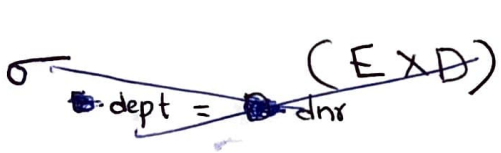
1.

(a). Relational algebra for given SQL:



$$\sigma_{dept = dnr}(E \times D)$$

(b).

enr	ename	dept	dnr	dname
1	Bill	A	A	marketing
2	Sarah	C	C	legal
3	John	A	A	marketing

* Select * means all attributes, so we perform E X D and take tuples were dept = dnr.

2. Given relation R_1 has N_1 tuples and R_2 has N_2 tuples.

Also $N_2 > N_1 > 0$.

a). $R_1 \cup R_2$

Assumption: R_1 and R_2 are union compatible.

Union Compatible:

Two relations R_1 and R_2 are said to be union compatible to each other if and only if

1. They have same number of attributes.
2. Domains of the respective attributes is same.

Possible Sizes:

1. Minimum: N_2 (if all tuples of R_1 also exist in R_2)
2. Maximum: $N_1 + N_2$ (if they are disjoint)

b). $R_1 \cap R_2$

Assumption: R_1 and R_2 are union compatible.

Possible Sizes:

1. Minimum: 0 (if they are disjoint)
2. Maximum: N_1 (if all tuples of R_1 exist in R_2)

c). $R_1 - R_2$

Assumption: R_1 and R_2 are union compatible.

Possible Sizes:

1. Minimum: 0 (if all tuples of R_1 exist in R_2)
2. Maximum: N_1 (if they are disjoint)

d). $R_1 \times R_2$

- No special assumptions except that duplicates are not being removed.

Possible Sizes :

1. Minimum : $N_1 * N_2$

2. Maximum : $N_1 * N_2$

e). $\sigma_{a=5}(R_1)$

Assumption : R_1 has attribute with name 'a'.

Possible Sizes :

1. Minimum : 0 (when no tuple of R_1 has value of attribute $a = 5$)

2. Maximum : N_1 (if all tuples of R_1 have value of attribute $a = 5$)

3. Assuming that there in an attribute named 'Salary' in Employee table. The following can be used to get 10th highest employee salary

```
select TOP(1) Salary From
{
        DISTINCT
    Select ^TOP(10) Salary From Employee ORDER by Salary DESC
} as Employeedata Order by Salary ASC
```

→ Take minimum of highest 10 salaries.

4. Given that table "TBL" has field "Nmbr".

To add 2 when Nmbr is 0 and 3 where Nmbr 1.

```
update TBL
    set Nmbr = case
            when Nmbr = 0 then Nmbr + 2
            else Nmbr + 3
            end
```

5.

$$(\sigma_{color = 'red'} Parts) \bowtie (\sigma_{cost < 100} Catalog)$$

↳ gives tuple (pid, pname, color, sid, cost) where all products have 'red' color and have a cost '100'.

$$(\sigma_{\cancel{color = 'green'}} \Pi_{sid}((\sigma_{color = 'red'} Parts) \bowtie (\sigma_{cost < 100} Catalog)))$$

gives supplier ids of suppliers who supply red part and whose cost is less than 100 dollars.

Similarly $\Pi_{sid}((\sigma_{color = 'green'} Parts) \bowtie (\sigma_{cost < 100} Catalog))$

gives supplier ids of suppliers who supply green parts and whose cost is less than 100 dollars.

$$\therefore \left(\pi_{sid}\left(\left(\sigma_{color='red'} Parts\right) \bowtie \left(\sigma_{cost<100} Catalog\right)\right)\right) \cap \left(\pi_{sid}\left(\left(\sigma_{color='green'} Parts\right) \bowtie \left(\sigma_{cost<100} Catalog\right)\right)\right)$$

Gives supplier id's of suppliers who supply both a red part and a green part where each costs less than 100 dollars.

6. In the given Query, we are getting eid's of employees with maximum salary.

To write a Relational Algebra for this since there is no 'MAX' operator, we can first find employees who don't have highest salary and then subtract these from total list of employees to get the list of highest paid employees.

$\rho_X(\text{Employees})$ [returns Employees under name X]

$\rho_Y(\text{Employees})$ [return Employees under name Y]

$\rho_Z(\pi_{Y.eid}(\sigma_{X.salary > Y.salary}(X \times Y)))$ → eid's of employees without max salary

$\rho_S(\pi_{Employees.eid}(Employees))$ → eid's of all employees

$S - Z$ → eid's of employees with max salary

7. Given SQL queries return eid's of employees with second maximum salary.

To do this with relational algebra, as it doesn't have 'MAX', first we find employees who don't have maximum salary. From this list say R, find highest paid employees. This gives us second highest paid employees list.

ρ_P (Employees) [returns employees under name P]

ρ_Q (Employees) [returns employees under name Q]

$\rho_R (\Pi_{Q.eid} (\sigma_{P.salary > Q.salary} (P \times Q)))$ [eid's of employee without max salary]

$\rho_S (P \bowtie R)$ [return natural join of P, R as S]

$\rho_T (P \bowtie R)$ [return natural join of P, R as T]

$\rho_U (\Pi_{T.eid} (\sigma_{S.salary > T.salary} (S \times T)))$

[gets eid's of employee without 2nd max salary]

$R - U$ [eid's of employee with maximum salary]

9. Select S.sname, MAX (C.cost) as MaximumPrice

From Suppliers S, Parts P, Catalog C

Where P.pid = C.pid and S.sid = C.sid

Group by

S.sid,

S.sname

Having any (P.color = 'red') and (P.color = 'green')

9. Select

```
    userdata . user_id,
    username,
    training_id,
    training_date
from users userdata JOIN trainingdetails tabledata ON userdata.user_id = tabledata.user_id
Group by user_id,
         username,
         training_id,
         training_date
Having count (user_training_id) > 1
Order by training_date DESC;
```

We just group by id, name, data and check if count > 1.

10. Given table A. To get required tuple

```
select sum(case when value > 0 then value else 0 end) as positive_sum,
       sum(case when value < 0 then value else 0 end) as negative_sum
from A.
```