*Sheffield* *Hallam* *University*



Faculty of Science, Technology and Arts

**SQL** **WORKBOOK**

**55-500998**

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

**Software** **Applications**

**SECTION B**

**2020/2021**

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 B

**Retrieving data - the SELECT statement**

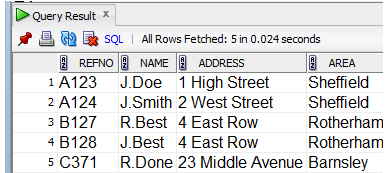
The SELECT statement is used to retrieve data from one or more tables. It can take several additional clauses, but the simplest form is:

SELECT \* FROM *table\_name*; where the \* means select ‘all columns’

or, more usefully,

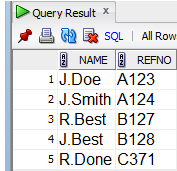
SELECT *column\_name*, *column\_name,* *.* *.* *.* FROM *table\_name*;

The two statements **SELECT** and **FROM** and the terminating **;** are mandatory.

This example displays all columns from the CUST table in the order defined in the Create statement.

**Example** **SELECT** \* **FROM** CUST;

This example specifies which columns are displayed, and the order of them.

**Example** **SELECT** NAME, REFNO **FROM** CUST;

NAME is displayed before REFNO even though they were initially defined the other way round.

Note the comma between column names, but not after the final column name.

It is also possible to manipulate the column values and set column titles before displaying them. This theoretical example:-

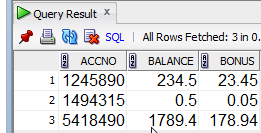
**SELECT** col1, col2\*2.5, col3+col4 **AS** column\_alias\_name **FROM** table-name;

. . . displays specified columns, performs arithmetic on some columns and specifies an alias name for the last column to use as a heading on output.

SQL Workbook 19 September 2020

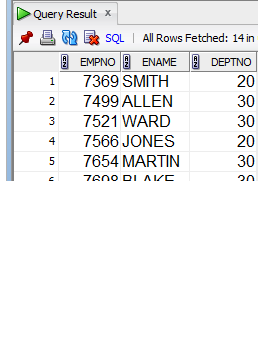
Section B

This actual example would produce the output shown on the right .



**Example** **SELECT** ACCNO, BALANCE, BALANCE \* 0.1 **AS** BONUS

**FROM** ACC;

To list all employees showing their

employee number, name, and department number – as shown on the right, we would use:

**Example** **SELECT** EMPNO, ENAME, DEPTNO **FROM** EMP ;

**Exercises**

For the following exercises record the statement that will give you the required output. Check against the given tables to ensure that the correct rows are selected and save the statements for future reference.

(For each section the output from some of the queries is reproduced for you, but after that you are on your own!)

**B1** Display all the contents of the DEPT, EMP and SALGRADE tables. The first one is done for you. Check your results against the tables listed.

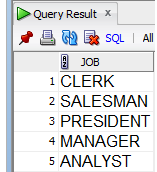
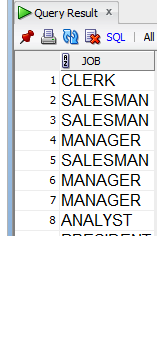
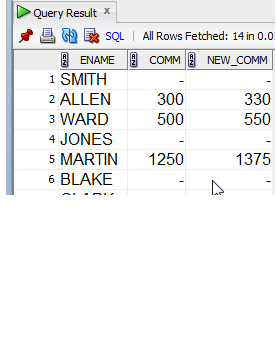
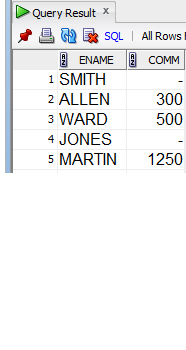
|  |
| --- |
| **SELECT** \* **FROM** EMP ; |
| **SELECT \* FROM DEPT;** |
| **SELECT \* FROM SALGRADE;** |

SQL Workbook 20 September 2020

Section B

**B2** Display the name and commission of all the employees.

**SELECT ENAME, COMM**



**FROM EMP;**

**B3** Display the name and commission of all the employees together with another column that shows their commission increased by 10%. Use an appropriate column alias for the new column.

**SELECT ENAME, COMM,**

**COMM \* 1.1 AS NEW\_COMM**

**FROM EMP;**

**B4** Display the job title of all the employees.

**SELECT JOB**

**FROM EMP;**

To remove duplicates within the result set table, use the DISTINCT

the first column)

**SELECT** **DISTINCT** col1, col2 etc e

**B5** Use DISTINCT to display all the different job titles which exist in the company.

**SELECT DISTINCT JOB FROM EMP;**

SQL Workbook 21 September 2020

Section B

**The WHERE clause**

The WHERE clause is used to extract only those records that fulfill a specified criterion. The syntax for the WHERE clause is:

SELECT *column\_name*, *column\_name* FROM *table\_name*

**WHERE** ***column\_name*** ***operator*** ***value*;**

Allowable Operators are:

Comparison operators: **=**, **<>**, **>**, **>=**, **<**, **<=**, (**!=** also means not equal (<>))

Value between an inclusive range: Value in a specified list:

Pattern matching:

**BETWEEN** value **AND** value **IN** (value, value, ..., ...)

**LIKE** '\_charstring%'

where, \_ matches any single character and, % matches 0 to n characters)

(The action of the BETWEEN, IN, and LIKE operators can be negated by preceding them with **NOT**)

Value is or is not null: **IS** NULL or **IS** NOT NULL

The logical operators **AND**, **OR** can be used to create multiple predicates. **AND** takes precedence over **OR**, - use brackets to make predicates unambiguous..

This example would display all columns for all customers in the Sheffield and Barnsley areas.

**Example** **SELECT** \* **FROM** CUST

**WHERE** (AREA = 'Sheffield') OR (AREA = ‘Barnsley’);

**Using Bind variables**

Queries which take parameters can be written using bind variables. Bind variables may be pre-declared and assigned a value, or be linked (bound) to a variable from another environment (eg. PHP) submitting a SQL query to the database. In the SQL Developer environment, a prompt is displayed for the bind variable value when the query is run. Bind variables begin with a colon – eg. :bvar

In this example the bind variables :area1 and :area2

will be prompted for at run time.

**Example** **SELECT** \* **FROM** CUST

**WHERE** (AREA = :area1) OR (AREA = :area2);

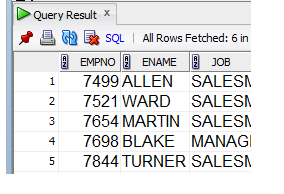
Some of the following exercises provide opportunities to use bind variables (eg. B6) to create more flexible queries.

SQL Workbook 22 September 2020

Section B

**Exercises**

**B6** Display the employee number, name and current job of all those who work in Department 30.

**SELECT** EMPNO, ENAME, JOB **FROM** EMP **WHERE DEPTNO = 30**

**B7** Display the names of all the clerks, showing their employee number and that of their manager. Note: String matching is case sensitive; strings must be enclosed within single quote marks.

**SELECT ENAME, EMPNO, MGR FROM EMP WHERE JOB = ‘CLERK’**

**B8** Display details of all clerks and analysts. Create two instructions to achieve this; each instruction must display both clerks and analysts.

**SELECT \* FROM EMP WHERE (JOB = ‘CLERK’) OR (JOB = ‘ANALYST’)**

**SELECT**

**B9** Display name, a twentieth of salary as 'Twentieth' and commission of all employees whose commission is greater than a twentieth of their salary.

**SELECT ENAME, SAL \* 0.05 AS TWENTIETH, COMM WHERE COMM / SAL > 0.05**

SQL Workbook 23 September 2020

Section B

**B10** Display employee name, job and department number for employees whose names begin with ‘M’.

**SELECT ENAME, JOB, DEPTNO FROM EMP WHERE NAME LIKE ‘M%’;**

**B11** Display details of all clerks and analysts. Create two different instructions to achieve this; each instruction must display both clerks and analysts.

|  |
| --- |
| **SELECT \* FROM EMP WHERE (JOB = ‘CLERK’) OR (JOB = ‘ANALYST’)** |
| **SELECT** |

**B12** Display details of all clerks, analysts and salesmen. Create two different instructions to achieve this; each instruction must display all jobs.

|  |
| --- |
| **SELECT \* FROM EMP WHERE (JOB = ‘CLERK’) OR (JOB = ‘ANALYST’) OR (JOB = ‘SALESMAN’);** |
| **SELECT** |

**B13** Display details of employees who are neither managers nor president; you must not use knowledge about other jobs that might exist. Write two different instructions to achieve this.

**SELECT \* FROM EMP WHERE (NOT(JOB = ‘MANAGER’) AND NOT(JOB = ‘PRESIDENT’));**

SQL Workbook 24 September 2020

Section B

**SELECT**

**B14** Display details of employees who are not clerks, analysts or salesmen; again you must not use knowledge about other jobs that might exist. Write two different instructions to achieve this.

|  |
| --- |
| **SELECT \* FROM EMP WHERE (NOT(JOB = ‘CLERK’) AND NOT(JOB = ‘ANALYST’) AND NOT(JOB = ‘SALESMAN’));** |
| **SELECT** |

**B15** Display details of employees whose salaries are between £1,200 and £1,400. Write two different instructions to achieve this.

|  |
| --- |
| **SELECT \* FROM EMP WHERE (SAL > 1200) AND (SAL < 1400);** |
| **SELECT** |

SQL Workbook 25 September 2020

Section B

**B16** Display details of employees whose salaries are **not** between £1,200 and £1,400. Write two different instructions to achieve this. Ensure you check your results!

|  |
| --- |
| **SELECT \* FROM EMP WHERE NOT((SAL > 1200) AND (SAL < 1400));** |
| **SELECT \* FROM EMP WHERE (SAL < 1200) OR (SAL > 1400);** |

**B17** Display details of those salesmen and managers in dept 30 whose salary is greater than or equal to £1,500. Check your results carefully.

Complex instructions may be written and tested in stages, building the complexity with each step.

Stage 1: Display details of salesmen and managers.

**SELECT \* FROM EMP WHERE (DEPTNO = 30) AND (JOB = ‘SALESMAN’ OR JOB = ‘MANAGER’) AND SAL >= 1500;**

Stage 2: Display details of those salesmen and managers in Department 30. Run the instruction and check your results.

**SELECT \* FROM EMP WHERE (DEPTNO = 30) AND (JOB = ‘SALESMAN’ OR JOB = ‘MANAGER’);**

Stage 3: Display details of those salesmen and managers in dept 30 whose salary is greater than or equal to £1,500. Check your results carefully.

**SELECT \* FROM EMP WHERE (DEPTNO = 30) AND (JOB = ‘SALESMAN’ OR JOB = ‘MANAGER’) AND (SAL >= 1500);**

SQL Workbook 26 September 2020

Section B

**The ORDER BY clause**

The ORDER BY clause is used to sort the result-set by one or more columns. The syntax for the ORDER BY clause is:

SELECT *column\_name*, *column\_name* FROM *table\_name*

WHERE *column\_name* *operator* *value*;

**ORDER** **BY** ***column\_name*,** ***column\_name*** **ASC|DESC;**

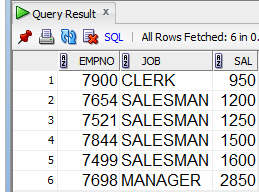
Note: The default ordering is ascending.

Null values are always displayed first regardless of sort sequence.

**Exercises**

**B18** Display the employee number, current job and salary of all those who work in Department 30, with the output in ascending salary order. As before, you may wish to write and test your instruction in a number of stages.

**SELECT** EMPNO, JOB, SAL **FROM** EMP **WHERE** DEPTNO = 30 **ORDER** **BY SAL ASC;**



**B19** Display the employee name and current job of all those who work in Department 30, with the output in descending salary order.

**SELECT ENAME, JOB FROM EMP WHERE DEPTNO = 30 ORDER BY SAL DESC;**

NB: You were asked to order the rows using a column that is not selected for display; maybe a little unusual but not impossible. How have you checked the accuracy of your result?

SQL Workbook 27 September 2020

Section B

**B20** Display the employee name, current job and salary of all those who work in Department 30, with the output in descending salary order within each job.

**SELECT ENAME, JOB, SAL FROM EMP WHERE DEPTNO = 30, ORDER BY JOB ASC, SAL DESC;**

The question asked for the selected data to be output in descending salary order within each job. Does this influence the best order for displaying the columns? Can you suggest a better solution?

**SELECT**

**B21** Display employee details for departments 10 and 30, in name order, within each department. Display the **columns** in the most appropriate order.

**SELECT DEPTNO, ENAME, EMPNO, JOB, MGR, HIREDATE, SAL, COMM FROM EMP WHERE (DEPTNO = 10 OR DEPTNO = 30) ORDER BY DEPTNO ASC, ENAME ASC;**

SQL Workbook 28 September 202