LAB BOOK 2

COMP41090 SQL PROGRAMMING

ANDREW DOYLE
STUDENT NUMBER: 12252388
MSc COMPUTER SCIENCE (CONVERSION)

02nd April 2013

Table of Contents

TABLE STRUCTURE

```
SQL> DESCRIBE employee;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 EMP_ID                                    NOT NULL NUMBER(10)
 FNAME                                              NVARCHAR2(20)
 LNAME                                              NVARCHAR2(20)
 MANAGER_EMP_ID                                     NUMBER(10)

SQL> DESCRIBE customer;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 CUST_NBR                                  NOT NULL NUMBER(10)
 FNAME                                              NVARCHAR2(20)
 LNAME                                              NVARCHAR2(20)

SQL> DESCRIBE cust_order;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 ORDER_NBR                                 NOT NULL NUMBER(10)
 CUST_NBR                                  NOT NULL NUMBER(10)
 SALES_EMP_ID                              NOT NULL NUMBER(10)
 SALE_PRICE                                         NUMBER(10,2)
```

TABLES CREATED

```
SQL> SELECT * FROM cust_order;

 ORDER_NBR   CUST_NBR SALES_EMP_ID SALE_PRICE
---------- ---------- ------------ ----------
      1000        100          300     400.99
      1001        100          301        800
      1002        100          302      90.99
      1003        101          303        200
      1004        101          300       1000
      1005        101          301       78.1
      1006        102          302     330.25
      1007        102          303      890.5
      1008        102          300        220
      1009        103          301       1300
      1010        103          302      99.99

 ORDER_NBR   CUST_NBR SALES_EMP_ID SALE_PRICE
---------- ---------- ------------ ----------
      1011        103          303      300.5
      1012        104          300     770.25
      1013        104          301        230
      1014        104          302       90.1
      1015        105          303        143
      1016        105          300     184.99
      1017        105          301      988.1
      1018        106          302       34.5
      1019        106          303      23.99
      1020        107          300     189.25
      1021        107          301        412

 ORDER_NBR   CUST_NBR SALES_EMP_ID SALE_PRICE
---------- ---------- ------------ ----------
      1022        108          302      231.5
      1023        108          303     444.99
      1024        109          300       12.1
      1025        109          301      129.5

26 rows selected.

SQL> SELECT * FROM employee;

    EMP_ID FNAME                LNAME                MANAGER_EMP_ID
---------- -------------------- -------------------- --------------
       304 Reno                 Lopez                           304
       305 Stewart              Fulbright                       305
       300 Jason                Chase                           304
       301 James                Mason                           304
       302 Mila                 Freeman                         305
       303 Michael              Berry                           305

6 rows selected.

SQL> SELECT * FROM customer;

  CUST_NBR FNAME                LNAME
---------- -------------------- --------------------
       100 John                 Smith
       101 David                Williams
       102 Angelina             Wolf
       103 Natalie              Clarrins
       104 Carl                 Sagan
       105 Renata               Jones
       106 Julie                DeValera
       107 Bruce                Ezell
       108 Mark                 Pruitt
       109 Nigel                Kennedy

10 rows selected.
```

QUERY 1

```
SELECT cust_order.order_nbr, employee.fname AS Employee_First_Name, employee.lname AS
Employee_Surname
FROM employee INNER JOIN cust_order ON employee.emp_id = cust_order.sales_emp_id
ORDER BY cust_order.order_nbr;
```

```
SQL> SELECT cust_order.order_nbr, employee.fname AS Employee_First_Name, employee.lname AS Employee_Surname
  2  FROM employee INNER JOIN cust_order ON employee.emp_id = cust_order.sales_emp_id
  3  ORDER BY cust_order.order_nbr;

 ORDER_NBR EMPLOYEE_FIRST_NAME  EMPLOYEE_SURNAME
---------- -------------------- --------------------
      1000 Jason                Chase
      1001 James                Mason
      1002 Mila                 Freeman
      1003 Michael              Berry
      1004 Jason                Chase
      1005 James                Mason
      1006 Mila                 Freeman
      1007 Michael              Berry
      1008 Jason                Chase
      1009 James                Mason
      1010 Mila                 Freeman

 ORDER_NBR EMPLOYEE_FIRST_NAME  EMPLOYEE_SURNAME
---------- -------------------- --------------------
      1011 Michael              Berry
      1012 Jason                Chase
      1013 James                Mason
      1014 Mila                 Freeman
      1015 Michael              Berry
      1016 Jason                Chase
      1017 James                Mason
      1018 Mila                 Freeman
      1019 Michael              Berry
      1020 Jason                Chase
      1021 James                Mason

 ORDER_NBR EMPLOYEE_FIRST_NAME  EMPLOYEE_SURNAME
---------- -------------------- --------------------
      1022 Mila                 Freeman
      1023 Michael              Berry
      1024 Jason                Chase
      1025 James                Mason

26 rows selected.
```

This query lists each order number (from the **cust_order** table) alongside the first name and surname (from the **employee** table). In the **SELECT** statement, the **AS** keyword is used to give descriptive column headings. Without the **AS** keyword, the column called *EMPLOYEE_FIRST_NAME* would instead be titled *FNAME* which is not descriptive enough.

The **FROM** statement selects the tables from which the information is to be taken from. The **employee** table is joined to the **cust_order** table (**INNER JOIN)** by the **emp_id** attributes of both tables (using the **ON** keyword).

Finally the results are ordered by the order number using the **ORDER BY** statement. This statement is not strictly required in this query.

QUERY 2

```
SELECT a.fname Emp_Name, a.lname Emp_Surname, b.fname Manager_Name, b.lname
Manager_Surname
FROM employee a JOIN employee b ON a.manager_emp_id = b.emp_id;
```

```
SQL> SELECT a.fname Emp_Name, a.lname Emp_Surname, b.fname Manager_Name, b.lname Manager_Surname
  2  FROM employee a JOIN employee b ON a.manager_emp_id = b.emp_id;

EMP_NAME             EMP_SURNAME          MANAGER_NAME         MANAGER_SURNAME
-------------------- -------------------- -------------------- --------------------
Reno                 Lopez                Reno                 Lopez
Stewart              Fulbright            Stewart              Fulbright
Jason                Chase                Reno                 Lopez
James                Mason                Reno                 Lopez
Mila                 Freeman              Stewart              Fulbright
Michael              Berry                Stewart              Fulbright

6 rows selected.
```

This query demonstrates a self join on the employee table, which outputs the employee`s name alongside their manager`s name. Since a join is being carried out within a table, table aliases are required.

For example, in the **SELECT** statement, *a.fname Emp_Name* selects fname (as table alias **a**) and gives it the descriptive column heading *Emp_Name*. The first name and last name of the managers are selected under the table alias **b**.

The **FROM** statement joins employee (alias **a**) with employee (alias **b**) by **manager_emp_id** and **emp_id** using the **ON** keyword.

QUERY 3

```
SELECT cust_order.order_nbr AS Order_Number, employee.fname AS Employee_First_Name,
employee.lname AS Employee_Surname
FROM employee FULL OUTER JOIN cust_order ON employee.emp_id = cust_order.sales_emp_id;
```

```
SQL> SELECT cust_order.order_nbr AS Order_Number, employee.fname AS Employee_Fi
st_Name, employee.lname AS Employee_Surname
  2  FROM employee FULL OUTER JOIN cust_order ON employee.emp_id = cust_order.s
les_emp_id;

ORDER_NUMBER EMPLOYEE_FIRST_NAME  EMPLOYEE_SURNAME
------------ -------------------- --------------------
        1000 Jason                Chase
        1001 James                Mason
        1002 Mila                 Freeman
        1003 Michael              Berry
        1004 Jason                Chase
        1005 James                Mason
        1006 Mila                 Freeman
        1007 Michael              Berry
        1008 Jason                Chase
        1009 James                Mason
        1010 Mila                 Freeman

ORDER_NUMBER EMPLOYEE_FIRST_NAME  EMPLOYEE_SURNAME
------------ -------------------- --------------------
        1011 Michael              Berry
        1012 Jason                Chase
        1013 James                Mason
        1014 Mila                 Freeman
        1015 Michael              Berry
        1016 Jason                Chase
        1017 James                Mason
        1018 Mila                 Freeman
        1019 Michael              Berry
        1020 Jason                Chase
        1021 James                Mason

ORDER_NUMBER EMPLOYEE_FIRST_NAME  EMPLOYEE_SURNAME
------------ -------------------- --------------------
        1022 Mila                 Freeman
        1023 Michael              Berry
        1024 Jason                Chase
        1025 James                Mason
             Reno                 Lopez
             Stewart              Fulbright

28 rows selected.
```

This query uses an outer join to list the order numbers alongside the employees who are associated with that order number. Employees who have never had a sale are also listed since it is a full outer join. (Simply using the **JOIN** keyword would not output Reno Lopez and Stewart Fulbright).

The relevant attributes are selected and renamed using the **AS** keyword. The **employee** table is joined to the **cust_order** table by **emp_id** (employee table) and **sales_emp_id** (cust_order table), using the **ON** keyword as before.

QUERY 4

```
SELECT Sum(cust_order.sale_price) AS Total_Sale_Price_Of_All_Orders
FROM cust_order;
```

```
SQL> SELECT Sum(cust_order.sale_price) AS Total_Sale_Price_Of_All_Orders
  2  FROM cust_order;

TOTAL_SALE_PRICE_OF_ALL_ORDERS
------------------------------
                       9595.59
```

This query demonstrates the ability to sum values of an attribute; using the syntax `Sum(attribute).` In this case, the *sale_price* of the **cust_order** table is totalled and the answer is given the descriptive column heading of **Total_Sale_Price_Of_All_Orders**.

QUERY 5

```
SELECT Avg(cust_order.sale_price) AS Average_Sale_Price
FROM cust_order;
```

```
SQL> SELECT Avg(cust_order.sale_price) AS Average_Sale_Price
  2  FROM cust_order;

AVERAGE_SALE_PRICE
------------------
        369.061154
```

This query demonstrates the ability to calculate the average value of an attribute; using the syntax `Avg(attribute).` The calculated average is output using the descriptive column heading of **Average_Sale_Price**.

QUERY 6

```
SELECT customer.fname AS Customer_First_Name, customer.lname AS Customer_Last_Name,
Sum(cust_order.sale_price) AS Total_Sale_Price_Per_Customer
FROM customer INNER JOIN cust_order ON customer.cust_nbr = cust_order.cust_nbr
GROUP BY customer.fname, customer.lname
ORDER BY customer.fname;
```

```
10 rows selected.

SQL> SELECT customer.fname AS Customer_First_Name, customer.lname AS Customer_Last_Name, Sum(cust_order.sale_price)
 AS Total_Sale_Price_Per_Customer
  2  FROM customer INNER JOIN cust_order ON customer.cust_nbr = cust_order.cust_nbr
  3  GROUP BY customer.fname, customer.lname
  4  ORDER BY customer.fname;

CUSTOMER_FIRST_NAME  CUSTOMER_LAST_NAME   TOTAL_SALE_PRICE_PER_CUSTOMER
-------------------- -------------------- ----------------------------
Angelina             Wolf                                     1440.75
Bruce                Ezell                                     601.25
Carl                 Sagan                                    1090.35
David                Williams                                  1278.1
John                 Smith                                    1291.98
Julie                DeValera                                    58.49
Mark                 Pruitt                                    676.49
Natalie              Clarrins                                 1700.49
Nigel                Kennedy                                     141.6
Renata               Jones                                    1316.09

10 rows selected.
```

This query displays the total sales price of all orders from each customer. Fname and lname are selected from the **customer** table and renamed to give a more appropriate column heading. The **sum** of the sale_price attribute from the **cust_order** table is also selected and renamed.

The **customer** and **cust_order** table are joined by the **cust_nbr** attribute. The result is grouped by the customers first and last name, and ordered by the customers first name.

QUERY 7

```
SELECT customer.fname AS Customer_First_Name, customer.lname AS Customer_Last_Name,
Sum(cust_order.sale_price) AS Total_Sale_Price_Per_Customer
FROM customer INNER JOIN cust_order ON customer.cust_nbr = cust_order.cust_nbr
GROUP BY customer.fname, customer.lname HAVING Sum(cust_order.sale_price) > 1000
ORDER BY customer.fname;
```

```
SQL> SELECT customer.fname AS Customer_First_Name, customer.lname AS Customer_Last_Name, Sum(cust_order.sale_price)
 AS Total_Sale_Price_Per_Customer
  2  FROM customer INNER JOIN cust_order ON customer.cust_nbr = cust_order.cust_nbr
  3  GROUP BY customer.fname, customer.lname HAVING Sum(cust_order.sale_price) > 1000
  4  ORDER BY customer.fname;

CUSTOMER_FIRST_NAME  CUSTOMER_LAST_NAME   TOTAL_SALE_PRICE_PER_CUSTOMER
-------------------- -------------------- ----------------------------
Angelina             Wolf                                     1440.75
Carl                 Sagan                                    1090.35
David                Williams                                  1278.1
John                 Smith                                    1291.98
Natalie              Clarrins                                 1700.49
Renata               Jones                                    1316.09

6 rows selected.
```

This query is almost identical to query 6, with an extra stipulation added which results in only customers who spent more than 1000 being included in the output. This is completed with the addition of the **HAVING** statement with the condition being the sum of the sale price (in the **cust_order** table) is greater than 1000.

The **HAVING** clause is similar to the **WHERE** clause (which is applied to rows); however **HAVING** is applied to groups and can use aggregates.

QUERY 8

```
SELECT employee.fname, cust_order.cust_nbr,SUM(cust_order.sale_price) AS Sale_Price
FROM employee JOIN cust_order ON employee.emp_id=cust_order.sales_emp_id
GROUP BY CUBE(employee.fname, cust_order.cust_nbr);
```

```
SQL> SELECT employee.fname, cust_order.cust_nbr,SUM(cust_order.sale_price) AS Sale_Price
  2  FROM employee JOIN cust_order ON employee.emp_id=cust_order.sales_emp_id
  3  GROUP BY CUBE(employee.fname, cust_order.cust_nbr);

FNAME                CUST_NBR SALE_PRICE
-------------------- ---------- ----------
                                  9595.59
                          100    1291.98
                          101     1278.1
                          102    1440.75
                          103    1700.49
                          104    1090.35
                          105    1316.09
                          106      58.49
                          107     601.25
                          108     676.49
                          109      141.6

FNAME                CUST_NBR SALE_PRICE
-------------------- ---------- ----------
Mila                              877.33
Mila                      100      90.99
Mila                      102     330.25
Mila                      103      99.99
Mila                      104       90.1
Mila                      106       34.5
Mila                      108      231.5
James                            3937.7
James                     100        800
James                     101       78.1
James                     103       1300

FNAME                CUST_NBR SALE_PRICE
-------------------- ---------- ----------
James                     104        230
James                     105      988.1
James                     107        412
James                     109      129.5
Jason                           2777.58
Jason                     100     400.99
Jason                     101       1000
Jason                     102        220
Jason                     104     770.25
Jason                     105     184.99
Jason                     107     189.25

FNAME                CUST_NBR SALE_PRICE
-------------------- ---------- ----------
Jason                     109       12.1
Michael                         2002.98
Michael                   101        200
Michael                   102      890.5
Michael                   103      300.5
Michael                   105        143
Michael                   106      23.99
Michael                   108     444.99

41 rows selected.
```

This query demonstrates the use of a **CUBE** query on two columns from two tables, namely *fname* from the **employee** table and *cust_nbr* from the **cust_order** table. The two tables are joined by the **emp_id** attribute in both tables .

QUERY 9

```
SELECT employee.manager_emp_id, cust_order.order_nbr,SUM(cust_order.sale_price) AS
Sale_Price
FROM employee JOIN cust_order ON employee.emp_id=cust_order.sales_emp_id
GROUP BY ROLLUP(employee.manager_emp_id, cust_order.order_nbr);
```

```
SQL> SELECT employee.manager_emp_id, cust_order.order_nbr,SUM(cust_order.sale_price) AS Sale_Price
  2  FROM employee JOIN cust_order ON employee.emp_id=cust_order.sales_emp_id
  3  GROUP BY ROLLUP(employee.manager_emp_id, cust_order.order_nbr);

MANAGER_EMP_ID  ORDER_NBR SALE_PRICE
-------------- ---------- ----------
           304       1000     400.99
           304       1001        800
           304       1004       1000
           304       1005       78.1
           304       1008        220
           304       1009       1300
           304       1012     770.25
           304       1013        230
           304       1016     184.99
           304       1017      988.1
           304       1020     189.25

MANAGER_EMP_ID  ORDER_NBR SALE_PRICE
-------------- ---------- ----------
           304       1021        412
           304       1024       12.1
           304       1025      129.5
           304                6715.28
           305       1002      90.99
           305       1003        200
           305       1006     330.25
           305       1007      890.5
           305       1010      99.99
           305       1011      300.5
           305       1014       90.1

MANAGER_EMP_ID  ORDER_NBR SALE_PRICE
-------------- ---------- ----------
           305       1015        143
           305       1018       34.5
           305       1019      23.99
           305       1022      231.5
           305       1023     444.99
           305                2880.31
                              9595.59

29 rows selected.
```

This query demonstrates the use of a **ROLLUP** query on *manager_emp_id* from the **employee** table and *order_nbr* from the **cust_order** table.