Experiment No. 5

Environment: Microsoft Windows
Tools/ Language: Oracle

OBJECTIVE: To implement the concept of subqueries & set theoretic functions.

Theory:

SET THEORETIC FUNCTIONS

The SQL UNION Operator

The UNION operator is used to combine the result-set of two or more SELECT statements.

Notice that each SELECT statement within the UNION must have the same number of columns. The columns must also have similar data types. Also, the columns in each SELECT statement must be in the same order.

SQL UNION Syntax:

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

Note: The UNION operator selects only distinct values by default. To allow duplicate values, use the ALL keyword with UNION.

SQL UNION ALL Syntax:

```
SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;
```

PS: The column names in the result-set of a UNION are usually equal to the column names in the first SELECT statement in the UNION.

SQL INTERSECT Operator

The following statement combines the results with the INTERSECT operator, which returns only those rows returned by both queries:

```
SELECT column_name(s) FROM table1
INTERSECT
SELECT column_name(s) FROM table2;
```

SQL Minus Operator

The following statement combines results with the MINUS operator, which returns only unique rows returned by the first query but not by the second:

```
SELECT column_name(s) FROM table1
MINUS
SELECT column_name(s) FROM table2;
```

Sub Queries :--> A sub query is a form of an SQL statement. It is also term as nested query. The statement containing a sub query is called a parent. The parent statement uses the row return by the sub query.

It can also use for following:-

- To insert records in a target tables.
- To create tables and insert records in the table created.
- To update records in a target table.
- To create views.
- To provide values for condition in WHERE, HAVING, IN and so on used with SELECT, UPDATE and DELETE statement.

Examples:

Q: Find the students name and sid whose GPA is equal to GPA of Amy & Craig?

A:
```
select sID, sName from student
where GPA IN(select GPA from student
where sName in('Amy', 'Craig'));
```

OUTPUT:

SID	SNAME
654	Amy
876	Irene
456	Doris
123	Amy
345	Craig
543	Craig
789	Gary

Q: Find the students name and sid whose GPA is equal to GPA of Amy & Craig but result should not include Amy and Craig?

A:
```
Select sID, sName
From Student
Where GPA IN (      Select GPA
                    From Student
                    Where sName IN ('Amy', 'Craig'))
And sName NOT IN ('Amy', 'Craig ');
```

Q: Find the name of student with their GPA and Sid whose high school size is not equal to high school size of Doris?

A:
```
select sID, sName, GPA
from student
where sizeHS NOT IN(        select sizeHS
                                from student
                                where sName='Doris');
```

OUTPUT: (Write any four only)

SID	SNAME	GPA
234	Bob	3.6
345	Craig	3.5
567	Edward	2.9
678	Fay	3.8
789	Gary	3.4
987	Helen	3.7
876	Irene	3.9
765	Jay	2.9
543	Craig	3.4

Q: Find the name and Sid of student who apply to same college where Irene has applied ?

A:
```
select sID, sName from student
where sID in(select sID from apply where Cname in(select Cname
from apply where sID in(select sID from student where sName='Irene')));
```

OUTPUT:

SID	SNAME
765	Jay
876	Irene
987	Helen
678	Fay
123	Amy
543	Craig
345	Craig

Q: Give the names of colleges who receive same number of applications or more than Stanford had received?

A:
```
select cName from apply group by cName
having count(sID)>=(select count(sID) from apply
where cName='Stanford')
```

OUTPUT:

CNAME
Cornell
Stanford

Q: Create a table having two columns college name and number of applications they receive?
A:

```
Create table collegeinfo
 (cName varchar(10), no_of_app number(5));

insert into collegeinfo
        selectcName, count(sID) from apply
        group by cName;
```

Q: Create table and load data from already created table
Create a Table having Student id, Student Name, GPA, College Name they applied to and decision i.e.

sID	sName	GPA	cName	decision

A: Create table ApplicationDetail
AS(SELECT S.sID, sName, GPA, cName, decision
FROM STUDENT S, Apply A where S.sID=A.sID);

Q: Update Decision to N for all students who applied to same major as sID 876 have applied.
A:

```
update APPLY
set decision='N'
WHERE major IN (    SELECT major
                    FROM APPLY
                    WHEREsID = 876);
```

Q: Delete the Application of student who have applied to same college as of 'Irene'
A:

```
delete from Apply
        where cName IN (Select cName from Apply
                        where sID IN (Select sID from Student
                                      where sName = 'Irene'));
```

Q:Find IDs and names of student who share same name with another student.
A:

```
Select *
From Student S1
Where exists ( Select * From student S2
                Where S1.sName=S2.sName)
```

Running above query will give WRONG result as every student of outer query will match (or each S1 student having same name as S2), we need ensure we are talking about **different person**. So,

```
Select *
From Student S1
Where exists ( Select * From student S2
               Where S1.sName=S2.sName
               and S1.sID <> S2.sID)
```

Q: Find IDs of students that applied to all colleges.

Insert these to get student who applied to all college.

```
insert into Apply values (123, 'MIT', 'CSE', 'N');
insert into Apply values (123, 'Harvard', 'CSE', 'N');
```

A:
```
Select s.sID, s.sName
From Student S
Where NOT Exists  ( (Select C.cName from College C)
                    MINUS
                    (Select A.cName from Apply A where A.sID=S.sID));
```

Kindly delete the above two rows

```
Delete from apply where major='CSE';
```

Practical Assignment - 5

Department: Computer Engineering & Applications

Course: B.Tech. (CSE)

Subject: Database Management System Lab (CSE3083)

Year: 2nd **Semester:** 3rd

SQL Script for this Experiment

```
BEGIN
 FOR cur_rec IN (SELECT object_name, object_type
                 FROM user_objects
                WHERE object_type IN
                      ('TABLE',
                       'VIEW',
                       'PACKAGE',
                       'PROCEDURE',
                       'FUNCTION',
                       'SEQUENCE'
                      ))
 LOOP
   BEGIN
     IF cur_rec.object_type = 'TABLE'
     THEN
       EXECUTE IMMEDIATE    'DROP '
                    || cur_rec.object_type
                    || ' "'
                    || cur_rec.object_name
                    || '" CASCADE CONSTRAINTS';
     ELSE
       EXECUTE IMMEDIATE    'DROP '
                    || cur_rec.object_type
                    || ' "'
                    || cur_rec.object_name
                    || '"';
     END IF;
   EXCEPTION
     WHEN OTHERS
     THEN
       DBMS_OUTPUT.put_line (   'FAILED: DROP '
                         || cur_rec.object_type
                         || ' "'
                         || cur_rec.object_name
                         || '"'
                        );
   END;
 END LOOP;
END;
/
commit;
```

```
drop table  College;
drop table  Student;
drop table  Apply;
create table College(cName varchar2(10) primary key, state
varchar2(10), enrollment int);
create table Student(sID int primary key, sName varchar2(10), GPA
real, sizeHS int);
create table Apply(sID int, cName varchar2(10), major varchar2(20),
decision char(1), primary key(sID, major, cName), constraint sID_fk
Foreign key(sID) references Student, constraint cName_fk Foreign
key(cName) references College);

delete from Student;
delete from College;
delete from Apply;

insert into Student values (123, 'Amy', 3.9, 1000);
insert into Student values (234, 'Bob', 3.6, 1500);
insert into Student values (345, 'Craig', 3.5, 500);
insert into Student values (456, 'Doris', 3.9, 1000);
insert into Student values (567, 'Edward', 2.9, 2000);
insert into Student values (678, 'Fay', 3.8, 200);
insert into Student values (789, 'Gary', 3.4, 800);
insert into Student values (987, 'Helen', 3.7, 800);
insert into Student values (876, 'Irene', 3.9, 400);
insert into Student values (765, 'Jay', 2.9, 1500);
insert into Student values (654, 'Amy', 3.9, 1000);
insert into Student values (543, 'Craig', 3.4, 2000);
insert into College values ('Stanford', 'CA', 15000);
insert into College values ('Berkeley', 'CA', 36000);
insert into College values ('MIT', 'MA', 10000);
insert into College values ('Cornell', 'NY', 21000);
insert into College values ('Harvard', 'MA', 50040);
insert into Apply values (123, 'Stanford', 'CS', 'Y');
insert into Apply values (123, 'Stanford', 'EE', 'N');
insert into Apply values (123, 'Berkeley', 'CS', 'Y');
insert into Apply values (123, 'Cornell', 'EE', 'Y');
insert into Apply values (234, 'Berkeley', 'biology', 'N');
insert into Apply values (345, 'MIT', 'bioengineering', 'Y');
insert into Apply values (345, 'Cornell', 'bioengineering', 'N');
insert into Apply values (345, 'Cornell', 'CS', 'Y');
insert into Apply values (345, 'Cornell', 'EE', 'N');
insert into Apply values (678, 'Stanford', 'history', 'Y');
insert into Apply values (987, 'Stanford', 'CS', 'Y');
insert into Apply values (987, 'Berkeley', 'CS', 'Y');
insert into Apply values (876, 'Stanford', 'CS', 'N');
insert into Apply values (876, 'MIT', 'biology', 'Y');
insert into Apply values (876, 'MIT', 'marine biology', 'N');
insert into Apply values (765, 'Stanford', 'history', 'Y');
insert into Apply values (765, 'Cornell', 'history', 'N');
insert into Apply values (765, 'Cornell', 'psychology', 'Y');
insert into Apply values (543, 'MIT', 'CS', 'N');
commit;
```

Student

sID	sName	GPA	sizeHS
123	Amy	3.9	1000
234	Bob	3.6	1500
345	Craig	3.5	500
456	Doris	3.9	1000
567	Edward	2.9	2000
678	Fay	3.8	200
789	Gary	3.4	800
987	Helen	3.7	800
876	Irene	3.9	400
765	Jay	2.9	1500
654	Amy	3.9	1000
543	Craig	3.4	2000

College

cName	state	enrollment
Stanford	CA	15000
Berkeley	CA	36000
MIT	MA	10000
Cornell	NY	21000
Harvard	MA	50040

Apply

sID	cName	major	decision
123	Stanford	CS	Y
123	Stanford	EE	N
123	Berkeley	CS	Y
123	Cornell	EE	Y
234	Berkeley	biology	N
345	MIT	bioengineering	Y
345	Cornell	bioengineering	N
345	Cornell	CS	Y
345	Cornell	EE	N
678	Stanford	history	Y
987	Stanford	CS	Y
987	Berkeley	CS	Y
876	Stanford	CS	N
876	MIT	biology	Y
876	MIT	marine biology	N
765	Stanford	history	Y
765	Cornell	history	N
765	Cornell	psychology	Y
543	MIT	CS	N

Write SQL queries for the following:

1. IDs and names of students who have applied to major in CS at some college.
2. Find ID and name of student having same high school size as *Jay*.
3. Find ID and name of student having same high school size as *Jay* but result should not include *Jay*.
4. Find the name of student with their GPA and Sid whose GPA not equal to GPA of Irene?
5. Find college where any student having their name started from J have applied?
6. Find all different major where *Irene* has applied?
7. Find IDs of student and major who applied in any of major *Irene* had applied to?
8. Find IDs of student and major who applied in any of major *Irene* had applied to? But this time exclude *Irene* sID from the list.
9. Give the number of colleges *Jay* applied to? (*Remember count each college once no matter if he applied to same college twice with different major*)
10. Find sID of student who applied to more or same number of college where *Jay* has applied?
11. Find details of Students who applied to major CS but not applied to major EE? (*sID* 987, 876, 543 *should only be include in result*)
12. All colleges such that some other college is in same state. (*Cornell* should not be part of result as no other college in New York *Hint:* use exists)
13. Find the college with highest enrollment.
14. Find name of student having lowest GPA.

15. Find the most popular major.
16. Find sID, sName, sizeHS of all students NOT from smallest HS
17. Find the name of student who applies to *all* the colleges where sID 987 has applied? (*Hint:* see Query Find IDs of student applied to all colleges)

Run the following SQL script before next query

```
insert into Apply
select s1.sID, 'Berkeley', 'CSE', 'Y'
from student s1
where s1.sID IN (select s.sID from student s
MINUS
select a.sID from apply a where a.cName = 'Berkeley');
```

18. Find college where *all* the student have applied.

For next <u>three</u> queries you are expected to solve using without joins and IN operators

19. Find sid of student who have not applied to Stanford.
20. Find sid of Student that applied to both Stanford and Berkeley.
21. Give list of all names including all names of colleges and students.

22. Create a table `ApplicationInfo` having columns sID: `int`, sName: `varchar2(10)` and number_of_applications: `number(2)` they *filed*? Populate this table with appropriate data using insert command.
23. Create table `ApplicationData` and load with ID, name and college where they applied with state of college (*remember to include details of ALL students that have applied or not applied*) on runtime using single query.
24. Stanford decide not to take any student who have also applied to its rival Berkeley turn their application decision to N.
25. Delete applications that are filed to city 'New York'.

Pre Experiment Questions

1. What is Union Compatibility?
2. What are correlated sub queries?
3. What are nested queries?

Post Experiment Questions

1. When we use sub queries?
2. What will be different from join results and sub queries?