information
8. Truncate table Customer.

4.

Consider following Relation: (Joins and View)

Companies(comp_id, name, cost, year)

C001	ONGC	2000	2010
C002	HPCL	2500	2012
C005	IOCL	1000	2014
C006	BHEL	3000	2015

Orders(comp_id, domain, quantity)

C001	Oil	109
C002	Gas	121
C007	Telecom	115
C008	IT	141

Execute the following query:

1. Find names, costs, domains and quantities for companies using inner join.
2. Find names, costs, domains and quantities for companies using left outer join.
3. Find names, costs, domains and quantities for companies using right outer join.
4. Find names, costs, domains and quantities for companies using Union operator.
5. Create View View1 by selecting both tables to show company name and quantities.
6. Create View2 on branch table by selecting any two columns and perform insert update delete operations.
7. Display content of View1, View2.

5.	Write a **Unnamed PL/SQL** of code for the following requirements:- **Consider Tables:** 1. Borrower(Roll_no, Name, Date of Issue, Name of Book, Status) 2. Fine(Roll_no, Date, Amt) • Accept Roll_no and Name of Book from user. • Check the number of days (from date of issue). • If days are between 15 to 30 then fine amount will be Rs 5per day. • If no. of days>30, per day fine will be Rs 50 per day and for days less than 30, Rs. 5 per day. • After submitting the book, status will change from I to R. • If condition of fine is true, then details will be stored into fine table. • Also handles the exception by named exception handler or user define exception handler.
6.	Write an **Unnamed PL/SQL** PL/SQL code block to calculate the area of a circle for a value of radius varying from 5 to 9. Store the radius and the corresponding values of calculated area in an empty table named areas, consisting of two columns, radius and area.
7.	**Named PL/SQL Block: PL/SQL Stored Procedure** Write a Stored Procedure namely proc_Grade for the categorization of student. If marks scored by students in examination is <=1500 and marks>=990 then student will be placed in distinction category if marks scored are between 989 and900 category is first class, if marks 899 and 825 category is Higher Second Class. Write a PL/SQLblock to use procedure created with above requirement. Stud_Marks(name, total_marks) Result(Roll,Name, Class)
8.	**Named PL/SQL Block: PL/SQL Stored Function.** Write a Stored Function namely proc_Grade for the categorization of student. If marks scored by students in examination is <=1500 and marks>=990 then student will be placed in distinction category if marks scored are between 989 and900 category is first class, if marks 899 and 825 category is Higher Second Class. Write a PL/SQLblock to use function created with above requirement. Stud_Marks(name, total_marks) Result(Roll,Name, Class

9.	Write a PL/SQL block of code using **Parameterized Cursor** that will merge the data available in the newly created table N_Roll Call with the data available in the table O_RollCall. If the data in the first table already exist in the second table then that data should be skipped.
10.	Write a PL/SQL block of code using **Explicit Cursor** that will merge the data available in the newly created table N_RollCall with the data available in the table O_RollCall. If the data in the first table already exist in the second table then that data should be skipped.
11.	Design and Develop MongoDB Queries using **CRUD operations**. **A. Create Empdb database and Employee collection for following Fields:** i. Empid: Number ii. Name: Embedded Doc (FName, LName) iii. Company Name: String iv. Salary: Number v. Designation: String vi. Age: Number vii. Expertise: Array viii. DOB: String or Date ix. Email id: String x. Contact: String xi. Address: Array of Embedded Doc (PAddr, LAddr) **C. Insert at least 05 documents in Employee Collection and execute following statements:** 1. Select all documents where the Designation field has the value "Programmer" and the value of the salary field is greater than 30000. 2. Creates a new document if no document in the employee collection contains {Designation: "Tester", Company_name: "TCS", Age: 25} 3. Selects all documents in the collection where the field age has a value less than 30 or the value of the salary field is greater than 40000. 4. Finds all documents with Company_name: "TCS" and modifies their salary field by 2000. 5. Find _id, Designation, Address and Name from all documents where Company_name is "Infosys". 6. Selects all documents in the employee collection where the value of the Designation is either "Developer" or "Tester".

12.	Write a program to implement **Mongo DB database connectivity** with any front end language (Python/Java) to implement Database navigation operations (add, delete, edit etc.)
13.	Design and Develop MongoDB Queries using **CRUD operations**. **A. Create Empdb database and Employee collection for following Fields:** i. Empid: Number ii. Name: Embedded Doc (FName, LName) iii. Company Name: String iv. Salary: Number v. Designation: String vi. Age: Number vii. Expertise: Array viii. DOB: String or Date ix. Email id: String x. Contact: String xi. Address: Array of Embedded Doc (PAddr, LAddr) **B. Insert at least 05 documents in Employee Collection and execute following statements:** 1. Select all documents where the Designation field has the value "Programmer" and the value of the salary field is greater than 30000. 2. Creates a new document if no document in the employee collection contains {Designation: "Tester", Company_name: "TCS", Age: 25} 3. Matches all documents where the value of the field Address is an embedded document that contains only the field city with the value "Pune" and the field Pin_code with the value "411001". 4. Finds all documents with Company_name: "TCS" and modifies their salary field by 2000. 5. Find documents where Designation is not equal to "Developer". 6. Find all document with Exact Match on an Array ['Mongodb','Mysql', 'Cassandra']
14.	Write a program to implement **MySQL/Oracle database connectivity** with any front end language (Python/Java) to implement Database navigation operations (add, delete, edit etc.)

15.

Design and Develop **MongoDB Queries using aggregation and indexing**.

A. Create Empdb database and Employee collection for following Fields:

i. Empid: Number
ii. Name: Embedded Doc (FName, LName)
iii. Company Name: String
iv. Salary: Number
v. Designation: String
vi. Age: Number
vii. Expertise: Array
viii. DOB: String or Date
ix. Email id: String
x. Contact: String
xi. Address: Array of Embedded Doc (PAddr, LAddr)

B. Perform Aggregation and Index Operation for statement below:

1. Return Designation with Total Salary is Above 200000
2. Find Total Salary of Employee with Designation="DBA" for Each Company
3. Return separates value in the Expertise array where Name of Employee="Swapnil"
4. Return Max and Min Salary for each company.
5. To Create Single Field Indexes on Designation
6. To Create Multikey Indexes on Expertise array
7. Return a List of All Indexes on Collection

16.

Implement MongoDB **Map reduces operation** for following database.

A. Create Empdb database and Employee collection for following Fields:

i. Empid: Number
ii. Name: Embedded Doc (FName, LName)
iii. Company Name: String
iv. Salary: Number
v. Designation: String
vi. Age: Number
vii. Expertise: Array
viii. DOB: String or Date
ix. Email id: String
x. Contact: String
xi. Address: Array of Embedded Doc (PAddr, LAddr)

B. Perform Map reduces operation for statement below:

1. Return the Total Salary of per Company
2. Return the Total Salary of Company Name:"TCS"
3. Return the Avg Salary of Company whose address is “Pune".
4. Return total count for “State=AP”.

17.	Design and Develop **MongoDB Queries using aggregation and indexing**. **A. Create Empdb database and Employee collection for following Fields:** i. Empid: Number ii. Name: Embedded Doc (FName, LName) iii. Company Name: String iv. Salary: Number v. Designation: String vi. Age: Number vii. Expertise: Array viii. DOB: String or Date ix. Email id: String x. Contact: String xi. Address: Array of Embedded Doc (PAddr, LAddr) **B. Perform Aggregation and Index Operation for statement below:** 1. Find Employee with Total Salary for Each City with Designation="DBA" 2. Find Total Salary of Employee with Designation="DBA" for Each Company 3. For each unique Designation, find avg Salary and output is sorted by AvgSal 4. Return separates value in the Expertise array and return sum of each element of array 5. To Create Compound Indexes on Name: 1, Age: -1 6. Return a List of All Indexes on Collection
18.	Write a **Row Level Before Trigger** on Library table. The System should keep track of the records that are being updated or deleted. The old value of updated or deleted records should be added in Library_Audit table.
19.	Write a **Row Level After Trigger** on Library table. The System should keep track of the records that are being updated or deleted. The old value of updated or deleted records should be added in Library_Audit table.