*Sheffield* *Hallam* *University*



Faculty of Science, Technology and Arts

**SQL** **WORKBOOK**

**55-500998**

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

**Software** **Applications**

**2020/21**

SQL Workbook September 2020

**BSc** **Computer Science & BSc Software Engineering**

**Database** **Systems** **for** **Software** **Applications**

|  |  |  |
| --- | --- | --- |
| **Section** | **Topics** | **Page** |
| Introduction | Resources & Workflow Example Tables | 3 4 |
| Week 1 Lab Getting Started | Create an Oracle account SQL Developer familiarisation Simple SQL statements | 7 |
| A | Data Types, Naming rules CREATE table DROP table  INSERT, UPDATE, DELETE data | 13 |
| B | SELECT … FROM WHERE  ORDER BY | 19 |
| C | Dates & Time Simple functions NULL & NVL() GROUP BY HAVING | 29 |
| D | Domain constraints  Primary & Foreign Key Constraints Naming constraints  Database Export/Import ALTER table | 39 |
| E | INNER JOIN  Equi & Non-equi-joins Views | 49 |
| F | Table Aliases SELF JOIN OUTER JOINS | 55 |
| G | Sub-queries | 63 |
| Appendix I | Oracle functions | 67 |
| Appendix II | Indexes | 71 |
| Appendix III | DCL & Object Privileges GRANT & REVOKE | 73 |
| Appendix X | Create Oracle Account | 75 |
| Appendix Y | Connect with SQL Developer | 77 |

SQL Workbook September 2020

**REFERENCES**

**-Colin Ritchie - Database Principles and Design - 3rd Edition**

**-**

**-Connolly, Begg & Strachan - Database Systems - 4th or 5th Edition**

**-ORACLE OCP Introduction to Oracle9i:SQL Exam Guide (Exam 1Z0-007)**

**-Lynn Beighley - Head First SQL/A Brain Friendly Guide**

**-Anthony Molinar0 - SQL Cookbook (O'Reilly Media)**

**-Ben Forta - Sams Teach Yourself SQL in 10 Minutes**

**-Jose A. Ramalho - Learn SQL (Wordware)**

**-Alan Beaulieu - Learning SQL (O'Reilly Media)**

SQL Workbook September 2020



*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 ;

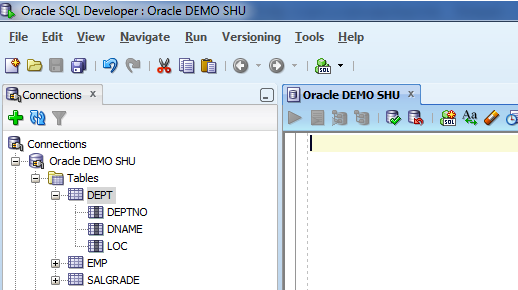
SQL Workbook 5 September 2020

Introduction

This page is intentionally blank.

SQL Workbook 6 September 2020

Lab Week 1



**SQL Lab Week 1**

**Objectives**

1. Create an Oracle account, and connect to it using Oracle SQL Developer. 2. Become familiar with the Oracle SQL Developer environment.

3. Practice the creation and execution of simple SQL statements and scripts.

**Activities**

**1.** **Create** **an** **Oracle** **account** – as described in Appendix X

**2.** **Connect** **to** **Oracle** **using** **SQL** **Developer** – as described in Appendix Y

**3.** **Open** **the** **connection**

Click on the + next to your Connection Name to open it. Click on the + next to Tables to show the tables.

Click on the + next to the DEPT table to show the columns

If a SQL Worksheet is not already open, then right-click your connection and select Open SQL Worksheet.

Your screen should look like this:

SQL Worksheet

Browser-style connections list showing tables and other objects

Worksheet controls

Worksheet code window

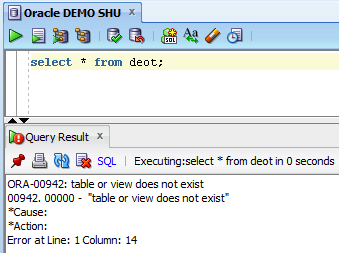
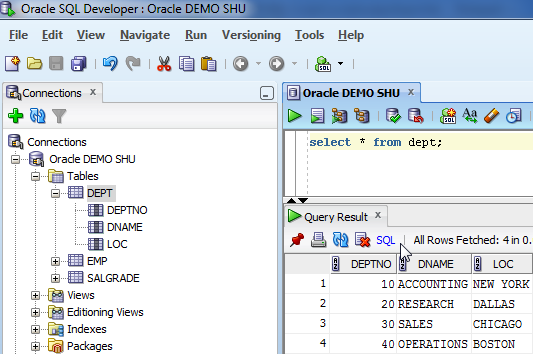
**4.** **Execute** **a** **simple** **SQL** **query,** **and** **observe** **error** **messages.**

4a) Place the cursor in the SQL Worksheet

Type: select \* from dept; and leave the cursor on the same line. Click the *Run* *Statement* button (green triangle).

Your screen should look like this:

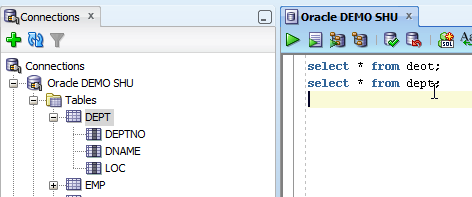
SQL Workbook 7 September 2020



|  |  |  |
| --- | --- | --- |
| Lab Week 1  Run Script | |  |
| Run Statement | Clear code window |  |
| Query Result pane  Line numbers – not part of data returned by the query.  4b) Edit the query to read: select \* from deot; (a deliberate error) and click *Run* *Statement* again.  Your screen should now look like this:  Errors are shown in the result pane. The messages are usually meaningful and also attempt to identify the location.  **Read** **them** carefully.  On the other hand sometimes messages can be rather vague. If they are, ask tutors, or Google the Oracle error number for help !  SQL Workbook 8 September 2020 | |  |

Lab Week 1

4c) Close the result pane, retype the correct query on the second line, and leave the cursor on the third line, so that you have this:

- Click *Run* *Statement* with cursor in line 3 – nothing should happen.

- Click *Run* *Statement* with cursor in line 1 – you should receive the previous error - Click *Run* *Statement* with cursor in line 2 – you should receive the query output

Explanation: *Run* *Statement* is designed to run single statements only. If there is more than one, it will run the first statement or the statement containing the cursor. To run a sequence of statements use *Run* *Script* (see later).

**5)** **Create,** **populate,** **query,** **and** **examine** **a** **table**

5a) Create, Populate & Query the Vehicle table.

Clear the code window and close the Query Result pane.

In the SQL Worksheet, enter and run **each** of the following statements in turn. Note the messages returned in the Statement Output pane. Do not proceed if you get an error message. Clear each statement before entering the next.

Create table Vehicle (VehicleType char(10), CurrentValue number(8,2)); Insert into Vehicle values ('Bus', 12500.00);

Insert into Vehicle values ('Car', 9987); Select \* from vehicle;

You should be able to relate the messages to the actions which caused them.

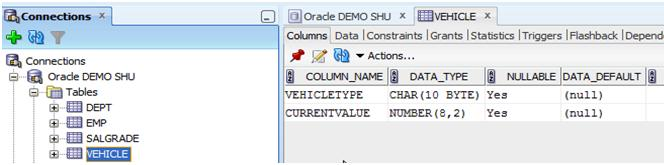
5b) Examine the Vehicle table

Click the Refresh icon in the Connections pane and see that the new table Vehicle appears in the list of tables.

Click on the Vehicle table, and see that a new tab (Vehicle) appears as below

SQL Workbook 9 September 2020

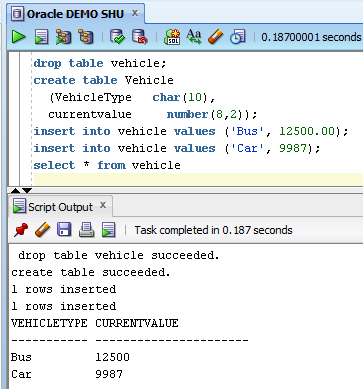
Lab Week 1

This tab contains several sub-tabs and the default is ‘Columns’ – showing the table column names and datatypes. Click on ‘Data’ to see the data rows in the table. Other tabs have little to show at the moment.

Click the red pushpin icon to ‘pin’ the Vehicle tab. Now click on the Dept table in the Connections pane, and see that another tab appears for the Dept table. Pinning these tabs is a useful way of flipping between multiple tables to compare data and or datatypes. Click the pushpin again to release the pin.

**6)** **Creating** **and** **running** **a** **script**

While it can be useful to run an individual statement when testing, applications (and DBA’s) often need to run a sequence of statements as a script.



1. Clear the code window.

2. Type: Drop table vehicle; on the first line, . . .

3. . . . and on the following lines type in the Create, Insert and Select statements used previously for the Vehicle table.

4. Click the *Run* *Script* button, and observe the output in the Script Output pane.

5. Click *Run* *Script* again, and notice that the script output is appended to previous output. Output can also be printed, or saved to disk.

Run Script button

SQL Workbook 10 September 2020

Lab Week 1

**7)** **Saving** **scripts,** **commenting** **scripts,** **and** **loading** **saved** **scripts**

If you intend to save a script, Oracle expects you to name it before you write it. So there is no ‘save’ option in the normal SQL Worksheet.

**To** **save** **a** **script**, select Save As from the File menu. Choose a filename (default is untitled1.sql) and a folder to store it in, then click Save. A new blank worksheet will be opened with the title of your chosen filename. Write (or copy from another worksheet) and/or run/test your script in this worksheet. To save changes either choose Save from the File menu, or try to close the sheet – it will prompt you to save changes.

**To** **load** an existing script, select Open from the File menu. Navigate to the script then click Open. The script will open in a new worksheet.

**Comments** can be included in a script to clarify and explain an SQL statement or procedure. The block comment starts with /\* and ends with \*/, preferably on their own lines, as:

/\*

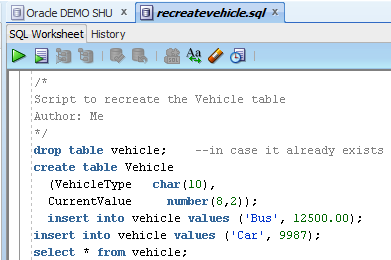
Comment out multiple full lines which are placed between the start and end points \*/

To comment out single lines, or the end of a line use a double-hyphen, e.g. -- this entire line is a comment

drop table vehicle -- this statement deletes the vehicle table

1. Choose Save As from the File menu, navigate to your F:\MyWork\Oracle folder. Set the filename as recreatevehicle.sql and click Save.

2. Copy/Paste the statements from the previous worksheet into this one.

3. Add some comments to the script.

4. Close the worksheet and Save Changes when prompted.

5. Choose Open from the File menu, navigate to the script and click Open – the script will be loaded in a new worksheet..

That completes an overview of the Oracle environment you will be using during this module. Take time to gain a clear understanding of it to save time and frustration later.

SQL Workbook 11 September 2020

Lab Week 1

You are now ready to start the exercises in this SQL Workbook.

1. Do not rush

2. Think what results you would expect from the statement(s) you are issuing and carefully check against the generated result.

3. Note your solutions, either in this workbook or in scripts (one script for each section of the workbook?). You will find it useful at times to refer back.

4. Make sure that you understand what is happening.

5. When you get errors, study your statement(s) and the data to determine a solution; try NOT to get into a 'hacking' mode.

6. Discuss problems and uncertainties with your tutors, they are there to help.

7. Maintain progress. It is very easy to lose contact with what is being discussed. Understanding one section is very dependent on understanding previous ones.

SQL Workbook 12 September 2020

Section A

**CREATING, POPULATING & DROPPING TABLES**

**Data Types**

Tables must be given a unique name (within your SQL database) and are made up of a number of columns. Each column must have a name and a data type assigned to it. Column names must be unique within the table although they can be duplicated in other tables. The declaration of the data type determines the operations that may be performed on the data. E.g. string variables will be sorted alphabetically, i.e. in ASCII code order, and numeric data will be sorted numerically.

The main data types supported by the version of SQL used on the course are shown in the table below:-

|  |  |
| --- | --- |
| Data Type | Description |
| NUMBER(p, s) or  NUMBER | Signed numerical value having size p as total number of digits of which s are decimal places.  Examples: NUMBER(7,2) will hold -99,999.99 to +99,999.99 NUMBER(7) will store -9,999,999 to + 9,999,999 |
| CHAR(n) | Fixed length character string of n characters. 0<n<2001, default n=1. |
| VARCHAR2(n) | Variable length character string, having a maximum length of n. 0<n<4001, no default value so n must be specified. |
| DATE | Dates ranging from 01/01/4712 BC to 31/12/4712 AD, and time. Default time is midnight (12:00:00 AM). |

**NB:** Single quote marks must be placed around any character string being inserted into or compared against a column of CHAR, VARCHAR2 or DATE data type.

**Naming Rules**

These naming rules apply to tables, views and columns. • must start with a letter.

• may be between 1 and 30 characters long.

• may contain alphabetic and numeric characters A to Z, a to z, 0 to 9. • they are NOT CASE SENSITIVE

• cannot be a reserved word • may contain underscores

• May contain $ and # (but not recommended).

SQL Workbook 13 September 2020

Section A

**The CREATE statement**

The CREATE statement creates the structure for a table.

The basic syntax for the CREATE statement is:

CREATE TABLE *table\_name* (

*column\_name1* *data\_type*(*size*), *column\_name2* *data\_type*(*size*), *column\_name3* *data\_type*(*size*), ....

);

**Example**

**CREATE** **TABLE** DEPT **(**

DEPTNO **NUMBER**(2) , DNAME **VARCHAR2**(14) , LOC **VARCHAR2**(13)

**)** ;

Note that this simple version of the CREATE statement does not define integrity constraints (neither entity integrity nor referential integrity). These are dealt with in a later section.

**Exercises**

**A1** Use this syntax to create the three tables which implement the Bank Account System described in the Introduction.

Note - column ACCNO should be defined as numeric.

|  |
| --- |
| **CREATE TABLE CUST**  **(**  **REFNO VARCHAR2(4),**  **NAME VARCHAR2(15),**  **ADDRESS VARCHAR2(30),**  **AREA VARCHAR2(20)**  **);** |
| **CREATE TABLE CUSTACC**  **(**  **REFNO VARCHAR2(4),**  **ACCNO NUMBER(9)**  **);** |
| **CREATE TABLE ACC**  **(**  **ACCNO NUMBER(9),**  **BALANCE NUMBER**  **);** |

SQL Workbook 14 September 2020

Section A

**The DROP statement**

This enables you to delete a table that already exists. The syntax is:

**DROP** **TABLE** table\_name ; .

**Oracle recyclebin**

**Example** **DROP** **TABLE** CUST ;

Information about objects which have been dropped is placed in a ‘recyclebin’ data dictionary table. A dropped object will have an object\_name like BIN$45azVqPcJrHgNAADugjeBA==$0:

To produce a list of all deleted tables with their original name:

**SELECT** \* **FROM** RECYCLEBIN;

To restore a table run the following command:

**FLASHBACK** **TABLE** tablename TO BEFORE DROP

To permanently remove all of them run the following command:

**PURGE** RECYCLEBIN;

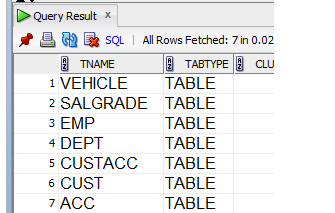
**The DESCRIBE command**

DESCRIBE (or DESC) can be used to check the structure of Tables and Views, once you have created them. The DESCRIBE command can also be used to display information about views, and other objects, as well as tables.

**DESCRIBE** object\_name (e.g.**DESC** DEPT;)

- but, it is easier to use the object browser and tabs in SQL Developer

To identify all your tables and views, use the special table TAB

**SELECT** \* **FROM** TAB ;

SQL Workbook 15 September 2020

Section A

**The INSERT statement**

The INSERT statement inserts a row of data into a table. There are two forms of the INSERT statement:

**Form** **1**

This form specifies data values only;

INSERT INTO *table\_name* VALUES (*value1*,*value2*,*value3*,...);

**Example** **INSERT** **INTO** DEPT

**VALUES** (60, 'PURCHASING', ’BOSTON’);

This form is designed to insert a value into every column in the table. In this form: - values must be present for each column

- values must be specified in the same sequence as the columns appear in the table definition

- values must be of the correct data type.

Both forms of the INSERT statement enter only **one** row at a time. Note the mandatory single quote marks around character strings; this also applies when entering dates.

**Form** **2**

The second form specifies both the column names and the values to be inserted:

INSERT INTO *table\_name* (*column1*,*column2*,*column3*,...) VALUES (*value1*,*value2*,*value3*,...);

This form can be used to insert values for some columns and not others.

**Example** **INSERT** **INTO** DEPT (DEPTNO, LOC, DNAME) **VALUES** (50, ‘NEW YORK’, ‘PERSONNEL’);

The order of column names and data values must match (but can be different from the order in the table definition).

This example will insert a new row with department number set to 70 and the remaining columns will be left empty (null).

**Example** **INSERT** **INTO** DEPT (DEPTNO) **VALUES** (70);

Note: To explicitly specify a null value for a column, use the keyword NULL.

In this example the department name is set to null.

. . . VALUES (70,,’BOSTON’) will not work.

**Example** **INSERT** **INTO** DEPT

**VALUES** (70, NULL, ‘BOSTON’');

SQL Workbook 16 September 2020

Section A

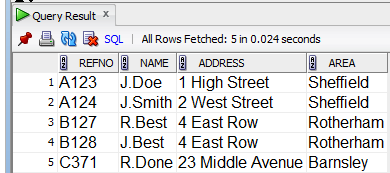
**Exercises**

**A2** Populate the Accounting System tables that you created in Exercise A1 with the data as given on Page 4. Give an example of one insert statement for each table in the boxes below.

|  |
| --- |
| **INSERT INTO CUST**  **VALUES('A123', 'J Doe', '1 High Street', 'Sheffield');** |
| **INSERT INTO CUSTACC**  **VALUES('A123', '1245890');** |
| **INSERT INTO ACC**  **VALUES('1245890', 234.50, 'Broomhill', TO\_DATE('12/11/2003', 'DD/MM/YYYY'), 100.00);** |

Retrieving the contents of your table is the subject of the next main section but a simple version for now is:

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

**SELECT** \* **FROM** CUSTACC;

**SELECT** \* **FROM** ACC;

SQL Workbook 17 September 2020

Section A

**The DELETE statement**

The DELETE statement is used to remove rows from a table. The syntax is

DELETE FROM *table\_name* WHERE *some\_column*=*some\_value*;

**Example** **DELETE** **FROM** DEPT **WHERE** DEPTNO = 60 ;

**NB:** DELETE will remove all of the rows that satisfy the WHERE clause. If there is no WHERE clause it will delete **ALL** **OF** **THE** **ROWS** **IN** **THE** **TABLE**.

**Exercise**

**A3** The bank has decided that J Smith is no longer a customer. Delete all the information about her from the database.

DELETE FROM CUST

WHERE NAME = “J Smith”;

**The UPDATE statement**

The **UPDATE** statement is used to alter data values for existing rows in a table. The syntax is:

UPDATE *table\_name*

SET *column1*=*value1*,*column2*=*value2*,... WHERE *some\_column*=*some\_value*;

**Example** **UPDATE** EMP

**SET** MGR = 7839 **WHERE** JOB = 'SALESMAN' ;

UPDATE is similar to DELETE in that it can operate on a set of rows that meet a specific condition, not just a single row.

**Exercise**

**A4** J Best has moved to 31 Hanover Street, Chapeltown. Amend the data in the Customer table to reflect this change.

UPDATE cust

SET address = '31 Hanover Street', area = 'Chapeltown'

WHERE NAME = 'J Best';

SQL Workbook 18 September 2