**EXPERIMENT NO: 04**

**===============================================================================**

**Author:** Yash Wankhedkar

**Roll No**: 77 [5A]

**Date**: 05-NOV-2022

**===============================================================================**

**PROBLEM STATEMENT:**

Use the SalesCo database established in Experiment-02 with the below mentioned schemata to execute the

listed queries involving join operations, sub-queries of different kinds and correlated queries.

CUSTOMER (C\_CODE, LNAME, FNAME, C\_AREA, C\_PHONE, BALANCE)

INVOICE (INV\_NUM, C\_CODE, INV\_DATE)

LINE (INV\_NUM, L\_NUM, P\_CODE, L\_UNITS, L\_PRICE)

PRODUCT (P\_CODE, DESCRIPT, P\_DATE, QTY, P\_MIN, P\_PRICE, P\_DISC, V\_CODE)

VENDOR (V\_CODE, V\_NAME, V\_CONTACT, V\_AREA, V\_PHONE, V\_STATE, V\_ORDER)

**QUERY SET**

**=========**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***QUERY–01**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write SQL code to create a table PART without any tuple from PRODUCT such that it includes product code-PT\_CODE, product description PT DESC, the unit price-PT\_PRICE and the supplier code. Now populate PART with the tuples fetching the contents from PRODUCT. For the PART table created, compare its schema with PRODUCT for the common attributes. Observe all the constraints on PART table (use USER CONSTRAINTS) and state your inferences.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL> CREATE TABLE PART

2 AS SELECT P\_CODE AS PT\_CODE, DESCRIPT AS PT\_DESC, P\_PRICE AS PT\_PRICE, V\_CODE

3 FROM PRODUCT

4 WHERE 1=2;

**Table created.**

SQL> ALTER TABLE PART

2 ADD CONSTRAINT PART\_PK PRIMARY KEY(PT\_CODE);

**Table altered.**

SQL> INSERT INTO PART

2 (SELECT P\_CODE, DESCRIPT, P\_PRICE, V\_CODE FROM PRODUCT);

**22 rows created.**

SQL> DESC PART;

**Name Null? Type**

**----------------------------------------- -------- ----------------------------**

**PT\_CODE NOT NULL CHAR(5)**

**PT\_DESC NOT NULL VARCHAR2(30)**

**PT\_PRICE NOT NULL NUMBER(6,2)**

**V\_CODE NUMBER(5)**

SQL> DESC PRODUCT;

**Name Null? Type**

**----------------------------------------- -------- ----------------------------**

**P\_CODE NOT NULL CHAR(5)**

**DESCRIPT NOT NULL VARCHAR2(30)**

**P\_DATE NOT NULL DATE**

**QTY NOT NULL NUMBER(4)**

**P\_MIN NOT NULL NUMBER(3)**

**P\_PRICE NOT NULL NUMBER(6,2)**

**P\_DISC NOT NULL NUMBER(2,2)**

**V\_CODE NUMBER(5)**

SQL> SELECT TABLE\_NAME, CONSTRAINT\_NAME

2 FROM USER\_CONSTRAINTS

3 WHERE TABLE\_NAME IN ('PART', 'PRODUCT');

**TABLE\_NAME CONSTRAINT\_NAME**

**------------------------------ ------------------------------**

**PART SYS\_C0010458**

**PART SYS\_C0010459**

**PART SYS\_C0010460**

**PART PART\_PK**

**PRODUCT SYS\_C008655**

**PRODUCT SYS\_C008656**

**PRODUCT SYS\_C008657**

**PRODUCT SYS\_C008658**

**PRODUCT SYS\_C008659**

**PRODUCT SYS\_C008660**

**PRODUCT SYS\_C008661**

**TABLE\_NAME CONSTRAINT\_NAME**

**------------------------------ ------------------------------**

**PRODUCT PRODUCT\_CK\_P\_MIN**

**PRODUCT PRODUCT\_PK\_P\_CODE**

**PRODUCT PRODUCT\_VENDOR\_FK\_V\_CODE**

**14 rows selected.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***QUERY–02**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write a SQL code that will list all vendors who have supplied a part (You must ensure that only unique V\_CODE values are displayed). Also retrieve information on vendors referenced in PRODUCT who have supplied products with prices in excess of 10 units.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL> SELECT DISTINCT V.V\_CODE, V.V\_NAME

2 FROM VENDOR V

3 INNER JOIN PRODUCT P ON V.V\_CODE = P.V\_CODE

4 WHERE P.QTY>10

5 ORDER BY V.V\_NAME;

**V\_CODE V\_NAME**

**---------- ------------------------------**

**23119 Blackman Sisters**

**21225 Bryson, Inc.**

**21231 GnB Supply**

**21344 Gomez Sons**

**25595 HighEnd Supplies**

**24992 Indian Masters**

**24288 Justin Stores**

**7 rows selected.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***QUERY–03**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write SQL code that will retrieve the product particulars for the parts with the highest and the lowest price. Use this query to retrieve the product particulars for the parts with the highest and the lowest inventory value (In both outputs the highest price products should be listed first).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL> SELECT P\_CODE, P\_PRICE FROM PRODUCT P

2 WHERE P\_CODE IN(

3 SELECT PT\_CODE FROM PART WHERE PT\_PRICE IN(

4 SELECT MAX(PT\_PRICE ) FROM PART UNION

5 SELECT MIN(PT\_PRICE ) FROM PART)

6 );

**P\_COD P\_PRICE**

**----- ----------**

**RF100 4.99**

**AB212 275**

SQL> SELECT P\_CODE, P\_PRICE, P\_PRICE\*QTY AS INV\_VAL FROM PRODUCT P

2 WHERE P\_CODE IN(

3 SELECT PT\_CODE FROM PART PT WHERE (PT.PT\_PRICE \* P.QTY) IN(

4 SELECT MAX(PT.PT\_PRICE \* P.QTY) FROM PART PT UNION

5 SELECT MIN(PT.PT\_PRICE \* P.QTY) FROM PART PT)

6 )

7 ORDER BY P\_PRICE DESC;

**P\_COD P\_PRICE INV\_VAL**

**----- ---------- ----------**

**AB212 275 4125**

**RF100 4.99 214.57**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***QUERY–04**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write SQL code that will retrieve the product particulars for all products whose prices (largest first) exceed the average product price of the inventory. Also list the number of products that are supplied by each vendor.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL> COLUMN DESCRIPT FORMAT A18;

SQL> COLUMN QTY FORMAT 999;

SQL> COLUMN P\_MIN FORMAT 999;

SQL> SELECT \* FROM PRODUCT

2 WHERE P\_PRICE >

3 (SELECT AVG(P\_PRICE) FROM

4 PRODUCT)

5 ORDER BY P\_PRICE DESC;

**P\_COD DESCRIPT P\_DATE QTY P\_MIN P\_PRICE P\_DISC V\_CODE**

**----- ------------------ --------- ---- ----- ---------- ---------- ----------**

**AB212 Power Drill 03-AUG-20 15 3 275 0**

**HC100 Hicut Chain Saw 07-FEB-20 11 5 256.99 .05 24288**

**AB111 Power Drill 25-SEP-22 15 5 125 .1 24992**

**JB012 Jigsaw 12in Blade 30-DEC-19 8 5 109.92 .05 24288**

**JB008 Jigsaw 8in Blade 24-DEC-19 6 5 99.87 .05 24288**

SQL> SELECT V.V\_CODE,V.V\_NAME,COUNT(P\_CODE)

2 FROM VENDOR V,PRODUCT

3 WHERE V.V\_CODE = PRODUCT.V\_CODE

4 GROUP BY V.V\_CODE,V.V\_NAME;

**V\_CODE V\_NAME COUNT(P\_CODE)**

**---------- ------------------------------ -------------**

**24288 Justin Stores 3**

**21344 Gomez Sons 3**

**25595 HighEnd Supplies 3**

**24992 Indian Masters 5**

**21225 Bryson, Inc. 2**

**23119 Blackman Sisters 2**

**21231 GnB Supply 1**

**7 rows selected.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***QUERY–05**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write SQL code to generate a listing of the number of products in the inventory supplied by each vendor that has prices average below 10. Extend this query to generate a listing of the total cost of products for each vendor - TOT\_COST, such that the total cost exceeds 400.00 and the high value vendor is placed last.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL> SELECT V\_CODE, COUNT(P\_CODE) COUNT, AVG(P\_PRICE) AVERAGE\_PRICE

2 FROM PRODUCT

3 GROUP BY V\_CODE

4 HAVING AVG(P\_PRICE)<10;

**V\_CODE COUNT AVERAGE\_PRICE**

**---------- ---------- -------------**

**21231 1 8.45**

**21225 2 8.47**

SQL> SELECT V\_CODE,SUM(P\_PRICE \* QTY) TOT\_COST

2 FROM PRODUCT

3 WHERE V\_CODE IS NOT NULL

4 GROUP BY V\_CODE

5 HAVING SUM(P\_PRICE\*QTY)>400;

**V\_CODE TOT\_COST**

**---------- ----------**

**25595 2311.16**

**23119 2311.16**

**24992 9045.5**

**21231 2002.65**

**21225 1431.13**

**24288 4305.47**

**21344 1009.07**

**7 rows selected.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***QUERY–06**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write SQL code to create a view - PRODUCT\_STATS from PRODUCT that generate a report that shows a summary of total product cost - TOT\_COST, and statistics on the quantity on hand [maximum - MX QTY, minimum - MN QTY, average- AV\_QTY] for each vendor.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL> CREATE OR REPLACE VIEW PRODUCT\_STATS AS

2 SELECT V\_CODE,SUM(P\_PRICE) TOT\_COST, MAX(QTY) MX\_QTY,

MIN(QTY) MN\_QTY,AVG(QTY) AV\_QTY

3 FROM PRODUCT

4 WHERE V\_CODE IS NOT NULL

5 GROUP BY V\_CODE;

**View created.**

SQL> SELECT \* FROM PRODUCT\_STATS;

**V\_CODE TOT\_COST MX\_QTY MN\_QTY AV\_QTY**

**---------- ---------- ---------- ---------- ----------**

**25595 182.46 18 8 12.6666667**

**23119 121.64 23 15 19**

**24992 209.05 200 10 67**

**21231 8.45 237 237 237**

**21225 16.94 172 23 97.5**

**24288 466.78 11 6 8.33333333**

**21344 37.47 43 18 31**

**7 rows selected.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***QUERY–07**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write a SQL query that will list for each customer who has made purchases, the customer number, the customer balance and the aggregate purchase amount.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL> SELECT C\_CODE,SUM(L\_UNITS\*L\_PRICE) PURCHASE,BALANCE

2 FROM INVOICE

3 NATURAL JOIN LINE NATURAL JOIN CUSTOMER

4 GROUP BY C\_CODE,BALANCE;

**C\_CODE PURCHASE BALANCE**

**---------- ---------- ----------**

**10011 479.57 0**

**10014 422.77 0**

**10012 408.85 345.86**

**10018 34.87 216.55**

**10015 34.97 0**

**10020 350 500**

**6 rows selected.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***QUERY–08**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Modify Query-07 to include the number of individual product purchases made by each customer. (If the customer's invoice is based on three products, one per LNUM, then count 3 product purchases. For example, customer 10011 generated 3 invoices, which contained a total of 5 lines, each representing a product purchase.)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL> SELECT C\_CODE,SUM(L\_UNITS) TOT\_UNITS FROM INVOICE

2 NATURAL JOIN LINE NATURAL JOIN CUSTOMER

3 GROUP BY C\_CODE

4 ORDER BY C\_CODE;

**C\_CODE TOT\_UNITS**

**---------- ----------**

**10011 23**

**10012 17**

**10014 8**

**10015 3**

**10018 5**

**10020 20**

**6 rows selected.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***QUERY–09**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write SQL query to produce the total purchase per invoice (The invoice total the sum of the product purchases in the LINE that corresponds to the INVOICE). Further, produce a listing showing invoice numbers with corresponding invoice total identified to a customer (Use GROUP BY on C\_CODE). Also generate a listing showing the number of invoices and the total purchase amounts by customer.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL> SELECT INV\_NUM,C\_CODE,SUM(L\_PRICE)

2 FROM INVOICE NATURAL JOIN LINE

3 GROUP BY C\_CODE,INV\_NUM;

**INV\_NUM C\_CODE SUM(L\_PRICE)**

**---------- ---------- ------------**

**1003 10012 119.39**

**1005 10011 5.87**

**1006 10014 383.85**

**1002 10011 4.99**

**1009 10020 17.5**

**1004 10018 14.94**

**1007 10015 19.98**

**1008 10011 135.77**

**1001 10014 24.94**

**9 rows selected.**

SQL> SELECT COUNT(INV\_NUM) INV\_TOT, C\_CODE, SUM(L\_PRICE) SUM

2 FROM INVOICE NATURAL JOIN LINE

3 GROUP BY C\_CODE

4 ORDER BY C\_CODE;

**INV\_TOT C\_CODE SUM**

**---------- ---------- ----------**

**5 10011 146.63**

**4 10012 119.39**

**6 10014 408.79**

**2 10015 19.98**

**2 10018 14.94**

**1 10020 17.5**

**6 rows selected.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***QUERY–10**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write SQL code to find the customer balance summary for all customers who have not made purchases during the current invoicing period. Use this query to generate a summary of the customer balance characteristics (the output should include the minimum, maximum and average balances over all purchases).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL> SELECT C\_CODE, BALANCE

2 FROM CUSTOMER WHERE C\_CODE IN (

3 SELECT C\_CODE FROM CUSTOMER

4 MINUS

5 SELECT C\_CODE FROM INVOICE

6 )

7 GROUP BY C\_CODE, BALANCE;

**C\_CODE BALANCE**

**---------- ----------**

**10010 0**

**10013 536.75**

**10019 0**

**10017 768.93**

**10016 221.19**

SQL> SELECT MAX(BALANCE) MX\_BAL,MIN(BALANCE) MN\_BAL,AVG(BALANCE) AV\_BAL

2 FROM CUSTOMER WHERE C\_CODE IN (

3 SELECT C\_CODE FROM CUSTOMER

4 MINUS

5 SELECT C\_CODE FROM INVOICE

6 );

**MX\_BAL MN\_BAL AV\_BAL**

**---------- ---------- ----------**

**768.93 0 305.374**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***QUERY–11**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write SQL code to create a table INV CUSTOMER that includes INV NUM as QUOTE\_ID, INV\_DATE as QUOTE DT and C\_NAME combining FNAME and LNAME with embedded space. Enforce the entity integrity constraint on QUOTE ID. (You may use subquery to create the table structure. Ensure that the created table is empty). Now, use SELECT subquery to populate INV\_CUSTOMER using the information contained in INVOICE and CUSTOMER.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL> CREATE TABLE INV\_CUSTOMER AS

2 SELECT I.INV\_NUM AS QUOTE\_ID,

3 I.INV\_DATE AS QUOTE\_DT,

4 C.FNAME ||' '||C.LNAME AS CNAME

5 FROM INVOICE I

6 JOIN CUSTOMER C

7 USING(C\_CODE)

8 WHERE 1=2;

**Table created.**

SQL> ALTER TABLE INV\_CUSTOMER

2 ADD PRIMARY KEY (QUOTE\_ID);

**Table altered.**

SQL> INSERT INTO INV\_CUSTOMER

2 ( SELECT I.INV\_NUM AS QUOTE\_ID,

3 I.INV\_DATE AS QUOTE\_DT,

4 C.FNAME ||' '||C.LNAME AS CNAME

5 FROM INVOICE I

6 JOIN CUSTOMER C

7 USING(C\_CODE));

**9 rows created.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***QUERY–12**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Modify Query-11 to create a view INV\_CUTOMER\_VW with the mentioned composition. Do not enforce entity integrity as in Query-11. Populate this view in similar manner. State the problem(s) are encountered. Try populating taking alternative approach you knew. Does that work? Now create the same view (use CREATE OR REPLACE VIEW) such that the view is populated at the creation time. Check the view contents. Now try inserting a record - 1011, Jagat Narayan, 12-Mar-

2020, and observe the result. Three non-discounted products - ZZ999 & AB212 (vendor 24992) and SH200 were added to the

inventory. The details are as below...

SH200, Sledge Hammer, 05-Jul-2020, 10, 3, 25.8

ZZ999, Cordless Drill, 10-Jul-2020, 200, 40, 25.5

AB212, Power Drill, 03-Aug-2020, 15, 3, 275.0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL> CREATE OR REPLACE VIEW INV\_CUSTOMER\_VW AS

2 SELECT I.INV\_NUM AS QUOTE\_ID,

3 I.INV\_DATE AS QUOTE\_DT,

4 C.FNAME ||' '||C.LNAME AS CNAME

5 FROM INVOICE I

6 JOIN CUSTOMER C

7 USING(C\_CODE)

8 WHERE 1=2;

**View created.**

SQL> INSERT INTO INV\_CUSTOMER\_VW

2 ( SELECT I.INV\_NUM AS QUOTE\_ID,

3 I.INV\_DATE AS QUOTE\_DT,

4 C.FNAME ||' '||C.LNAME AS CNAME

5 FROM INVOICE I

6 JOIN CUSTOMER C

7 USING(C\_CODE));

**INSERT INTO INV\_CUSTOMER\_VW**

**\***

**ERROR at line 1:**

**ORA-01733: virtual column not allowed here**

Since cname is a virtual column, i.e. a combination of fname and lname, we cannot directly change it. If we want to change it, we have to change the fname and lname from customer.

SQL> DROP VIEW INV\_CUSTOMER\_VW;

**View dropped.**

SQL> CREATE OR REPLACE VIEW INV\_CUSTOMER\_VW AS

2 SELECT I.INV\_NUM AS QUOTE\_ID,

3 I.INV\_DATE AS QUOTE\_DT,

4 C.FNAME ||' '||C.LNAME AS CNAME

5 FROM INVOICE I

6 JOIN CUSTOMER C

7 USING(C\_CODE);

**View created.**

SQL> SELECT \* FROM INV\_CUSTOMER\_VW;

**QUOTE\_ID QUOTE\_DT CNAME**

**---------- --------- ---------------------**

**1008 17-JAN-20 Elena Johnson**

**1005 17-JAN-20 Elena Johnson**

**1002 16-JAN-20 Elena Johnson**

**1003 16-JAN-20 Kathy Smith**

**1006 17-JAN-20 Bill Johnson**

**1001 16-JAN-20 Bill Johnson**

**1007 17-JAN-20 Julia Samuels**

**1004 17-JAN-20 Ming Lee**

**1009 22-JUN-20 Yash Wankhedkar**

**9 rows selected.**

SQL> INSERT INTO INV\_CUSTOMER\_VW VALUES(1011, '12-Mar-2020','Jagat Narayan');

**INSERT INTO INV\_CUSTOMER\_VW VALUES(1011, '12-Mar-2020','Jagat Narayan')**

**\***

**ERROR at line 1:**

**ORA-01733: virtual column not allowed here**

SQL> INSERT INTO PRODUCT VALUES('SH200', 'Sledge hammer', '05-Jul-2020', 10, 3, 25.8, 0, 24992);

**1 row created.**

SQL> INSERT INTO PRODUCT VALUES('ZZ999', 'Cordless Drill', '10-Jul-2020', 200, 40, 25.5, 0, 24992);

**1 row created.**

SQL> INSERT INTO PRODUCT VALUES('AB212', 'Power Drill', '03-Aug-2020', 15, 3, 275.0, 0, NULL);

**1 row created.**

SQL> SELECT \* FROM PRODUCT WHERE P\_CODE IN ('SH200','ZZ999','AB212');

**P\_COD DESCRIPT P\_DATE QTY P\_MIN P\_PRICE P\_DISC V\_CODE**

**----- ------------------ --------- ---- ----- ---------- ---------- ----------**

**SH200 Sledge hammer 05-JUL-20 10 3 25.8 0 24992**

**ZZ999 Cordless Drill 10-JUL-20 200 40 25.5 0 24992**

**AB212 Power Drill 03-AUG-20 15 3 275 0**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***QUERY–13**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write SQL code using subquery to list the supplier number and name of only those suppliers who supply some products.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL> SELECT V\_CODE, V\_NAME FROM VENDOR WHERE V\_CODE IN (

2 SELECT DISTINCT V\_CODE FROM PRODUCT

3 );

**V\_CODE V\_NAME**

**---------- ------------------------------**

**25595 HighEnd Supplies**

**21344 Gomez Sons**

**23119 Blackman Sisters**

**24288 Justin Stores**

**21225 Bryson, Inc.**

**21231 GnB Supply**

**24992 Indian Masters**

**7 rows selected.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***QUERY–14**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write SQL code using subquery that will compute the average price of all products. Modify the query to compute the average price of all products based on the product description.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL> SELECT AVG(P\_PRICE)

2 FROM PRODUCT;

**AVG(P\_PRICE)**

**------------**

**60.8209091**

SQL> SELECT DESCRIPT,COUNT(DESCRIPT) COUNT,AVG(P\_PRICE) AVG\_PRICE

2 FROM PRODUCT

3 GROUP BY DESCRIPT;

**DESCRIPT COUNT AVG\_PRICE**

**------------------ ---------- ----------**

**Hrd. Spring 1/4in 1 60.82**

**Hicut Chain Saw 1 256.99**

**Sledge hammer 1 25.8**

**7.25in Saw Blade 1 14.99**

**9.00 in Saw Blade 1 17.49**

**Hrd. Spring 1/2in 1 60.82**

**Jigsaw 12in Blade 1 109.92**

**Jigsaw 8in Blade 1 99.87**

**Metal Screw 1 6.99**

**2.5in wide Screw 1 8.45**

**Rat Tail File 1 4.99**

**Sledge Hammer 1 14.4**

**PVC Pipe 1 5.87**

**Power Drill 3 153.606667**

**Claw Hammer 1 9.95**

**Steel Malting Mesh 1 60.82**

**HiVeld Hammer 1 17.5**

**PVC PIPE 1 15.25**

**Cordless Drill 2 43.16**

**19 rows selected.**

SQL> SELECT DESCRIPT, AVG(P\_PRICE)

2 FROM PRODUCT

3 GROUP BY DESCRIPT;

**DESCRIPT AVG(P\_PRICE)**

**------------------ ------------**

**Hrd. Spring 1/4in 60.82**

**Hicut Chain Saw 256.99**

**Sledge hammer 25.8**

**7.25in Saw Blade 14.99**

**9.00 in Saw Blade 17.49**

**Hrd. Spring 1/2in 60.82**

**Jigsaw 12in Blade 109.92**

**Jigsaw 8in Blade 99.87**

**Metal Screw 6.99**

**2.5in wide Screw 8.45**

**Rat Tail File 4.99**

**Sledge Hammer 14.4**

**PVC Pipe 5.87**

**Power Drill 153.606667**

**Claw Hammer 9.95**

**Steel Malting Mesh 60.82**

**HiVeld Hammer 17.5**

**PVC PIPE 15.25**

**Cordless Drill 43.16**

**19 rows selected.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***QUERY–15**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write SQL code using subquery that will list product code, product description and unit product price for all products having the unit price higher than or equal to the average product price.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL> SELECT P\_CODE, DESCRIPT, P\_PRICE

2 FROM PRODUCT

3 WHERE P\_PRICE >= (

4 SELECT AVG(P\_PRICE)

5 FROM PRODUCT);

**P\_COD DESCRIPT P\_PRICE**

**----- ------------------ ----------**

**JB012 Jigsaw 12in Blade 109.92**

**JB008 Jigsaw 8in Blade 99.87**

**HC100 Hicut Chain Saw 256.99**

**AB111 Power Drill 125**

**AB212 Power Drill 275**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***QUERY–16**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write SQL code that will list supplier number, name and contact person for suppliers who do not supply any product in current season.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL> SELECT V\_CODE,V\_NAME,V\_CONTACT

2 FROM VENDOR WHERE V\_CODE IN(

3 SELECT V\_CODE FROM VENDOR

4 MINUS

5 SELECT V\_CODE FROM PRODUCT);

**V\_CODE V\_NAME V\_CONTACT**

**---------- ------------------------------ --------------------**

**21226 SuperLoo, Inc. Ching Ming**

**22587 Downing, Inc. Simon Singh**

**24004 Almeda House Almeda Brown**

**25443 Super Systems Ted Hwang**

**25501 Silvermines Ltd. Anne White**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***QUERY–17**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write SQL code using subquery to update the product price to the average product price, but only for the products that are supplied by vendors not belonging to the state TN' and 'KY'. Add a line for invoice number 1003 to include a 10 items of the product named ZZ999 -1003, 4, ZZ999, 10, 25.5

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL> UPDATE PRODUCT SET P\_PRICE=(SELECT AVG(P\_PRICE) FROM PRODUCT)

2 WHERE V\_CODE IN(

3 SELECT V\_CODE FROM VENDOR

4 WHERE V\_STATE NOT IN ('TN', 'KY'));

**5 rows updated.**

SQL> INSERT INTO LINE VALUES(1003,4,'ZZ999',10,25.5);

**1 row created.**

SQL> SELECT COUNT(\*) FROM LINE;

**COUNT(\*)**

**----------**

**20**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*QUERY–18\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write SQL code using subquery to find all the customers (include customer numbers, first name and last name) who have ordered some kind of a blade. Now find the customers who have ordered the part "Power Drill".

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL> SELECT C\_CODE,FNAME,LNAME FROM CUSTOMER

2 WHERE C\_CODE IN(

3 SELECT C\_CODE FROM INVOICE

4 WHERE INV\_NUM IN(

5 SELECT INV\_NUM FROM LINE

6 WHERE P\_CODE IN (

7 SELECT P\_CODE FROM PRODUCT

8 WHERE LOWER(DESCRIPT) LIKE '%blade')

9 )

10 );

**C\_CODE FNAME LNAME**

**---------- ---------- ----------**

**10014 Bill Johnson**

**10012 Kathy Smith**

**10015 Julia Samuels**

SQL> SELECT C\_CODE,FNAME,LNAME

2 FROM CUSTOMER

3 WHERE C\_CODE IN(

4 SELECT C\_CODE FROM INVOICE

5 WHERE INV\_NUM IN(

6 SELECT INV\_NUM FROM LINE

7 WHERE P\_CODE IN(

8 SELECT PT\_CODE FROM PART

9 WHERE LOWER(PT\_DESC) LIKE '%power drill'

10 )

11 )

12 );

**no rows selected**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*QUERY–19\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write SQL code using subquery to find all the customers who have purchased a drill or a hammer or a saw.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL> SELECT C\_CODE,FNAME,LNAME

2 FROM CUSTOMER

3 WHERE C\_CODE IN(

4 SELECT C\_CODE FROM INVOICE

5 WHERE INV\_NUM IN (

6 SELECT INV\_NUM FROM LINE

7 WHERE P\_CODE IN(

8 SELECT P\_CODE FROM PRODUCT

9 WHERE UPPER(DESCRIPT) LIKE '%HAMMER' OR

10 UPPER(DESCRIPT) LIKE '%DRILL' OR

11 UPPER(DESCRIPT) LIKE '%SAW%'

12 )

13 )

14 );

**C\_CODE FNAME LNAME**

**---------- ---------- ----------**

**10015 Julia Samuels**

**10012 Kathy Smith**

**10014 Bill Johnson**

**10011 Elena Johnson**

**10018 Ming Lee**

**10020 Yash Wankhedkar**

**6 rows selected.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*QUERY–20\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write SQL code using subquery to list all products with the total quantity sold greater than the average quantity sold.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL> SELECT \* FROM PRODUCT WHERE P\_CODE IN (

2 SELECT P\_CODE FROM (

3 SELECT P\_CODE,SUM(L\_UNITS)

4 FROM LINE

5 GROUP BY P\_CODE

6 HAVING SUM(L\_UNITS)>AVG(L\_UNITS)

7 )

8 );

**P\_COD DESCRIPT P\_DATE QTY P\_MIN P\_PRICE P\_DISC V\_CODE**

**----- ------------------ --------- ---- ----- ---------- ---------- ----------**

**SB725 7.25in Saw Blade 13-DEC-19 32 15 14.99 .05 21344**

**RF100 Rat Tail File 15-DEC-19 43 20 4.99 0 21344**

**CD00X Cordless Drill 20-JAN-20 12 5 60.82 .05 25595**

**CH10X Claw Hammer 20-JAN-20 23 10 9.95 .1 21225**

**PP101 PVC Pipe 20-FEB-20 188 75 5.87 0**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*QUERY–21\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write SQL code using subquery to list all customers who have purchased products HC100 and JB012.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL> SELECT C\_CODE,FNAME,LNAME FROM CUSTOMER NATURAL JOIN INVOICE

2 WHERE INV\_NUM IN(

3 SELECT INV\_NUM FROM LINE

4 WHERE P\_CODE IN ('HC100','JB012')

5 );

**C\_CODE FNAME LNAME**

**---------- ---------- ----------**

**10014 Bill Johnson**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*QUERY–22\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write SQL code using subquery that will for all products list the product price and the difference between each product's price and the average product price. Ensure that the average product price is also displayed.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL> SELECT P\_CODE,P\_PRICE,(

2 P\_PRICE-(

3 SELECT AVG(P\_PRICE)

4 FROM PRODUCT)

5 ) DIFF,

6 (SELECT AVG(P\_PRICE) FROM PRODUCT) AVERAGE

7 FROM PRODUCT;

**P\_COD P\_PRICE DIFF AVERAGE**

**----- ---------- ---------- ----------**

**AB112 60.82 -.00090909 60.8209091**

**SB725 14.99 -45.830909 60.8209091**

**SB900 17.49 -43.330909 60.8209091**

**CL025 60.82 -.00090909 60.8209091**

**CL050 60.82 -.00090909 60.8209091**

**JB012 109.92 49.0990909 60.8209091**

**JB008 99.87 39.0490909 60.8209091**

**CD00X 60.82 -.00090909 60.8209091**

**CH10X 9.95 -50.870909 60.8209091**

**SH100 14.4 -46.420909 60.8209091**

**RF100 4.99 -55.830909 60.8209091**

**HC100 256.99 196.169091 60.8209091**

**PP101 5.87 -54.950909 60.8209091**

**MC001 6.99 -53.830909 60.8209091**

**WC025 8.45 -52.370909 60.8209091**

**SM48X 60.82 -.00090909 60.8209091**

**HW15X 17.5 -43.320909 60.8209091**

**AB111 125 64.1790909 60.8209091**

**PP102 15.25 -45.570909 60.8209091**

**SH200 25.8 -35.020909 60.8209091**

**ZZ999 25.5 -35.320909 60.8209091**

**AB212 275 214.179091 60.8209091**

**22 rows selected.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*QUERY–23\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write SQL code using correlated query to list all product sales in which the units sold value is greater than the average units sold value for that product (as opposed to the average for all products).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL> SELECT P\_CODE, SUM(L\_UNITS) FROM LINE L

2 GROUP BY P\_CODE

3 HAVING SUM(L.L\_UNITS)>(

4 SELECT AVG(L\_UNITS) FROM LINE LA

5 WHERE LA.P\_CODE=L.P\_CODE);

**P\_COD SUM(L\_UNITS)**

**----- ------------**

**SB725 8**

**RF100 6**

**CD00X 2**

**CH10X 5**

**PP101 17**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*QUERY–24\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Write SQL, code using correlated query to list all customers who have placed an order. (Use EXISTS clause in SELECT statement).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SQL> SELECT \* FROM CUSTOMER C

2 WHERE EXISTS(

3 SELECT \* FROM INVOICE I

4 WHERE I.C\_CODE=C.C\_CODE

5 );

**C\_CODE LNAME FNAME C\_AREA C\_PHONE BALANCE**

**---------- ---------- ---------- ---------- ---------- ----------**

**10014 Johnson Bill 615 2455533 0**

**10011 Johnson Elena 713 2753455 0**

**10012 Smith Kathy 615 2873453 345.86**

**10018 Lee Ming 713 2323234 216.55**

**10015 Samuels Julia 713 2345432 0**

**10020 Wankhedkar Yash 904 3562098 500**

**6 rows selected.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*