Name: Anvit Gupta   Enroll: 22114009   Batch: CSE-02   Year: 3rd year
Course Code: CSN-351: DBMS.

Assignment-3

① Query (c) will produce different answer.

Example:- $R = \{(3,4)\}$ and $S = \{(1,2)\}$

Query (a) and (b) will produce empty result while (c) produce $\{(3,2)\}$

where $\pi_X(R)$ = Projection of Relation R on set of Attributes X

$\sigma_{<cond>}(R)$ = Relation containing only that tuples of Relation R which satisfy <cond>.

② (a) Minimum no. of tuples will occur when one relation is subset of another.

$\boxed{\text{Minimum} = \max(r,s)}$ Ans

Maximum will occur when both the relations are disjoint.

$\boxed{\text{Maximum} = (r+s)}$ Ans

(b) Minimum will occur if there are shared B values, and will produce no. of tuples = 0.

$\boxed{\text{Minimum} = 0}$ Ans

Maximum, if all the B values are same in both relation R and S

$\boxed{\text{Maximum} = (r \times s)}$ Ans

(c) If no shared B values, then minimum no. of tuples will be in output relation.

$\boxed{\text{Minimum} = 0}$

Maximum will occur when the below conditions are satisfied:

(i) All B values are distinct

(ii) one relation's B values are a subset of the other's.

$\boxed{\text{Maximum} = \min(r, s)}$ Ans

(d) $(R \bowtie R) \bowtie R \equiv R \bowtie R \equiv R$

Hence: $\boxed{\text{minimum} = \text{maximum} = r}$ Ans

(e) If $A = B$, $\forall$ tuples of R, then minimum no. of tuples will be in output relation.

$\boxed{\text{minimum} = 0}$ Ans

If $A \neq B$ $\forall$ tuples of R, then maximum no. of tuples and

$\boxed{\text{maximum} = r}$ Ans

3. (a) $\Pi_{pizzeria}(\sigma_{age<18}(Person) \bowtie Frequents)$

(b) $\Pi_{name}(\sigma_{gender=\text{"female"} \land (pizza=\text{"mushroom"} \lor pizza=\text{"pepperoni"})}(Person \bowtie Eats))$

or

~~#~~ $\Pi_{name}(\sigma_{<condition>}(Person \bowtie Eats))$

where $<condition> = (gender=\text{"female"} \land (pizza=\text{"mushroom"} \lor pizza=\text{"pepperoni"}))$

(c) $\Pi_{name}(\sigma_{gender=\text{"female"} \land pizza=\text{"mushroom"}}(Person \bowtie Eats))$
$\cap\ \Pi_{name}(\sigma_{gender=\text{"female"} \land pizza=\text{"pepperoni"}}(Person \bowtie Eats))$

(d) $\Pi_{pizzeria}(\sigma_{name=\text{"Amy"}}(Eats) \bowtie \sigma_{price<10}(Serves))$

(e) $\Big(\Pi_{pizzeria}(\sigma_{gender=\text{"female"}}(Person) \bowtie Frequents) - \Pi_{pizzeria}(\sigma_{gender=\text{"male"}}(Person) \bowtie Frequents)\Big)$
$\cup \Big(\Pi_{pizzeria}(\sigma_{gender=\text{"male"}}(Person) \bowtie Frequents) - \Pi_{pizzeria}(\sigma_{gender=\text{"female"}}(Person) \bowtie Frequents)\Big)$

(f) Eats − $\pi_{name, pizza}$ (Frequents ⋈ Serves)

(4.) 1.)

```
SELECT DISTINCT S.name
FROM Student S
WHERE S.snum NOT IN (SELECT E.snum FROM Enrolled E);
```

2.)

```
SELECT MAX(S.age)
FROM Student S
WHERE (S.major = "History")
OR S.snum IN (SELECT E.snum
FROM class C, Enrolled E, Faculty F
WHERE E.cname = C.name AND c.fid = F.fid.
AND F.fname = "I. Teach");
```

3.)

```
SELECT C.name
FROM class C
WHERE c.room = "R128" OR C.name IN
(SELECT E.cname FROM Enrolled E GROUP BY
E.cname HAVING COUNT(*) >= 5);
```

4.)

```
SELECT DISTINCT S.sname FROM Student S
WHERE S.snum IN (SELECT E1.snum
FROM Enrolled E1, Enrolled E2, Class C1, Class C2
WHERE E1.snum = E2.snum AND E1.cname
<> E2.cname AND E1.cname = C1.name
AND E2.cname = C2.name AND c1.meets at =
C2.meets at);
```

5.)

```
SELECT DISTINCT F.fname FROM faculty F
WHERE NOT EXISTS ((SELECT * FROM class C)
EXCEPT (SELECT C1.room FROM class C1
WHERE C1.fid = F.fid));
```

```
6.) SELECT DISTINCT F.fname FROM Faculty F
    WHERE 5 > (SELECT COUNT (E.snum) FROM Class C,
    Enrolled E  WHERE C.name = E.cname AND C.fid=F.fid);

7.) SELECT S.level, AVG(S.age)
    FROM Student S GROUP BY S.level.;

8.) SELECT S.level, AVG(S.age) FROM Student S
    WHERE S.level <> "JR"
    GROUP BY S.level ;

9.) SELECT F.fname, COUNT(*) AS CourseCount
    FROM Faculty F, Class C
    WHERE F.fid= C.fid GROUP BY F.fid, F.fname
    HAVING EVERY (C.room = "R128") ;

10.) SELECT DISTINCT S.sname
     FROM Student S
     WHERE S.snum IN (SELECT E.snum
      FROM Enrolled E GROUP BY E.snum
      HAVIN COUNT(*) >= ALL (SELECT COUNT(*) FROM
       Enrolled E2 GROUP BY E2.snum));
```