*Sheffield* *Hallam* *University*



Faculty of Science, Technology and Arts

**SQL** **WORKBOOK**

**55-500998**

**Database** **Systems** **For**

**Software** **Applications**

**SECTION F**

**2020/2021**



*Sheffield* *Hallam* *University*

Faculty of Science, Technology and Arts

**STRUCTURED QUERY LANGUAGE** **(SQL)**

**2020/2021**

SQL Workbook 1 September 2020

SQL Workbook 2 September 2020

Introduction

**INTRODUCTION**

The purpose of this book is to provide practical exercises in the use of SQL to create, populate and maintain a relational database.

**SQL**

SQL (Structured Query Language) is an ISO and ANSI standard *database* *query* *language.* Most relation databases are SQL-compliant, but in spite of SQL being a standard, SQL code is not completely portable among different database management systems.

**Oracle Relational Database**

For this module we will be using Oracle Database 11g Enterprise Edition.

Students will need to create a user account for Oracle, - how to do this is shown in Appendix X. If you have an account from a previous module, it will still be valid – to reset the password or unlock an account, also see Appendix X

**Oracle SQL Developer**

Students will need to connect to their Oracle database account using SQL Developer which is a **free** graphical tool for database development. How to do this is shown in Appendix Y.

With SQL Developer, you can create, browse and manage database objects, execute SQL statements and SQL scripts, and import, manipulate, and export data.

**Sample Tables**

All new accounts have a set of default tables (EMP, DEPT and SALGRADE) already created and populated with data. These are referred to as the **PERSONNEL** **SYSTEM**. Many of the **exercises** throughout this workbook are based on the Personnel System tables.

Some lecture material and various **worked** **examples** in this text are based on part of a simple **BANK** **ACCOUNTING** **SYSTEM** and use the tables CUST, CUSTACC and ACC. These tables represent the fact that a bank customer may have many accounts, and that an account may be held jointly by more than one customer.

Details of these tables can be found in the following pages. If your Personnel System tables get changed/corrupted the instructions to delete and recreate them can be found at the bottom of page 5.

**Workflow**

Topics should be tackled in sequence. This is because exercises in later sections may rely on changes you make to your data or data structures in earlier sections to work properly. Topics must be completed before the scheduled session of the next topic.

SQL Workbook 3 September 2020

Introduction

**The Bank Accounting System**

CUST Owns CUSTACC Allocated ACC

Table: **CUST**

|  |  |  |  |
| --- | --- | --- | --- |
| REFNO | NAME | ADDRESS | AREA |
| A123 A124 B127 B128  C371 | J Doe J Smith R Best J Best  R Done | 1 High Street 2 West Street 4 East Row  4 East Row  23 Middle Avenue | Sheffield Sheffield Rotherham Rotherham  Barnsley |

Table: **CUSTACC**

|  |  |
| --- | --- |
| REFNO | ACCNO |
| A123 A123 B127  B128 | 1245890 1494315 5418490  5418490 |

Table: **ACC**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ACCNO | BALANCE | BRANCH | OPENED | BONUS |
| 1245890 1494315  5418490 | 234.50 0.50  1789.40 | Broomhill Tinsley  Broomhill | 12 Nov 2003 15 Dec 1999  6 May 1988 | 100.00 0.00 |

SQL Workbook 4 September 2020

Introduction

**THE PERSONNEL SYSTEM**

Table: **EMP**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |
| 7369 7499 7521 7566 7654 7698 7782 7788 7839 7844 7876 7900 7902  7934 | SMITH ALLEN WARD JONES MARTIN BLAKE CLARK SCOTT KING TURNER ADAMS JAMES FORD  MILLER | CLERK SALESMAN SALESMAN MANAGER SALESMAN MANAGER MANAGER ANALYST PRESIDENT SALESMAN CLERK CLERK ANALYST  CLERK | 7902 7698 7698 7839 7698 7839 7839 7566  7698 7788 7698 7566  7782 | 17-DEC-80 20-FEB-81 22-FEB-81 02-APR-81 28-SEP-81 01-MAY-81 09-JUN-81 09-DEC-82 17-NOV-81 08-SEP-81 12-JAN-83 03-DEC-81 03-DEC-81  23-JAN-82 | 800.00 1600.00 1250.00 2975.00 1200.00 2850.00 2450.00 3000.00 5000.00 1500.00 1100.00 950.00 3000.00  1300.00 | 300.00 500.00  1250.00  0.00 | 20 30 30 20 30 30 10 20  30 20 30 20  10 |

Table: **DEPT**

|  |  |  |
| --- | --- | --- |
| DEPTNO | DNAME | LOC |
| 10 20 30 40 | ACCOUNTING RESEARCH SALES OPERATIONS | NEW YORK DALLAS CHICAGO BOSTON |

Table: **SALGRADE**

|  |  |  |
| --- | --- | --- |
| GRADE | LOSAL | HISAL |
| 1 2 3 4  5 | 700.00 1201.00 1401.00 2001.00  3001.00 | 1200.00 1400.00 2000.00 3000.00  9999.00 |

Should data in the tables become corrupt, they may be restored to their original status by issuing each of the following statements for the appropriate table:

**DROP** **TABLE** EMP ;

**CREATE** **TABLE** EMP **AS** **SELECT** \* **FROM** EXAMPLE.EMP ;

SQL Workbook 5 September 2020

Introduction

This page is intentionally blank.

SQL Workbook 6 September 2020

Section F

**TABLE ALIASING, SELF JOINS, OUTER JOINS**

**TABLE ALIAS NAMES**

It is possible within a SELECT statement to use temporary labels, called **aliases**, for the table names; this can be a useful shorthand to reference the tables. Single letters may be used although it is advisable to use meaningful aliases to aid readability. Once an alias is declared for a table it must be used consistently throughout the query; using the table name will generate an error.

Using table aliases, this example:

**SELECT** **FROM** **INNER** **JOIN** **ON**

**INNER** **JOIN** **ON**

becomes:

**SELECT** **FROM** **INNER** **JOIN** **ON**

**INNER** **JOIN** **ON**

CUST.REFNO, NAME, ACC.ACCNO, BALANCE CUST

CUSTACC

CUST.REFNO = CUSTACC.REFNO ACC

CUSTACC.ACCNO = ACC.ACCNO ;

**C**.REFNO, NAME, **A**.ACCNO, BALANCE CUST **C**

CUSTACC **CA** **C**.REFNO = **CA**.REFNO ACC **A**

**CA.**ACCNO = **A**.ACCNO ;

Note that the alias is set by writing it after the table name in the FROM or JOIN clause, so ‘FROM CUST C’ sets C as the alias for the CUST table.

Aliases are a useful option in most cases, but they are essential for Self-Joins – covered in the next section.

**SELF JOINS**

Table EMP contains details of Employees. One piece of data is the Employee Number of their manager (the MGR column). How do we obtain the name of their manager? This is a common occurrence, where a table needs to 'link back' to itself.

We cannot do this directly, and we cannot reference two tables with the same name. But we can have a second copy of a table which is referenced by a (sensible) alias.

If we give one copy of table EMP the alias 'Manager' (and consider it to be the list of managers), and the second 'Staff' (and consider it to be the list of staff, some of whom are also managers), we can now treat them as two separate tables. Note that all column names are duplicated and need to be referenced using the *table.column* notation.

SQL Workbook 55 September 2020

Section F

What is the relationship between the data in these two copies of table EMP? (You may find it helpful to place two appropriately headed copies of the printed tables next to one another and work out who is Smith's manager). The MGR column of the Staff table equates to the EMPNO column of the Manager table.

**Exercises**

**F1** Show the name, employee number and manager's name of those who are managed by either Blake or Jones. **Carefully** **check** **your** **results.**

**SELECT** E.ENAME, E.EMPNO, M.ENAME **FROM** EMP E **INNER** **JOIN** EMP M **ON E.MGR = M.EMPNO**

**WHERE M.ENAME = ‘BLAKE’ OR M.ENAME = ‘JONES’;**

**F2** Display the names of all the people in a management position, their department number and the number of staff for whom they have direct responsibility. (This is **not** a case of asking for those with job 'MANAGER'. We are looking for those employees that manage one or more employees, ie where employees have them as their manager (column MGR).

**SELECT** M.ENAME, M.DEPTNO, E.COUNT(M.EMPNO) **FROM** EMP E **INNER** **JOIN** EMP M **ON E.MGR = M.EMPNO**

**GROUP** **BY M.ENAME, M.DEPTNO;**

**F3** Show only those managers, as defined above, who are directly responsible for the management of more than two people.

**SELECT** M.ENAME, M.DEPTNO, COUNT(M.EMPNO) **FROM** EMP E **INNER** **JOIN** EMP M **ON E.MGR = M.EMPNO**

**GROUP** **BY M.ENAME, M.DEPTNO**

**HAVING COUNT(M.EMPNO) > 2;**

Note that the above examples of self-joins are still INNER JOINS, as each returned row contains columns from each table involved.

SQL Workbook 56 September 2020

Section F

**OUTER JOINS**

While an inner join returns a result set composed of columns from rows in both tables, sometimes we want to return the rows from one table even if they aren’t matched by rows in the other table (as well as those that are matched),

The following queries illustrate such situations.

**F4** Display the employee name, location and department number of those whose salary is greater than £1500.

**SELECT E.ENAME, D.LOC, E.DEPTNO FROM EMP E**

**INNER JOIN DEPT D ON E.DEPTNO = D.DEPTNO**

**WHERE E.SAL > 1500;**

How many rows have you selected? Is that correct? This problem will be looked into soon.

**F5** For all employees list their name, salary and location.

**SELECT** E.ENAME, E.SAL, D.LOC **FROM** EMP E **INNER** **JOIN** DEPT D **ON E.DEPTNO = D.DEPTNO;**

**F6** List the names of Employees, and their manager's name.

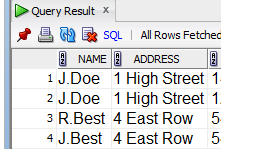
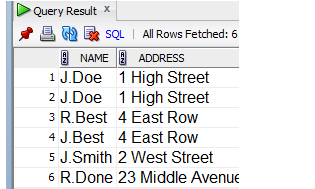
**SELECT** E.ENAME, M.ENAME AS MNAME **FROM** EMP E **INNER** **JOIN** EMP M **ON E.MGR = M.EMPNO;**

What is missing from the output, - and why ?

In the above examples the resulting rows are correct as far as they go, but not totally correct as some rows are missing. That is because in INNER joins rows are only selected where the join is satisfied, ie where matching values are found in both tables.

SQL Workbook 57 September 2020

Section F



In this example, . . .

**Example** **SELECT** NAME, ADDRESS, ACCNO

**FROM** CUST

**INNER** **JOIN** CUSTACC

**ON** CUST.REFNO **=** CUSTACC.REFNO ;

. . . no information is displayed about Smith or Done, yet they are both in the CUST table. We often want to include/select such rows. Knowing which customers don't have a certain type of account can be a very valuable piece of information.

These situations are addressed by another form of join - the **OUTER** **JOIN** - which returns rows as for the INNER JOIN but, in addition, returns rows from one table which have no match in the other table – the extra unmatched rows.

There are three forms of OUTER JOIN, the **LEFT** OUTER JOIN, **RIGHT** OUTER JOIN and **FULL** OUTER JOIN, and the correct one to use is determined by which table we want to select the extra rows from.

Using the same example again, but with an outer join.

**Example** **SELECT** NAME, ADDRESS, ACCNO

**FROM** CUST

**LEFT** **OUTER** **JOIN** CUSTACC

**ON** CUST.REFNO **=** CUSTACC.REFNO ;

The CUST table contains the extra rows we want,

and we indicate that by using LEFT OUTER JOIN, - because the CUST table is on the LEFT hand side of the join clause (“FROM **CUST** LEFT OUTER JOIN **CUSTACC**”).

This will list **all** the names and addresses in the customer table, including customers who do not currently have an account number assigned to them.

If we had wanted extra rows from CUSTACC instead of CUST we would say RIGHT OUTER JOIN. And if we want extra rows from both of them, we say FULL OUTER JOIN.

Note that the following two examples will produce exactly the same results.

**Example** **SELECT** NAME, ADDRESS, ACCNO

**FROM** CUST

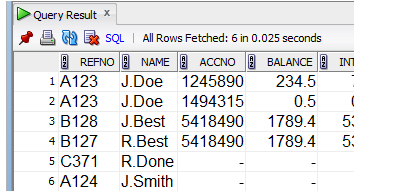
**LEFT** **OUTER** **JOIN** CUSTACC **ON** CUSTACC.REFNO **=** CUST.REFNO ;

**Example** **SELECT** NAME, ADDRESS, ACCNO

**FROM** CUSTACC

**RIGHT** **OUTER** **JOIN** CUST **ON** CUSTACC.REFNO **=** CUST.REFNO ;

Section F



If we join three or more tables where any of the relationships are optional we must use an outer join in each of the join statements. Failure to do so will produce an error message saying that the join is ambiguous.

**Example** **SELECT** CUST.REFNO, NAME, ACC.ACCNO, BALANCE,

BALANCE \*0.03 **AS** INTEREST **FROM** CUST

**LEFT** **OUTER** **JOIN** CUSTACC

**ON** CUST.REFNO **=** CUSTACC.REFNO **LEFT** **OUTER** **JOIN** ACC

**ON** CUSTACC.ACCNO **=** ACC.ACCNO;

Note the column titles, particularly ACCNO. This column name exists in both the ACC and CUSTACC tables yet the display does not identify this, even though we have specified ACC.ACCNO.

The **value** in the ACCNO column for CUST.REFNO A124 is not zero but **NULL**.

It can be treated like any other value and used within Criteria (column IS NULL) - a very powerful facility. Any calculations performed on a column containing a null value will also return a null.

Here is the same query but with a WHERE clause designed to select only customers with no accounts.

**Example** **SELECT** CUST.REFNO, NAME, ACC.ACCNO, BALANCE, BALANCE \*0.03 AS INTEREST

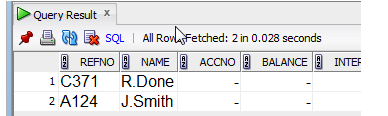
**FROM** CUST

**LEFT** **OUTER** **JOIN** CUSTACC

**ON** CUST.REFNO **=** CUSTACC.REFNO **LEFT** **OUTER** **JOIN** ACC

**ON** CUSTACC.ACCNO **=** ACC.ACCNO **WHERE** **ACC**.ACCNO **IS** **NULL** ;

SQL Workbook 59 September 2020



Section F

**Exercises**

**F7** Display the employee name, location, and department name of those whose salary is greater than £1500. Ensure all relevant employees are selected.

**SELECT** E.ENAME, D.LOC, D.DNAME **FROM EMP** **OUTER** **JOIN DEPT** **ON E.DEPTNO = D.DEPTNO**

**WHERE E.SAL > 1500**

How many rows have you selected? Is that correct?

**F8** For **all** employees (i.e. to include the President) display their own name and employee number and their manager's name and number.

(Check this output very carefully against the sample tables).

**SELECT E.ENAME, E.EMPNO, M.ENAME, M.EMPNO FROM EMP E**

**LEFT OUTER JOIN M.EMP M ON E.MGR = M.EMPNO;**

**F9** List the names of all employees that are not in the RESEARCH department. Ensure you check the results generated by your solution.

**SELECT ENAME FROM EMP RIGHT OUTER JOIN DEPTNO ON EMP.DEPTNO = DEPT.DEPTNO WHERE DEPT.DNAME != ‘RESEARCH’;**

Ensure you have displayed nine rows, including KING.

**F10** List the names of those departments that have no employees.

SQL Workbook 60 September 2020

Section F

**SELECT DNAME FROM DEPT LEFT OUTER JOIN EMP ON EMP.DEPTNO = DEPT.DEPTNO WHERE EMP.DEPTNO IS NULL;**

**F11** List the departmental names and number of employees in departments with fewer than 6 employees.

**SELECT D.DNAME, COUNT(\*) FROM DEPT D LEFT OUTER JOIN EMP E ON E.DEPTNO = D.DEPTNO GROUP BY D.DNAME HAVING COUNT(E.DEPTNO) < 6;**

SQL Workbook 61 September 2020

Section F