QUESTION1)

EXISTS(...UNION...):

select L(r1,...,rk) from R1 r1,...Rk rk, S1 s1,...Sm sm where C1(r1,....,rk) and C2(s1,...,sm,r1,...,rk)

union

select L(r1,...,rk) from R1 r1,...,Rl rk, T1 t1,..., Tn tn where C1(r1,...,rk) and C3(t1,...tn,r1,...,rk);

IN RA:

$$\boldsymbol{\pi}_{L(\boldsymbol{r1},\ldots,\boldsymbol{rk})}(\boldsymbol{\sigma}_{\boldsymbol{C1}(\boldsymbol{r1},\ldots,\boldsymbol{rk})\wedge \boldsymbol{C2}(\boldsymbol{s1},\ldots\boldsymbol{sm},\boldsymbol{r1},\ldots,\boldsymbol{rk})}(\boldsymbol{R1} \times \ldots \times \boldsymbol{Rk} \times \boldsymbol{S1} \times \ldots \times \boldsymbol{Sm})) \cup \boldsymbol{\pi}_{L(\boldsymbol{r1},\ldots,\boldsymbol{rk})}(\boldsymbol{\sigma}_{\boldsymbol{C1}(\boldsymbol{r1},\ldots,\boldsymbol{rk})\wedge \boldsymbol{C3}(\boldsymbol{t1},\ldots\boldsymbol{tn},\boldsymbol{r1},\ldots,\boldsymbol{rk})}(\boldsymbol{R1} \times \ldots \times \boldsymbol{Rk} \times \boldsymbol{T1} \times \ldots \times \boldsymbol{Tn}))$$

not EXISTS(...UNION...):

select Lq(r1,...,rk) from

(select r1.*,...,rk.* from R1 r1,...Rk rk where C1(r1,....,rk)

except

(select r1.*,...,rk.* from R1 r1,...Rk rk, S1 s1,...Sm sm where C1(r1,....,rk) and C2(s1,...,sm,r1,...,rk)

UNION

select r1.*,...,rk.* from R1 r1,...,Rl rk, T1 t1,..., Tn tn where C1(r1,...,rk) and C3(t1,...tn,r1,...,rk))) q;

IN RA:

$$\boldsymbol{\pi}_{L(\boldsymbol{r1},\ldots,\boldsymbol{rk})}(\boldsymbol{\pi}_{(\boldsymbol{r1}.*,\ldots,\boldsymbol{rk}.*)}(\boldsymbol{\sigma}_{\boldsymbol{C1}(\boldsymbol{r1},\ldots,\boldsymbol{rk})}(\boldsymbol{R1} \times \ldots \times \boldsymbol{Rk})) - (\boldsymbol{\pi}_{(\boldsymbol{r1}.*,\ldots,\boldsymbol{rk}.*)}(\boldsymbol{\sigma}_{\boldsymbol{C1}(\boldsymbol{r1},\ldots,\boldsymbol{rk})\wedge \boldsymbol{C2}(\boldsymbol{s1},\ldots\boldsymbol{sm},\boldsymbol{r1},\ldots,\boldsymbol{rk})}(\boldsymbol{R1} \times \ldots \times \boldsymbol{Rk} \times \boldsymbol{S1} \times \ldots \times \boldsymbol{Sm})) \cup \boldsymbol{\pi}_{(\boldsymbol{r1}.*,\ldots,\boldsymbol{rk}.*)}(\boldsymbol{\sigma}_{\boldsymbol{C1}(\boldsymbol{r1},\ldots,\boldsymbol{rk})\wedge \boldsymbol{C3}(\boldsymbol{t1},\ldots\boldsymbol{tn},\boldsymbol{r1},\ldots,\boldsymbol{rk})}(\boldsymbol{R1} \times \ldots \times \boldsymbol{Rk} \times \boldsymbol{T1} \times \ldots \times \boldsymbol{Tn}))))$$

EXISTS(...INTERSECT...):

select Lq(r1,...,rk) from

(select r1.*,...,rk.* from R1 r1,...Rk rk, S1 s1,...Sm sm where C1(r1,....,rk) and C2(s1,...,sm,r1,...,rk)

INTERSECT

select r1.*,...,rk.* from R1 r1,...,Rl rk, T1 t1,..., Tn tn where C1(r1,...,rk) and C3(t1,...tn,r1,...,rk)) q;

IN RA:

$$\boldsymbol{\pi_{L(r1,\dots,rk)}(\pi_{L(r1.*,\dots,rk.*)}(\sigma_{C1(r1,\dots,rk)\wedge C2(s1,\dots sm,r1,\dots,rk)}(R1\times\dots\times Rk\times S1\times\dots\times Sm))}$$
$$\boldsymbol{\cap\ \pi_{L(r1.*,\dots,rk.*)}(\sigma_{C1(r1,\dots,rk)\wedge C3(t1,\dots tn,r1,\dots,rk)}(R1\times\dots\times Rk\times T1\times\dots\times Tn)))}$$

not EXISTS(...INTERSECT...):

select Lq(r1,...,rk) from

(select r1.*,...,rk.* from R1 r1,...Rk rk where C1(r1,....,rk)

except

(select r1.*,...,rk.* from R1 r1,...Rk rk, S1 s1,...Sm sm where C1(r1,....,rk) and C2(s1,...,sm,r1,...,rk)

INTERSECT

select r1.*,...,rk.* from R1 r1,...,Rl rk, T1 t1,..., Tn tn where C1(r1,...,rk) and C3(t1,...tn,r1,...,rk))) q;

IN RA:

$$\boldsymbol{\pi_{L(r1,\dots,rk)}(\pi_{(r1.*,\dots,rk.*)}\left(\sigma_{C1(r1,\dots,rk)}(R1\times\dots\times Rk)\right)}$$
$$\boldsymbol{-\ (\pi_{L(r1.*,\dots,rk.*)}(\sigma_{C1(r1,\dots,rk)\wedge C2(s1,\dots sm,r1,\dots,rk)}(R1\times\dots\times Rk\times S1\times\dots\times Sm))}$$
$$\boldsymbol{\cap\ \pi_{L(r1.*,\dots,rk.*)}(\sigma_{C1(r1,\dots,rk)\wedge C3(t1,\dots tn,r1,\dots,rk)}(R1\times\dots\times Rk\times T1\times\dots\times Tn))))}$$

EXISTS(...EXCEPT...):

select Lq(r1,...,rk) from

(select r1.*,...,rk.* from R1 r1,...Rk rk, S1 s1,...Sm sm where C1(r1,....,rk) and C2(s1,...,sm,r1,...,rk)

EXCEPT

select r1.*,...,rk.* from R1 r1,...,Rl rk, T1 t1,..., Tn tn where C1(r1,...,rk) and C3(t1,...tn,r1,...,rk)) q;

IN RA:

$$\boldsymbol{\pi_{L(r1,\dots,rk)}(\pi_{L(r1.*,\dots,rk.*)}(\sigma_{C1(r1,\dots,rk)\wedge C2(s1,\dots sm,r1,\dots,rk)}(R1 \times \dots \times Rk \times S1 \times \dots \times Sm))}$$
$$\boldsymbol{- \ \pi_{L(r1.*,\dots,rk.*)}(\sigma_{C1(r1,\dots,rk)\wedge C3(t1,\dots tn,r1,\dots,rk)}(R1 \times \dots \times Rk \times T1 \times \dots \times Tn)))}$$

not EXISTS(...EXCEPT...):

select Lq(r1,...,rk) from

(select r1.*,...,rk.* from R1 r1,...Rk rk where C1(r1,....,rk)

except

(select r1.*,...,rk.* from R1 r1,...Rk rk, S1 s1,...Sm sm where C1(r1,....,rk) and C2(s1,...,sm,r1,...,rk)

EXCEPT

select r1.*,...,rk.* from R1 r1,...,Rl rk, T1 t1,..., Tn tn where C1(r1,...,rk) and C3(t1,...tn,r1,...,rk))) q;

IN RA:

$$\boldsymbol{\pi_{L(r1,\dots,rk)}(\pi_{(r1.*,\dots,rk.*)}\left(\sigma_{C1(r1,\dots,rk)}(R1 \times \dots \times Rk)\right)}$$
$$\boldsymbol{- (\pi_{L(r1.*,\dots,rk.*)}(\sigma_{C1(r1,\dots,rk)\wedge C2(s1,\dots sm,r1,\dots,rk)}(R1 \times \dots \times Rk \times S1 \times \dots \times Sm))}$$
$$\boldsymbol{- \ \pi_{L(r1.*,\dots,rk.*)}(\sigma_{C1(r1,\dots,rk)\wedge C3(t1,\dots tn,r1,\dots,rk)}(R1 \times \dots \times Rk \times T1 \times \dots \times Tn))))}$$

QUESTION 2)

$SET1 \rightarrow \pi_{a,d}(R \bowtie_{c=d} S)$

$SET2 \rightarrow \ \pi_{a,d}\left(\pi_{a,c}(R) \bowtie_{c=d} \pi_d(S)\right).\boldsymbol{Let\ us\ represent\ this\ as\ \pi_{A_L}(\pi_{A_{L_R}}(R) \bowtie_{A_C} \pi_{A_{L_S}}(S))}$

$Where\ A_L = \{a, d\}, A_C = \{c, d\}$

$A_{L_R} = A_R \cap (A_L \cup A_C)\ (\because\ Given\ A_R = \{a, b, c\}, A_S = \{d, e\})$

$A_{L_S} = A_S \cap (A_L \cup A_C)$

$Hence,\ we\ can\ say\ that\ A_{L_R} \subseteq A_R\ and\ A_{L_S} \subseteq A_S. So, \boldsymbol{\pi_{A_{L_R}}(R) \subseteq R\ and\ \pi_{A_{L_S}}(S) \subseteq S}$...Statement 1

Since we are not using the attributes $\{b\} \in A_R\ \ and\ \{e\}\ \in$
$A_S\ \ in\ \ A_L or\ A_C\ \ we\ can\ eliminate\ them\ in\ the\ join\ condition$ ....Statement 2

From Statement1 and Statement2, we can say that $\boldsymbol{SET2 \subseteq SET1}$**. And** $\boldsymbol{SET1 \subseteq SET2}$

QUESTION 3)

The expression $\pi_{a,d}\left(\pi_{a,c}(R) \bowtie_{c=d} \pi_d(S)\right)$ can be simplified to $\boldsymbol{\pi_{a,c}(R)}$**.**

Below is the justification:

The final projection is on the attributes a from relation R and d from relation S. The join between R to its parent table uses only the foreign key (FK) attribute c and Primary KEY(PK) attribute d. Hence, it is notable to observe that the values observed in attribute d will be observed in attribute c too. And since there are no other attributes from S that are present in the projection of the join, we can eliminate the parent relation S all together.

QUESTION 4a)

```
select distinct s.sid, s.sname
from student s,buys t, cites c, book b1, book b2, major m
where s.sid = m.sid and m.major = 'CS' AND s.sid = t.sid and t.bookno = c.citedbookno and
c.citedbookno = b1.bookno and c.bookno = b2.bookno and
b1.price > b2.price;
```

Translated query:

$$\pi_{s.sid,s.sname}(\sigma_{\substack{s.sid=m.sid \land m.major='CS' \land s.sid=t.sid \land t.bookno=c.citedbookno \\ \land b1.price>b2.price \land c.citedbookno=b1.bookno \land c.bookno=b2.bookno}} (S \times T \times C \times B1 \times B2 \times M))$$

Optimized query:

$$\pi_{sid,sname}(S \ltimes \pi_{m.sid}\left(\sigma_{m.major='CS'} M\right) \ltimes \pi_{t.sid}(T \ltimes_{t.bookno=c.citedbookno} \pi_{c.citedbookno}(C \ltimes (\pi_{bookno,price}(B2) \bowtie_{price<price1} \rho_{bookno,price \rightarrow citedbookno,price1}(\pi_{bookno,price}(B1))))))$$

QUESTION 4b)

```
select distinct s.sid, m.major
from student s, major m
where s.sid = m.sid and m.major <> 'CS' and
s.sid = SOME (select t.sid
from buys t, book b
where t.bookno = b.bookno and b.price > 30) and
s.sid not in (select t.sid
from buys t, book b
where t.bookno = b.bookno and b.price > 50);
```

Translated query:

$$\pi_{sid,major}(\pi_{s1.sid,s1.sname,m.sid,m.major}\left(\sigma_{s1.sid=m.sid \wedge m.major<>'CS' \wedge s1.sid=t1.sid \wedge t1.bookno=b1.bookno \wedge b1.price>30}(S1 \times M \times T1 \times B1)\right)$$
$$- \pi_{s.sid,s.sname,m.sid,m.major}(\sigma_{s.sid=m.sid \wedge m.major<>'CS' \wedge s.sid=t.sid \wedge t.bookno=b.bookno \wedge b.price>50}(S \times M \times T \times B)))$$

Optimized query:

$$\pi_{sid,major}\left(S \bowtie \left(\left(\sigma_{m.major<>'CS'}(M)\right) \ltimes \pi_{t.sid}\left(T \ltimes \pi_{b.bookno}\left(\sigma_{b.price>30}(B)\right)\right)\right)\right)$$
$$- \pi_{sid,major}\left(S \bowtie \left(\left(\sigma_{m.major<>'CS'} M\right) \ltimes \pi_{t.sid}\left(T \ltimes \pi_{b.bookno}\left(\sigma_{b.price>50}(B)\right)\right)\right)\right)$$

QUESTION 4c)

select distinct t.sid, b.bookno
from buys t, book b
where t.bookno = b.bookno and
b.price <= ALL (select b1.price
from buys t1, book b1
where t1.bookno = b1.bookno and t1.sid = t.sid);

Translated query:

$$\pi_{t.sid,b.bookno}((T \bowtie B)$$
$$- (\pi_{t.sid,t.bookno,b.bookno,b.title,b.price}(\sigma_{t1.bookno=b1.bookno \wedge t1.sid=t.sid \wedge t.bookno=b.bookno \wedge \neg(b.price \le b1.price)}(T \times B \times T1 \times B1))))$$

Optimized query:

$$T - (T \ltimes \pi_{b1.sid,b.bookno}(\pi_{bookno,price}(B) \bowtie_{\neg(price \le price1)} \rho_{price \rightarrow price1}(\pi_{sid,price}(\pi_{bookno,price}(B1) \bowtie (T1)))))$$

QUESTION 4d)

select b.bookno
from book b
where not exists (select s.sid
from student s
where s.sid in (select m.sid from major m
where m.major = 'CS'
INTERSECT
select m.sid from major
where m.major = 'Math') and
s.sid not in (select t.sid
from buys t
where t.bookno = b.bookno));

Translated query:

$$\pi\_bookno(B$$
$$-\pi_{bookno,title,price}((\pi_{s1.sid,s1.sname,b1.bookno,b1.title,b1.price}\left(\sigma_{s1.sid=m1.sid \wedge m1.major='CS'}(B1 \times S1 \times M1)\right) \cap \pi_{s2.sid,s2.sname,b2.bookno,b2.title,b2.price}\left(\sigma_{s2.sid=m2.sid \wedge m2.major='Math'}(B2 \times S2 \times M2)\right)\right)$$
$$-\pi_{s3.sid,s3.sname,b3.bookno,b3.title,b3.price}(\sigma_{s3.sid=t.sid \wedge t.bookno=b3.bookno}(B3 \times S3 \times T))))$$

Optimized query:

$$\pi_{bookno}(B) - \pi_{bookno}\left(\pi_{bookno}(B1) \times \left(\left(\pi_{sid}(S1) \ltimes \pi_{m1.sid}\left(\sigma_{m1.major='CS'}(M1)\right)\right) \bigcap (\pi_{sid}(S2) \ltimes \pi_{m2.sid}\left(\sigma_{m2.major='CS'}(M2)\right)\right)\right) - \rho_{bookno,sid}(T))$$

QUESTION 5)

makerandomR	Q3	Q4
(100,100,1000)	11.396 ms	1.235 ms
(500,500,25000)	5222.318 ms	17.319 ms
(1000,1000,100000)	86380.686 ms	59.736 ms
(2000,2000,400000)	----	282.519 ms
(5000,5000,2500000)	----	260.194 ms

Below is the query Q3:

```
select distinct r1.a
from R r1, R r2, R r3
where r1.b = r2.a and r2.b = r3.a;
```

Below is the query Q4:

```
select distinct r1.a
from R r1 join (select  distinct r2.a from R r2 join (select distinct a from R) r3 on (r2.b = r3.a)) r2 on (r1.b = r2.a );

explain analyze select distinct r1.a
from R r1 join (select  distinct r2.a from R r2 join (select distinct a from R) r3 on (r2.b = r3.a)) r2 on (r1.b = r2.a );
```

Obervation:

From the above, we observe significant reduction in execution time upon optimization. The original query Q3 is more than 100 times slower than the query Q4. The execution time for Q3 is 4690.443 ms which is 100 times that of Q4 (44.077 ms).

The optimized query Q4 runs $O(|R|)$ times while the original query Q3 has a cubic execution time $O(|R|^3)$. This is because each join involves hashing to buckets of the join attribute values. The unnecessary attributes are removed. This means when R1 joins with R2 , the attributes of R3 are not involved , even the attributes of R2 that are not involved in the join with R1 are removed. Everytime the distinct values are taken, hence this also reduces the execution time significantly. There are two joins – Between R2 and R3 which takes $O(|R|)$, another Between R1 and R2 which takes $O(|R|)$ as well.

For higher sized data |R|>30000 the explain analyze for Q3 takes too much time. For lower sizes of R, the execution time for Q3 is almost same as Q1 i.e. the optimization makes a huge difference for only big data.

QUESTION 6)

makerandomR	makerandomS	Q5	Q6
(100,100,1000)	(100,100)	1.797 ms	0.513 ms
(500,500,25000)	(500,500)	10.259 ms	4.747 ms
(1000,1000,100000)	(1000,1000)	13.496 ms	17.035 ms
(2000,2000,400000)	(2000,2000)	54.209 ms	63.145 ms
(5000,5000,2500000)	(5000,5000)	416.378 ms	411.062 ms

Below is Q5:

```
select ra.a
from Ra ra where not exists (select r.b
from R r where r.a = ra.a and r.b not in (select s.b from S s));
```

Below is Q6:

```
with
S as (select distinct b from S)
Select distinct a from Ra
except
Select distinct a from R natural join (Select distinct b from R except select b from S) q;
```

Observation:

The optimized query Q6 runs faster than Q5 although for some higher datasets it runs somewhat same time as the original query Q5. Guess is that the POSTGRES is able to optimize this problem. In query Q6, we use distinct values of b from S, we remove the join of R with Ra in inner query since R.a is subset of Ra, we do not join R with S for 'not in' since attributes of S has only b. But we do not replace Ra with R in outer most query since entries not in R but in Ra can still be part of the output.

QUESTION 7)

makerandomR	makerandomS	Q7	Q8
(100,100,1000)	(100,100)	6.646 ms	6.463 ms
(500,500,25000)	(500,500)	546.869 ms	192.024 ms
(1000,1000,100000)	(1000,1000)	4366.922 ms	823.606 ms

(2000,2000,400000)	(2000,2000)	48111.768 ms	4232.733 ms
(5000,5000,2500000)	(5000,5000)	42495.532 ms	3571.576 ms

Below is Q7:

```
select ra.a
from Ra ra
where not exists (select s.b
from S s
where s.b not in (select r.b
from R r
where r.a = ra.a));
```

Below is Q8:

```
with
RRaa as (select distinct a from R),
S as (select distinct b from S)
select a from RRaa
except
select distinct q.a from (select rraa.a,s.b from RRaa rraa,S s
except
select a,b from R r natural join S s) q;
```

Observation:

The execution time of Q8 is much less than Q7. This is because the run time of Q7 is cubic $O(|R|^3)$ while that of Q7 is quadratic $O(|R|^2)$. We use distinct a from R instead of Ra since the query will not output the values of attribute a not present in relation R (since S is not null set). It will appear in the cross join of Ra with S in inner query and will be exempt from the final output query and since R.a is subset of Ra.a we remove the natural join with Ra. Also, we use distinct values of b in S hence we get higher execution time.

Secondly, with increase in size of data, the run time also increases exponentially and the difference in execution time between the two queries is more prominent. Q8 runs significantly faster than Q7 for higher data sizes of more than 100000.

Furthermore the query for ALL (Q7, Q8) runs almost 10 times slower than query for ONLY (Q5, Q6). This is due to the cross join between relations R and S in ALL which leads to quadratic time whereas in case of ONLY it is linear time only.

QUESTION 8)

makerandomR	makerandomS	Q9	Efficient Q9
(100,100,1000)	(100,100)	7.586 ms	1.746 ms
(500,500,25000)	(500,500)	38.108 ms	25.587 ms
(1000,1000,100000)	(1000,1000)	170.772 ms	157.324 ms
(2000,2000,400000)	(2000,2000)	1208.839 ms	1059.147 ms

(5000,5000,2500000)	(5000,5000)	925.599 ms	881.522 ms

Below is the Q9 query:
with NestedR as (select r.a, array_agg(r.b order by 1) as Bs
from R r
group by (r.a)
union
select q.a, '{}' as Bs
from (select a from ra
except
select distinct a from R) q),
SetS as (select array(select s.b from S s order by 1) as Bs)
select r.a
from NestedR r, SetS s
where r.Bs <@ s.Bs;

How does it work:
The ONLY means that all of the values of attribute b for output Ra.a must come from S. Hence we do r.Bs <@ s.Bs. But the output of our query is Ra and not R and there are values of a in Ra that are not in R.a. These must also be included into the output since the b for Ra.a is null set {} which is a subset of any set and hence is a subset of set S as well. Hence, we add "select ra.a, '{}' as Bs
from (select a from ra
except
select distinct a from R) q)" to the set from R. The set from R is "select r.a, array_agg(r.b order by 1) as Bs
from R r
group by (r.a)" which creates a set (array literally) of values of attribute b for every r.a value.
eg.

```
a  |                    bs
-----+------------------------------------------------------
   1 | {73,52,7,56,58,49,99,64,24,77,20,54,55,36}
   2 | {95,45,18,76,38,3,53}
```

It is then checked for each value of a, if its set b in Ra is subset of b in S , if yes then the corresponding value of a is output. The values of a not in R but in Ra are output nevertheless.

In the efficient query we are doing the union with Ra.a not in R.a after the subset condition on R and S. This avoids unnecessary comparison of these Ra.a values with S which we know will be part of the output query.

Comparison with Q5 and Q6:
Q5 and Q6 run much faster than Q9 including the efficient query Q9.
This could be because the subset condition r.Bs <@ s.Bs scans through each element in S to ensure if r.Bs is part of s.Bs. This includes duplicates as well hence it is clearly inefficient than our optimized and even our original query. Hence, the query Q9 is quadratic while the optimized query is linear.

QUESTION 9)

makerandomR	makerandomS	Q10	Efficient Q10
(100,100,1000)	(100,100)	2.058 ms	1.400 ms
(500,500,25000)	(500,500)	19.102 ms	14.023 ms
(1000,1000,100000)	(1000,1000)	90.873 ms	72.239 ms
(2000,2000,400000)	(2000,2000)	334.584 ms	301.785 ms
(5000,5000,2500000)	(5000,5000)	300.866 ms	232.639 ms

makerandomR	makerandomS	Q8	Q10
(100,100,1000)	(100,100)	6.463 ms	2.058 ms
(500,500,25000)	(500,500)	192.024 ms	19.102 ms
(1000,1000,100000)	(1000,1000)	823.606 ms	90.873 ms
(2000,2000,400000)	(2000,2000)	4232.733 ms	334.584 ms
(5000,5000,2500000)	(5000,5000)	3571.576 ms	300.866 ms

Below is the Q10 query:

```
with NestedR as (select r.a, array_agg(r.b order by 1) as Bs
from R r
group by (r.a)
union
select q.a, '{}' as Bs
from (select a from ra
except
select distinct a from R) q),
SetS as (select array(select s.b from S s order by 1) as Bs)
select r.a
from NestedR r, SetS s
where s.Bs <@ r.Bs;
```

How it works:

The ALL query requires that all the values b in set S must be present in set b of output Ra.a. The query "select r.a, array_agg(r.b order by 1) as Bs
from R r
group by (r.a)" returns values of a and its corresponding set b from relation R. The values of a in Ra that are not in R have a null set {} as set b. Hence, the query "select q.a, '{}' as Bs
from (select a from ra
except

select distinct a from R) q),". Though we are considering the Ra in output , the values of a in Ra but not in R.a will not be part of the output since the set b for those a values is a {} null set which is not a superset of S.b (if S is not null), hence it is removed in the efficient query mentioned for Q10.

Finally, for every a in Ra, it is checked if S.b is a subset of a's corresponding set b, if yes then a is output.

Comparison with Q7 and Q8:
Q10 runs much more efficiently than Q7 or Q8. This is because it scans through all values of R instead of S which is a much smaller set than S. Hence it is more efficient.

```
\qecho 'QUESTION 10'
--union (GIVEN)
create or replace function setunion(A anyarray, B anyarray) returns anyarray as
$$
with
Aset as (select UNNEST(A)),
Bset as (select UNNEST(B))
select array( (select * from Aset) union (select * from Bset) order by 1);
$$ language sql;

--intersection
create or replace function setintersection(A anyarray, B anyarray) returns anyarray as
$$
with
Aset as (select UNNEST(A)),
Bset as (select UNNEST(B))
select array( (select * from Aset) intersect (select * from Bset) order by 1);
$$ language sql;

--setdifference
create or replace function setdifference(A anyarray, B anyarray) returns anyarray as
$$
with
Aset as (select UNNEST(A)),
Bset as (select UNNEST(B))
select array( (select * from Aset) except (select * from Bset) order by 1);
$$ language sql;

--memberof
create or replace function memberof(x anyelement, S anyarray)
returns boolean as
$$
select x = SOME(S)
$$ language sql;


\qecho 'QUESTION 11'
```

```
--GIVEN:
create or replace view student_books as
select s.sid, array(select t.bookno
from buys t
where t.sid = s.sid order by bookno) as books
from student s order by sid;

create or replace view book_students as
select b.bookno, array(select sid
from buys
where bookno=b.bookno order by sid) as sids
from book b order by 1;

create or replace view book_citedbooks as
select b.bookno, array(select citedbookno
from cites
where bookno=b.bookno) as citedbook
from book b order by 1;

create or replace view book_citingbooks as
select b.bookno as citedbook,array(select bookno
from cites
where citedbookno=b.bookno) as bookno
from book b order by 1;

create or replace view major_Students as
select m.major, array_agg(sid) as sid
from major m
group by m.major order by 1;

create or replace view student_majors as
select s.sid, array(select m.major
from major m where m.sid=s.sid) as major
from student s order by 1;

\qecho 'QUESTION 12a'
select citedbook as bookno from book_citingbooks where bookno && (select array(select bookno from
book where price<50));

\qecho 'QUESTION 12b'
select distinct bookno,title from student_books s join book b on (memberof(b.bookno,s.books)) where
memberof(s.sid,(select array(select sid from student_majors where array['Math','CS'] <@ major))) order
by 1;

\qecho 'QUESTION 12c'
select citedbook as bookno from book_citingbooks where cardinality(bookno)=1;

\qecho 'QUESTION 12d'
```

```
select sid from student_books where (select array(select bookno from book where price >50))<@ books;

\qecho 'QUESTION 12e'
-- considers students who did not buy any books too
select sid from student_books where not (select array(select bookno from book where price >50)) &&
books;

\qecho 'QUESTION 12f'
-- considers students who did not buy any books too
select sid from student_books where books<@(select array(select bookno from book where price >50));

\qecho 'QUESTION 12g'
select sm.sid from student_majors sm join student_books sb on (sm.sid=sb.sid) where
memberof('CS',sm.major) and not( sb.books && array(select bookno from book_students where sids &&
(select sid from major_Students where major='Physics')));

\qecho 'QUESTION 12h'
select sb.sid as s,bci.citedbook as b from student_books sb join book_citingbooks bci on ( not
(sb.books<@bci.bookno));

\qecho 'QUESTION 12i'
-- considers students who did not buy any books too
select sb.sid as s,bci.citedbook as b from student_books sb join book_citingbooks bci on (
sb.books<@bci.bookno);

\qecho 'QUESTION 12j'
select sb1.sid as s1,sb2.sid as s2 from student_books sb1 join student_books sb2 on (sb1.books <@
sb2.books and sb2.books <@ sb1.books and sb1.sid !=sb2.sid);

\qecho 'QUESTION 12k'
select sb1.sid as s1,sb2.sid as s2 from student_books sb1 join student_books sb2 on
(cardinality(sb1.books) = cardinality(sb2.books) and sb1.sid !=sb2.sid);

\qecho 'QUESTION 12l'
select bookno from book_citedbooks bcd where cardinality(setdifference(array(select distinct bookno
from book),bcd.citedbook))=2;
```