# Laboratory Worksheet Four

**NOTE:** *You should complete Worksheet Three before beginning Worksheet Four*

## Part 1: Creating SQL Queries from a Template

Using the **projemp** database, create and run five **SQL Queries,** based on the five Template Queries below, and state the **Natural Language Queries** which they answer.

**projemp** relational schema:

**DEPT(dno, dname, location)**
**EMP(eno, ename, salary, age, supno, dno)**
**WORKS(eno, pno, role)**
**PROJ(pno, pname, ptype, budget)**

A **Natural Language Query** is an end user perspective of what is required in the query, **NOT** a description of how the query works. As such, it should not mention implementation words such as table, join or other query implementation details. For example:

| Natural Language Query | SQL Query |
|---|---|
| ***"Find the average salary in each department"*** | SELECT dno, AVG(salary)<br>FROM EMP<br>GROUP BY dno; |

***"This query uses an aggregate function and a group by to calculate the average salary in each department"* is not a valid Natural Language Query.**

The five **Template Queries** are:

1. Single table query with an aggregate function in the SELECT clause and one WHERE clause
   [e.g. SELECT COUNT(*) FROM EMP WHERE dno = 'd1';]
2. Single table query with a GROUP BY clause plus an aggregate function in the SELECT clause
   [e.g. SELECT pno, COUNT(*) FROM WORKS GROUP BY pno;]
3. Single table query with a GROUP BY clause and a valid HAVING clause
   [e.g. SELECT eno FROM WORKS GROUP BY eno HAVING COUNT(pno) > 2]
4. Single query block join selecting at least one attribute from each table in the SELECT clause linked by a Primary/Foreign Key match, plus one extra selection clause (WHERE/AND) from each table
   [e.g. SELECT ename, dname FROM EMP, DEPT WHERE emp.dno = dept.dno AND
   salary > 25000 AND location = 'belfast';]
5. Nested query with two query blocks linked across the Primary/Foreign Key where the lower query block includes an OR clause
   [e.g. SELECT ename FROM EMP WHERE dno IN (SELECT dno FROM dept WHERE
   location = 'belfast' OR location = 'newry');]

**Part 2: Relational Table Creation and Population**

Create and query a family tree database containing just one table:

**FAMILYTREE (person_id, name, mother_id, father_id, spouse_id, dob, dod, gender)Create the tree database and make it the current database.**

1. In a new query window create the **FAMILYTREE table** and save the working SQL file
   a. Use the data below to make sensible choices concerning **datatypes** and **sizes**
   b. A dash (-) in the data below means there is no data for that field
   c. Remember to add an appropriate **primary key** constraint

| person_id | name | mother_id | father_id | spouse_id | dob | dod | gender |
|---|---|---|---|---|---|---|---|
| 101 | Moe | 104 | 105 | 107 | 12/12/1994 | - | M |
| 102 | Larry | 104 | 106 | - | 08/09/1996 | - | M |
| 103 | Curly | 104 | 106 | 110 | 02/04/1997 | - | M |
| 104 | Minnie | 109 | 108 | 106 | 05/06/1975 | - | F |
| 105 | Micky | - | - | - | 04/08/1972 | 16/02/1995 | M |
| 106 | Goofy | - | - | 104 | 05/09/1970 | - | M |
| 107 | Kylie | - | - | 101 | 10/12/1997 | - | F |
| 108 | Bob | - | - | 109 | 04/04/1953 | 05/08/2000 | M |
| 109 | Rita | - | - | 108 | 01/02/1955 | - | F |
| 110 | Brittany | - | - | 103 | 02/02/1995 | - | F |
| 111 | Alvin | 107 | 101 | - | 05/06/2014 | - | M |
| 112 | Simon | 107 | 101 | - | 05/06/2014 | - | M |
| 113 | Theodore | 107 | 101 | - | 05/06/2014 | - | M |
| 114 | Brittany | 110 | 103 | - | 07/07/2015 | - | F |
| 115 | Jeanette | 110 | 103 | - | 07/07/2015 | - | F |
| 116 | Eleanor | 110 | 103 | - | 07/07/2015 | - | F |

2. Refresh the **tree** database to ensure the **FAMILYTREE** table has been created
3. **Insert the data** from the table above into the **FAMILYTREE** table and **save the SQL statements** you used.

***See next page for Part 3…***

**Part 3: Complex Queries on FAMILYTREE**

***Warning*** *–These queries are particularly difficult. Only attempt them if you have been able to implement all of the queries in the worksheets to date.*

1. Using a suitable JOIN approach, list the names of each individual parent with each of their individual child(ren). That is, names of parents with more than one child will occur more than once and children's names can occur twice - once for each parent.
   (20 records returned)

   e.g. Moe, Minnie
   Moe, Micky
   Larry, Minnie
   Larry, Goofy etc.

2. Using a suitable JOIN approach, list each child with the names of their mother and father (10 records returned)

3. List each child with their parents and their grandparents –your answer should not list NULLS.

4. Using a suitable nested query in the FROM clause, list each child with the name of their father where the father is not married. You could use an INNER or OUTER JOIN, but your answer should not list NULLS.