

# **Table of Contents**

Question 1	2
Figure 1.1 – Inserting new information	2
Figure 1.2 – Display new information added	3
Question 2	4
Figure 2.1 – Demonstration of rollback functionality	4
Question 3	6
Figure 3.1 - Before Rollback:	6
Figure 3.2 – Demonstration that information was not added	7
Figure 3.3 – Clarification that after rollback the information is the same	8
Figure 3.4 – Commiting the information	8
Figure 3.5 – Option 2 Before Rollback	9
Figure 3.7 – Table after rollback	10
Figure 3.5 – Table before Rollback	10
Figure 3.6 - Rollback	10
Question 4	11
Figure 4.1 – Creating a new user	11
Figure 4.2 – Granting connect and session privilege to new user	11
Figure 4.3 – Granting required privileges and demonstration of their use	11
Figure 4.4 – Double Session	12
Question 5	13
Figure 5.1 – Revoking Privilege	13
Figure 5.2 – Attempting to delete	13
Question 6	14
Figure 6.1 - Output of customer with smallest order	14
Figure 6.2 - Cust_order table to verify correct information obtained:	14

Figure 1.1 - Inserting new information

```
SQL> SET AUTOCOMMIT OFF;
SQL> INSERT INTO customer (cust_nbr, fname, lname) VALUES ( 110, 'Wayne', 'Roone
y');
SQL> INSERT INTO customer (cust_nbr, fname, lname) VALUES ( 111, 'David','Beckha
m');
  row created.
SQL> INSERT INTO employee (emp_id, fname, lname, MANAGER_EMP_ID> VALUES( 401,'Ga
reth', 'Bale', 300);
1 row created.
SQL> INSERT INTO cust_order (order_nbr, cust_nbr, sales_emp_id, sale_price) VALU
ES(1026, 110, 401, 1200.23);
1 row created.
SQL> INSERT INTO cust_order (order_nbr, cust_nbr, sales_emp_id, sale_price) VALU
ES<1027, 111, 401, 1100.49);
1 row created.
SQL> COMMIT;
Commit complete.
SQL> SELECT * FROM customer;
   CUST_NBR FNAME
                                            LNAME
         100 John
101 David
                                            Smith
Williams
Wolf
         100 John
101 David
102 Angelina
103 Natalie
104 Carl
105 Renata
106 Julie
107 Bruce
108 Mark
                                            Clarrins
                                            Sagan
                                            Jones
                                            DeValera
Ezell
Pruitt
         109
110
               Nigel
                                             Kennedy
              Wayne
                                             Rooney
   CUST_NBR FNAME
                                            LNAME
          111 David
                                            Beckham
12 rows selected.
SQL> SELECT * FROM employee;
     EMP_ID FNAME
                                                                          MANAGER_EMP_ID
                                            LNAME
                                                                                         304
305
304
304
                                             Lopez
Fulbright
               Reno
          305 Stewart
300 Jason
301 James
302 Mila
303 Michael
                                             Chase
                                            Mason
                                                                                         305
                                             Freeman
                                             Berry
                                                                                         305
          401 Gareth
                                             Bale
                                                                                         300
  rows selected.
```

### **SQL USED**

Please Note: SQL is also included within the text file "sql.txt" included with the zip file.

```
1 | SET AUTOCOMMIT OFF;
2 | INSERT INTO customer (cust_nbr, fname, lname) VALUES ( 110, 'Wayne', 'Rooney');
3 | INSERT INTO customer (cust_nbr, fname, lname) VALUES ( 111, 'David', 'Beckham');
4 | INSERT INTO employee (emp_id, fname, lname, MANAGER_EMP_ID) VALUES( 401, 'Gareth', 'Bale', 300);
5 | INSERT INTO cust_order (order_nbr, cust_nbr, sales_emp_id, sale_price) VALUES(1026, 110, 401, 1200.23);
6 | INSERT INTO cust_order (order_nbr, cust_nbr, sales_emp_id, sale_price) VALUES(1027, 111, 401, 1100.49);
7 | COMMIT;
8 | SELECT * FROM customer;
9 | SELECT * FROM employee;
10 | SELECT * FROM cust_order;
```

In line 1, **AUTOCOMMIT** is set to **OFF** in order to prevent the SQL statements from automatically implementing changes to the database. Two new customers, one employee, and two orders are inserted on Lines 2 – 6. Line 7 commits the data. Lines 8 - 10 demonstrate that the data was added successfully (see Figure 1.2).

Figure 1.2 - Display new information added

SQL> SELECT	* FROM cus	t_order;		
ORDER_NBR	CUST_NBR	SALES_EMP_ID	SALE_PRICE	CATEGORY
1000 1001	100 100		400.99 800	
1002	100		90.99	
1003	101	303	200	
1004	101		1000	
1005	101	301	78.1	
1006	102	302	330.25	
1007	102	303	890.5	
1008	102	300	890.5 220 1300	
1009	103	301	1300	
1010	103	302	99.99	
ORDER_NBR	CUST_NBR	SALES_EMP_ID	SALE_PRICE	CATEGORY
1011	103	303	300.5	
1012	104	300	770.25	
1013	104	301 302	230 90.1	
1014	104	302	90.1	
1015	105	303	143	
1016	105	300	184.99	
1017	105	101 202	988.1 34.5 23.99	
1018 1019	106 106	30Z	24.5	
	107	383 383	189.25	
1021	107	301		
1621	101	201	112	
ORDER_NBR	CUST_NBR	${\tt SALES\_EMP\_ID}$	SALE_PRICE	CATEGORY
1022			231.5	
1023	108	303	444.99	
1024	109	300	12.1 129.5	
1025	109	301	129.5	
	110		1200.23	
1027	111	401	1100.49	
28 rows selected.				

Figure 2.1 - Demonstration of rollback functionality

```
SQL>
        SET AUTOCOMMIT OFF;
INSERT INTO customer (cust_nbr, fname, lname) VALUES ( 112, 'Kim', 'Jong-un
  ۶;
   row created.
SQL> INSERT INTO cust_order (order_nbr, cust_nbr, sales_emp_id, sale_price) VALU
ES<1028, 112, 401, 200.20);
1 row created.
SQL> ROLLBACK;
Rollback complete.
SQL > SELECT * FROM customer;
    CUST_NBR FNAME
                                                       LNAME
            100 John
101 David
102 Angelina
103 Natalie
104 Carl
105 Renata
                                                       Smith
Williams
Wolf
Clarrins
                                                       Sagan
                                                       Jones
DeValera
Ezell
Pruitt
            106
107
108
                  Julie
Bruce
            108 Mark
109 Nigel
110 Wayne
                                                       Kennedy
                                                       Rooney
    CUST_NBR_FNAME
                                                       LNAME
            111 David
                                                       Beckham
12 rows selected.
SQL> SELECT * FROM cust_order;
                      CUST_NBR SALES_EMP_ID SALE_PRICE CATEGORY
  ORDER_NBR
                                                                   400.99
800
90.99
200
1000
78.1
          1000
1001
1002
1003
1004
1005
                               100
100
100
101
101
                                                     300
301
302
303
300
                               \overline{101}
                                                     301
          1006
1007
1008
                               \frac{102}{102}
                                                     302
                                                                    890.5
220
1300
99.99
                                                     303
300
                               \overline{102}
                                                     301
302
                               103
           1009
           1010
                               103
  ORDER_NBR
                      CUST_NBR SALES_EMP_ID SALE_PRICE CATEGORY
          1011
1012
1013
1014
1015
                               103
104
104
104
105
                                                     303
300
301
                                                                   300.5
770.25
230
                                                     302
303
300
301
                                                                      90.1
143
           1016
                                                                   184.99
988.1
                               105
105
           1017
                                                     302
303
300
          1018
1019
                               106
106
107
           1020
                               107
           1021
                                                     301
  ORDER_NBR
                      CUST_NBR SALES_EMP_ID SALE_PRICE CATEGORY
                                                                 231.5
444.99
12.1
129.5
1200.23
1100.49
                               108
108
109
109
           1022
                                                     302
          1023
1024
1025
                                                     303
300
301
                               110
111
          1026
1027
                                                     401
401
28 rows selected.
```

# **SQL USED**

1| SET AUTOCOMMIT OFF;
2| INSERT INTO customer (cust\_nbr, fname, Iname) VALUES ( 112, 'Kim', 'Jong-un');
3| INSERT INTO cust\_order (order\_nbr, cust\_nbr, sales\_emp\_id, sale\_price) VALUES(1028, 112, 401, 200.20);
4| ROLLBACK;
5| SELECT \* FROM customer;
6| SELECT \* FROM cust\_order;

Autocommit is set to off to prevent information being automatically saved to the database. Line 2-3 inserts a new customer and a new order. Line 4 uses the **ROLLBACK** keyword to disregard the previous **INSERT** statements. Lines 5-6 demonstrate that the information was not inserted into the tables (see Figure 2.1).

### **SQL USED:**

```
1 | INSERT INTO cust_order (order_nbr, cust_nbr, sales_emp_id, sale_price) VALUES(1036, 111, 401, 333.44);
2 | INSERT INTO cust_order (order_nbr, cust_nbr, sales_emp_id, sale_price) VALUES(1037, 110, 401, 76.65);
3 | SAVEPOINT qthree;
4 | INSERT INTO cust_order (order_nbr, cust_nbr, sales_emp_id, sale_price) VALUES(1036234567864, 110, 401, 22.54);
5 | SELECT * FROM cust_order;
6 | ROLLBACK TO qthree;
7 | SELECT * FROM cust_order;
8 | COMMIT;
```

- In lines 1-2, two new orders are inserted.
- On line 3, a SAVEPOINT is created which saves a snapshot of the state of the database.
- In line 4, an order number is inserted that exceeds the maximum number length, therefore the order fails (see Figure 3.1).
- Line 5 demonstrates that the order was not entered (Figure 3.2).
- Line 6 returns the state of the database to the SAVEPOINT qthree using ROLLBACK.

# Figure 3.1 - Before Rollback:

```
SQL> INSERT INTO cust_order (order_nbr, cust_nbr, sales_emp_id, sale_price) VALUES(1036, 111, 401, 333.44);

1 row created.

SQL> INSERT INTO cust_order (order_nbr, cust_nbr, sales_emp_id, sale_price) VALUES(1037, 110, 401, 76.65);

1 row created.

SQL> SAVEPOINT qthree;

Savepoint created.

SQL> INSERT INTO cust_order (order_nbr, cust_nbr, sales_emp_id, sale_price) VALUES(1036234567864, 110, 401, 22.54);
INSERT INTO cust_order (order_nbr, cust_nbr, sales_emp_id, sale_price) VALUES(1036234567864, 110, 401, 22.54);
INSERT INTO cust_order (order_nbr, cust_nbr, sales_emp_id, sale_price) VALUES(1036234567864, 110, 401, 22.54)

ERROR at line 1:

ORA-01438: value larger than specified precision allowed for this column
```

Figure 3.2 - Demonstration that information was not added

Kuri SQL Com	mand Line			_ D X
SQL> SELECT	* FROM cu	st_order;		A
ORDER_NBR	CUST_NBF	SALES_EMP_ID	SALE_PRICE	CATEGORY
1000	100	300	400.99	
1001	100		800	
1002 1003	100 101		90.99 200	
1004	101		1000	
1005	101		78.1	
1006 1007	102 102		330.25	
1008	102		890.5 220	
1009	103	301	1300	
1010	103	302	99.99	
ORDER_NBR	CUST_NBF	SALES_EMP_ID	SALE_PRICE	CATEGORY
1011	103		300.5 770.25	
1012 1013	104 104			
1013 1014	104 104		230 90.1	
1015	105		143	=
1016	105		184.99	
1017	105		988.1	
1018 1019	106 106		34.5 23.99	
1020	107		189.25	
1021	107	301	412	
ORDER_NBR	CUST_NBF	SALES_EMP_ID	SALE_PRICE	CATEGORY
1022	108	302	231.5	
1023	108			
1024 1025	109 109		12.1 129.5	
1025	110		1200.23	
1027	111		1100.49	
1028	111		222.25	
1029	119		100.99	
1030 1031	110 111		100.99 666.44	
1032	110		88.77	
ORDER_NBR	CUST_NBF	SALES_EMP_ID	SALE_PRICE	CATEGORY
1033	110		66.55	
1034	111		333.44	
1035 1036	110 111			
1037	116		76.65	
38 rows selected.				
SQL> ROLLBAC	K TO qthr			₹
1		!!!		

Figure 3.3 - Clarification that after rollback the information is the same

SQL> SELECT	* FROM cus	st_order;		
ORDER_NBR	CUST_NBR	SALES_EMP_ID	SALE_PRICE	CATEGORY
1000	100	300	400.99	
1001	100	301	_800	
1002	100	302	90.99	
1003	101	101 200	200	
1005	101	300 301	78.1	
1006	102	302	330.25	
1007	102	303	890.5	
1008	102	300	220	
1009	103	301	1300	
		300 301 302 303 300 301 302 303 300 301 302		
ORDER_NBR	CUST_NBR	SALES_EMP_ID	SALE_PRICE	CATEGORY
1011 1012	103 104 104	303	300.5	
1012	104	300	770.25	
1013	104	301	230	
1014 1015	104	302	70.1	
1016	105	300 202	194 99	
1017	105	301	988 1	
	106	302	34.5	
1018 1019	106	303	23.99	
1020	107	300	189.25	
1021	107	303 300 301 302 303 300 301 302 303 300	412	
ORDER_NBR	CUST_NBR	SALES_EMP_ID	SALE_PRICE	CATEGORY
1022	108	302	231.5	
1023	108	303	444.99	
1024	109	300	12.1	
1025	109	301	129.5	
1026	110	401	1200.23	
1027	111	401 401	2200.47	
1020	110	401 401	100.99	
1030	110	401	100.99	
1031	111	401	666.44	
1032	110	302 303 300 301 401 401 401 401 401 401	88.77	
ORDER_NBR	CUST_NBR	SALES_EMP_ID	SALE_PRICE	CATEGORY
1033	110	401 401 401 401 401 401	66.55	
1034	111	401	333.44	
1035	110	401	76.65	
1036	111	401	333.44	
1037	110	401	76.65	
38 rows sel	ected.			

Figure 3.4 - Committing the information

SQL> COMMIT; Commit complete.

#### **OPTION 2**

In option 2, outlined below, the savepoint is created after the failed insert (on line 4). This may be the more logical sequence of SQL code. Similarly, the database is returned to the state of the savepoint. Lines 5 and 7 demonstrate that the tables were not updated with the failed insert (see Figures 3.5 and 3.7).

```
1 | INSERT INTO cust_order (order_nbr, cust_nbr, sales_emp_id, sale_price) VALUES(1038, 111, 401, 333.44);
2 | INSERT INTO cust_order (order_nbr, cust_nbr, sales_emp_id, sale_price) VALUES(1039, 110, 401, 76.65);
3 | INSERT INTO cust_order (order_nbr, cust_nbr, sales_emp_id, sale_price) VALUES(1036234567864, 110, 401, 22.54);
4 | SAVEPOINT qthree3;
5 | SELECT * FROM cust_order;
6 | ROLLBACK TO qthree3;
7 | SELECT * FROM cust_order;
8 | COMMIT;
```

# Figure 3.5 - Option 2 Before Rollback

```
SQL> INSERT INTO cust_order (order_nbr, cust_nbr, sales_emp_id, sale_price) VALU
ES(1036234567864, 110, 401, 22.54);
INSERT INTO cust_order (order_nbr, cust_nbr, sales_emp_id, sale_price) VALUES(10
36234567864, 110, 401, 22.54)

ERROR at line 1:
ORA-01438: value larger than specified precision allowed for this column

SQL> SAVEPOINT qthree3;
Savepoint created.
```

Figure 3.5 - Table before Rollback

SQL> SELECT	* FROM cus	st_order;	
ORDER_NBR	CUST_NBR	SALES_EMP_ID	
1000 1001	100 100	300	400.99
1001	100	301	800
1002	100 101	302	90.99
1003	101	303	200
1004	101 101	300	1000
1005 1006	101	202 201	220 25
1007	102	302 303	230.23 290 5
1008	102	300	220
1009	103	301	1300
1010	103	302	400.99 800 90.99 200 1000 730.25 890.5 220 1300 99.99
ORDER_NBR	CUST_NBR	SALES_EMP_ID	SALE_PRICE
1011	103 104	303	300.5
1011 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 1019	106	30Z	34.5
1020	106 107	300 303	100 25
1021	107	301	300.5 770.25 230 90.1 143 184.99 988.1 34.5 23.99 189.25
ORDER_NBR	CUST_NBR	SALES_EMP_ID	SALE_PRICE
1022	108	302	231 5
1023	108 108	303	444 99
1024	109	300	12.1
1025	109	301	129.5
1026	110	401	1200.23
1027	$\overline{1}\overline{1}\overline{1}$	401	1100.49
1028	111	401	222.25
1029	1111	401	100.99
1030	110 111	401	100.99
1031 1032	111 110	401 401	231.5 444.99 12.1 129.5 1200.23 1100.49 222.25 100.99 100.99 666.44 88.77
ORDER_NBR	CUST_NBR	SALES_EMP_ID	SALE_PRICE
1033	110	401	66 55
1034	110 111	401 401	333.44
1035	110	401	76 - 65
1036	111	401	333.44
1037	110	401	76.65
1038	111	401	333.44
1039	110	401	66.55 333.44 76.65 333.44 76.65 333.44 76.65

Figure 3.7 - Table after rollback

SQL> SELECT	* FROM cus	st_order;	
		SALES_EMP_ID	
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	300 301 302 303 300 301 302 303 300 301 302	99.99
ORDER NBR	CHST NBR	SALES EMP ID	SALE PRICE
1011	103	303 309 301 302 303 309 301 302 303 300 301 302	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
1017	100	393 300	400 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
1026	110	401	1200.23
1027	111	401	1100.49
1028	111	401	222.25
1029	110	401	100.99
1030	110	401	100.99
1031	111	401	666.44
1032	110	302 303 300 301 401 401 401 401 401 401	88.77
		SALES_EMP_ID	
1033	110 111 110 111	401 401 401 401 401 401 401 401	66.55
1034	111	401	333.44
1035	110	401	76.65
1036	111	401	333.44
1037	111 110 111 110	401	76.65
1038	111	401	333.44
1039	110	401	76.65

Figure 3.6 - Rollback

SQL> ROLLBACK TO qthree3; Rollback complete.

#### **SQL USED:**

- 1 | connect / as sysdba;
- 2 | CREATE USER q4 IDENTIFIED BY q4;
- 3 | GRANT CONNECT, CREATE SESSION TO q4;
- 4 | GRANT SELECT, INSERT, UPDATE, DELETE ON adoyle.employee TO q4;
- 5 | connect q4;
- 6 | //enter password
- 7 | SELECT \* FROM adoyle.employee;

This question demonstrates the creation of a user and the granting of privileges. In order to create the user, sysdba was logged onto. On line 2 the user is created, the *identified by* keyword assigns a password for the user to logon with. On line 3, the user is granted *connect* and *create session* privileges (as user adoyle was).

Line 4 grants the user the required privileges, with the **ON** keyword also added to limit the privileges to a specific table. Note the inclusion of **adoyle** before **employee**, since **adoyle** is the owner of the table.

Lines 5-7 demonstrate how the user can now **select** from the table (see Figure 4.3).

### Figure 4.1 - Creating a new user

```
SQL> connect / as sysdba;
Connected.
SQL> CREATE USER q4 IDENTIFIED BY q4;
User created.
```

# Figure 4.2 - Granting connect and session privilege to new user

```
SQL> GRANT CONNECT, CREATE SESSION TO q4;
Grant succeeded.
```

### Figure 4.3 - Granting required privileges and demonstration of their use

```
SQL> GRANT SELECT, INSERT, UPDATE, DELETE ON adoyle.employee TO q4;
Grant succeeded.
SQL> connect q4;
Enter password:
SQL> SELECT * FROM adoyle.employee;
    EMP_ID FNAME
                                  LNAME
                                                        MANAGER_EMP_ID
           Reno
                                  Lopez
                                  Fulbright
           Stewart
       300
           Jason
                                  Chase
                                                                    304
           James
                                  Mason
                                  Freeman
           Michael
           Gareth
 rows selected.
```

# Figure 4.4 - Double Session

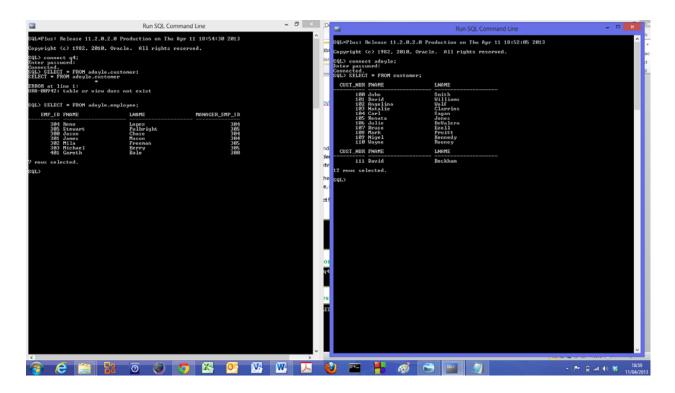


Figure 4.4 above shows a double instance of the SQL command line, which demonstrates how **q4** is not allowed access to the customer table, but can access the employee table (left-hand instance of Figure 4.4). The right-hand instance of the command line shows the user **adoyle** accessing the customer table.

### **SQL USED:**

- 1 | REVOKE DELETE ON adoyle.employee FROM q4;
- 2 | connect q4;
- 3| //enter password
- 4 | DELETE FROM adoyle.employee
- 5 | WHERE Iname = 'Monster';

This question demonstrates the use of the **REVOKE** keyword, which allows you to remove privileges from users. Again, the **ON** keyword is used to specify the table for the **revoke** to be carried out on, and the **FROM** keyword allows you to specify which user is affected.

Lines 4-5 demonstrate an attempt to delete content from the **employee** table whilst logged in as **q4**. The **revoke** has been successful because the error message "insufficient privileges" is output to the screen (see Figure 5.2).

Figure 5.1 - Revoking Privilege

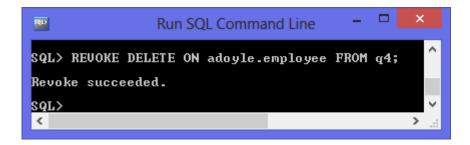
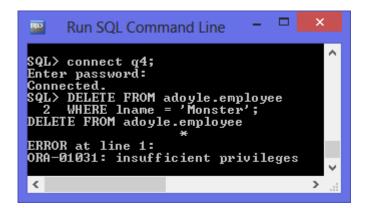


Figure 5.2 - Attempting to delete



### **SQL USED:**

- 1| SELECT cust\_order.order\_nbr, cust\_order.sale\_price, customer.fname AS first\_name, customer.lname AS last name
- 2 FROM customer INNER JOIN cust order ON customer.cust nbr = cust order.cust nbr
- 3 | WHERE SALE\_PRICE =(SELECT min(sale\_price) from cust\_order);

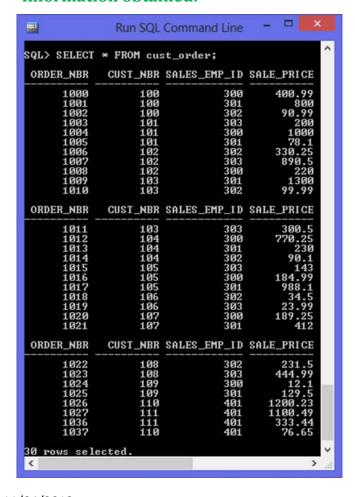
Figure 6.1 - Output of customer with smallest order

This question demonstrates how to display information based on a specific criteria, in this case being the customer who had place the smallest value order.

The relevant headings are selected on line 1 of the SQL code above. The **customer** and **cust\_order** tables are joined by the attribute **cust\_nbr** (using the **ON** keyword).

The criteria is specified using the **WHERE SALE PRICE** = statement. As shown on line 3 of the SQL code, the minimum value is selected from the **cust\_order** table using the **min** function.

Figure 6.2 - Cust\_order table to verify correct information obtained:



To be submitted at a later stage, as discussed in class on  $09^{th}$  April 2013.