COEN 178 Intro to Database Systems Spring 2022

Lab 4 (75 pts)

Objectives: Learn

- **SQL queries using joins.**
- **Integrity Constraints**
- **Using Date and exists**

Part 1 (20 pts)

Step1: Create a folder structure called COEN178\labs\lab4.

Create the following tables:

Table **Club** (**clubName** – a variable string of 25 characters, **fee** – an Integer)

Table **Members** (**id** – a variable string of 5 characters, **name** – a variable string of 20 characters, **clubName** – a variable string of 25 characters).

Insert the values given below into the tables (from a script file)

```
insert into Club values ('tennis',1000);
insert into Club values ('hockey',1200);
insert into Club values ('dancing',300);
insert into Club values ('videogames',200);
insert into Club values ('gymnastics',500);
insert into Club values ('polo',5000);
insert into Members values ('m1','smith','tennis');
insert into Members values ('m2','jones','tennis');
insert into Members values ('m3','shiner','dancing');
insert into Members values ('m4','winger','dancing');
insert into Members values ('m5','toner','videogames');
insert into Members values ('m6','clark','videogames');
insert into Members values ('m22','mason','rockclimbing');
```

In the next 4 exercises, you will learn how to use the different joins in SQL.

Exercise 1.1 (10 pts)

Show the names of clubs and their fees, only if the club has members.

Approach 1 using Natural Join:

Run the query:

```
Select clubname,fee from Members Natural Join Club;
```

Q1) How many rows are displayed?

A) Execute the query to eliminate the duplicates.

Approach 2 using Inner Join:

Run the query:

```
Select members.clubname,fee from Members INNER JOIN Club
ON Members.clubname = Club.clubname;
```

B) Rewrite the query to eliminate duplicates.

Q2) Which approach do you like better? Give your reasons.

Exercise 1.2 (5 pts)

Show the names of members, their clubnames and the fees (if that information is available).

```
Select name,members.clubname,fee from Members Left Join Club
ON Members.clubname = Club.clubname;
```

Q1) Are all the members shown?

Q2) Are fees for all clubs shown?

Exercise 1.3 (5 pts)

Show the names of members, their clubnames and the fees (if that information is available) and also club names and fees that do not have any members.

```
Select name,club.clubname,fee from Members Right Join Club
ON Members.clubname = Club.clubname;
```

Q1) Are all the clubs shown?

Part 2 (30 pts)

In this part, you will learn to define integrity constraints, primary key and foreign key constraints, on tables.

Exercise 2.1 (5 pts)

Create a table, **AlphaCoEmp** as follows.

```
CREATE TABLE AlphaCoEmp(Name VARCHAR(25) Primary Key, Title
VARCHAR(20) DEFAULT NULL, Salary Number(10,2) DEFAULT 0);
```

Inserting values:

You will load the table, **AlphaCoEmp**, with employee last names from the **Staff table** you created in Lab1.

Run the query below to insert the last names from table, **Staff**.

```
INSERT INTO AlphaCoEmp (name) SELECT last from Staff;
```

Did you see any errors? The errors are because of the primary key constraint which is being violated by some duplicate last names in the Staff table.

A) How do you fix this query so that it loads unique last names from Staff?

(**Hint**: Think **distinct**, modify and rerun the query)

Q1) Did it run now? How many rows are created?

Q2) Do a Select * from AlphaCoEmp and check the results displayed. What was displayed for title and salary?

Exercise 2.2 (10 pts)

Let us define a table called **Emp_Work** as follows:

```
Create Table Emp_Work(name VARCHAR(25) Primary Key, Project
VARCHAR(20) default NULL,
Constraint FK_AlphaCo
Foreign Key (name) REFERENCES AlphaCoEmp(name));
```

Note the foreign key constraint defined in this table with a name.

Let us insert employee names that start with **A** or **G** or **S** from the **AlphaCoEmp** table into **Emp_Work** table. We will use **regular expressions** to grab names that start with A or G or S.

```
insert into Emp_Work(name)
(Select Name from AlphaCoEmp where
REGEXP_LIKE(name,'(^[ags])','i'));
```

Study the Regex that describes a pattern where the string starts with (^) a [ags] (a or g or s).

Q1) What is the 'i' for?

Regex – A reference

Run the query.

Q2) How many rows are inserted into Emp_Work table?

Run a query to display the names from Emp_Work table.

Display the list of names in AlphaCoEmp. Pick a name from the displayed list and delete it from the table, AlphaCoEmp**.**

```
Delete from AlphaCoEmp
```

```
Where name='Smith';
```

Q3) Did your deletion work? Explain why it did not work.

Exercise 2.3 (10 pts)

Let us change the **Emp_Work** table definition such that when a primary key that has a reference from a foreign key is deleted, it should delete the row with that foreign key as well.

We will use the **alter Table** command and define the constraint, on **delete cascade**.

Remember how you defined this table earlier.

```
Create Table Emp_Work(name VARCHAR(20) Primary Key, Project
VARCHAR(20) default NULL,

Constraint FK_AlphaCo

Foreign Key (name) REFERENCES AlphaCoEmp(Name));
```

Now, we need to drop this constraint **FK_AlphaCo** and add the constraint with **delete cascade**. We drop and add the new constraint with **Alter Table** statement.

Run this statement to drop the constraint.

```
Alter table Emp_Work

drop constraint FK_AlphaCo;
```

Add the new constraint as follows:

```
Alter table Emp_Work

add constraint FK_AlphaCo

FOREIGN KEY (name)

references AlphaCoEmp(name)

on delete cascade;
```

Q1) Is the table altered?

Now try to delete the name you tried earlier, from AlphaCoEmp table.

Example:

```
DELETE from AlphaCoEmp
Where name='Smith';
```

Q2) Did you succeed this time?

Q3) Check if the name "Smith" is in the Emp_Work table. It should have been deleted automatically if our constraint worked. Did it work?

Part 3 (25 pts)

In this part, you will write queries using **SQL Date type** and **subqueries using exists**.

Create the following tables:

```
create table L_Person(id Integer Primary key, name
varchar(30),age Integer);
create table L_Member(id Integer, clubname varchar(25),
joinedDate Date);
```

Insert the values below into L_Person table.

```
insert into L_Person values(10,'smith',25);
insert into L_Person values(15,'jones',20);
insert into L_Person values(16,'grey',30);
insert into L_Person values(17,'green',43);
insert into L_Person values(18,'sander',50);
insert into L_Person values(19,'brown',62);
insert into L_Person values(25,'mart',21);
```

Insert the values below into L_Member table.

```
insert into L_Member values(10,'tennis',TO_DATE('March
11,2015','MONTH DD,YYYY'));
insert into L_Member values(15,'videogames',TO_DATE('April
14,2020','MONTH DD,YYYY'));
insert into L_Member values(17,'hockey',TO_DATE('October
20,2021','MONTH DD,YYYY'));

insert into L_Member values(19,'hockey',TO_DATE('September
20,2021','MONTH DD,YYYY'));
insert into L_Member values(25,'polo',TO_DATE('may
2,2019','MONTH DD,YYYY'));
```

Please note how the Date value is inserted into the table.

Now, you will run a few queries that involve the attribute of Date type.

Exercise 3.1

We will show the id of the member and the months of membership (months between the joinedDate of the member and the currentDate (SYSDATE) in the L_Member table.

We will use the MONTHS_BETWEEN function on Dates to get the no. of months between 2 dates.

```
Select id, MONTHS_BETWEEN(SYSDATE,JoinedDate) Months
from L_Member;
```

Q1) Are the months shown with fractional parts?

Write the query using the FLOOR(n) function(returns the largest INTEGER that is equal to or less than n).

```
Select id, FLOOR(MONTHS_BETWEEN(SYSDATE,JoinedDate)) Months
from L_Member;
```

What if we want to show the time between the current date and the joined date in Years and months? To convert to years, you should divide by 12 and to get the months, you should use the MOD function (modulo division).

a) **Write and execute the the SQL query (on L_MEMBER table) to show the results as below:**

ID	JOINEDDATE	CURRENTDATE	YEARS	MONTHS
10	11-MAR-15	23-APR-22	7	1
10	14-APR-20	23-APR-22	2	0
17	20-OCT-21	23-APR-22	0	6
19	20-SEP-21	23-APR-22	0	7
25	02-MAY-19	23-APR-22	2	11

b) The next query is to find the ID,clubname and the joined date of the person who has been a member for the longest time (from L_Member table). To get the longest time, we should compare the current date (SYSDATE with the joined date. The query is written as a query connected to a subquery.

Complete the query and execute it.

```
Select id,clubname,joinedDate from L_Member
where joinedDate <= ----(Select joinedDate from L_Member);
```

Exercise 3.2

In this exercise, you will use exists and not exists to write correlated subqueries.

The EXISTS operator is used to test for the existence of any record in a subquery.

The EXISTS operator returns TRUE if the subquery returns one or more records.

We will use the L_Person and L_Member tables for the following queries.

To find the persons(id, name) who are members of a club, we write the query using exists as follows:

```
Select id,name from L_Person
where exists (Select * from L_Member
      where L_Person.id = L_Member.id);
```

Note the highlighted condition in the subquery which correlates the outer and the inner queries. Execute the query and check if it returns the correct result.

Now modify the query as below and run it.

```
Select id,name from L_Person
where exists (Select * from L_Member);
```

Q1) Does it show the correct results (remember it should return only people that are members of a club)?

The way that exists works is that if the inner query returns a non-empty relation the exists is evaluated to true and the selected values in the outer query are included in the result.

Since the inner query “Select * from L_Member” returns a non-empty table, exists evaluates to true and every value in the outer query is included in the result. To make it work correctly, we need to include the condition, where L_Person.id = L_Member.id in the inner query.

a) Can you write a query to show **the persons (id,name) who are not members of a club (they are present in the L_Person table but not in the L_Member table)?**

Think of Not Exists (which returns a true if the table returned by the subquery is empty).

Exercise 3.3

Can you write the same queries in 3.2 (shown below) without using exists or not exists? You can use joins or any other operator that you have used before.

a) **Find the persons(id,name) who are members of a club.**

b) **Find the persons(id,name) who are not members of a club.**