Experiment 6B

1. For the order table shown above compute the following

a. Total number of tuples

b. Average purchase amount

c. Maximum purchase amount

d. Sum of purchase amount

e. Minimum purchase amount

SQL> select count(ord\_no) as Number\_of\_tuples from orders;

NUMBER\_OF\_TUPLES

----------------

12

SQL> select avg(purch\_amt) as Average\_Purchase\_Amount from orders;

AVERAGE\_PURCHASE\_AMOUNT

-----------------------

1461.765

SQL> select max(purch\_amt) as Max\_Purchase\_Amount from orders;

MAX\_PURCHASE\_AMOUNT

-------------------

5760

SQL> select min(purch\_amt) as Min\_Purchase\_Amount from orders;

MIN\_PURCHASE\_AMOUNT

-------------------

65.26

SQL> select sum(purch\_amt) as Total\_Purchase\_Amount from orders;

TOTAL\_PURCHASE\_AMOUNT

---------------------

17541.18

2. Write a SQL statement to find the number of salesmen currently listing for all of their customers.

SQL> select salesman\_id, count(customer\_id) from customer group by(salesman\_id);

SALESMAN\_ID COUNT(CUSTOMER\_ID)

----------- ------------------

5006 1

5003 1

5007 1

5001 2

5005 1

5002 2

6 rows selected.

3. Write a SQL statement know how many orders a customer have placed.

SQL> select customer\_id, count(ord\_no) from orders group by(customer\_id);

CUSTOMER\_ID COUNT(ORD\_NO)

----------- -------------

3004 1

3008 1

3002 3

3005 2

3001 1

3009 2

3007 1

3003 1

8 rows selected.

4. Write a SQL statement find the number of customers who gets at least a gradation for his/her performance.

SQL> select count(customer\_id) from customer where grade IS NOT NULL;

COUNT(CUSTOMER\_ID)

------------------

7

5. Write a SQL statement to find the highest purchase amount ordered by the each customer on a particular date with their ID, order date and highest purchase amount.

SQL> select customer\_id,ord\_date,max(purch\_amt) from orders group by(customer\_id,ord\_date);

CUSTOMER\_ID ORD\_DATE MAX(PURCH\_AMT)

----------- --------------- --------------

3009 2012-10-10 2480.4

3001 2012-09-10 270.65

3009 2012-08-17 110.5

3003 2012-08-17 75.29

3005 2012-10-05 150.5

3002 2012-09-10 5760

3007 2012-07-27 2400.6

3002 2012-10-05 65.26

3004 2012-10-10 1983.43

3002 2012-04-25 3045.6

3005 2012-09-10 948.5

CUSTOMER\_ID ORD\_DATE MAX(PURCH\_AMT)

----------- --------------- --------------

3008 2012-06-27 250.45

12 rows selected.

6. Write a SQL statement to find the highest purchase amount with their ID and order date, for only those customers who have highest purchase amount in a day is more than 2000.

SQL> select customer\_id,ord\_date,max(purch\_amt) from orders group by(customer\_id,ord\_date) having max(purch\_amt)>2000;

CUSTOMER\_ID ORD\_DATE MAX(PURCH\_AMT)

----------- --------------- --------------

3009 2012-10-10 2480.4

3002 2012-09-10 5760

3007 2012-07-27 2400.6

3002 2012-04-25 3045.6

7. Write a SQL statement to find the highest purchase amount with their ID and order date, for those customers who have a higher purchase amount in a day is within the range 2000 and 6000.

SQL> select customer\_id,ord\_date,max(purch\_amt) from orders group by(customer\_id,ord\_date) having max(purch\_amt) between 2000 and 6000;

CUSTOMER\_ID ORD\_DATE MAX(PURCH\_AMT)

----------- --------------- --------------

3009 2012-10-10 2480.4

3002 2012-09-10 5760

3007 2012-07-27 2400.6

3002 2012-04-25 3045.6

8. Create a order view order over500 to display all order details above 500$

SQL> create view orderover500 as select \* from orders where purch\_amt>500;

View created.

SQL> select \* from orderover500;

ORD\_NO PURCH\_AMT ORD\_DATE CUSTOMER\_ID SALESMAN\_ID

---------- ---------- --------------- ----------- -----------

70007 948.5 2012-09-10 3005 5002

70005 2400.6 2012-07-27 3007 5001

70008 5760 2012-09-10 3002 5001

70010 1983.43 2012-10-10 3004 5006

70003 2480.4 2012-10-10 3009 5003

70013 3045.6 2012-04-25 3002 5001

6 rows selected.

9. Display all the records of order over 500

SQL> select \* from orderover500;

ORD\_NO PURCH\_AMT ORD\_DATE CUSTOMER\_ID SALESMAN\_ID

---------- ---------- --------------- ----------- -----------

70007 948.5 2012-09-10 3005 5002

70005 2400.6 2012-07-27 3007 5001

70008 5760 2012-09-10 3002 5001

70010 1983.43 2012-10-10 3004 5006

70003 2480.4 2012-10-10 3009 5003

70013 3045.6 2012-04-25 3002 5001

6 rows selected.

10. Write a query to display all the orders from the orders table issued by the salesman 'Paul Adam'.

SQL> select \* from orders where salesman\_id=(select salesman\_id from salesman where name='Paul Adams');

ORD\_NO PURCH\_AMT ORD\_DATE CUSTOMER\_ID SALESMAN\_ID

---------- ---------- --------------- ----------- -----------

70011 75.29 2012-08-17 3003 5007

11. Write a query to display all the orders for the salesman who belongs to the city London.

SQL> select \* from orders where salesman\_id=(select salesman\_id from salesman where name='Paul Adams');

ORD\_NO PURCH\_AMT ORD\_DATE CUSTOMER\_ID SALESMAN\_ID

---------- ---------- --------------- ----------- -----------

70011 75.29 2012-08-17 3003 5007

SQL> select \* from orders where salesman\_id in (select salesman\_id from salesman where city='London');

ORD\_NO PURCH\_AMT ORD\_DATE CUSTOMER\_ID SALESMAN\_ID

---------- ---------- --------------- ----------- -----------

70009 270.65 2012-09-10 3001 5005

12.  Write a query to find all the orders issued against the salesman who works for customer whose id is 3007.

SQL> select \* from orders where salesman\_id in (select salesman\_id from customer where customer\_id=3007);

ORD\_NO PURCH\_AMT ORD\_DATE CUSTOMER\_ID SALESMAN\_ID

---------- ---------- --------------- ----------- -----------

70002 65.26 2012-10-05 3002 5001

70005 2400.6 2012-07-27 3007 5001

70008 5760 2012-09-10 3002 5001

70013 3045.6 2012-04-25 3002 5001

13. Write a query to display all the orders which values are greater than the average order value for 10th October 2012

SQL> select \* from orders where purch\_amt> (select avg(purch\_amt) from orders where ord\_date='2012-10-10');

ORD\_NO PURCH\_AMT ORD\_DATE CUSTOMER\_ID SALESMAN\_ID

---------- ---------- --------------- ----------- -----------

70005 2400.6 2012-07-27 3007 5001

70008 5760 2012-09-10 3002 5001

70003 2480.4 2012-10-10 3009 5003

70013 3045.6 2012-04-25 3002 5001