# Practical 5

## Working with more than one table (JOINs)

*Last update: 23 August 2022*

## Learning objectives

1. Use different forms of joins such as, INNER JOIN, OUTER JOIN, NATURAL JOIN to query more than one table.
2. Sort and group results of queries based on different requirements (ORDER BY, GROUP BY).
3. Use HAVING clause to retrieve data with a condition when grouping data
4. Use CREATE DATABASE statement to create a new database.
5. Use DELETE TABLE statement with IF EXISTS clause.
6. Run multiple SQL scrip files using a single SQL file.

### Setting-up

- Do this practical is DBS/Prac05 directory. If Prac05 directory is not there, create a one.
- **Task 1** of this practical does not require MySQL. Write the answers in a file named *Prac05Task1.txt* file
- **Task 2 and Task 3**:
  - Download and copy the given nine sql files from the practical-5 link to your Prac05 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 *Prac05Workings.out* file
  - Open another Terminal while you are in the same directory and open a text file with name *Prac05Commands* in Vim (> vim Prac02Commands) or using any text editor you wish to use. Create your commands in this file before pasting them on MySQL prompt. Comment each command.

### 1. Task 1

The tables for the activities in task are not going to be created in MySQL. Write the answers in the *Prac05Task1.txt* file

1. Consider two relations *Actors(actor_name, address, age)*, and *Performs(movie_name, star, date)* where *Actors* store information of actors and *Performs* stores the main star actor of movies. *Performs* store the name of the movie, the name of the main star and the date that the movie released.
   A list of actors who have been a main star, and the movies that they have been the main star of is required.
   Decide on the most appropriate join to use for answering the above query that requires linking the two relations and what attributes to join on.

2. Consider two relations *Properties(address, lot, type)* and *Offers(customer, address, date, agent)* where *address* is the address of the property concerned and the *customer* and *agent* are the names of the people involved in the offer.

A list of the address and type of each property along with the date offers were made is required. A property may be shown multiple times in the results, if there are multiple offers, but should be shown even if no offers have been made. If there are no offers, the date field should be NULL.

Decide on the most appropriate join to use for answering the above query that requires linking the two relations and what attributes to join on.

3. Consider the two relations *Cruises(shipname, origin, destination, departure, duration, cost)* and *Ships(shipname, line, cabins, rating)*.
   The aim of a query is to list all cruises with the origin and destination ports, their cost and the star rating of the ship involved.
   Decide on the most appropriate join to use for the above query that requires linking the two relations and what attributes to join on.

## 2. Task 2: Creating the required tables

This activity use several tables showing employees and their departments, projects etc. *Emp, Dept, Proj* and *Pworks* are the four tables involved, which holds information about employees, departments, projects, and the employee's work on projects respectively. (*Emp* and *Dept* tables are already used in the practical -2).

SQL script files (.sql files) which contain the statements for creating the above tables and inserting sample data to the tables are given:
*createemp.sql :* creating Emp table.
*createdept.sql* : creating Dept table.
*createproj.sql* : creating Proj table.
*createpworks.sql*: creating Pworks table.
*Insemp.sql, insdept.sql, insproj.sql* and *inspworks.sql* scripts contains statements to insert sample data to *Emp, Dept, Proj* and *Pwork* tables respectively.

Descriptions of the tables are given below.

**Emp (Employee Table)**

| COL NAME | TYPE | SIZE | NULL | DESCRIPTION |
|---|---|---|---|---|
| empno | CHAR | 6 | no | Employee number, unique |
| firstname | VARCHAR | 12 | no | First name |
| midinit | CHAR | 1 | no | Middle initial |
| lastname | VARCHAR | 15 | no | Last name |
| workdept | CHAR | 3 | | Employee's dept number |
| phoneno | CHAR | 4 | | Employee's telephone number |
| hiredate | DATE | | | Date hired |
| job | CHAR | 8 | | Job held by employee |
| edlevel | INT | 2 | | No. of years of formal educ. |
| sex | CHAR | 1 | | M=male, F=female |
| birthdate | DATE | | | Date of birth |
| salary | DECIMAL | (8,2) | | Annual salary |
| bonus | DECIMAL | (8,2) | | Annual bonus |
| comm | DECIMAL | (8,2) | | Annual commission |

**Dept (Department Table)**

| COL NAME | TYPE | SIZE | NULL | DESCRIPTION |
|---|---|---|---|---|
| deptno | CHAR | 3 | no | Department number, unique |
| deptname | VARCHAR | 36 | no | Department name |
| mgrno | CHAR | 6 | | Dept manager's employee no. |
| admrdept | CHAR | 3 | no | ID of administrative dept |

**Proj (Project Table)**

| COL NAME | TYPE | SIZE | NULL | DESCRIPTION |
|---|---|---|---|---|
| projno | CHAR | 6 | no | Project number, unique |
| projname | VARCHAR | 24 | no | Project name |
| deptno | CHAR | 3 | no | Department number of responsible dept |
| respemp | CHAR | 6 | no | EMPNO of responsible employee |
| prstdate | DATE | | | Project start date |
| prendate | DATE | | | Project end date |
| majproj | CHAR | 6 | | PROJNO of the major project for a sub-project |

If a project is not a sub-project of any other project, then it has a MAJPROJ value of null.

**Pworks (Employees working on projects)**

| COL NAME | TYPE | SIZE | NULL | DESCRIPTION |
|---|---|---|---|---|
| empno | CHAR | 6 | no | Employee number |
| projno | CHAR | 6 | no | Project number |
| hours | SMALLINT | 6 | | Number of hours per week |

The combination of employee number and project number is unique for rows of this table.

4. 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.

- Note -1 : Note that these scripts first delete any existing tables of those names. Normally, a DELETE TABLE command will generate an error message if the table doesn't exist, however the scripts use the IF EXISTS option, which means that the command only executes if the given object already exists.

5. Create a new database called *company* to store the above four tables using the following command.

```
mysql> CREATE DATABASE company;
```

6. When there are many SQL scripts to run, all the commands can be added to a separate SQL script file in the correct order and that file can be run so that all the other SQL scripts are executed one by one.
   *rebuild.sql* script file contains command to create all the tables and inserting sample data. Open the *rebuild.sql* file and have a look to verify the order the commands are executed would be correct.
   Run the *rebuild.sql* file ( source the file) to create the tables and fill them with sample data in the newly created *company* database.
   Your database should now have all four tables in it with sample data.

7. Use suitable commands and verify all four tables are created and data is inserted.

## 3. Task 3 : Writing queries

Write and execute SQL SELECT statements to answer the queries given below. Prepare a script of all the queries by writing them into the *prac05Commands.sql* before pasting them into the MySQL window.

1. Write a join query to display the *last name, job* and *salary* of employees who are responsible for projects where the project name includes the string *'Design'*. Display the project names also in the query results to verify the results.

- Note 2: In queries involving more than one table, it is a good practice to write the column names with the table name (e.g., Students.id) even when the column name is not the same in other tables. This way, users need not to look at table schemas to know from which table the attributes are taken.

2. Use a join query to show the *firstname, lastname job, managing department number* and the *name of the managing department* of each department manager. Use suitable column names for the managing department number and the name of the managing department.

3. Revise the above query (Q2) to show the *department number, department name, manager's firstname, lastname* and *job* for each department.

4. Use a join query to display the *last name, job* and *education level* of all employees who have at least the education level of the employee whose job description is '*Pres'.*

5. Show the *minimum salary, maximum salary* and the number of employees in the job category for each job.

6. For each *workdept* that has more than 5 employees, find the *average salary of employees*. Display *workdept* and the number of employees in each *workdept* also in the query results.

Hint: You'll have to look up how to use the HAVING keyword:
https://dev.mysql.com/doc/refman/8.0/en/select.html

- Note-3: The HAVING clause, like the WHERE clause, specifies selection conditions. The WHERE clause specifies conditions on columns in the select list but cannot refer to aggregate functions. The HAVING clause specifies conditions on groups, typically formed by the GROUP BY clause.

7. For all employees who work on projects, get the *empno, lastname, salary*, and the *total number of hours per week they work on projects*. Display the total number of hours per week they work under a column '*TotalHrs'*.

8. For all employees, show the *empno, lastname, salary, total number of hours per week they work on projects* and the *number of projects they work*. Display the total number of hours per week they work under a column '*TotalHrs'*. If any employee is not working in projects, total number of hours per week they work on projects should be shown as NULL.

9. In the query results of Q8, identify by which column the results are sorted in ascending order. (This is the default ordering)
   Show the same results of the Q8, sorted by *TotalHrs''* in descending order.

10. For each project, list the *project number* and *project name* along with the *project name* and the *project number* of any sub-projects. All the project, irrespective whether they have sub-projects or not should be listed.

11. For projects with sub-projects, list the *project number* and *project name* along with the *project name* and *project number* of any sub-projects.
    (Only the projects having sub-projects should be listed)

## 4. Submitting your work

Your prac05 directory should have *prac05Task1*, *Prac05Commands.sql* and *Prac05Workings,out* files.

Zip your Prac05 directory and upload it to Blackboard under 'Assessments/In Class Practical Submissions'

- Go to your DBS directory and type:

```
> zip -r Prac05_<your student ID> Prac05
```

**Check whether you have achieved learning outcomes:**

I am confident that I can,

| | |
|---|---|
| Use appropriate joins (INNER JOIN, OUTER JOIN, NATURAL JOIN with LEFT/RIGHT clauses ) to retrieve data using more than one table | |
| Sorting the query results using ORDER BY clause | |
| Use suitable aggregate functions (MIN, MAX, SUM, COUNT, AVG) | |
| Group query results using GROUP BY clause. | |
| Use HAVING clause when grouping data. | |
| Create a new database with the CREATE DATABASE statement. | |
| Execute several SQL scripts by calling them in a separate SQL script file. | ✓ |

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.