Practical 6-Part 1

Working with more than one table (Sub-queries) and manipulating tables with sub-queries

Last update: 30August 2022

Learning objectives

1. Use sub queries and retrieve data from tables.
2. Use correlated sub-queries.
3. Use sub-queries to manipulate tables:
   CREATE TABLE
   ALTER TABLE
   UPDATE TABLE
   DELETE TABLE

1. Setting-up

- Do this practical is DBS/Prac06 directory. If Prac06 directory is not there, create a one.
- **Task 1:** Same .sql files used in Practical 05 will be used in this task.
    - If you have downloaded them earlier, please use them. If not, download and copy the given nine sql files from the practical-05 link to your Prac06 directory.

- **Task 2:** Same .sql files used in Practical 02 will be used in this task.
    - If you have downloaded them earlier, please use them. If not, download and copy the three sql files from the practical-02 link to your Prac07 directory.
- Go to Prac05 directory, open a Terminal and connect to MySQL server using correct username and password.
- Use a suitable command record all the commands and query results to *Prac06Workings.out* file
- Open another Terminal while you are in the same directory and open a text file with name *Prac06Commands* in Vim (> vim *Prac06Commands*) or using any text editor you wish to use. Create your commands in this file before pasting them on MySQL prompt. Use a comment line with the question number before starting each task.
- Comment each command/ query you type.

2. Task 1: Sub-queries to retrieve data

This activity use several tables showing employees and their departments, projects etc. *Emp, Dept, Proj* and *Pworks* are the four tables involved, which hold information about employees, departments, projects and the employee's work on projects respectively. These tables are already used in Practical 05.
Follow the instructions given in Practical05 Task 2 and create all four tables and then insert the given set of values to created tables. Please refer practical 5 for description of the tables.

1. Use a select statement with a subquery to display the *names of all departments* that are responsible for *at least one project* that has already started.

2. Find the *name, start date* and *duration in months* of projects that have the earliest end date.

3. Produce a list of all female employees whose education level is higher than the average education level of all employees in their departments. Display *employee number, first name, last name* and *gender*.

4. Find the *lastname, job* and *salary* of employees who do not work on any project.

5. List *all employees* who have a higher education level than all designers. Display the *first name, last name* and *edlevel of the employee*. Assume that designers are indicated by job = 'Designer'.

6. Get the *names of departments* that are currently responsible for only one project each.

7. In the PROJ table, a row with a null value in the MAJPROJ column indicates that the project represented by that row is not a sub-project of any other.
   Display the *project number, name* and *end date* of all projects along with a remark of 'Sub-project' if the MAJPROJ value is not null and 'Not a sub-project' otherwise. Order the result by project end date.

8. Look at the SQL scripts which creates tables and identify the order in which these files are needed to be run to create the four given tables and fill them with sample data.

3. Task 2: Table modifications with sub-queries

In this task, you will first create the tables given below and populate them with data by completing Questions 3-7. Then you will use these tables to answer Questions 8-11. **The new tables will be created using the some other tables already exists.**

An employee can possibly work in more than one department; the *pct_time* field of the Works relation shows the percentage of time that a given employee works in a given department.

Emp2 (New Employee Table)

COL NAME	TYPE	SIZE	DESCRIPTION
eID	CHAR	6	Employee number, unique
eName	VARCHAR	17	Employee name
age	NUMBER		Age of the Employee
gender	CHAR	1	M=male, F=female
salary	DECIMAL	(8,2)	Annual salary

Dept2 (New Department Table)

COL NAME	TYPE	SIZE	DESCRIPTION
dID	CHAR	3	Department number, unique
dName	VARCHAR	36	Department name
Budget	DECIMAL	(12,2)	Department Budget
managerID	CHAR	6	Dept manager's employee no.

Work2 (Time work in departments Table)

COL NAME	TYPE	SIZE	NULL	DESCRIPTION
eID	CHAR	6		Employee number
dID	CHAR	3		Department number
Pct_time	INTEGER	3		Percentage of time

1. Create a new database “*NewCompany* and use it as the database for all activities in this task.

2. Create the *Emp* and *Dept* tables on the *NewCompany* database using the *create_tables.sql* script. Then populate the two tables *Emp* and *Dept* using *insemp.sql* and *insdept.sql* files. Look at the table structure and the data and be familiar with the tables.

3. Create the table *Emp2* from table *Emp*, using a statement of the form:

```
CREATE TABLE Emp2 <select_expression>;
```

The select_expression should select the appropriate table, and the results are stored in *Emp2*, instead of being displayed and discarded.
The *employee name* in *Emp2* should be in the format <lastname>,<first initial>. **For example, Eileen Henderson becomes Henderson,E.**
Note also that the table contains the age of the employee, not the birthdate, so you will need to calculate that.

Make sure that your new table has the same number of tuples as the original one.

- HINT 1: You probably want to test that the SELECT query gives the correct result before using it to populate a table.
- HINT 2: You will want to look up the **SUBSTRING** command.

4. Make *eid* the primary key of *Emp2*, by using an *ALTER TABLE* statement of the form:

```
ALTER TABLE <tablename> ADD PRIMARY KEY (attributenames);
```

5. Create the table Dept2 in your database using the following statement:

```
CREATE TABLE Dept2 (
      did CHAR(3),
      dname VARCHAR(36),
      budget DECIMAL(12,2),
      managerid CHAR(6),
      PRIMARY KEY(did),
      FOREIGN KEY (managerid) REFERENCES Emp2 (eid)
);
```

6. Insert data into *Dept2* from the *Emp* and *Dept* tables of Practical 2. Take the budget of each department as 20% more than the total salary of all employees in the department. Use an insert statement of the same form as above. Note the renaming of some attributes.

7. Use a CREATE TABLE statement with a subquery to create the table *Works2*, assuming that the percentage of time each employee works in his/her workdept is 100.

8. Add an attribute called '*since*' of TIMESTAMP type to Works2 table, using an ALTER TABLE statement. Give CURRENT_TIMESTAMP as the default value for this attribute. Display the rows of the table to check the changes.

HINT 3: Look up the TIMESTAMPE type and default values.

9. Delete all managers of departments from the Works2 table.

10. Update the *since* column of Works2 with the hiredate of each employee from the EMP table. In the update statement of the form:

```
UPDATE <table name>  AS <corr_var> SET <column name = expression>;
```

Use a correlated subquery as the expression to assign the corresponding hiredate of Emp table to each Works2 tuple.

11. Define (eid, did) as the primary key of Works2, using ALTER TABLE in the form:

```
ALTER TABLE <table name> ADD PRIMARY KEY (attribute name);
```

Remember to separate attribute names of the key by commas.

4. Submitting your work

Your Prac06 directory should have *Prac06Task1*, *Prac06Commands.sql* and *Prac06Workings.out* files together with all given .sql files.

Zip your Prac06 directory and upload it to Blackboard under 'Assessments/In Class Practical Submissions'

- Go to your DBS directory and type:

```
    > zip -r Prac06_<your student ID> Prac06
```

Check whether you have achieved learning outcomes:

I am confident that I can,

Use sub-queries appropriately to retrieve data from a multiple tables	✓
Use sub-queries with IN, ANY, ALL, EXISTS keywords	
Write correlated sub queries	
Use sub-queries to CREATE tables	
Use sub-queries to UPDATE tables	
Use sub-queries to ALTER tables	
Use sub-queries to DELETE tables	

Please refer lecture slides, reading materials, and online resources and attempt again, if all the learning outcomes were not achieved. Ask your tutor and get help if you need any clarification.

It's always a good practise to try to finish the practical of a particular week, before attempting the next practical worksheet as your work will be building upon the previous week's tasks.