*Sheffield* *Hallam* *University*



Faculty of Science, Technology and Arts

**SQL** **WORKBOOK**

**55-500998**

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

**Software** **Applications**

**SECTION E**

**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 E

**INNER JOINS and VIEWS**

An SQL JOIN clause is used when we need to select columns from rows in two or more tables, based on a common field between them.

**INNER JOINS**

These joins are the most common form and return rows from two joined tables when there are rows in BOTH tables which satisfy the ON condition in the JOIN clause.

In other words, if we have Table A and Table B involved in an inner join, each row in the result set will contain columns from a row in A and columns from a row in B, and each pair of these rows will have been matched by the join ON condition. This is true because rows in A not matched by any rows in B are discarded, and vice versa.

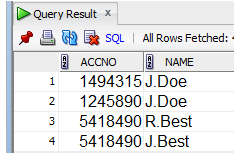
The syntax is:

**SELECT** *select-list* **FROM** *table\_name1* **INNER** **JOIN** *table\_name2*

**ON** **t***able\_name1.column* *operator* **t***able\_name2.column* **WHERE** . . . *other* *criteria* (optional)

**Equi-Joins**

Most commonly the join condition will require that the columns should be equal (an equi-join), as in this example:

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

**FROM** CUST

**INNER** **JOIN** CUSTACC

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

The column names in the ON clause should include the table name, even if the column names are different.

**Exercises**

**E1** Write a query to display ENAME and DNAME for employees in the EMP table. As EMP only has DEPTNO, this requires joining EMP to DEPT using the common attribute DEPTNO.

SELECT ENAME, DNAME FROM EMP INNER JOIN DEPT ON EMP.DEPTNO = DEPT.DEPTNO;

SQL Workbook 49 September 2020

Section E

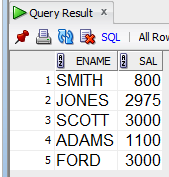
E2 Display each employees name, and the city where they work.

SELECT ENAME, LOC FROM EMP INNER JOIN DEPT ON EMP.DEPTNO = DEPT.DEPTNO;

**E3** Display the name of each department and the names of the staff who work in each of them.

SELECT DNAME, ENAME FROM DEPT INNER JOIN EMP ON DEPT.DEPTNO = EMP.DEPTNO;

**E4** Find the name and salary of employees in DALLAS.

SELECT ENAME, SAL FROM EMP INNER JOIN DEPT ON EMP.DEPTNO = DEPT.DEPTNO

WHERE LOC = ‘DALLAS’;

**E5** Display details of employees in ACCOUNTING. (Don’t answer this by using 'WHERE Deptno = 10'. Use two tables to improve that solution.)

SELECT \* FROM EMP INNER JOIN DEPT ON DEPT.DNAME = ‘ACCOUNTING’;

**E6** Display the department name along with the lowest and highest salaries in each department.

SELECT DNAME, MIN(SAL), MAX(SAL) FROM EMP INNER JOIN DEPT ON EMP.DEPTNO = DEPT.DEPTNO GROUP BY DNAME;

SQL Workbook 50 September 2020

Section E

**E7** What are the highest and lowest incomes (including commission) in the Sales department?

SELECT DNAME, MIN(SAL+NVL(COMM,0)) AS MIN\_INCOME, MAX(SAL+NVL(COMM,0)) AS MAX\_INCOME FROM EMP INNER JOIN DEPT ON DNAME = 'SALES' GROUP BY DNAME;

**E8** Display the department name and number of employees in departments with 5 or more employees.

SELECT DNAME, COUNT(EMP.DEPTNO) FROM EMP INNER JOIN DEPT ON EMP.DEPTNO = DEPT.DEPTNO

GROUP BY DNAME HAVING COUNT(EMP.DEPTNO) > 5;

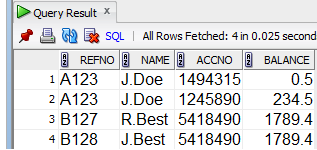
**E9** Calculate the total income for all employees in each city. This will include commission.

SELECT SUM(SAL + NVL(COMM, 0) FROM EMP INNER JOIN DEPT ON EMP.DEPTNO = DEPT.DEPTNO

GROUP BY DEPT.DNAME;

In many cases there will be more than two tables involved. If there are n tables involved in the instruction, then there are (n-1) table relationships to be defined using JOIN and ON. In the next example, there are 3 tables, and 2 join on clauses:

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



**FROM** CUST

**INNER** **JOIN** CUSTACC

**ON** CUST.REFNO = CUSTACC.REFNO **INNER** **JOIN** ACC

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

Notice that in this example the SELECT statement specifies REFNO and ACCNO for display. Because each of these columns exists in two tables, they must be qualified with the tablename to avoid ambiguity in the select clause (as well as in the ON clause).

SQL Workbook 51 September 2020

Section E

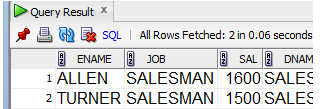
**Non-Equi Joins**

Although most ON clauses will use = for the relationship, other operators may be used, such as ‘less than’, ‘greater than’, BETWEEN, LIKE etc.

For example, table EMP contains details of Employees. One piece of data is their Salary. The SALGRADE table holds the Grade and the salary range (stated as the low and high values) applicable to that grade. How do we find the Grade that any employee is on? We cannot do this directly as the SALGRADE table does not contain a list of all possible salaries. But we can identify the grade by selecting the row where the employee's salary is **BETWEEN** LOSAL and HISAL (assuming the grades do not overlap).

**Exercises**

**E10** Display employee name, job, salary and department name for those people on grade 3.

SELECT ENAME, JOB, SAL, DNAME, FROM EMP INNER JOIN DEPT ON EMP.DEPTNO = DEPT.DEPTNO

INNER JOIN SALGRADE ON SAL BETWEEN LOSAL AND HISAL

WHERE GRADE = 3;

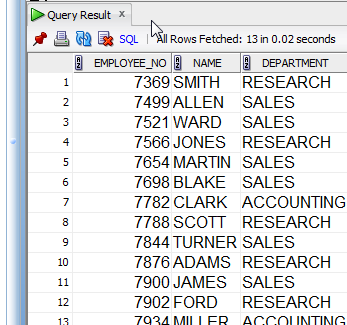
**E11** Produce a list that will show the salary grade each employee is on. Display the name, job, salary and grade with the output in ascending order of salary, and with those on the same salary ordered alphabetically.

SELECT ENAME, JOB, SAL, GRADE FROM EMP INNER JOIN SALGRADE ON SAL BETWEEN LOSAL AND HISAL

ORDER BY SAL, ENAME;

Inner joins create a dataset of related information comprised of columns from two or more tables. If a particular dataset is frequently required, the same join may appear many times in different queries, and if the join involves many tables it makes the code complex. An alternative approach is to use the join to create a named ‘View’ which stores the resulting rows in memory. The View can then be queried as if it were a table, using a query which needs no join.

SQL Workbook 52 September 2020



|  |  |  |
| --- | --- | --- |
| Section E  **VIEWS**  A view is a SELECTION of columns from one or more tables. It is a virtual table, and the rows appearing in the view are derived from the underlying base tables at run time, so that changes in the data in the base tables are reflected directly in the views. The view definition is stored in the schema as if it were a table, and needs to be defined only once.  Views are are useful for :   restricting access to the database   providing a simple table for users to retrieve the results of complex queries  providing data independence for users.  The syntax to create a view is:  **CREATE** **VIEW** *view\_name* *[(column\_name* *alias,* *column\_name* *alias,,,)]* **AS**  **SELECT**...  **Example** **CREATE** **VIEW** EMP\_VIEW (EMPLOYEE\_NO, NAME, DEPARTMENT)  **AS**  **SELECT** E.EMPNO, E.ENAME, D.DNAME **FROM** EMP E  **INNER** **JOIN** DEPT D  **ON** E.DEPTNO = D.DEPTNO;  and then  **SELECT** \* **FROM** EMP\_VIEW; will return: | |  |
| This view provides employee and department information without having to perform the join each time the request has to be met.  Select queries may now be run against this view as though it were a single table in the database. |  |  |
| SQL Workbook 53 September 2020 | |  |

Section E

**E12** Create a view called CUSTOMER which contains customer name, address and area.

**CREATE VIEW CUSTOMER (NAME, ADDRESS, AREA)**

**AS**

**SELECT NAME, ADDRESS, AREA FROM CUST;**

**E13** Create a view called ACCOUNT\_BALANCE which contains customer reference number, account number and balance.

**CREATE VIEW ACCOUNT\_BALANCE (REFNO, ACCNO, BALANCE)**

**AS**

**SELECT C.REFNO, CA.ACCNO, A.BALANCE**

**FROM CUST C**

**INNER JOIN CUSTACC CA ON C.REFNO = CA.REFNO**

**INNER JOIN ACC A ON CA.ACCNO = A.ACCNO;**

**E14** Create a view to display details of customers, who have balances greater than £500. Can this result be produced using a previously created view?

**CREATE VIEW FIVE\_HUNDRED(REFNO, ACCNO, BALANCE)**

**AS**

**SELECT \* FROM ACCOUNT\_BALANCE AB**

**WHERE AB.BALANCE > 500;**

SQL Workbook 54 September 2020