```
                ****************Experiment no:-04*****************
        ------------------------------------------------------------------------
                                Author:Saurabh Khandagale
                                Roll No:46
                                Date :16- September-2020
        ------------------------------------------------------------------------
```

EXPERIMENT:-04

AIM:-

TO Execute different sql join operation ,sub-queries and correlated queries ona multi-relation database.

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.

```
=================================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.
 =======================================================================
```

```
CREATE TABLE PART AS(SELECT P_CODE AS PT_CODE,DESCRIPT AS PT_DESC,P_PRICE AS
PT_PRICE,V_CODE AS V_CODE FROM PRODUCT );

Table created.

SQL> TRUNCATE TABLE PART;

Table truncated.

SQL> INSERT INTO PART
       (PT_CODE,PT_DESC,PT_PRICE,V_CODE)
     SELECT P_CODE,DESCRIPT,P_PRICE,V_CODE FROM PRODUCT;

19 rows created.

SQL> COMMIT;

Commit complete.
```

```
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> SELECT * FROM PART;

PT_CO PT_DESC                          PT_PRICE     V_CODE
----- ------------------------------ ---------- ----------
AB112 Power Drill                        109.99      25595
SB725 7.25in Saw Blade                    14.99      21344
SB900 9.00 in Saw Blade                   17.49      21344
JB012 Jigsaw 12in Blade                  109.92      24288
JB008 Jigsaw 8in Blade                    99.87      24288
CD00X Cordless Drill                      38.95      25595
CH10X Claw Hammer                          9.95      21225
SH100 Sledge Hammer                        14.4
RF100 Rat Tail File                        4.99      21344
HC100 Hicut Chain Saw                    256.99      24288
PP101 PVC Pipe                             5.87
MC001 Metal Screw                          6.99      21225
WC025 2.5in wide Screw                     8.45      21231
SM48X Steel Malting Mesh                 119.95      25595
HW15X HiVeld Hammer                        15.5      24992
AB111 Power Drill                           125      24992
PP102 PVC PIPE                            15.25      24992
CL025 Hrd. Spring 1/4in                   39.95      23119
CL050 Hrd. Spring 1/2in                   43.99      23119

19 rows selected.
```

```
===============================Query02=================================
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.

========================================================================
PART A:-

SQL> SELECT V.V_CODE ,V.V_NAME FROM VENDOR V
     INNER JOIN PART P
     ON V.V_CODE=P.V_CODE;

    V_CODE V_NAME
---------- ------------------------------
     25595 HighEnd Supplies
     21344 Gomez Sons
     21344 Gomez Sons
     24288 Justin Stores
     24288 Justin Stores
     25595 HighEnd Supplies
     21225 Bryson, Inc.
     21344 Gomez Sons
     24288 Justin Stores
     21225 Bryson, Inc.
     21231 GnB Supply
     25595 HighEnd Supplies
     24992 INDIAN MASTER
     24992 INDIAN MASTER
     24992 INDIAN MASTER
     23119 Blackman Sisters
     23119 Blackman Sisters

17 rows selected.
PART B:

SQL> SELECT  V.V_CODE,V.V_NAME FROM VENDOR V
INNER JOIN PRODUCT P
ON P.V_CODE=V.V_CODE WHERE P_PRICE>10;

    V_CODE V_NAME
---------- ------------------------------
     25595 HighEnd Supplies
     21344 Gomez Sons
     21344 Gomez Sons
     24288 Justin Stores
     24288 Justin Stores
     25595 HighEnd Supplies
     24288 Justin Stores
     25595 HighEnd Supplies
     24992 INDIAN MASTER
     24992 INDIAN MASTER
     24992 INDIAN MASTER
     23119 Blackman Sisters
     23119 Blackman Sisters

13 rows selected.
```

```
============================Query03==========================================
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).
=============================================================================

[PART -A]

SQL> SELECT PT_PRICE,PT_CODE,PT_DESC
   FROM PART
    WHERE PT_PRICE = (SELECT MAX(PT_PRICE) FROM PART) OR
    PT_PRICE = (SELECT MIN(PT_PRICE) FROM PART)
    ORDER BY 1 DESC;




  PT_PRICE PT_CO PT_DESC
---------- ----- ------------------------------
    256.99 HC100 Hicut Chain Saw
      4.99 RF100 Rat Tail File
[PART -B]

SQL> SELECT MAX(P_PRICE*QTY) AS INV_VALUE FROM PRODUCT UNION SELECT
MIN(P_PRICE*QTY) FROM PRODUCT ORDER BY 1 DESC;

 INV_VALUE
----------
      5100
     115.2

===========================================Query04============================
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.
=========================================================================

SELECT P_CODE, DESCRIPT, P_DATE, QTY, P_PRICE

FROM PRODUCT

WHERE (SELECT AVG(P_PRICE) FROM PRODUCT) < P_PRICE

ORDER BY P_PRICE DESC;


             P_COD DESCRIPT            P_DATE           QTY    P_PRICE

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

             HC100 Hicut Chain Saw     07-FEB-20         11     256.99

             AB111 Power Drill         20-AUG-20         15        125

             SM48X Steel Malting Mesh  17-JAN-20         18     119.95

             AB112 Power Drill         03-NOV-19          8     109.99

             JB012 Jigsaw 12in Blade   30-DEC-19          8     109.92

             JB008 Jigsaw 8in Blade    24-DEC-19          6      99.87


             6 rows selected.
```

```
SQL> SELECT COUNT(*) AS "P_NO_PRD",V_CODE
     FROM PRODUCT JOIN VENDOR USING(V_CODE)
     GROUP BY V_CODE;

  P_NO_PRD     V_CODE
---------- ----------
         3      25595
         2      23119
         6      24992
         1      21231
         2      21225
         3      24288
         3      21344

7 rows selected.
```

```
==================================Query05=====================================

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.
==============================================================================
SELECT COUNT(*) AS "NUMBER OF PRODUCTS",V_CODE
    FROM PRODUCT
    GROUP BY V_CODE
    HAVING 10 > AVG(P_PRICE);

NUMBER OF PRODUCTS     V_CODE
------------------ ----------
                 1      21231
                 2      21225


SQL> SELECT SUM(P_PRICE) AS "TOT_COST",V_CODE
    FROM VENDOR JOIN PRODUCT
    USING(V_CODE)
    GROUP BY V_CODE;

  TOT_COST     V_CODE
---------- ----------
    268.89      25595
     83.94      23119
    482.05      24992
      8.45      21231
     16.94      21225
    466.78      24288
     37.47      21344

7 rows selected.
```

```
======================================Query06==================================
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 VIEW PRODUCT_STATS AS
   SELECT SUM(P_PRICE) AS "TOT_COST",MAX(QTY) AS MX_QTY,
    MIN(QTY) AS MN_QTY,AVG(QTY) AS AV_QTY
    FROM VENDOR JOIN PRODUCT
    USING(V_CODE)
    GROUP BY V_CODE;

View created.

SQL> SELECT * FROM PRODUCT_STATS;

  TOT_COST     MX_QTY     MN_QTY     AV_QTY
---------- ---------- ---------- ----------
    268.89         18          8 12.6666667
     83.94         23         15         19
    482.05        200         10 58.3333333
      8.45        237        237        237
     16.94        172         23       97.5
    466.78         11          6 8.33333333
     37.47         43         18         31

7 rows selected.
```

```
======================================Query07==================================
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(P_PRICE*QTY) AS AG_AMT,AVG(BALANCE) AS BALANCE
    FROM CUSTOMER NATURAL JOIN PRODUCT NATURAL JOIN LINE
    NATURAL JOIN INVOICE
    GROUP BY C_CODE;

    C_CODE     AG_AMT    BALANCE
---------- ---------- ----------
     10015     694.25          0
     10014    5845.91          0
     10012    1414.48     345.86
     10011    4809.64          0
     10018     443.42     216.55
     10020        930        500

6 rows selected.
```

```
======================================Query08=================================
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
L_NUM, 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,LNAME||' '||FNAME AS NAME,COUNT(L_NUM),
    SUM(P_PRICE*QTY) AS AG_AMT,AVG(BALANCE) AS BALANCE
    FROM CUSTOMER NATURAL JOIN PRODUCT
    NATURAL JOIN LINE NATURAL JOIN INVOICE
    GROUP BY C_CODE,LNAME||' '||FNAME;

    C_CODE NAME                 COUNT(L_NUM)     AG_AMT    BALANCE
---------- -------------------- ------------ ---------- ----------
     10018 Lee Ming                        2     443.42     216.55
     10020 Khandagale Saurabh              1        930        500
     10011 Johnson Elena                   5    4809.64          0
     10014 Johnson Bill                    6    5845.91          0
     10015 Samuels Julia                   2     694.25          0
     10012 Smith Kathy                     3    1414.48     345.86

6 rows selected.

======================================Query09=================================

 Write SQL query to produce the total purchase per invoice (The invoice total
is 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 SUM(L_UNITS*L_PRICE) AS "TOTAL_PCH",INV_NUM
    FROM LINE WHERE INV_NUM IN
    (SELECT INV_NUM FROM INVOICE JOIN LINE USING(INV_NUM))
    GROUP BY INV_NUM;

 TOTAL_PCH    INV_NUM
---------- ----------
    153.85       1003
       310       1009
    397.83       1006
     24.94       1001
      9.98       1002
     34.97       1007
     34.87       1004
     70.44       1005
    399.15       1008

9 rows selected.
```

[PART-2]

```
SQL> SELECT C_CODE,SUM(L_UNITS*L_PRICE) AS "TOTAL PURCHASE PER INVOICE"
    FROM LINE JOIN INVOICE USING(INV_NUM)
    WHERE INV_NUM IN
    (SELECT INV_NUM FROM INVOICE JOIN LINE USING(INV_NUM))
    GROUP BY C_CODE;

    C_CODE TOTAL PURCHASE PER INVOICE
---------- --------------------------
     10015                      34.97
     10014                     422.77
     10011                     479.57
     10012                     153.85
     10018                      34.87
     10020                        310

6 rows selected.
```

[PART-3]

```
SQL> SELECT C_CODE,SUM(L_UNITS*L_PRICE) AS "TOT_INVOICE",COUNT(INV_NUM) AS
"TOTAL INVOICES"
    FROM LINE JOIN INVOICE USING(INV_NUM)
    WHERE INV_NUM IN
    (SELECT INV_NUM FROM INVOICE JOIN LINE USING(INV_NUM))
    GROUP BY C_CODE,INV_NUM;

    C_CODE TOT_INVOICE TOTAL INVOICES
---------- ----------- --------------
     10012      153.85              3
     10011       70.44              1
     10014      397.83              4
     10011        9.98              1
     10020         310              1
     10018       34.87              2
     10015       34.97              2
     10011      399.15              3
     10014       24.94              2

9 rows selected.
```

```
======================================Query10================================
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).
==============================================================================
[PART-A]

SQL> SELECT C_CODE,BALANCE FROM CUSTOMER WHERE C_CODE NOT IN(SELECT C.C_CODE
FROM CUSTOMER C INNER JOIN INVOICE I ON C.C_CODE=I.C_CODE);

    C_CODE    BALANCE
---------- ----------
     10016     221.19
     10017     768.93
     10013     536.75
     10010          0
     10019          0

[PART-B]


SQL> SELECT MIN(BALANCE) AS MIN_BALANCE,
    MAX(BALANCE) AS MAX_BALANCE, AVG(BALANCE) AS AVG_BALANCE
      FROM CUSTOMER;

MIN_BALANCE MAX_BALANCE AVG_BALANCE
----------- ----------- -----------
          0      768.93  235.389091



======================================Query11================================
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(SELECT I.INV_NUM AS QUOTE_ID,I.INV_DATE AS
QUOTE_DT,C.FNAME ||' '||C.LNAME AS C_NAME FROM CUSTOMER C INNER JOIN INVOICE I
ON C.C_CODE=I.C_CODE);

Table created.



INSERT INTO INV_CUSTOMER_VW(QUOTE_ID,QUOTE_DT,C_NAME)(SELECT
I.INV_NUM,I.INV_DATE,C.FNAME||' '||C.LNAME FROM INVOICE I INNER JOIN CUSTOMER
C ON C.C_CODE=I.C_CODE) ;

9 rows created.
```

```
SQL> SELECT * FROM INV_CUSTOMER_VW;

  QUOTE_ID QUOTE_DT  C_NAME
---------- --------- ---------------------
      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 Saurabh Khandagale

9 rows selected.
9 rows selected.
```

====================================Query12================================

Modify **Query-11** to create a view INV_CUTOMER_VW with the mentioned composition. Do not enforce entity integrity as in **<u>Query-11</u>**. 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.

==

```
SQL> CREATE VIEW INV_CUSTOMER_VW AS (SELECT I.INV_NUM AS QUOTE_ID,I.INV_DATE
AS QUOTE_DT, C.FNAME||' '||C.LNAME AS C_NAME FROM CUSTOMER C INNER JOIN
INVOICE I ON C.C_CODE=I.C_CODE);

View created.



SQL> INSERT INTO INV_CUSTOMER_VW(QUOTE_ID,QUOTE_DT,C_NAME)(SELECT
I.INV_NUM,I.INV_DATE, C.FNAME||' '||C.LNAME FROM INVOICE I INNER JOIN CUSTOMER
C ON C.C_CODE=I.C_CODE) ;
INSERT INTO INV_CUSTOMER_VW(QUOTE_ID,QUOTE_DT,C_NAME)(SELECT
I.INV_NUM,I.INV_DATE, C.FNAME||' '||C.LNAME FROM INVOICE I INNER JOIN CUSTOMER
C ON C.C_CODE=I.C_CODE)
                                                *
ERROR at line 1:
ORA-01733: virtual column not allowed here
```

Explain:-See while creating view we are creating logical schema,and in logical schema we are inserting values from physical schema i.e(table with combine column) which lead to error.

```
SQL> CREATE OR REPLACE VIEW INV_CUSTOMER_VW AS (SELECT * FROM INV_CUSTOMER);

View created.

SQL> SELECT * FROM INV_CUSTOMER_VW;

  QUOTE_ID QUOTE_DT  C_NAME
---------- --------- --------------------
      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 Saurabh Khandagale

9 rows selected.


SQL> INSERT INTO INV_CUSTOMER VALUES(1011,'12-Mar-2020','Jagat Narayan');

1 row created.

SQL> SELECT * FROM INV_CUSTOMER_VW;

  QUOTE_ID QUOTE_DT  C_NAME
---------- --------- --------------------
      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 Saurabh Khandagale
      1011 12-MAR-20 Jagat Narayan

10 rows selected.

SQL>  SELECT * FROM PRODUCT;

P_COD DESCRIPT           P_DATE    QTY    P_MIN    P_PRICE     P_DISC   V_CODE
----- ------------------ --------- ---------- ---------- ---------- ----------
AB112 Power Drill        03-NOV-19    8     5     109.99        0       25595
SB725 7.25in Saw Blade   13-DEC-19   32    15      14.99      .05       21344
SB900 9.00 in Saw Blade  13-NOV-19   18    12      17.49        0       21344

CL025 Hrd. Spring 1/4in  15-JAN-20   15     8      39.95         0       23119
CL050 Hrd. Spring 1/2in  15-JAN-20   23     5      43.99         0       23119

19 rows selected.


INSERT INTO PRODUCT
(P_CODE,DESCRIPT,P_DATE,QTY,P_MIN,P_PRICE,V_CODE)VALUES('SH200','Sledge
Hammer','05-Jul-2020',10,3,25.8,24992);
```

```
1 row created.


INSERT INTO PRODUCT
     (P_CODE,DESCRIPT,P_DATE,QTY,P_MIN,P_PRICE,V_CODE)
   VALUES('ZZ999','Cordless Drill','10-Jul-2020',200,40,25.5,24992);

1 row created.

SQL> INSERT INTO PRODUCT
    (P_CODE,DESCRIPT,P_DATE,QTY,P_MIN,P_PRICE,V_CODE)
     VALUES('AB212','Power DrilL','03-Aug-2020',15,3,275.0,24992);

1 row created.


SQL> SELECT * FROM PRODUCT;

P_COD DESCRIPT     P_DATE     QTY      P_MIN    P_PRICE     P_DISC     V_CODE
----- ------------------ --------- ---------- ---------- ---------- ----------
AB112 Power Drill         03-NOV-19   8       5      109.99       0      25595
SB725 7.25in Saw Blade  13-DEC-19   32      15       14.99     .05      21344

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      24992


22 rows selected.

======================================Query13================================
Write SQL code using subquery to list the supplier number and supplier
name of only those suppliers who supply some products.

==============================================================================



SQL> SELECT V_CODE,V_NAME
     FROM VENDOR
     WHERE V_CODE IN (SELECT V_CODE FROM PRODUCT);

    V_CODE V_NAME
---------- ------------------------------
     25595 HighEnd Supplies
     21344 Gomez Sons
     24288 Justin Stores
     21225 Bryson, Inc.
     21231 GnB Supply
     24992 INDIAN MASTER
     23119 Blackman Sisters

7 rows selected.
```

```
======================================Query14=================================
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) FROM PRODUCT GROUP BY DESCRIPT;

AVG(P_PRICE)
------------
      256.99
       39.95
       14.99
       17.49
      109.92
       99.87
       43.99
        6.99
        8.45
        4.99
        20.1
       10.56
  169.996667
        9.95
      119.95
        15.5
      32.225

17 rows selected.
[PART-B]

SQL> SELECT AVG(P_PRICE) AS "AVG PRICE",COUNT(DESCRIPT) FROM PRODUCT
     GROUP BY DESCRIPT;

 AVG PRICE COUNT(DESCRIPT)
---------- ---------------
    256.99               1
     39.95               1
     14.99               1
     17.49               1
    109.92               1
     99.87               1
     43.99               1
      6.99               1
      8.45               1
      4.99               1
      20.1               2
     10.56               2
169.996667               3
      9.95               1
    119.95               1
      15.5               1
    32.225               2

17 rows selected.
```

```
======================================Query15================================

 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
        FROM PRODUCT
        WHERE P_PRICE >(SELECT AVG(P_PRICE) FROM PRODUCT);

P_COD DESCRIPT              P_PRICE
----- ------------------ ----------
AB112 Power Drill            109.99
JB012 Jigsaw 12in Blade      109.92
JB008 Jigsaw 8in Blade        99.87
HC100 Hicut Chain Saw        256.99
SM48X Steel Malting Mesh     119.95
AB111 Power Drill               125
AB212 Power Drill               275

7 rows selected.
```

```
======================================Query16================================
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
    FROM VENDOR
    WHERE V_CODE NOT IN (SELECT V_CODE
    FROM VENDOR V JOIN PRODUCT P
    USING(V_CODE));

    V_CODE V_NAME                         V_CONTACT
---------- ------------------------------ --------------------
     21226 SuperLoo, Inc.                 Ching Ming
     24004 Almeda House                   Almeda Brown
     22587 Downing, Inc.                  Simon Singh
     25501 Silvermines Ltd.               Anne White
     25443 Super Systems                  Ted Hwang
```

```
======================================Query17================================
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 P SET P.P_PRICE=(SELECT AVG(P_PRICE) FROM PRODUCT P)
WHERE P.V_CODE IN(SELECT V.V_CODE FROM VENDOR V WHERE V.V_STATE !='KY' AND
V.V_STATE!='TN') ;

5 rows updated.



SQL> SELECT P_CODE,P_PRICE
            FROM PRODUCT NATURAL JOIN VENDOR
            WHERE V_STATE IN  ('FL', 'GA');

P_COD    P_PRICE
----- ----------
AB112      62.95
CD00X      62.95
SM48X      62.95
CL025      62.95
CL050      62.95

[BEFORE]
SQL> SELECT * FROM LINE;

   INV_NUM      L_NUM P_COD    L_UNITS    L_PRICE
---------- ---------- ----- ---------- ----------
      1001          1 SB725          1      14.99
      1001          2 CH10X          1       9.95
      1002          1 RF100          2       4.99
      1003          1 CD00X          1      38.95
      1003          2 CD00X          1      39.95
      1003          3 SB725          5      14.99
      1004          1 RF100          3       4.99
      1004          2 CH10X          2       9.95
      1005          1 PP101         12       5.87
      1006          1 MC001          3       6.99
      1006          2 JB012          1     109.92
      1006          3 CH10X          1       9.95
      1006          4 HC100          1     256.99
      1007          1 SB725          2      14.99
      1007          2 RF100          1       4.99
      1008          1 PP101          5       5.87
      1008          2 SM48X          3     119.95
      1008          3 CH10X          1       9.95
      1009          1 HW15X         20       15.5

19 rows selected
```

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

1 row created.
SQL> SELECT * FROM LINE;

   INV_NUM      L_NUM P_COD    L_UNITS    L_PRICE
---------- ---------- ----- ---------- ----------
      1001          1 SB725          1      14.99
      1001          2 CH10X          1       9.95
      1002          1 RF100          2       4.99
      1003          1 CD00X          1      38.95
      1003          2 CD00X          1      39.95
      1003          3 SB725          5      14.99
      1004          1 RF100          3       4.99
      1004          2 CH10X          2       9.95
      1005          1 PP101         12       5.87
      1006          1 MC001          3       6.99
      1006          2 JB012          1     109.92
      1006          3 CH10X          1       9.95
      1006          4 HC100          1     256.99
      1007          1 SB725          2      14.99
      1007          2 RF100          1       4.99
      1008          1 PP101          5       5.87
      1008          2 SM48X          3     119.95
      1008          3 CH10X          1       9.95
      1009          1 HW15X         20       15.5
      1003          4 ZZ999         10       25.5

20 rows selected.

======================================Query18================================
 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.C_CODE,C.LNAME,C.FNAME FROM CUSTOMER C WHERE C.C_CODE IN (SELECT
I.C_CODE FROM INVOICE I WHERE I.INV_NUM IN (SELECT L.INV_NUM FROM LINE L WHERE
L.P_CODE IN(SELECT P.P_CODE  FROM PRODUCT P WHERE P.DESCRIPT LIKE '%Blade%'
))) ;

    C_CODE LNAME      FNAME
---------- ---------- ----------
     10014 Johnson    Bill
     10012 Smith      Kathy
     10015 Samuels    Julia


[PART-B]
SQL> SELECT C.C_CODE,C.LNAME,C.FNAME FROM CUSTOMER C
  WHERE C.C_CODE IN
(SELECT I.C_CODE FROM INVOICE I WHERE I.INV_NUM IN
 (SELECT L.INV_NUM FROM LINE L WHERE L.P_CODE IN
 (SELECT P.P_CODE  FROM PRODUCT P WHERE P.DESCRIPT LIKE '%POWER DRILL%' ))) ;

no rows selected
```

```
======================================Query19================================
Write SQL code using subquery to find all the customers who have purchased
a drill or a hammer or a saw.
==============================================================================

SQL> SELECT C.C_CODE,C.LNAME,C.FNAME FROM CUSTOMER C WHERE C.C_CODE IN (SELECT
I.C_CODE FROM INVOICE I WHERE I.INV_NUM IN (SELECT L.INV_NUM FROM LINE L WHERE
L.P_CODE IN(SELECT P.P_CODE  FROM PRODUCT P WHERE P.DESCRIPT LIKE '%Drill%' OR
P.DESCRIPT LIKE '%Hammer%' OR P.DESCRIPT LIKE '%Saw%' ))) ;

    C_CODE LNAME      FNAME
---------- ---------- ----------
     10014 Johnson    Bill
     10012 Smith      Kathy
     10018 Lee        Ming
     10015 Samuels    Julia
     10011 Johnson    Elena
     10020 Khandagale Saurabh

6 rows selected.

======================================Query20================================
Write SQL code using subquery to list all products with the total quantity
sold greater than the average quantity sold.

==============================================================================

SQL> SELECT P_CODE,DESCRIPT,P_PRICE FROM PRODUCT
    WHERE P_CODE IN (SELECT P_CODE FROM LINE
    GROUP BY P_CODE
    HAVING SUM(L_UNITS)> AVG(L_UNITS));

P_COD DESCRIPT              P_PRICE
----- ----------------- ----------
SB725 7.25in Saw Blade       14.99
RF100 Rat Tail File           4.99
CD00X Cordless Drill         62.95
CH10X Claw Hammer             9.95
PP101 PVC Pipe                5.87


======================================Query21================================
 Write SQL code using subquery to list all customers who have purchased
products HC100 and JB012
==============================================================================
SELECT C.C_CODE,C.LNAME,C.FNAME FROM CUSTOMER C WHERE C.C_CODE IN (SELECT
I.C_CODE FROM INVOICE I WHERE I.INV_NUM IN (SELECT L.INV_NUM FROM LINE L WHERE
L.P_CODE IN(SELECT P.P_CODE  FROM PRODUCT P WHERE  P.P_CODE LIKE '%JB012%' OR
P.DESCRIPT LIKE '%HC100%' ))) ;

    C_CODE LNAME      FNAME
---------- ---------- ----------
     10014 Johnson    Bill
```

```
======================================Query22=================================
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,(SELECT AVG(P_PRICE) FROM PRODUCT)-P_PRICE AS
     "DIFFERENCE",AVG(P_PRICE) AS "AVG PRICE"
      FROM PRODUCT
      GROUP BY P_CODE,P_PRICE;

P_COD    P_PRICE DIFFERENCE  AVG PRICE
----- ---------- ---------- ----------
CH10X       9.95     52.995       9.95
SH200       25.8     37.145       25.8
ZZ999       25.5     37.445       25.5
SB900      17.49     45.455      17.49
JB012     109.92    -46.975     109.92
HW15X       15.5     47.445       15.5
PP102      15.25     47.695      15.25
SH100       14.4     48.545       14.4
WC025       8.45     54.495       8.45
AB212        275   -212.055        275
SB725      14.99     47.955      14.99
JB008      99.87    -36.925      99.87
PP101       5.87     57.075       5.87
SM48X     119.95    -57.005     119.95
HC100     256.99   -194.045     256.99
CL025      39.95     22.995      39.95
CL050      43.99     18.955      43.99
CD00X      38.95     23.995      38.95
RF100       4.99     57.955       4.99
MC001       6.99     55.955       6.99
AB112     109.99    -47.045     109.99
AB111        125    -62.055        125

22 rows selected.

======================================Query23=================================
 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)

==============================================================================

SELECT DESCRIPT, P_PRICE,  AVG(L_PRICE) AS AVG_UNIT_SOLD_VALUE
  2      FROM PRODUCT NATURAL JOIN LINE
  3      WHERE L_PRICE>(SELECT AVG(L_PRICE) FROM  LINE)
  4      GROUP BY DESCRIPT, P_PRICE;

DESCRIPT              P_PRICE AVG_UNIT_SOLD_VALUE
------------------ ---------- -------------------
Hicut Chain Saw        256.99              256.99
Steel Malting Mesh      62.95              119.95
Cordless Drill          62.95               39.45
Jigsaw 12in Blade      109.92              109.92
```

```
======================================Query24=================================
 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
  2  WHERE EXISTS (SELECT * FROM INVOICE
  3  WHERE CUSTOMER.C_CODE = INVOICE.C_CODE);

    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 Khandagale Saurabh           904    3562098        500

6 rows selected.
******************************************************************************
```

Viva Voice
========
****************************** VIVA-VOCE **********************************

Q 01.What is a correlated query?

ANS :

1. A correlated subquery is a subquery that uses the values of the outer query.
2. Because of this dependency, a correlated subquery cannot be executed independently as a simple subquery.
3. Moreover, a correlated subquery is executed repeatedly, once for each row evaluated by the outer query.The correlated subquery is also known as a repeating subquery.

****

Q 02.What are the three types of results that a subquery can return?

ANS :

1. The first SQL statement is known as the outer query, the second is known as the inner query or subquery.
2. The inner query or subquery is normally executed first.
3. The output of the inner query is used as the input for the outer Query.

1. A subquery can return
   1) a single value (one row, one column)
   2) a list of values(many rows, one column)
   3) a virtual table (many rows, many columns).

****

Q 03.What do you understand by an inline subquery?

ANS :

1. The subquery specified in the FROM clause of a query is called an inline view.
2. Because an inline view can replace a table in a query, it is also called a derived table.

****

Q 04.What do you understand by Theta Join and Self-Join?

ANS :

1. THETA JOIN allows you to merge two tables based on the condition represented by theta.
2. Theta joins work for all comparison operators. It is denoted by symbolθ.
3. The general case of JOIN operation is called a Theta join.

****

Q 05.List the execution differences while including an USING clause and an ON clause with JOIN query.

ANS :

1.In Joins, we use ON in a set of columns.

```
2.USING is useful when both the tables share a column of the exact same
name on which they join.

************************************************************************
==========
Inferences
==========
1.Subqueries execution time is less than that of join.
2.Correlated queries are executed more than one time if condition
becomes tre.
3.Aggrrigate functions like avg,min,max are helpful to compute
operations.
```