QB

Q.1 Explain various relational algebra operators

⇒ • Fundamental operations of Relational Algebra :

1) Unary Relational Operations

⇒

a) Project Operation (π)
b) Select operation (σ)
c) Rename operation (ρ)

2) SET Theory Operations

⇒

a) Union Operation ($\cup$)
b) Difference Operation (-)
c) Intersection Operation ($\cap$)

3) Binary Operation

⇒

a) Join Operation ($\bowtie$)
b) Cartesian product Operation ($\times$)
c) Division operation (%)

• Unary Relational Operation :

1) Select Operation

⇒ • This operator is used to select some rows from table which satisfy particular selection condition given in the selection operation.

• Selection operator selects a set of tuples that satisfy a ~~particular condi~~ selection condition

- The symbol σ (sigma) is used to denote the select operator.
- Its syntax is –

$\sigma_{<attribute\_name> < comparison\_operator> <constant\_values>}$

- For e.g.,
SELECT the Employee tuples whose salary is greater than ₹.75,000/-

$\sigma_{salary > 75,000}$ (Employee)

2) PROJECT Operation (π)

⇒ • This operator is used for selecting some of many columns in table to displayed in result set.

- It is denoted by π symbol.
- Its syntax –

$\pi_{<column\_list>}$ (Input_table_name)

- For e.g.,
To list each employee's first, and last name and salary

⇒

$\pi_{LNAME, FNAME, SALARY}$ (EMPLOYEE)

3) RENAME Operation

⇒

- We can give alternative name to any column or any table of query expressions using operator called RENAME operator.
- It is denoted by ρ (rho)
- It's syntax is -

$\rho_{<New\_names>}$ (Input_table_name)

- For e.g.,

Find salary and age of all Employees

⇒

$\Pi_{e.age,\ e.salary} (\rho_e (\text{Empolyee}))$

- SET Operation

1) Union Operator

⇒ This operator finds out all combined rows in table 1 and table2.

Duplicates ~~are~~ tuples are eliminated.

It's syntax is -

(Query Expression 1) U (Query Expression 2)

For e.g.,

Find all students in comps & IT dept.

⇒

(COMPS_STUDENT) U (IT_STUDENT)

2) Intersection Operator

⇒ This operator finds out all rows that are common in table 1 and table.

It is denoted by $\cap$.

It's syntax is -

(Query Expression 1) $\cap$ (Query Expression 2)

For e.g., Find all members ~~of~~ who are in both CSI & codecell Comittee.

⇒ (CSI) $\cap$ (CODECELL)

3) Difference Operator

⇒ This operator finds out all rows that are present in table 1 and not in table2.

It is denoted by (-).

It's syntax is -

(Query Expression 1) - (Query Expression 2)

For e.g.,

Find all members of CSI comittee.

⇒

(CSI_member) - (Other_committe_member)

- Binary Operation

=>

1) Cartesian Product

=> This operation is used to combine tuples from two relations in a combinational fashion.

Denoted by -

$$R(A_1, A_2, \ldots, A_n) \times S(B_1, B_2, \ldots B_m)$$

For e.g.,

Find ~~all~~ combinations all Employee and departments.

=> (Employee) × (Department)

- A cartesian product is formed when-

i) A Join condition is omitted

ii) A join condition is invalid

iii) To avoid a Cartesian product, always include a valid join condition in a WHERE clause.

iii) Different types of joins.

⇒ • Join operations helps us to retrieve data from multiple tables or relations.

i) Natural Join

⇒ Natural can join tables based on the common columns in the tables being joined.

For e.g.,

E_no	E-name	Address
1	A	W
2	B	X
3	C	Y
4	D	Z

Dept-no	Name	E-no
D1	I	1
D2	J	2
D3	K	5
D4	H	4

Query

⇒ find all Emp-names who is working in a department.

Solution

⇒

```
SELECT   E-name
FROM     EMP  NATURAL JOIN  Dept
```

Output

⇒

E-no	Ename	Address	Dept-no	Name	E-no
1	A	W	D1	I	1
2	B	X	D2	J	2
4	D	Z	D3	H	4

ii) Self Join

⇒ SQL SELF JOIN is used to join a table to itself as if the table were two tables.

For e.g;

S_id	C_id	Since
S1	C1	2016
S2	C2	2017
S1	C2	2017

→ study

Query

⇒ Find student id who is enrolled in at least two courses.

Solution

⇒

```
SELECT   T1.S_id
FROM     study as T1 , Study as T2
WHERE    T1.S_id = T2.S_id      and
         T1.C_id <> T2.C_id
                  ↑
              Not equal to
```

Output

⇒

S_id
S1

~~C_id~~

iii) EQUI JOIN (INNER)

⇒ This will combine tuples from multiple relations if they satisfy ~~join~~ the specified join condition.

For e.g.,

E_No	E-name	Address	Dept_No	location	E-No
1	AB	Delhi	D1	Delhi	1
2	B	Pune	D2	Pune	2
3	C	Mumbai	D3	Harayana	4
4	D	Bandra			

Query

⇒ find the Emp_name who worked in a department having location same as address

Solution

⇒

```
SELECT   E-name
 FROM    Emp, Dept
 WHERE   Emp. E-No = Dept. E-No   and
         Emp. Address   =   Dept. location
```

Output

⇒

E-NO	E-Name	Address	Dept-No	location	E-No
1	A	Delhi	D1	Delhi	1
2	B	Pune	D2	Pune	2

iv) Outer Joins

⇒

a) Left Outer Join

⇒ Returns all the rows from the left table, even if there are no match in the right table.

For e.g.,

Emp

Eid	EName	Did
1	A	D1
2	B	D2
3	C	D3

Dept

Did	Dname
D1	IT
D2	HR
D4	TIS

Query

⇒ Find all departments with emp data.

Solution

⇒ ~~SELECT~~

Emp = ⋈ emp.did = Dept.did Dept.

Output

⇒

Eid	EName	Did	Did	Dname
1	A	D1	D1	IT
2	B	D2	D2	HR
3	C	D3	NULL	NULL

b) Right outer join
⇒ Returns all the rows from the right table, even if there are no matches in the ~~right~~ left table.

c) Full outer join
⇒ Combines the result of both left and right outer join.