Laboratory Worksheet Three

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

SQL RETRIEVAL EXERCISES

Use database file **projemp** on SQL Server - Execute **USE projemp;** in a New Query window

projemp relational schema:

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

Attributes eno, dno and pno are all text values (e.g, e1, e2, etc). All other attributes are text except salary [decimal(8,2)], age [tinyint], duration [tinyint] and budget [decimal(9,2)]

Click on ***New Query*** on the top menu to create a new query.

```
USE projemp;
GO
```
Placed before each query ensures the correct database is current

COMPLEX NESTED QUERIES USING GROUPING

1. Get the employee numbers of employees that work on more than two projects (where the count of projects is greater than 2)

```
SELECT eno
FROM WORKS
GROUP BY eno
HAVING COUNT(pno) > 2;
```

2. Get the names and salaries of employees working on more than two projects**.**

```
SELECT ename, salary
FROM EMP
WHERE eno IN
   (SELECT eno
   FROM WORKS
   GROUP BY eno
   HAVING COUNT(pno) > 2);
```

3. How many employees work on more than two projects.

```
SELECT COUNT(eno)
FROM EMP
WHERE eno IN
   (SELECT eno
   FROM WORKS
   GROUP BY eno
   HAVING COUNT(pno) > 2);
```

NOW ANSWER THE FOLLOWING QUERIES AND SAVE THEM

a. **Get the projects which have more than seven employees working on them**
   This query can be answered using the **WORKS** table.
   Expected result: p13, p19, p23

b. **Get the names and budgets of projects which have more than seven employees working on them.**
   This can be answered using Q1. as nested query passing the selected project numbers into a table where project number is unique.
   Expected result: payroll 520000; graphics 650000; registration 790000

c. **How many projects have more than seven employees working on them?**
   Same approach as Q2. above.
   Expected result: 3

d. **How many departments have an average salary above 25000?**
   Expected result: 2

COMPLEX NESTED QUERIES USING NEGATION

Suppose you want to get the employees that don't work as a consultant (i.e. role = 'consultant') on any project.

Intuitively you might try a query such as this:

```
SELECT DISTINCT eno
FROM WORKS
WHERE role <> 'consultant';
```
(20 rows in result table)

However if you inspect the **WORKS** table you will see, for example, that 'e12' and 'e18' are in the result table and yet they do work as consultants on some projects.

The correct approach is by double negation:

- Find the employees that work on *ANY* project as a consultant
- Use the **NOT IN** link to negate this (get the employees **NOT IN** that list)

```
SELECT eno
FROM EMP
WHERE eno NOT IN
   (SELECT eno
   FROM WORKS
   WHERE role = 'consultant');
```
(15 rows in result table)

CREATE and SAVE SQL STATEMENTS TO ANSWER THE FOLLOWING

e. Get the projects that have a programmer working on them.

Requires **WORKS** table.

Expected result: p13, p19, p23, p26

f. Get the projects that have no programmer working on them.

Use Q5. as a lower level nested query passing the selected project numbers into a table in which project number is unique using **NOT IN** to negate the list i.e. Find the projects that *DO* have a programmer in them then ignore them and take the others (those that must have *NO* programmer in them).

Expected result: p15, p31

g. How many departments have no employee earning exactly £33,000?

Same approach as Q6

Expected result: 1

VIEWS IN SQL

A view is a virtual table created from existing database tables. When it is created it can be used as if it were a normal table. It only exists as a definition and when used it is materialised (actually created as a table) from the definition working on the table(s) on which it is created.

The general syntax of the **VIEW** statement is as follows:

```
CREATE VIEW viewname (<new attribute list>)
   AS SELECT <attribute list>
      FROM <table list>
      [other clauses];
```

Create the following **VIEW** for high-paid employees:

```
CREATE VIEW RICHEMP (empnum, empname, pay)
   AS SELECT eno, ename, salary
      FROM EMP
      WHERE salary > 45000;
```

NOTE: the attribute names have been changed from those in **EMP** *AND* there are no age, supno or dno attributes.

Expand the Views subfolder in **projemp** to see that **RICHEMP** is listed like a table.

Now use the **RICHEMP** view as a table (views can be used as a normal table although the system translates the query on the view – **RICHEMP** - into a query on the table on which it is defined – **EMP**)

```
SELECT *
FROM RICHEMP;

SELECT empname, pay
FROM RICHEMP;
```

Run the following **INSERT** command:

INSERT INTO RICHEMP VALUES ('e40', 'young', 60000);

Now examine the **RICHEMP** view

Examine the underlying **EMP** table to see what has been entered (there should be a **NULL** value for the age, supno and dno values).

NOW TRY THESE QUERIES, SAVE THEM AND CHECK THE DATABASE

RICHEMP (empnum, empname, pay)

h. **Update the pay attribute (i.e. salary in EMP) for empname 'young' through the view RICHEMP to give a new value of £75,000**
   use an **UPDATE…SET…WHERE..** command then examine the **RICHEMP** view and **EMP** table.

i. **Delete employee with empname 'young' through the view.**

j. **Now drop the view**
   General syntax; **DROP VIEW viewname;**
   Check that the view has been dropped by expanding the Views subfolder (and Refresh)

Remaining queries can be found on the next page…

NOW – TRY THESE.

Using the **projemp** relational schema:

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

Create and save SQL statements to answer the following data retrieval problems:

k. **Get a list of employee numbers and salaries for employees whose department number is 'd2'.**
Expected result: e11 - 28000.00, e12 - 33000.00, e6 - 24000.00, e7 - 16000.00, e9 - 23000.00

l. **What is the average budget for projects whose project type is 'access'?**
Expected result: 616666.66

m. **How many employees work on each project (by project number)?**
Expected result: p13 – 8, p15 – 5, p19 – 10, p23 – 9, p26 - 7

n. **Get the departments (by department number) that have a total salary below £200,000.**
Expected result: d2, d3

o. **Get the employee names, and roles for employees working on project number 'p13'.**
Use a single query block join.

Expected result:

pearse	programmer
Roberts	analyst
smyth	consultant
vance	programmer
evans	manager
greer	administrator
hamill	programmer
irwin	programmer

p. **Get the names of projects worked on by employees whose role is a 'consultant'.**
Use a nested query
Expected result: payroll, graphics, registration, examination

q. **How many employees in department number 'd2' have a salary below the average salary for employees in department number 'd3'?**
Expected result: 3