CSE 532 Assignment #1

Group Members:
1. Jatin Garg (111462753)
2. Rahul Bhansali (111401451)

1. Consider the tables Frequents(drinker, bar), Likes(drinker, beer), Sells(bar, beer), with the following meanings: A tuple (d,b) in Frequents means that d frequents the bar b, a tuple (d,b) in Likes means that d likes the beer b, and a tuple (B,b) in Sells means that the bar B sells the beer b. Assume the tables do not have any duplicates. Answer the following queries in Relational Algebra. Do not use the aggregate operator.

a) Find pairs of drinkers that frequent the SAME SET of bars.

Two pairs of drinkers (D1, D2) frequent the same set of bars, if D1 frequents all bars that D2 frequents and D2 frequents all bars that D1 frequents. So,to find pairs of drinkers (D1, D2) that frequent the same set of bars, we do the following:
1. Find all pairs of drinkers (D1, D2) such that D1 frequents all the bars that D2 frequents.
2. Find all pairs of drinkers (D2, D1) such that D2 frequents all the bars that D1 frequents.
3. Take intersection of the result of above two queries.

Query for Step 1:

$$AllDrinkers = \Pi_{drinker}(Frequents)$$
$$AllBars = \Pi_{bar}(Sells)$$
$$NotFrequents = AllDrinkers\ X\ AllBars - Frequents$$
$$NotAnswer = \Pi_{NotFrequents.drinker,\ Frequents.drinker}(\sigma_{Frequents.drinker <> NotFrequents.drinker\ AND\ Frequents.bar = NotFrequents.bar})(Frequents\ X\ NotFrequents)$$

$$Superset(D1,\ D2) = AllDrinkers\ X\ AllDrinkers - NotAnswer$$

Query for Step 2:

$$Subset(D1,\ D2) = \rho_{D1/D2,\ D2/D1}(Superset)$$

Query for Step 3:

$Answer\ =\ Superset \cap Subset$

b) A beer is called "popular" if it is sold by at least 4 different bars. Find drinkers that like all the popular beers.

Steps:

1) In this, we first find the set of popular beers iteratively by first finding bars sold by at least 2 bars, then finding beers sold by at least 3 bars and then beers sold by at least 4 bars.
2) Then, we will find beers that a drinker doesn't likes and make a table for that.
3) We will subtract those drinkers who doesn't like atleast 1 popular beer(by using natural join) from all drinkers which will give us the answer.

Queries:

1. Make copies S1, S2, S3 and S4 of Sells tables.
2. $S5\ = \Pi(\sigma(_{S1.bar <> S2.bar\ and\ S1.beer = S2.beer}))(S1\ X\ S2)$
3. $S6\ =\ \Pi(\sigma\ _{(S5.S1.bar <> S3.bar\ and\ S5.S2.bar <> S3.bar\ and\ S5.S1.beer = S3.beer)}\ (S5\ X\ S3)$
4. $PopBeer\ =\ \Pi(_{Beer})(\sigma(_{S6.S5.S1.bar <> S4.bar\ and\ S6.S5.S2.bar <> S4.bar\ and\ S6.S5.S3.bar <> S4.bar\ and\ S6.S3.beer = S4.beer}))(\ S5\ X\ S4\ )$
5. $AllDrinker\ =\ \Pi\ _{drinker}(Frequents)$
6. $AllBeer\ =\ \Pi\ _{beer}(Sells)$
7. $NotLikes\ =\ (AllDrinker\ X\ AllBeer)\ -\ Likes(Drinker,\ Beer)$
8. $Answer\ =\ AllDrinker\ -\ \Pi(_{Drinker})(NotLikes \bowtie PopBeer)$

c) We say that a bar b is redundant if there exists a bar b1 such that (i) b1 sells all the beers that b sells, and (ii) All the drinkers that frequent b *and* like some beer that b sells, also frequent b1. Find all redundant bars.

Let's find:

(i) b1 sells all the beers that b sells -
So in first step we are just finding pair of bars (b1,b2) s.t b1 sells all beers that b2 sells.

$AllBeers = \Pi_{beer}(Sells)$

$AllBars = \Pi_{bar}(Sells)$

$NotSells = AllBars\ X\ AllBeers - Sells$

$$NotAnswer = \Pi_{NotSells.bar,\ Sells.bar}(\sigma_{Sells.bar <> NotSells.bar\ AND\ Sells.beer = NotSells.beer})(Sells\ X\ NotSells)$$

$Superset(NBar,\ Bar) = AllBars\ X\ AllBars - NotAnswer$

(ii) All the drinkers that frequent b *and* like some beer that b sells, also frequent b1
So in this we are finding pair of bars(b1,b2) s.t a drinker does not frequents b1 and frequents b2 and also likes a beer in b2.

$LikesBar = \Pi_{drinker,\ bar}(Likes \bowtie Sells \bowtie Frequents)$

$AllDrinkers = \Pi_{drinker}(Frequents)$

$AllBars = \Pi_{bar}(Sells)$

$NF = AllDrinkers\ X\ AllBars - Frequents$

$\varrho_F(Frequents)$ $(creating\ a\ copy\ of\ Frequents\ named\ F)$

$$FNF = \Pi_{drinker/F.drinker,\ bar/F.bars,\ NBar/NF.bar}(\sigma_{F.drinker = NF.drinker\ AND\ F.bar <> NF.bar})(F\ X\ NF)$$

$NotAnswer = \Pi_{NBar,\ Bar}(LikesBar \bowtie FNF\ )$

Now we subtract (ii) from (i) :
Finally, the answer is:

$Answer = \Pi_{Bar}(Superset - NotAnswer1\ )$

2) Consider the following two queries over R(a,c) and S(a,b).

Q1:

```
Select  r.a, r.c
From    R r
Where   NOT EXISTS (Select s.a
               From  S s
               Where s.b > '4' and r.a = s.a)
```

Q2:

```
Select  r.a, r.c
From    R r,  (select distinct S.a, S.b From  S) s
Where   NOT (r.a = s.a and s.b > '4')
```

- **Give a relational algebra expression that is equivalent to Q1. Assume bag semantics, i.e., assume that the relational algebra operators are bag operators.**
- **Are Q1 and Q2 equivalent (return the same bags, i.e, same tuples with the same number of duplicates)? If yes, PROVE it. If not, show a counter example.**

Sol) a.) We will find those tuples in R X S where S.b > 4 and R.a equals to S.a and then we will subtract them from the R table to get our answer.

1.) $A\ =\ \Pi_{(R.a,\ R.c)}\ \sigma_{(R.a\,=\,S.a\ AND\ S.b\,>\,4)}\ (R\ X\ S)$

2.) $Answer\ =\ R\ -\ A$

b.) The 2 queries are not equal because: in first query a tuple in R cannot occur more than once in result while in second query a tuple can come more than once if it satisfies the condition as we are taking cartesian product in second query. Lets see the example:
Consider 2 tables :
R =

a	c
Y	6

S =

a	b
X	3
X	1
X	2
X	3

In query 2, the result would be :

a	c
Y	6
Y	6
Y	6

While in query 1, the result is:

a	c
Y	6

Hence, the two queries are not equal.

Question .3)
Consider the following two queries over R(a,b) and S(b,c).

```
Q1:     SELECT   A
        FROM     R NATURAL JOIN S
        GROUPBY  A
        HAVING   COUNT(*) < 2

Q2:     SELECT   A
        FROM     R
        WHERE    B NOT IN (SELECT   B
                           FROM     S
                           GROUPBY  B
                           HAVING   COUNT(*) > 1)
```

Treating Q1 and Q2 as multisets (bags), which one of the following is true? Prove your answer (perhaps, by using a counter example or otherwise).
. Q1 and Q2 always produce the same answer for all databases.
. The answer to Q1 is always contained in the answer to Q2 for all databases.
. The answer to Q2 is always contained in the answer to Q1 for all databases.
. None of the above.

Sol) The answer is option b. **The answer to Q1 is always contained in the answer to Q2 for all databases.**
So first let's disapprove a and c by proving that q1 not equal q2 and also q2 is not subset of q1.

R =

a	b
X	Y
X	P

S =

b	c
Y	Z

So q1 will give output as :

a
X

While q2 will give output as :

a
X
X

Because the subquery in q2 will not return anything and we would get X from both rows as the operators are following bag semantics. So a and c are not possible.

Proof for b.)
q1) Due to natural join all those rows having no common values in the other table will get eliminated. Query 1 will give us only copy of R.a like it cannot give us 2 X's in R.a. **Query 1 will give R.a of only those rows whose R.b is found only once in S.b** and exact duplicate rows in q1 will also be eliminated.
q2.) **Query 2 will give R.a of those rows whose R.b is found only once in S.b due to subquery and it will also give R.a of those tuple whose R.b is not present in S at all which will not be given by query1 due to natural join**. Also as it is bag semantics, the number of R.a given by query2 will always be more than R.a given by query1 as we are selecting all R.a in query2

whose R.b is present only once in relation S while in query 1 as we have done group by so only one copy of R.a can be received.

4. Do the following exercises from the textbook.

2.4.1 (i). Note that you can't use the aggregate operator here.

Exercise 2.4.1: This exercise builds upon the products schema of Exercise 2.3.1. Recall that the database schema consists of four relations, whose schemas are:

Product(maker,model, type)
PC(model, speed, ram, hd, price)
Laptop(model, speed, ram, hd, screen, price)
Printer(model, color, type, price)

2.4.1 (i) Find the manufacturer(s) of the computer (PC or laptop) with the highest available speed.

$$A = \Pi_{model,\, speed}(PC)$$
$$B = \Pi_{model,\, speed}(Laptop)$$
$$C = A \cup B$$
$$D = \Pi_{model}(\sigma_{C1.speed < C2.speed}(\rho_{C1}(C)\; X\; \rho_{C2}(C)))$$
$$E = \Pi_{maker}(Product \bowtie D)$$
$$Answer = \Pi_{maker}(Product) - E$$

2.4.10:

$$Given:\; R(A_1, A_2, ..., A_n, B_1, B_2, ..., B_m)$$
$$S(B_1, B_2 ..., B_m)$$

$$Solution:$$
$$Q = \Pi_{A_1, A_2, ..., A_n}(R)$$
$$T = Q\,X\,S$$
$$P = \Pi_{T.A_1, T.A_2, ..., T.A_n}(\sigma_{T.A_1 <> R.A_1\; OR\; T.A_2 <> R.A_2\; OR ... T.A_n <> R.A_n ... OR\; T.B_m <> R.B_m}(TXR)\,)$$
$$Answer = Q - P$$

Exercise 6.3.1: Write the following queries, based on the database schema

Product(maker,model, type)
PC(model, speed, ram, hd, price)
Laptop(model, speed, ram, hd, screen, price)
Printer(model, color, type, price)

You should use at least one subquery in each of your answers and write each query in two significantly different ways (e.g., using different sets of the operators EXISTS,IN,ALL,and ANY).

f) Find the maker(s) of the PC(s) with the fastest processor among all those PC's that have the smallest amount of RAM.

Solution 1:

```
SELECT p.maker
FROM Product p
WHERE p.model IN (SELECT pc2.model
                  FROM PC pc2
                  WHERE pc2.speed >= ALL (SELECT pc1.speed
                                          FROM PC pc1
                                          WHERE pc1.ram <= ALL (SELECT pc.ram
                                                                FROM PC pc)));
```

Solution 2:

```
SELECT p.maker
FROM Product p
WHERE p.model = ANY (SELECT pc2.model
                  FROM PC pc2
                  WHERE pc2.speed NOT IN(
                              SELECT pc3.speed
                              FROM PC pc3
                              WHERE pc3.speed <
                                   ANY( SELECT pc4.speed
                                        FROM PC pc4
                                        WHERE pc4.ram
                                             NOT IN( SELECT pc5.ram
                                                     FROM PC pc5
                                                     WHERE pc5.ram >
                                                          ANY( SELECT ram
                                                               FROM PC)))));
```

6.3.6

Let S be a Relation:

EXISTS can be replaced by IN as follows:

```
SELECT …..
FROM S
WHERE EXISTS ( R );
```

becomes:

```
SELECT ….
FROM S
WHERE ‘Constant’  IN (SELECT ‘Constant’ AS DummyColumn
                              FROM R);
```

NOT EXISTS can be replaced by NOT IN as follows:

```
SELECT ….
FROM S
WHERE NOT EXISTS ( R );
```

becomes:

```
SELECT ….
FROM S
WHERE ‘Constant’ NOT IN (SELECT ‘Constant’ AS DummyColumn
                                   FROM R);
```

So in general to prove we can say that when we use EXISTS operator, we put the condition that we want to filter inside EXISTS operator so that if that condition is true , it will return some rows and EXISTS operator will return TRUE so that row will be picked of outside query. So equivalently in IN operator, we use the LHS or RHS of condition in EXISTS operator as LHS of WHERE clause in IN operator and put the other part in IN clause, so LHS is in RHS we will still get the row of the relation. So both IN and EXISTS can be used interchangeably. Similarly NOT EXISTS can be replaced by NOT IN.

6.4.6 (j)

```
SELECT pc.maker, AVG(hd)
```

```
FROM PC pc NATURAL JOIN (SELECT p1.maker, p1.model
                         FROM Product p1
                         WHERE p1.maker IN (SELECT p2.maker
                                            FROM Product p2
                                             WHERE p2.model IN (SELECT printer.model
                                                                FROM Printer printer)))
GROUP BY pc.maker;
```