**\*\*\*\*\*\*\*\*\*\* EXPERIMENT: 04 \*\*\*\*\*\*\*\*\*\***

**Problem Statement**:

Use the SalesCo database established in experiment 2 with the below mentioned

schemata to execute the listed queries involving join operations,sub-queries of

different kinds and corelated 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)

**Author** : Mehul Y Khandhadiya

**Roll no** : 55

**Date** : 15-Sep-2020

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

**Write a 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 FROM PRODUCT WHERE 1 = 2);**

Table created.

**SELECT \* FROM PART;**

no rows selected

**INSERT INTO PART(SELECT P\_CODE,DESCRIPT,P\_PRICE,V\_CODE FROM PRODUCT);**

19 rows created.

**SELECT \* FROM PART;**

PT\_CODE PT\_DESC PT\_PRICE V\_CODE

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

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

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

19 rows selected.

**SELECT CONSTRAINT\_NAME, CONSTRAINT\_TYPE, TABLE\_NAME FROM**

**USER\_CONSTRAINTS WHERE TABLE\_NAME IN ('PRODUCT','PART');**

CONSTRAINT\_NAME C TABLE\_NAME

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

SYS\_C005700 C PART

SYS\_C005701 C PART

SYS\_C005702 C PART

SYS\_C005679 C PRODUCT

SYS\_C005680 C PRODUCT

SYS\_C005681 C PRODUCT

SYS\_C005682 C PRODUCT

SYS\_C005683 C PRODUCT

SYS\_C005684 C PRODUCT

SYS\_C005685 C PRODUCT

PRODUCT\_CK\_P\_MIN C PRODUCT

PRODUCT\_PK\_P\_CODE P PRODUCT

PRODUCT\_VENDOR\_FK\_V\_CODE R PRODUCT

13 rows selected.

**INFERENCE** - THE KEY CONSTRAINTS I.E REFERENTIAL INTEGRITY AND

ENITITY INTEGRITY CONSTRAINTS FROM PRODUCT ARE NOT IMPOSED ON PART TABLE.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 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.**

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

**SELECT DISTINCT V\_CODE, V\_NAME FROM PRODUCT NATURAL JOIN VENDOR;**

V\_CODE V\_NAME

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

24288 Justin Stores

21344 Gomez Sons

24992 INDIAN MASTERS

25595 HighEnd Supplies

21225 Bryson, Inc.

23119 Blackman Sisters

21231 GnB Supply

7 rows selected.

**SELECT V\_CODE, V\_NAME FROM PRODUCT NATURAL JOIN VENDOR;**

V\_CODE V\_NAME

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

25595 HighEnd Supplies

21344 Gomez Sons

21344 Gomez Sons

23119 Blackman Sisters

23119 Blackman Sisters

24288 Justin Stores

24288 Justin Stores

25595 HighEnd Supplies

21225 Bryson, Inc.

21344 Gomez Sons

24288 Justin Stores

V\_CODE V\_NAME

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

21225 Bryson, Inc.

21231 GnB Supply

25595 HighEnd Supplies

24992 INDIAN MASTERS

24992 INDIAN MASTERS

24992 INDIAN MASTERS

17 rows selected.

**SELECT V\_CODE, V\_NAME**

**FROM PRODUCT P NATURAL JOIN VENDOR**

**WHERE P.P\_PRICE > 10;**

V\_CODE V\_NAME

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

25595 HighEnd Supplies

21344 Gomez Sons

21344 Gomez Sons

23119 Blackman Sisters

23119 Blackman Sisters

24288 Justin Stores

24288 Justin Stores

25595 HighEnd Supplies

24288 Justin Stores

25595 HighEnd Supplies

24992 INDIAN MASTERS

V\_CODE V\_NAME

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

24992 INDIAN MASTERS

24992 INDIAN MASTERS

13 rows selected.

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

**Write a 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).**

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

**SELECT \* FROM PRODUCT**

**WHERE P\_PRICE = (SELECT MIN (P\_PRICE) FROM PRODUCT) OR**

**P\_PRICE = (SELECT MAX (P\_PRICE) FROM PRODUCT) ORDER BY P\_PRICE DESC;**

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

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

HC100 Hicut Chain Saw 07-FEB-20 11 5 256.99 0 24288

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

2 rows selected.

**SELECT \* FROM (SELECT \* FROM PRODUCT**

**WHERE P\_PRICE = (SELECT MIN (P\_PRICE) FROM PRODUCT) OR**

**P\_PRICE = (SELECT MAX (P\_PRICE) FROM PRODUCT))**

**NATURAL JOIN**

**(SELECT P\_CODE, P\_PRICE \* QTY AS INVEN\_VAL FROM PRODUCT**

**WHERE P\_PRICE = (SELECT MIN (P\_PRICE) FROM PRODUCT) OR**

**P\_PRICE = (SELECT MAX (P\_PRICE) FROM PRODUCT))**

**ORDER BY P\_PRICE DESC;**

P\_COD DESCRIPT P\_DATE QTY P\_MIN P\_PRICE P\_DISC V\_CODE INVEN\_VAL

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

AB212 Power Drill 03-AUG-20 15 3 275 0 24992 4125

RF100 Rat Tail File 15-DEC-19 43 20 4.99 0 21344 214.57

2 rows selected.

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

**Write a 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 which are supplied by each vendor.**

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

**SELECT \* FROM PRODUCT**

**WHERE P\_PRICE > (SELECT AVG (P\_PRICE) FROM PRODUCT)**

**ORDER BY P\_PRICE DESC;**

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

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

HC100 Hicut Chain Saw 07-FEB-20 11 5 256.99 0 24288

AB111 Power Drill 17-AUG-20 15 5 125 0 24992

JB012 Jigsaw 12in Blade 30-DEC-19 8 5 109.92 0 24288

JB008 Jigsaw 8in Blade 24-DEC-19 6 5 99.87 0 24288

SM48X Steel Malting Mesh 17-JAN-20 18 5 62.93 0 25595

AB112 Power Drill 03-NOV-19 8 5 62.93 0 25595

6 rows selected.

**SELECT V\_CODE, COUNT (P\_CODE)**

**FROM VENDOR NATURAL JOIN PRODUCT**

**GROUP BY V\_CODE;**

V\_CODE COUNT(P\_CODE)

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

25595 3

23119 2

24992 3

21231 1

21225 2

24288 3

21344 3

7 rows selected.

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

**Write a 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 V\_CODE, COUNT (P\_CODE), AVG(P\_PRICE) AS AGP**

**FROM PRODUCT**

**GROUP BY V\_CODE**

**HAVING AVG (P\_PRICE) < 10;**

V\_CODE COUNT(P\_CODE) AGP

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

21231 1 8.45

21225 2 8.47

2 rows selected.

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

**Write a 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.**

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

**CREATE OR REPLACE VIEW PRODUCT\_STATS AS**

**SELECT V\_CODE, SUM (P\_PRICE) AS TOT\_COST,**

**MAX (QTY) AS MX\_QTY, MIN(QTY) AS**

**MN\_QTY, AVG (QTY) AS AV\_QTY**

**FROM PRODUCT**

**GROUP BY V\_CODE;**

View created.

SELECT \* FROM PRODUCT\_STATS;

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

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

25595 268.89 18 8 12.67

23119 83.94 23 15 19

20.27 188 8 98

21231 8.45 237 237 237

21225 16.94 172 23 97.5

24288 466.78 11 6 8.33

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.**

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

**SELECT C\_CODE, SUM (BALANCE), AVG (LP)**

**FROM CUSTOMER NATURAL JOIN INVOICE NATURAL JOIN**

**(SELECT AVG (L\_PRICE) AS LP, INV\_NUM FROM LINE**

**GROUP BY INV\_NUM)**

**GROUP BY C\_CODE;**

C\_CODE SUM(BALANCE) AVG(LP)

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

10015 0 9.99

10014 0 54.21

10012 345.86 31.29

10011 0 18.70

10018 216.55 7.47

10020 500 15.5

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 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.)**

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

**SELECT C\_CODE, SUM (BALANCE), SUM (CI) AS CNT**

**FROM CUSTOMER NATURAL JOIN INVOICE NATURAL JOIN**

**(SELECT COUNT (INV\_NUM) AS CI, INV\_NUM FROM LINE**

**GROUP BY INV\_NUM)**

**GROUP BY C\_CODE;**

C\_CODE SUM(BALANCE) CNT

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

10015 0 2

10014 0 6

10012 345.86 3

10011 0 5

10018 216.55 2

10020 500 1

6 rows selected.

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

**Write a 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.**

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

**SELECT INV\_NUM, SUM (L\_PRICE) AS TOT**

**FROM LINE**

**GROUP BY INV\_NUM;**

INV\_NUM TOT

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

1003 93.89

1009 15.5

1006 383.85

1001 24.94

1002 4.99

1007 19.98

1004 14.94

1005 5.87

1008 135.77

9 rows selected.

**SELECT C\_CODE, SUM (TOT) AS TOTAL,**

**COUNT (INV\_NUM) AS NO FROM**

**(INVOICE NATURAL JOIN**

**(SELECT INV\_NUM, SUM (L\_PRICE) AS TOT**

**FROM LINE GROUP BY INV\_NUM))**

**GROUP BY C\_CODE;**

C\_CODE TOTAL NO

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

10015 19.98 1

10014 408.79 2

10012 93.89 1

10011 146.63 3

10018 14.94 1

10020 15.5 1

6 rows selected.

**SELECT C\_CODE, TOT, INV\_NUM FROM**

**CUSTOMER NATURAL JOIN (INVOICE**

**NATURAL JOIN (SELECT INV\_NUM, SUM (L\_PRICE) AS TOT**

**FROM LINE GROUP BY INV\_NUM));**

C\_CODE TOT INV\_NUM

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

10012 93.89 1003

10020 15.5 1009

10014 383.85 1006

10014 24.94 1001

10011 4.99 1002

10015 19.98 1007

10018 14.94 1004

10011 5.87 1005

10011 135.77 1008

9 rows selected.

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

**Write a 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 across all purchases)**.

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

**SELECT C.C\_CODE, BALANCE FROM**

**(SELECT C\_CODE FROM CUSTOMER**

**MINUS**

**SELECT C\_CODE FROM INVOICE) A**

**JOIN CUSTOMER C ON**

**A.C\_CODE=C.C\_CODE;**

C\_CODE BALANCE

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

10010 0

10013 536.75

10016 221.19

10017 768.93

10019 0

**SELECT MIN (BALANCE) AS MIN, MAX (BALANCE) AS MAX, AVG (BALANCE) AS AVG FROM**

**(SELECT C\_CODE FROM CUSTOMER**

**MINUS**

**SELECT C\_CODE FROM INVOICE) A**

**JOIN CUSTOMER C ON**

**A.C\_CODE = C.C\_CODE;**

MIN MAX AVG

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

0 768.93 305.374

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

**Write a 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.**

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

**CREATE TABLE INV\_CUSTOMER (**

**QUOTE\_ID NUMBER (4) NOT NULL,**

**QUOTE\_DT DATE NOT NULL,**

**C\_NAME VARCHAR (25) NOT NULL,**

**CONSTRAINT CONSTRAINT\_INV\_CUST\_PK PRIMARY KEY (QUOTE\_ID));**

**INSERT INTO INV\_CUSTOMER**

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

**I.INV\_DATE AS QUOTE\_DT,**

**C.FNAME||' '||C.LNAME AS C\_NAME FROM INVOICE I JOIN CUSTOMER C**

**ON I.C\_CODE = C.C\_CODE;**

**SELECT \* FROM INV\_CUSTOMER;**

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 UPLANCHWAR ATHARVA

9 rows selected.

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

**Modify Query-11 to create a view INV\_CUSTOMER\_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 and 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**

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

**CREATE OR REPLACE VIEW INV\_CUSTOMER\_VW**

**AS (SELECT INV\_NUM AS QUOTE\_ID, INV\_DATE AS QUOTE\_DT,**

**FNAME||' '||LNAME AS C\_NAME FROM CUSTOMER**

**NATURAL JOIN INVOICE WHERE 1=2);**

View created.

**INSERT INTO INV\_CUSTOMER\_VW**

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

**I.INV\_DATE AS QUOTE\_DT,**

**C.FNAME||' '||C.LNAME AS C\_NAME FROM INVOICE I JOIN CUSTOMER C**

**ON I.C\_CODE = C.C\_CODE;**

ERROR at line 1:

ORA-01733: virtual column not allowed here

**INFERENCE -** WHEN WE TRY TO INSERT TUPLES INTO VIEW THEY ARE ALSO INSERTED INTO THE TABLE FROM WHICH THE VIEW IS CREATED, BUT HERE BVIEW IS CREATED USING JOIN OPERATION BETWEEN TWO TABLES WHICH IS NOT ITSELF A TABLE SO IF WE TRY INSERTING INTO VIEW THERE WILL BE NO TABLE TO TAKE THESE TUPLES AND HENCE IT IS INVALID OPERATION.

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

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

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

**INSERT INTO PRODUCT VALUES ('&P\_CODE','&DESCRIPT','&P\_DATE', &QTY, &P\_MIN, &P\_PRICE, &P\_DISC, &V\_CODE);**

**SELECT V\_NAME, V\_CODE**

**FROM VENDOR**

**WHERE V\_CODE IN (SELECT V\_CODE FROM PRODUCT);**

V\_NAME V\_CODE

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

HighEnd Supplies 25595

Gomez Sons 21344

Blackman Sisters 23119

Justin Stores 24288

Bryson, Inc. 21225

GnB Supply 21231

INDIAN MASTERS 24992

7 rows selected.

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

**Write a 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**.

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

**SELECT SUM (P\_PRICE)/COUNT (P\_CODE) AS AVG**

**FROM PRODUCT;**

AVG

----------

62.93

**SELECT SUM (P\_PRICE)/COUNT (P\_CODE) AS AVG, DESCRIPT**

**FROM PRODUCT WHERE GROUP BY DESCRIPT;**

AVG DESCRIPT

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

39.95 Hrd. Spring 1/4in

256.99 Hicut Chain Saw

14.99 7.25in Saw Blade

17.49 9.00 in Saw Blade

43.99 Hrd. Spring 1/2in

109.92 Jigsaw 12in Blade

99.87 Jigsaw 8in Blade

6.99 Metal Screw

8.45 2.5in wide Screw

4.99 Rat Tail File

20.1 Sledge Hammer

AVG DESCRIPT

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

10.56 PVC Pipe

169.99 Power Drill

9.95 Claw Hammer

119.95 Steel Malting Mesh

15.5 HiVeld Hammer

32.075 Cordless Drill

17 rows selected.

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

**Write a 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.**

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

**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.

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

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

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

**SELECT V\_CODE, V\_NAME,V\_CONTACT**

**FROM VENDOR WHERE**

**V\_CODE IN ((SELECT V\_CODE FROM VENDOR**

**MINUS**

**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

5 rows selected.

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

**Write a 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**

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

**UPDATE PRODUCT**

**SET P\_PRICE = (SELECT AVG (P\_PRICE) FROM PRODUCT)**

**WHERE V\_CODE IN (SELECT V\_CODE FROM VENDOR**

**WHERE V\_STATE != 'TN' AND V\_STATE != 'KY');**

5 rows updated.

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

1 row created.

**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

INV\_NUM L\_NUM P\_COD L\_UNITS L\_PRICE

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

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.

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

**Write a 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".**

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

**SELECT C\_CODE, FNAME||' '||LNAME AS NAME FROM CUSTOMER**

**WHERE C\_CODE IN (SELECT C\_CODE FROM INVOICE**

**WHERE INV\_NUM IN (SELECT INV\_NUM FROM LINE**

**WHERE P\_CODE IN (SELECT P\_CODE**

**FROM PRODUCT WHERE DESCRIPT**

**LIKE '%Blade%')));**

C\_CODE NAME

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

10012 Kathy Smith

10014 Bill Johnson

10015 Julia Samuels

**SELECT C\_CODE, FNAME||' '||LNAME AS NAME FROM CUSTOMER**

**WHERE C\_CODE IN (SELECT C\_CODE FROM INVOICE**

**WHERE INV\_NUM IN (SELECT INV\_NUM FROM LINE**

**WHERE P\_CODE IN (SELECT P\_CODE**

**FROM PRODUCT WHERE**

**DESCRIPT = 'Power Drill')));**

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.**

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

**SELECT C\_CODE, FNAME||' '||LNAME AS NAME FROM CUSTOMER**

**WHERE C\_CODE IN (SELECT C\_CODE FROM INVOICE**

**WHERE INV\_NUM IN (SELECT INV\_NUM FROM LINE**

**WHERE P\_CODE IN (SELECT P\_CODE**

**FROM PRODUCT WHERE DESCRIPT**

**LIKE '%Drill%' OR DESCRIPT**

**LIKE '%Hammer%' OR DESCRIPT**

**LIKE '%Saw%')));**

C\_CODE NAME

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

10015 Julia Samuels

10014 Bill Johnson

10012 Kathy Smith

10018 Ming Lee

10011 Elena Johnson

10020 Mehul Khandhadiya

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.**

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

**SELECT P\_CODE FROM LINE WHERE**

**(SELECT AVG (L\_UNITS) FROM LINE) < L\_UNITS;**

P\_COD

-----

SB725

PP101

PP101

HW15X

ZZ999

5 rows selected.

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

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

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

**SELECT FNAME||' '||LNAME AS NAME**

**FROM CUSTOMER WHERE C\_CODE IN**

**(SELECT C\_CODE FROM INVOICE**

**WHERE INV\_NUM IN (SELECT INV\_NUM FROM**

**LINE WHERE P\_CODE IN ('HC100','JB012')));**

NAME

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

Bill Johnson

1 row selected.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 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.**

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

**SELECT P\_CODE, P\_PRICE, P\_PRICE-(SELECT (SUM (P\_PRICE)/COUNT (P\_CODE))**

**FROM PRODUCT) AS DIFF, (SELECT (SUM (P\_PRICE)/COUNT (P\_CODE))**

**FROM PRODUCT) AS AVG FROM PRODUCT;**

P\_COD P\_PRICE DIFF AVG

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

AB112 62.93 1.73 61.19

SB725 14.99 -46.20 61.19

SB900 17.49 -43.70 61.19

CL025 62.93 1.73 61.19

CL050 62.93 1.73 61.19

JB012 109.92 48.72 61.19

JB008 99.87 38.67 61.19

CD00X 62.93 1.73 61.19

CH10X 9.95 -51.24 61.19

SH100 14.4 -46.79 61.19

RF100 4.99 -56.20 61.19

P\_COD P\_PRICE DIFF AVG

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

HC100 256.99 195.79 61.19

PP101 5.87 -55.32 61.19

MC001 6.99 -54.20 61.19

WC025 8.45 -52.74 61.19

SM48X 62.93 1.73 61.19

HW15X 15.5 -45.69 61.19

AB111 125 63.80 61.19

PP102 15.25 -45.94 61.19

SH200 25.8 -35.39 61.19

ZZ999 25.2 -35.99 61.19

AB212 275 213.80 61.19

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).

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

**SELECT DISTINCT P\_CODE FROM LINE WHERE**

**(SELECT AVG (L\_UNITS) FROM LINE) < LINE.L\_UNITS;**

P\_COD

-----

SB725

ZZ999

PP101

HW15X

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

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

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

**SELECT C\_CODE FROM CUSTOMER**

**WHERE EXISTS (SELECT \* FROM INVOICE**

**WHERE CUSTOMER.C\_CODE = INVOICE.C\_CODE);**

C\_CODE

----------

10011

10012

10014

10015

10018

10020

6 rows selected.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* VIVA-VOCE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Q1 – What is a corelated query?**

Correlated subqueries are used for row-by-row processing. Each subquery is executed once for every row of the outer query. A correlated subquery is one way of reading every row in a table and comparing values in each row against related data. It is used whenever a subquery must return a different result or set of results for each candidate row considered by the main query. In other words, you can use a correlated subquery to answer a multipart question whose answer depends on the value in each row processed by the parent statement.

**Q2 – What are three types of results that a subquery can return?**

A subquery can return

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

**Q3 – What do you understand by an inline query? Give example.**

Inline query/inline view is a query that is being pasted instead of table name.

Example:

select c1, c2 from tab1;

select c1,c2 from (select c1,c2 from tab1);

here in second query , we have used a inline query from table tab1 .

output will be same here.

Uses:

To get limited columns.

To get the output from any complex query.

**Q4 – What do you understand by theta join and self join ?**

**Theta Join**:

In theta join we apply the condition on input relation(s) and then only those selected rows are used in the cross product to be merged and included in the output. It means that in normal cross product all the rows of one relation are mapped/merged with all the rows of second relation, but here only selected rows of a relation are made cross product with second relation. rovided they satisfy the theta condition. The join condition is denoted by the symbol θ.

**Self join**:

A self join is a join in which a table is joined with itself (which is also called Unary relationships), especially when the table has a FOREIGN KEY which references its own PRIMARY KEY. To join a table itself means that each row of the table is combined with itself and with every other row of the table.

**Q5 – List the execution differences while including an USING clause and an ON clause with JOIN query.**

**Using clause** -

The USING clause is used if several columns share the same name but you don’t want to join using all of these common columns. The columns listed in the USING clause can’t have any qualifiers in the statement, including the WHERE clause.

**On clause** –

The ON clause is used to join tables where the column names don’t match in both tables. The join conditions are removed from the filter conditions in the WHERE clause:

**INFERENCE**: Database was created with the help of tables present in salesco schema.

From this expt, we were able to learn about subqueries, aggregate

functions and their implementation. All the queries were studied

thoroughly and output was compiled and analysed.

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