Database Management Systems

Questions for Practise

Question 1

- Discuss relational DBMS, considering a simple example.

Question 2

- Explain aggregation taking an example.

Question 3

- Discuss the difference between specialization and generalization

Question 4

- Make a simple ER diagram for a test module, where students can take up the test and the teacher can put the result on the module itself. Consider including multiple-choice questions as well as subjective questions in all courses.

Question 5

- ***Consider the relation:***

student(sid, sname, address, gender, age)

Write relational algebra(RA) for the following:

1. To display the name of student whose age is greater than 15.
2. To remove the record of the student who are from Kathmandu.
3. To update address of student to Kathmandu whose sid is 'S1101'.

Questions

a. To display name of student whose age is greater than 15.

π sname (σ age > 15 (student))

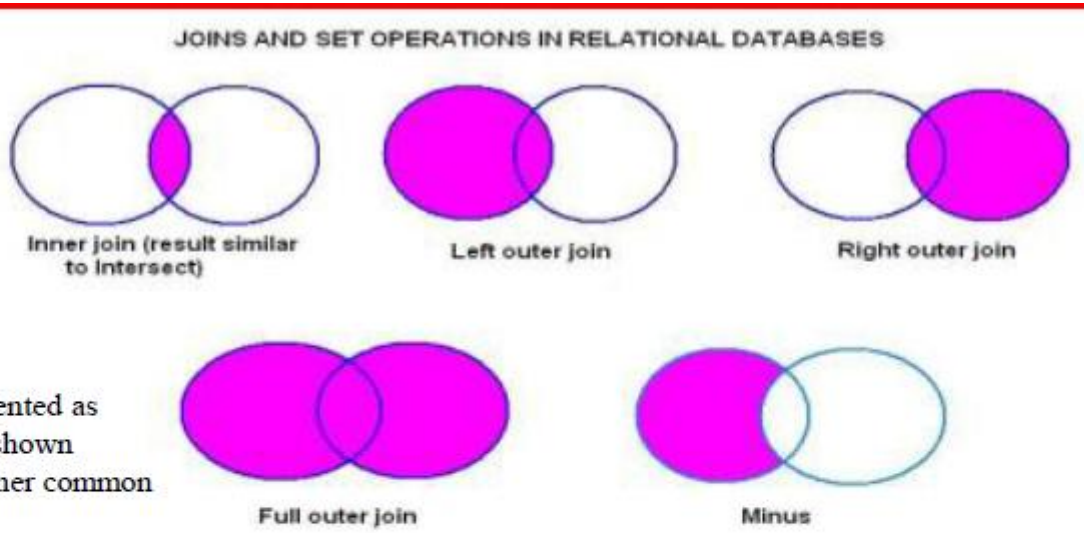
b. To remove the record of the student who are from Kathmandu.

student $\leftarrow$ student - (σ address = 'Kathmandu' (student))

c. To update address of student to Kathmandu whose sid is 'S1101'.

student $\leftarrow$ π sid, sname, address = 'Kathmandu', gender, age (σ sid = 'S1101' (student)) $\cup$ (σ sid $\neq$ 'S1101' (student))

JOINS



Joins may be represented as Venn diagrams, as shown above along with other common set operations:

Result of applying these joins in a query:
INNER JOIN: Select only those rows that have values in common in the columns specified in the ON clause.
LEFT, RIGHT, or FULL OUTER JOIN: Select all rows from the table on the left (or right, or both) regardless of whether the other table has values in common and (usually) enter NULL where data is missing.

Left Outer Join

EmployeeID	Name	DepartmentID
1	John	101
2	Sarah	102
3	Michael	NULL
4	Emma	103

DepartmentID	DepartmentName
101	HR
102	IT
103	Marketing
104	Finance

```
SELECT Employees.Name, Employees.DepartmentID, Departments.DepartmentName
FROM Employees
LEFT JOIN Departments
ON Employees.DepartmentID = Departments.DepartmentID;
```

Right Outer Join

EmployeeID	Name	DepartmentID
1	John	101
2	Sarah	102
3	Michael	NULL
4	Emma	103

DepartmentID	DepartmentName
101	HR
102	IT
103	Marketing
104	Finance

```
SELECT Employees.Name, Employees.DepartmentID, Departments.DepartmentName
FROM Employees
RIGHT JOIN Departments
ON Employees.DepartmentID = Departments.DepartmentID;
```

Full Outer Join

EmployeeID	Name	DepartmentID
1	John	101
2	Sarah	102
3	Michael	NULL
4	Emma	103

DepartmentID	DepartmentName
101	HR
102	IT
103	Marketing
104	Finance

```
SELECT Employees.Name, Employees.DepartmentID, Departments.DepartmentName
FROM Employees
FULL JOIN Departments
ON Employees.DepartmentID = Departments.DepartmentID;
```

Inner Join

EmpID	Name	DeptID	Salary
1	Alice	10	60000
2	Bob	20	55000
3	Carol	10	70000
4	Dave	30	50000
5	Eva	NULL	65000

DeptID	DeptName
10	HR
20	Finance
30	IT
40	Marketing

Inner Join

List all employees with their department names.

```
SELECT Name, DeptName
FROM Employees e
INNER JOIN Departments d
ON e.DeptID = d.DeptID;
```

Left Join

EmpID	Name	DeptID	Salary
1	Alice	10	60000
2	Bob	20	55000
3	Carol	10	70000
4	Dave	30	50000
5	Eva	NULL	65000

DeptID	DeptName
10	HR
20	Finance
30	IT
40	Marketing

Left Join

Show all employees and their department names (including those without a department).

```
SELECT e.Name, d.DeptName
FROM Employees e
LEFT JOIN Departments d
ON e.DeptID = d.DeptID;
```

Right Join

EmpID	Name	DeptID	Salary
1	Alice	10	60000
2	Bob	20	55000
3	Carol	10	70000
4	Dave	30	50000
5	Eva	NULL	65000

DeptID	DeptName
10	HR
20	Finance
30	IT
40	Marketing

Right Join

Show all departments and the employees in them (including departments with no employees).

```
SELECT e.Name, d.DeptName
FROM Employees e
RIGHT JOIN Departments d
ON e.DeptID = d.DeptID;
```

Full Outer Join

EmpID	Name	DeptID	Salary
1	Alice	10	60000
2	Bob	20	55000
3	Carol	10	70000
4	Dave	30	50000
5	Eva	NULL	65000

DeptID	DeptName
10	HR
20	Finance
30	IT
40	Marketing

Right Join

List all employees and departments, even if there's no match.

```
SELECT e.Name, d.DeptName
FROM Employees e
FULL OUTER JOIN Departments d
ON e.DeptID = d.DeptID;
```

Join Questions

Table 1 — Students

StudentID	StudentName	CourseID	Marks
1	Alice	101	85
2	Bob	102	75
3	Carol	103	92
4	David	101	66
5	Eva	NULL	78

Table 2 — Courses

CourseID	CourseName	Instructor
101	Computer Science	Dr. Adams
102	Mathematics	Prof. Green
103	Physics	Dr. Lee
104	Chemistry	Dr. Brown

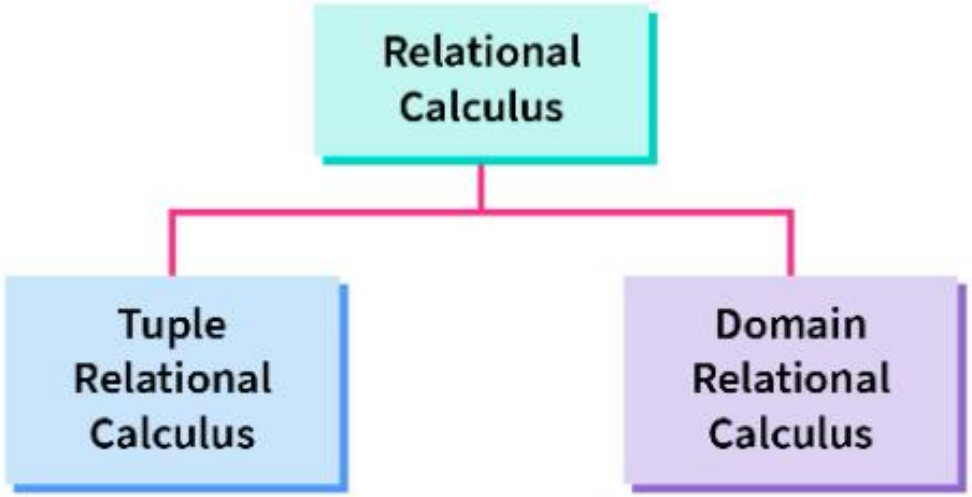
List all students with their course names.

Show all students with their course names (even if they are not enrolled in any course).

Show all courses and the students taking them (including courses with no students).

List all students and courses, whether or not they are linked.

Relational Calculus



1. **Procedural Language** - Those Languages which clearly define how to get the required results from the Database are called Procedural Language. **Relational algebra** is a Procedural Language.

2. **Declarative Language** - Those Language that only cares about What to get from the database without getting into how to get the results are called Declarative Language. **Relational Calculus** is a Declarative Language.

Tuple Calculus

$$\{t \mid P(t)\}$$

Customer Table

Customer_id	Name	Zip code
1	Rohit	12345
2	Rahul	13245
3	Rohit	56789
4	Amit	12345.

Write a TRC query to get all the data of customers whose zip code is 12345.

TRC Query: $\{t \mid t \in Customer \wedge t.Zipcode = 12345\}$ or TRC Query: $\{t \mid Customer(t) \wedge t[Zipcode] = 12345\}$

Domain Calculus

{<x1,x2,x3,x4...> \| P(x1,x2,x3,x4...)}

Customer Table

Customer_id	Name	Zip code
1	Rohit	12345
2	Rahul	13245
3	Rohit	56789
4	Amit	12345.

Write a DRC query to get the data of all customers with Zip code 12345.

DRC query: {<x1,x2,x3> \| <x1,x2> ∈ Customer ∧ x3 = 12345 }

Sub Queries

Find employees earning more than the average salary:

```
SELECT name, salary
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);
```

Write an SQL query to display the names and salaries of employees who earn more than the average salary of their respective department.

```
SELECT e1.name, e1.salary
FROM employees e1
WHERE e1.salary > (
   SELECT AVG(e2.salary)
   FROM employees e2
   WHERE e1.department_id = e2.department_id
);
```

Questions for Practise

employees

emp_id	emp_name	department_id	salary	manager_id

departments

department_id	dept_name	location

Find employees whose department is one of the departments located in London or Paris.

List all employees who work in the same department as Alice.

Find employees who are the highest paid in their department.

Practice all the set operation queries

emp_id	emp_id
emp_name	name
department	department

UNION — Employees who worked in 2023 or 2024

```
SELECT emp_id, emp_name, department FROM emp_2023
UNION
SELECT emp_id, emp_name, department FROM emp_2024;
```

Practice all the set operation queries

emp_id
emp_name
department

emp_id
name
department

UNION ALL — Including duplicates

```
SELECT emp_id, emp_name, department FROM emp_2023
UNION ALL
SELECT emp_id, emp_name, department FROM emp_2024;
```

Practice all the set operation queries

emp_id
emp_name
department

emp_id
name
department

INTERSECT — Employees who worked in both 2023 and 2024

```
SELECT emp_id, emp_name, department FROM emp_2023
INTERSECT
SELECT emp_id, emp_name, department FROM emp_2024;
```

Practice all the set operation queries

emp_2023	emp_2024
emp_id	emp_id
emp_name	name
department	department

MINUS — Employees who worked in 2023 but not in 2024

```
SELECT emp_id, emp_name, department FROM emp_2023
MINUS
SELECT emp_id, emp_name, department FROM emp_2024;
```

Practice Questions Schema

students_2023

student_id
student_name
course

students_2024

student_id
student_name
course

sales_north

sales_id
employee_name
region

sales_south

sales_id
employee_name
region

Practice Questions

List all employees (unique names) from both north and south regions.

Show all distinct employee IDs from both regions.

Combine students from both years and count total records before removing duplicates.

Find employees who work in both North and South regions.

Find employees who are only in the South region.

Find students who enrolled in exactly one year (2023 or 2024, but not both).

Practice Questions

STUDENT(student_id, name, dept_id)
DEPARTMENT(dept_id, dept_name)
COURSE(course_id, course_name, dept_id)
ENROLLMENT(student_id, course_id, grade)

- List all students and their department names, including students not yet assigned to a department.
- Show all departments and students in them, including departments that have no students.
- List all students and the courses they are enrolled in (if any).

Practice Solutions

List all students and their department names, including students not yet assigned to a department.

```
SELECT s.name, d.dept_name
FROM STUDENT s
LEFT OUTER JOIN DEPARTMENT d
ON s.dept_id = d.dept_id;
```

Practice Solutions

Show all departments and students in them, including departments that have no students.

```
SELECT s.name, d.dept_name
FROM STUDENT s
RIGHT OUTER JOIN DEPARTMENT d
ON s.dept_id = d.dept_id;
```

Practice Solutions

List all students and the courses they are enrolled in (if any).

```
SELECT s.name, e.course_id
FROM STUDENT s
LEFT OUTER JOIN ENROLLMENT e
ON s.student_id = e.student_id;
```

Practice Questions

EMPLOYEE_SALES(emp_id, emp_name, dept, salary)
EMPLOYEE_MARKETING(emp_id, emp_name, dept, salary)

List all employees who work in either the Sales or Marketing department.

Find employees who work in both Sales and Marketing departments.

List employees who work in Sales but not in Marketing.

Practice Questions (Subquery (Single Row Subquery))

EMPLOYEE(emp_id, emp_name, dept, salary, hire_date)

DEPARTMENT(dept_id, dept_name)

- Display the employee(s) who earn the **highest salary** in the company.

```
SELECT emp_name, salary
FROM EMPLOYEE
WHERE salary = (SELECT MAX(salary) FROM EMPLOYEE);
```

Practice Questions (Subquery (Single Row Subquery))

EMPLOYEE(emp_id, emp_name, dept, salary, hire_date)

DEPARTMENT(dept_id, dept_name)

- Display the employee names in uppercase along with the length of each name.

```
SELECT UPPER(emp_name) AS NAME_IN_UPPERCASE,
LENGTH(emp_name) AS NAME_LENGTH
FROM EMPLOYEE;
```

Practice Questions (Subquery (Single Row Subquery))

EMPLOYEE(emp_id, emp_name, dept, salary, hire_date)

DEPARTMENT(dept_id, dept_name)

- Find employees who joined before the earliest hired employee in the 'HR' department.