**Subject: DBMSL**

**List of Practical**

|  |  |
| --- | --- |
|  | 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. List all customers in alphabetical order who have loan from Akurdi branch. 5. Find all customers who have both account and loan at bank. 6. Find the average account balance at each branch 7. Find no. of depositors at each branch. 8. Calculate total loan amount given by bank. 9. Find name of Customer and city where customer name starts with Letter P. 10. Display distinct cities of branch. |
|  | 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. |
|  | 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. |
|  | Consider following Relation:  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. |
|  | **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. |
|  | Write a 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. |
|  | **Named PL/SQL Block: PL/SQL Stored Procedure and Stored Function.**  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) |
|  | **Cursors:(All types: Implicit, Explicit, Cursor FOR Loop, Parameterized Cursor)**  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. |
|  | **Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers).**  Write a database 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. |
|  | Write a program to implement MySQL/Oracle database connectivity with any front end language to implement Database navigation operations (add, delete, edit etc.) |
|  | Write a program to implement Mongo DB database connectivity with any front end language to implement Database navigation operations (add, delete, edit etc.) |
|  | Design and Develop MongoDB Queries using **CRUD operations**.(Use CRUD operations, SAVE method, logical operators etc.).  **A. Create Empdb database**  **B. Create Employee collection by considering 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  3. {Designation: "Tester", Company\_name: "TCS", Age: 25}  4. 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.  5. 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".  6. Finds all documents with Company\_name: "TCS" and modifies their salary field by  2000.  7. Find documents where Designation is not equal to "Developer".  8. Find \_id, Designation, Address and Name from all documents where Company\_name  is "Infosys".  9. Selects all documents in the employee collection where the value of the Designation  is either "Developer" or "Tester".  10. Find all document with Exact Match on an Array ['Mongodb','Mysql', 'Cassandra']  11. Drop Single documents where designation="Developer" |
|  | Design and Develop **MongoDB Queries using aggregation and indexing** with suitable example using MongoDB.  **A. Create Empdb database**  **B. Create Employee collection by considering 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. Aggregation Operation**  1. Return Designation with Total Salary is Above 200000  2. Find Employee with Total Salary for Each City with Designation="DBA"  3. Find Total Salary of Employee with Designation="DBA" for Each Company  4. Returns names and \_id in upper case and in alphabetical order.  5. Count all records from collection  6. For each unique Designation, find avg Salary and output is sorted by AvgSal  7. Return separates value in the Expertise array where Name of Employee="Swapnil"  8. Return separates value in the Expertise array and return sum of each element of array  9. Return Array for Designation whose address is "Pune"  10. Return Max and Min Salary for each company.  **D. Index Operation**::  1. To Create Single Field Indexes on Designation  2. To Create Compound Indexes on Name: 1, Age: -1  3. To Create Multikey Indexes on Expertise array  4. Return a List of All Indexes on Collection  5. Rebuild Indexes  6. Drop Index on Remove Specific Index  7. Remove All Indexes except for the \_id index from a collection |
|  | Implement **Map reduces operation** with suitable example using MongoDB.  **A. Create Empdb database**  **B. Create Employee collection by considering 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. Use Employee database and perform Map reduces operation for following statements**:  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 the Total Salary for each Designation of Infosys.  5. Return total count for “State=AP”  6. Return Count for State AP and Age greater than 40. |