List of variation Programs in the Practical Exam

Note: Only these problem statements will not be asked in practical exam, similar types of problem statements may be asked.

Collection "city" which contains the documents given as below(Perform on Mongo Terminal)

```
{
        city: "pune",
        type: "urban",
        state: "MH",
        population: 5600000
}
```

-using mapreduce, find statewise population

-using mapreduce, find citywise population

-using mapreduce, find typewise population.

Collection "books" which contains the documents given as below(Perform on Mongo Terminal)

```
{
        name: "understanding  JAVA"
        pages:400
}
```

-using MapReduce, categerize books into small books and big books, based on number of pages display and categorize total number of books.

Collection "movies" which contains the documents given as below(Perform on Mongo Terminal)

```
{
        name: "Movie1"
        type: "action"
        budget:1000000
        producer:{
                        name: "producer1"
                        address: "PUNE"
                        }
}
```

i. Find the name of the movie having budget greater than 1,00,000.
ii. Find the name of producer who lives in Pune
iii. Update the type of movie "action" to "horror"
iv. Find all the documents produced by name "producer1" with their address

Collection "orderinfo" which contains the documents given as below(Perform on Mongo Terminal)

{

cust_id:123
cust_name: "abc",
status: "A",
price: 250

}

i. Find the total price for each customer and display in the order of total price.
ii. Find the distinct customer names
iii. Display the "price" of customers whose status is 'A'
iv. Delete the customers whose status is 'A'

Collection "orderinfo" which contains the documents given as below(Perform on Mongo Terminal)

{

cust_id:123
cust_name:"abc"
status:"A"
price:250

}

i. find the average price for each customers having status 'A'
ii. Display the status of the customers whose amount/price lie between 100 and 1000
iii. Display the customers information without "_id" .
iv. create a simple index on onderinfo collection and fire the queries. Also explain the impact of index before and after.

Collection "orderinfo" which contains the documents given as below(Perform on Mongo Terminal)

{

cust_id:123
cust_name:"abc"
status:"A"
price:250

}

i. Add "Age" field to the orderinfo collection
ii. Create the complex index on orderinfo collection and fire the queries and drop the duplicates.
iii. Display the average price for each customer group by status
iv. Change the customer"s name whose status is "B"

Collection "orderinfo" which contains the documents given as below(Perform on Mongo Terminal)

{

cust_id:123
cust_name:"abc"
status:"A"
price:250

}

i. Display the name of the customer having the price between 250 and 450
ii. Increment the price by 10 for cust_id: 123 and decrement the price by 5 for cust_id: 124
iii. Remove any one of the field from the orderinfo collection.
iv. Find the name of the customer whose status is either A or price is 250 or both.

(Perform on MYSQL Terminal)
teaches(T_ID, course_id, sec_id, semester, year)
student(S_ID, name, dept_name, tot_cred)
instructor(T_ID, name, dept_name, salary)
course(course_id, title, dept_name, credits)

i. Find the names of the instructor in the university who have taught the courses semester wise.
ii. Create View on single table which retrieves student details.
iii. Rename the column of table student from dept_name to department_name
iv. Delete student name whose department is NULL

(Perform on MYSQL Terminal)
teaches(T_ID, course_id, sec_id, semester, year)
student(S_ID, name, dept_name, tot_cred)
instructor(T_ID, name, dept_name, salary)
course(course_id, title, dept_name, credits)

i. Find the names of all instructor whose salary is greater than 25000.
ii. Update Instructor dept_name from CSE to IT
iii. Create Simple Index on course table on the attribute dept_name and find out department wise courses.
iv. Create a View to retrieve data of instructor with details of course, section, semester and year he conducted lectures on.

(Perform on MYSQL Terminal)
teaches(T_ID,course_id,sec_id,semester,year)
section(course_id,sec_id,semester,year,building,room_no)
instructor(T_ID,name, dept_name,salary)
course(course_id,title,dept_name,credits)

i. Display the name of the instructor whose salary is between 30000 and 60000 in descending order.
ii. Find the average, minimum, maximum salary in each department.
iii. Display the name of the instructor, his department name and salary who teaches the course_id: 101
iv. Find the name of the instructor whose average salary in greater then 60000.

(Perform on MYSQL Terminal)
student(S_ID,name,dept_name,tot_cred)
instructor(T_ID,name,dept_name,salary)
course(course_id,title,dept_name,credits)

i. Find the average salary of instructor in those departments where the average salary is more than Rs. 42000/-.
ii. Increase the salary of each instructor in the computer department by 10%.
iii. Find the names of instructors whose names are neither „Amol‟ nor „Amit‟.
iv. Find the names of student which contains „am‟ as its substring.
v. Find the name of students from computer department that "DBMS" courses they take.

(Perform on MYSQL Terminal)
teaches(T_ID,course_id,sec_id,semester,year)
student(S_ID,name,dept_name,tot_cred)
instructor(T_ID,name,dept_name,salary)
course(course_id,title,dept_name,credits)

i. Find the total number of instructor who teach a course in the First semester 2010.
ii. Find the names of all instructors from computer department whose name includes the substring „shree‟.
iii. Find the names of all instructor that have a salary greater than average salary of the Civil Department.
iv. Find the set of all courses taught either in Fifth semester 2014 or in First semester 2010 or both.

(Perform on MYSQL Terminal)
Emp(emp_id,ename, street, city)
works(emp_id,c_id,ename, cname, sal)
Company(c_id,cname, city)
Manager(mgr_id, mgrname)

i. Modify the database so that a particular company (eg. ABC) now in Pune
ii. Give all managers of Mbank a 10% raise. If salary is >20,000, give only 3% raise.
iii. Find out the names of all the employees who works in „Bosch‟ company in city Pune
iv. Delete all records in the works table for employees of a particular company (Eg, SBC Company) whose salary>50,000.

(Perform on MYSQL Terminal)

Empl(e_no, e_name, post, pay_rate)
Position(pos_no, post)
Duty-alloc (pos_no, e_no, month,year, shift)

Implement the following SQL queries

i. Get duty allocation details for e_no 123 for the first shift in the month of April 2003
ii. Get the employees whose rate of pay is > or equal rate of pay of employees 'Sachin'.
iii. Create a view for displaying minimum, maximum and average salary for all the posts.
iv. Get a count of different employees on each shift having post „manager".

Using MySQL and JAVA connectivity (Two Tier) perform the following queries

i. Create a table of employee details , Employee(SSN, Ename,state, salary)
ii. Insert Records into Employee table
iii. Retrieve the details based on Social Security Number(SSN).
iv. Update the employee state from 'MH' to 'TN'
v. Delete all the employees from „Gujrat"

Use Java and MongoDB connectivity and perform the following operations
Student(Roll_no, S_name, Semester,Year, No_of_backlogs)

i. Create a database and student collection
ii. Insert the documents in to the student collection
iii. Delete the students who have got backlogs>4.
iv. Retrieve the documents based on student "Roll_no"

Use Java and MongoDB connectivity and Write a MapReduce program for categorising books as "Big book" and "Small book" based on number of pages.

```
Books {
        Name:"Understanding mongoDB",
        Pages: 200
        }
```

Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View, Index using Client-Data sever(two tier)

Employee(e_no, e_name, post, pay_rate)
Position(pos_no, post)
Duty-alloc (pos_no, e_no, month,year, shift)

i. Create view that displays first three columns of employee table.
ii. Create view that displays left outer joins on employee and Position tables on post criteria.
iii. Create index on pos_no attribute of Position table.
iv. Based on indexed column fire the queries.

Collection "orderinfo" which contains the documents given as below(Perform on Mongo Terminal)

{
cust_id:123
cust_name:"abc"
status:"A"
price:250
}

i. Find the total price for each customer and display in the order of total price.
ii. Display the average price for each customer group by status
iii. Create simple index on orderinfo collection.
iv. Create the complex index on orderinfo collection and fire the queries and drop the duplicates.

- Write a PL/SQL block of code for the following requirements:-
  Schema:
  1. Borrower(Rollin, Name, DateofIssue, NameofBook, Status)
  2. Fine(Roll_no,Date,Amt)
     - Accept roll_no& 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 & 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.

- Write a PL/SQL block to implement all type of cursor(Implicit,Explicit, Cursor FOR Loop,Parameterized Cursor). Also write a PL/SQL block of code using parameterized 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.

- 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 and 900 category is first class, if marks 899 and 825 category is Higher Second Class. Write a PL/SQL block for using procedure created with above requirement.

  Stud_Marks(rollno, name, total_marks)
  Result(Roll,Name, Class)

- Write a PL/SQL block to implement database trigger on Library Table.