COEN 178 Intro to Database Systems Spring 2020

Lab 2 (100 pts)

Objectives:

Learn writing SQL queries with

- Orderby,
- aggregate functions
- Joining tables,
-

Note: In this lab, you will create a few new tables in addition to the Employee data file you created and loaded last time.

Capturing the session and output:

You should capture the session (queries and output) into a file called, **lab2_session.lst**.

Capturing and Recording your Session

One way to capture your SQL session is to use the spool command that SQLPLUs provides to save query results to a file. At the SQL> prompt, type spool <filename>. For example,

SQL> spool foo;

This will create a foo.lst file in the current directory and will capture all input and output during your session at SQLPLUS (until you exit SQL).

You can terminate the capture with the command, spool off;

You can print/submit the foo.lst file for grading.

Oracle SQL Datatypes, Reference: http://www.ss64.com/orasyntax/datatypes.html

PART 1 (20 pts)

In this part, you will create a few tables and load them with given values and values of your choice.

Creating Tables

Assume that a customer can schedule a variety of home delivery services (grocery, movies, books etc).

Create the following tables based on the description given below **(primary keys are underlined**):

a) **Customer**: **custid** (a string of at most 5vcharacters),first name (a string of at most 10 characters) lastname (a string of at most 15 characters), **city** (a string of at most, 10 characters). Primary key: custid.

b) **DeliveryService**: **serviceid** (a string of at most, 10 characters), **item** (a string of 15 characters), **location** (a string of at most 15 characters), **servicefee** (a number with two points of precision after decimal point). Primary key: serviced.

c) **Schedule**: **service id, custid, day** (a string of up to 2 char and cannot be null). We will define a rule/constraint for the day attribute, that the day has to be Mon('m') day through Fri ('f'). The rule is defined using a CHECK clause and **in** to indicate the set of values that are allowed. Use **'m','t','w','r','f'** to represent the day values.

See the example table below and use the same approach to define the rule for day in Schedule table.

Example table (this is an example to show the usage of CHECK. Do not create this table):

```
Create table Trip (id varchar(5) PRIMARY KEY,
source varchar(15), dest varchar(15)
CHECK (dest in ('NY','Paris','London')));
```

Foreign key: **custid** referencing the table Customer.

Foreign key: **serviceid** referencing the table DeliveryService.

Inserting Tuples

Add the following tuples into the tables (use a script file to add the data).

Customer:

'c1','John','Smith','SJ'
'c2','Mary', 'Jones','SFO'
'a1','Vincent','Chen','SJ';
'a12','Greg', 'King','SJ';

'c7',’James’,'Bond','LA';
'x10',”Susan’,'Blogg','SFO';
Add a few more tuples of your choice.

DeliveryService:

'dsg1','grocery','SJ',25.0
'dsb1','books','SJ',10.0
'dsm2','movies','LA',10.0
'dby3','babygoods','SFO',15.0
'dsg2','grocery','SFO',20.0
'dg5','greengoods','SFO',30.0
Add a few more tuples of your choice.

Schedule

'dsg1','c1','m'
'dsg1','a12','w'
'dby3','x10','f'
'dg5','c1','r'
'dg5','c1','t'
'dg5','c32','t'

a) Did you successfully insert all the above tuples? If not, explain the reason for error. **(5 pts**)

Now, try to insert the following tuple into Schedule table.

'dsg2','c1','s'

b) Did you successfully insert all the above tuples? If not, explain the reason for error. **(5 pts)**

Add a few more tuples of your choice.

Part 2 (65 pts)

In this part, you will execute SQL commands from a script file.

Step 1: Create a folder structure called **COEN178/labs/lab2**.

Step 2: Create a text file called lab2_**queries1.sql.** This file will contain the SQL queries that you want to execute.

Write the SQL for the following queries (5 pts each) ***(You are free to refer to the class notes and examples posted).***

Show all the data in the tables, **Customer, DeliveryService** and **Schedule.**

a) Show the data in the Customer table as follows:

- **i.** custid, fullname, city, where fullname is the result of concatenating First name and last ('**||**' is the concatenation operator). Rename the column name as the fullname.

b) Show data in the Customer, sorted by Customer last name (use ***order by***).

c) Show data in the Schedule table, sorted by service id and then by customer id in descending order (use ***order by***).

d) Show service ids of delivery services that are not in the schedule table (think **set difference**).

e) The following query is given to show the names of customers who ordered a delivery service on Monday ('M'). Will it work? If not, fix it and show the query and results.

Select name from customer, schedule where day = 'M';

f) Show the last names of the customers that are scheduled delivery services. (What tables is the data coming from?)

g) Show the highest servicefee in Delivery Services (think of the aggregate function, max) renaming the result as highest_Servicefee.

h) Show the number of delivery services scheduled by day (think of aggregate function, **count()** and **group by (day)).**

i) The incomplete query below is given to show pairs of customer ids from the same city. Complete it.

```
Select A.custid,B.custid,A.city
from Customer A, Customer B
where A.city = B.city;
```

Will it work? If not fix it and run it again. Show the result.

j) Write a query to show the customers (who scheduled delivery services) where the customer city and location of the delivery services are in the same city.

Do the following two queries against staff_2010 table that you have created and loaded in lab1.

k) Write a query to show the minimum salary and maximum salary of staff members in **staff-_2010** table. Display appropriate headings.

Part 3 (15 pts)

Create two tables and load them with values given .

Creating Tables

Create the tables, L_EMP and L_DEPT using the DDL statements below:

Create table L_EMP (empNo Integer Primary Key, empname VARCHAR(10),deptId VARCHAR(5));

Create table L_DEPT (deptId VARCHAR(5) Primary Key, deptname VARCHAR(10));

Note: We have not defined any foreign key constraint in these tables.

Inserting Tuples

Add the following tuples into the tables (use a script file to add the data).

```
insert into L_EMP values(1,'smith','d1');
insert into L_EMP values(2,'jones','d2');
insert into L_EMP values(3,'wayne','d1');
```

```
insert into L_EMP values(4,'moor','d3');
insert into L_EMP values(5,'king','d1');
insert into L_EMP values(6,'chen','d1');
insert into L_EMP values(7,'winger','d3');
insert into L_DEPT values('d1','Research');
insert into L_DEPT values('d2','Devt');
insert into L_DEPT values('d3','Testing');
insert into L_DEPT values('d4','Advert');
```

Create a text file called lab2_**queries2.sql and include the queries for the exercises below.**

Using a simple Select query, show all the rows in the tables, L_EMP and L_DEPT.

Exercise 1 (15 pts)

Show the names of employees (from L_EMP) who work in the dept with name, 'testing'.

Approach 1: You can use L_EMP and L_Dept in the "from" clause, use a bridge (L_EMP.deptid = L_Dept.deptid) and retrieve the dept.name.

Approach 2: You will learn another approach in the next lab.