Imaginary Purchase Order System

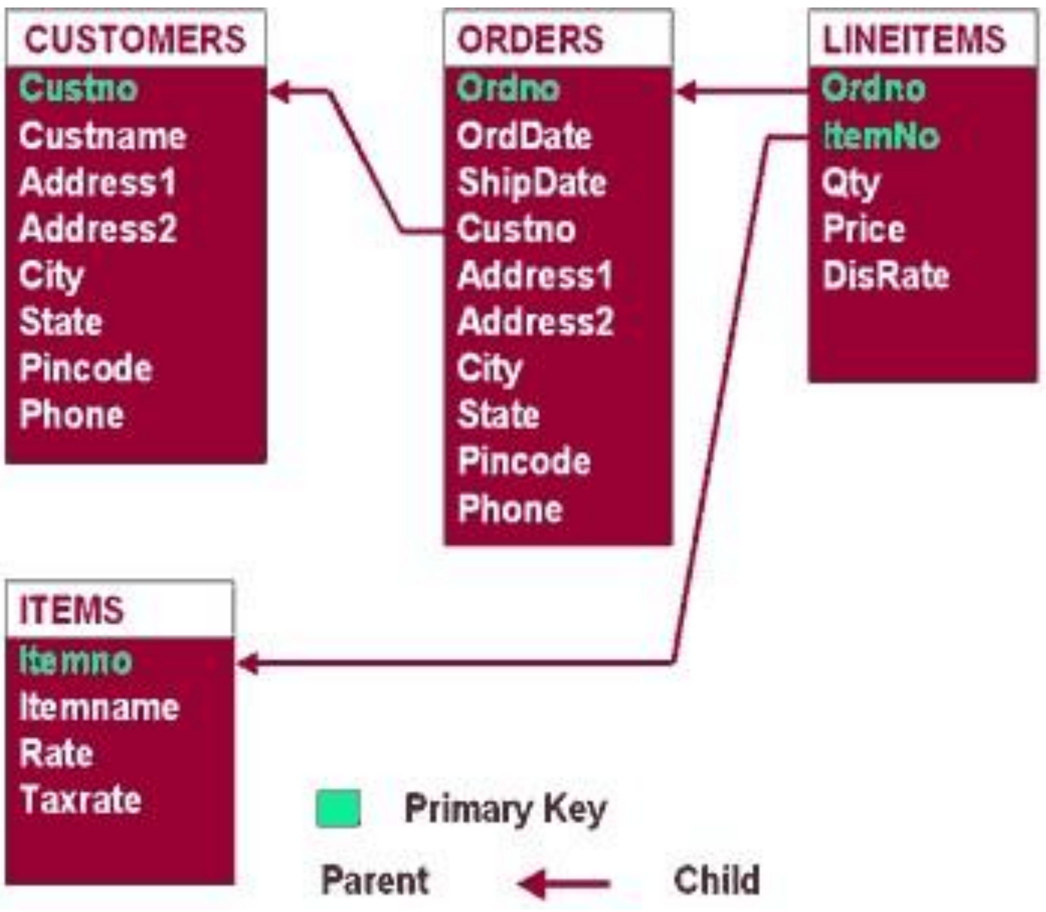
This project report gives you introduction to a typical and imaginary purchase order system. This project report explains the data to be stored in different tables, how to create those tables and also provides sample data so that you can start working with that data. The following are the topics of this Syestem.

Required Tables

This is a simple purchase order system in which customers place orders and each order contains one or more items. The data related to this application will be stored in the following tables.

Table	Meaning
Items	Stores information about products that are offered by company
Customers	Contains information about customer who place orders.
orders	Stores information about all orders placed by customers
lineitems	Contains information about items in each order.

The following picture shows the relationship between these four tables.



ITEMS table

This table stores information about all the items that are offered by compnay. The structure of the table is as follows:

Column	Datatype	Meaning
Itemno	Number(5)	A unique number assigned to each item.
ItemName	Varchar2(20)	Name of the item.
Rate	Number(8,2)	Rate of the item.
taxrate	Number(4,2)	Sales tax rate for this item.

The following are the constraints related to ITEMS table:

- **ITEMNO is primary key**
- **RATE and TAXRATE must be >= 0**
- **Default value for TAXRATE is 0**

```
create table ITEMS
(
 itemno   number(5)   constraint items_pk  primary key,
itemname varchar2(20),
 rate     number(8,2) constraint items_rate_chk check( rate >= 0),
taxrate  number(4,2) default 0 constraint items_rate_chk check( rate
>= 0)
);

insert into items values(1,'Samsung 14" monitor',7000,10.5);
insert into items values(2,'TVS Gold Keyboard',1000,10);
insert into items values(3,'Segate HDD 20GB',6500,12.5);
insert into items values(4,'PIII processor',8000,8); insert
into items values(5,'Logitech Mouse',500,5); insert into
items values(6,'Creative MMK',4500,11.5);
```

CUSTOMERS Table

This table contains information about customers who have placed one or more orders. The following is the structure of the table.

Column	Datatype	Meaning
Custno	Number(5)	A unique number assigned to each customer.
CustName	Varchar2(20)	Complete name of the customer.
Address1	varchar2(50)	First line of address.
Address2	varchar2(50)	Second line of address.
City	varchar2(30)	Name of the city where customer lives.
state	varchar2(30)	Name of the state where customer lives.
PinCode	varchar2(10)	Pincode of the city.
Phone	varchar2(30)	One or more phone numbers separated using comma(,).

The following are the constraint related to CUSTOMERS table.

- **CUSTNO is primary key**
- **CUSTNAME is not null column**

```
create table CUSTOMERS
(
 custno    number(5)    constraint customers_pk  primary key,
custname  varchar2(20) constraint customers_custname_nn not null,
address1  varchar2(50),  address2  varchar2(50),   city
varchar2(30),  state     varchar2(30),  pin       varchar2(10),
phone     varchar2(30)
);


insert into customers values(101,'Raul','12-22-29','Dwarakanagar',
           'Vizag','AP','530016','453343,634333');
insert into customers values(102,'Denilson','43-22-22','CBM Compound',
           'Vizag','AP','530012','744545');
insert into customers values(103,'Mendiator','45-45-52','Abid Nagar',
           'Vizag','AP','530016','567434');
insert into customers values(104,'Figo','33-34-56','Muralinagar',
'Vizag','AP','530021','875655,876563,872222'); insert into
customers values(105,'Zidane','23-22-56','LB Colony',
           'Vizag','AP','530013','765533');
```

ORDERS Table

Contains information about all orders placed by customers. Contains one row for each order. The details of items ordered in an order will be found in LINEITEMS table. The following is the structure of the table.

Column	Datatype	Meaning
OrdNo	Number(5)	A unique number assigned to each order.
OrdDate	Date	Date on which order is placed.
ShipDate	Date	Date on which goods are to be shipped to customer.
Address1	varchar2(50)	First line of shipping address.
Address2	varchar2(50)	Second line of shipping address.
City	varchar2(30)	City name in shipping address.
state	varchar2(30)	State name in shipping address.
PinCode	varchar2(10)	Pincode of the city in shipping address.

Phone	varchar2(30)	One or more phone numbers separated using comma(,) of shipping place.

The following are the constraint related to ORDERS table.

- **ORDNO is primary key**
- **CUSTNO is foreign key referencing CUSTNO of CUSTOMERS table.**
- **SHIPDATE must be >= ORDDATE.**

```
create table ORDERS
(
 ordno     number(5)  constraint orders_pk  primary key,
orddate   date,  shipdate  date,
 custno    number(5) constraint orders_custno_pk references customers,
address1  varchar2(50),  address2  varchar2(50),  city
varchar2(30),  state    varchar2(30),  pin      varchar2(10),  phone
varchar2(30),
 constraint order_dates_chk  check( orddate <= shipdate)
);

insert into orders values(1001,'15-May-2001','10-jun-2001',102,   '43-
22-22','CBM Compound','Vizag','AP','530012','744545');

insert into orders values(1002,'15-May-2001','5-jun-2001',101,    '12-
22-29','Dwarakanagar','Vizag','AP','530016','453343,634333');

insert into orders values(1003,'17-May-2001','7-jun-2001',101,    '12-
22-29','Dwarakanagar','Vizag','AP','530016','453343,634333');

insert into orders values(1004,'18-May-2001','17-jun-2001',103,
'45-45-52','Abid Nagar', 'Vizag','AP','530016','567434');

insert into orders values(1005,'20-May-2001','3-jun-2001',104,
'33-34-
56','Muralinagar','Vizag','AP','530021','875655,876563,872222');

insert into orders values(1006,'23-May-2001','11-jun-2001',104,
'54-22-12','MVP Colony','Vizag','AP','530024',null);
```

LINEITEMS Table

Contains details of items ordered in each order. For each item in each order this table contains one row. The following is the structure of the table.

Column	Datatype	Meaning
OrdNo	Number(5)	Refers to the order number of the order.
Itemno	Number(5)	Refers to the item number of the item.
qty	number(3)	Howmany units of this item arerequired in this order.
price	Number(8,2)	Selling price of the item for this order.
DisRate	Number(4,2)	Discount Rate for this item in this order.

The following are the constraint related to ORDERS table.

- **Primary key is ORDNO and ITEMNO.**
- **ORDNO is a foreign key referencing ORDNO of ORDERS table.**
- **ITEMNO is a foreign key referencing ITEMNO of ITEMS table.**
- **Default DISRATE is 0**
- **QTY must be >= 1**
- **DISRATE must be >= 0**

```
create table LINEITEMS
(
 ordno   number(5)   constraint LINEITEMS_ORDNO_FK references ORDERS,
itemno  number(5)   constraint LINEITEMS_itemno_FK references ITEMS,
qty     number(3)   constraint LINEITEMS_qty_CHK CHECK( qty >= 1),
price   number(8,2),  disrate number(4,2) default 0
                     constraint LINEITEMS_DISRATE_CHK CHECK( disrate >=
0),
 constraint lineitems_pk primary key (ordno,itemno)
);


insert into lineitems values(1001,2,3,1000,10.0); insert
into lineitems values(1001,1,3,7000,15.0); insert into
lineitems values(1001,4,2,8000,10.0); insert into
lineitems values(1001,6,1,4500,10.0);

insert into lineitems values(1002,6,4,4500,20.0); insert
into lineitems values(1002,4,2,8000,15.0); insert into
lineitems values(1002,5,2,600,10.0);
```

```
insert into lineitems values(1003,5,10,500,0.0); insert
into lineitems values(1003,6,2,4750,5.0);

insert into lineitems values(1004,1,1,7000,10.0); insert
into lineitems values(1004,3,2,6500,10.0); insert into
lineitems values(1004,4,1,8000,20.0);

insert into lineitems values(1005,6,1,4600,10.0); insert
into lineitems values(1005,2,2,900,10.0);

insert into lineitems values(1006,2,10,950,20.0); insert
into lineitems values(1006,4,5,7800,10.0); insert into
lineitems values(1006,3,5,6600,15.0);
```

Queries

DISPLAY DETAILS OF ITEMS WHERE ITEMNAME CONTAINS LETTER 'O' TWICE

```
SELECT * FROM ITEMS
WHERE ITEMNAME LIKE '%O%O%';
```

DISPLAY ITEMNO,NAME,PRICE AND SELLING PRICE(PRICE+TAX) ROUND SELLING PRICE TO 100

```
SELECT ITEMNO, ITEMNAME, RATE, ROUND(RATE + RATE * TAXRATE /100)
"SPRICE"
FROM ITEMS;
```

DISPLAY DETAILS OF ITEMS BY PADDING ITEMNAME TO 20 CHARACTERS WITH '.' AND IN UPPERCASE

```
SELECT ITEMNO, UPPER(RPAD(ITEMNAME,20,'.')) ITEMNAME, RATE, TAXRATE
FROM ITEMS;
```

DISPLAY CUSTNO,NAME AND ADDRESS

```
COLUMN ADDRESS FORMAT A40
COLUMN PHONE  FORMAT A15
SELECT CUSTNO, CUSTNAME, TRIM(ADDRESS1 || ',' || ADDRESS2 || ',' ||
CITY ||
      ',' || STATE || ',' || PIN) ADDRESS , PHONE
FROM  CUSTOMERS;
```

DISPLAY ORDERDATE,APPROXIMATE SHIPDATE, WHICH WILL BE COMMING MONDAY AFTER 7 DAYS FROM ORDERDATE

```
SELECT ORDNO,ORDDATE, NEXT_DAY(ORDDATE+7,'MON') SHIPDATE
FROM ORDERS;
```

DISPLAY ALL THE ORDERS THAT ARE PLACED IN THE CURRENT MONTH

```
SELECT * FROM ORDERS
WHERE TO_CHAR(ORDDATE,'MMYY') = TO_CHAR(SYSDATE,'MMYY');
```

DISPLAY THE ORDERS THAT WERE PLACED IN THE LASTWEEK OF PREVIOUS MONTH

```
SELECT * FROM ORDERS WHERE  ORDDATE BETWEEN  LAST_DAY(
ADD_MONTHS(SYSDATE,-1)) - 7
       AND  LAST_DAY( ADD_MONTHS(SYSDATE,-1));
```

DISPLAY ORDERNO,ORDERDATE IN DD-MM HH24:MI FORMAT,SHIPDATE IF NOT AVAILABLE TAKE IT AS 15 DAYS FROM THE DAY OF ORDER

```
SELECT ORDNO, TO_CHAR(ORDDATE,'DD-MM HH24:MI') ORDDATE,
NVL(SHIPDATE,ORDDATE + 15)  SHIPDATE
FROM  ORDERS;
```

DISPALY TOTAL NO OF ORDERS

```
SELECT COUNT(*) "TOTAL NO. ORDERS"
FROM  ORDERS;
```

DISPLY ORDERNO,NO.OF ITEMS IN AN ORDER AND AVG RATE OF ORDERS

```
SELECT ORDNO, COUNT(*) "NO ITEMS", ROUND(AVG(PRICE),2) "AVERAGE RATE"
FROM LINEITEMS
GROUP BY ORDNO;
```

DISPLAY ORDERNO FOR ORDERS WHERE ATLEAST ONE PRODUCT IS HAVING RATE MORE THAN 5000 AND TOTAL NO.OF UNITS IS MORE THAN 10

```
SELECT ORDNO
FROM   LINEITEMS
GROUP  BY ORDNO
```

```
HAVING MAX(PRICE) > 5000 AND SUM(QTY) > 10;
```

DISPLAY MONTH NAME AND NO.OF ORDERS RECEIVED IN THE MONTH

```
SELECT  TO_CHAR(ORDDATE,'MONTH') MONTH, COUNT(*) "NO. ORDERS"
FROM  ORDERS
GROUP BY TO_CHAR(ORDDATE,'MONTH');
```

DISPLAY CUSTNO WHO HAVE PLACED MORE THAN 2 ORDERS IN THE LAST 3 MONTHS

```
SELECT CUSTNO
FROM   ORDERS
WHERE  ORDDATE >  ADD_MONTHS(SYSDATE,-3)
GROUP  BY CUSTNO
HAVING COUNT(*) > 2;
```

DISPLAY CUSTNO,NO.OF ORDERS ,DATE OF MOST RECENT ORDER

```
SELECT  CUSTNO, COUNT(*) "NO. ORDERS", MAX(ORDDATE) "RECENT ORDER ON"
FROM  ORDERS
GROUP BY CUSTNO;
```

DISPLAY CUSTNO,DATE ON WHICH FIRST ORDER WAS PLACED AND THE GAP BETWEEN FIRST ORDER AND LAST ORDER IN DAYS

```
SELECT CUSTNO, MIN(ORDDATE) "FIRST ORDER", MAX(ORDDATE)  -
MIN(ORDDATE) "GAP IN DAYS"
FROM   ORDERS
GROUP BY CUSTNO;
```

DISPLAY ORDERNO,MAX PRICE IN THE ORDER FOR THE ORDERS WHERE THE AMOUNT OF ITEMS IS MORE THAN 10000

```
SELECT  ORDNO, MAX(PRICE) "MAX PRICE"
FROM    LINEITEMS
GROUP BY ORDNO
HAVING SUM(PRICE * QTY) > 10000;
```

DISPLAY ITEMNO,TOTAL NO.OF UNITS SOLD,MAXPRICE,MINPRICE

```
SELECT ITEMNO, SUM(QTY) "TOTAL NO. UNITS", MAX(PRICE), MIN(PRICE)
FROM   LINEITEMS
GROUP  BY  ITEMNO;
```

DISPLAY CUSTNO,DATE,NO.OF ORDERS PLACED

```
SELECT CUSTNO, ORDDATE, COUNT(*) "NO. ORDRES"
FROM   ORDERS
GROUP BY CUSTNO, ORDDATE;
```

DISPLAY ORDERNO,CUSTNAME,ORDERDATE,NO.OF DATE BETWEEN SHIPDATE AND ORDERDATE FOR ORDERS THAT HAVE BEEN SHIPPED

```
SELECT ORDNO, CUSTNAME, ORDDATE, SHIPDATE - ORDDATE "DAYS"
FROM ORDERS O, CUSTOMERS C
WHERE  SHIPDATE IS NOT NULL AND  O.CUSTNO  = C.CUSTNO;
```

DISPLAY ORDERNO,ORDERDATE,CUSTNO,NAME FOR ALL THE ORDERS WHERE THE ORDER CONTAINS ORDER FOR ITEMNO 5.

```
SELECT O.ORDNO, ORDDATE, O.CUSTNO, CUSTNAME
FROM   ORDERS O, CUSTOMERS C, LINEITEMS L
WHERE  ITEMNO = 5 AND  L.ORDNO = O.ORDNO AND O.CUSTNO = C.CUSTNO;
```

The above query can also be written as follows.

```
SELECT ORDNO, ORDDATE, O.CUSTNO, CUSTNAME

FROM  ORDERS O, CUSTOMERS C

WHERE  O.CUSTNO = C.CUSTNO

    AND  ORDNO IN

        ( SELECT ORDNO FROM LINEITEMS WHERE ITEMNO = 5);
```

DISPLAY ITEMNO,NAME,ORDERNO,CUSTNAME AND AMOUNT.

```
SELECT I.ITEMNO, ITEMNAME, O.ORDNO, CUSTNAME, PRICE * QTY "AMOUNT"
FROM   CUSTOMERS C, ORDERS O, LINEITEMS L, ITEMS I
WHERE  O.CUSTNO = C.CUSTNO AND  O.ORDNO = L.ORDNO
       AND I.ITEMNO  = L.ITEMNO
```

DISPLAY DETAILS OF ORDEERS IN WHICH ORDERDATE IS AS MONDAY AND CUSTOMER RESIDES IN VSP

```
SELECT * FROM ORDERS
WHERE TO_CHAR(ORDDATE,'fmDAY') = 'MONDAY'
  AND  CUSTNO IN (SELECT CUSTNO FROM CUSTOMERS WHERE  CITY  LIKE
'%VIS%');
```

DISPLAY DETAILS OF CUSTOMERS WHO PLACED ANY ORDERS WORTH MORE THAN 30000

```
SELECT * FROM CUSTOMERS
WHERE  CUSTNO IN
  ( SELECT CUSTNO
    FROM ORDERS
    WHERE  ORDNO IN
        ( SELECT  ORDNO
          FROM   LINEITEMS
          GROUP BY ORDNO
          HAVING  SUM(QTY*PRICE) > 30000)
  );
```

DISPLAY DETAILS OF ITEMS FOR WHICH THERE IS AN ORDER IN THE CURRENT MONTH

```
SELECT * FROM ITEMS
WHERE ITEMNO IN
    ( SELECT ITEMNO
      FROM LINEITEMS
      WHERE ORDNO IN
       ( SELECT ORDNO
         FROM   ORDERS          WHERE  TO_CHAR(ORDDATE,'MMYY') =
TO_CHAR(SYSDATE,'MMYY')
       )
    );
```

DISPLAY DETAILS OF ORDER IN WHICH WE SOLD ITEM 3 FOR MAX PRICE

```
 SELECT * FROM ORDERS
 WHERE ORDNO IN
    (
       SELECT ORDNO
       FROM   LINEITEMS
       WHERE  PRICE =
         ( SELECT MAX(PRICE) FROM  LINEITEMS
           WHERE  ITEMNO = 3)
       AND  ITEMNO = 3
    );
```

DISPLAY DETAILS OF ITEMS FOR WHICH THERE IS AN ORDER IN THE LAST 7 DAYS OR TOTAL NO.OF UNITS ORDERED IS MORE THAN 10.

```
SELECT * FROM ITEMS
WHERE ITEMNO IN
   (SELECT ITEMNO
    FROM   LINEITEMS
    WHERE  ORDNO IN
       (SELECT ORDNO FROM ORDERS WHERE SYSDATE-ORDDATE <= 7)
   )
   OR ITEMNO IN
     ( SELECT ITEMNO
       FROM   LINEITEMS
       GROUP BY ITEMNO
       HAVING SUM(QTY) > 10
     );
```

DISPLAY ALL THE LINEITEMS IN WHICH THE RATE OF THE ITEM IS MORE THAN AVG RATE OF THE ITEMS

```
SELECT * FROM LINEITEMS L
WHERE  PRICE >
  (SELECT  AVG(PRICE)
   FROM LINEITEMS
   WHERE  ITEMNO = L.ITEMNO);
```

DISPLAY DETAILS OF CUSTOMER WHO HAS PLACED MAX NO OF ORDERS

```
SELECT * FROM CUSTOMERS
WHERE CUSTNO IN
  ( SELECT  CUSTNO
    FROM    ORDERS
    GROUP   BY CUSTNO
    HAVING  COUNT(*) =
      (
        SELECT MAX(COUNT(*))
        FROM   ORDERS
        GROUP BY CUSTNO
      )
   );
```

DISPLAY DETAILS OF ORDERS IN WHICH ATLEAST ONE ITEM IS SOLD FOR HIGHER RATE THAN ACTUAL RATE

```
SELECT * FROM ORDERS
WHERE  ORDNO  IN
 ( SELECT  ORDNO
   FROM    LINEITEMS L, ITEMS I
   WHERE   L.ITEMNO = I.ITEMNO
           AND PRICE > RATE );
```

DETAILS OF CUSTOMERS WHO HAVE NOT PLACED ANY ORDER FOR THE LAST 15 DAYS

```
SELECT * FROM CUSTOMERS
WHERE  CUSTNO NOT IN
 (SELECT CUSTNO
  FROM   ORDERS
  WHERE  SYSDATE - ORDDATE <= 15);
```

DISPLAY DETAILS OF ITEMS FOR WHICH THERE WAS NO ORDER IN THE PREVIOUS MONTH

```
SELECT * FROM ITEMS
WHERE ITEMNO NOT IN
 (
    SELECT ITEMNO
    FROM   LINEITEMS
    WHERE  ORDNO IN
       ( SELECT ORDNO
         FROM   ORDERS
         WHERE  TO_CHAR(ORDDATE,'MMYY') = TO_CHAR(
ADD_MONTHS(SYSDATE,-1),'MMYY')
       )
 );
```

DISPLAY ORDERS WHERE ORDDATE IS IN THE CURRENT MONTH OR AFTER ORDER 1004.

```
SELECT O1.*
```

```
FROM ORDERS O1, ORDERS O2
WHERE TO_CHAR( O1.ORDDATE,'MMYY') = TO_CHAR(SYSDATE,'MMYY')
      OR (O2.ORDNO = 1004 AND O1.ORDDATE > O2.ORDDATE);
```

DISPLAY DETAILS OF ITEMS THAT ARE PURCHASED BY CUSTOMER 102

```
SELECT * FROM ITEMS
WHERE ITEMNO IN
       (  SELECT ITEMNO
          FROM   LINEITEMS
          WHERE  ORDNO IN
             ( SELECT ORDNO
               FROM ORDERS
               WHERE  CUSTNO = 102
             )
       );
```

DISPLAY DETAILS OF ITEMS THAT ARE PURCHASED BY CUSTOMER 102

```
SELECT * FROM ITEMS
WHERE ITEMNO IN
       (  SELECT ITEMNO
          FROM   LINEITEMS
          WHERE  ORDNO IN
             ( SELECT ORDNO
               FROM ORDERS
               WHERE  CUSTNO = 102
             )
       );
```

CHANGE SHIPDATE OF ORDER 1004 TO THE ORDER DATE OF MOST RECENT ORDER

```
UPDATE ORDERS
    SET SHIPDATE = ( SELECT  MAX(ORDDATE)
                     FROM ORDERS)
WHERE  ORDNO = 1004;
```

DISPLAY THE DETAILS OF ITEMS WHERE ITEMNAME CONTAINS LETTER O OR M

```
SELECT * FROM ITEMS
WHERE ITEMNAME LIKE '%O%' OR ITEMNAME LIKE '%M%';
```

DISPLAY DETAILS OF ORDERS THAT WERE PLACED IN THE MONTH OF JUNE 2000.

```
SELECT * FROM ORDERS
WHERE ORDDATE BETWEEN '01-JUN-2000' AND '30-JUN-2000';
```

DISPLAY ORDERNO,ORDERDATE AND APPROXIMATE SHIPDATE(15 DAYS FROM ORDDATE) FOR ALL ORDERS THAT ARE NOT SHIPPED.

```
SELECT ORDNO, ORDDATE, ORDDATE + 15 "SHIPDATE"
FROM ORDER WHERE SHIPDATE IS NULL;
```

DISPLAY ITEMNO,ORDERNO AND TOTAL AMOUNT AFTER ROUNDING THE VALUE TO 100'S FOR ALL THE ITEMS WHERE THE QUANTITY IS MORE THAN 5 UNITS OR PRICE IS LESS THAN 5000.

```
SELECT ITEMNO, ORDNO, ROUND(QTY*PRICE,-2) "TOTAL"
FROM LINEITEMS
WHERE  QTY > 5 OR PRICE < 5000;
```

DISPLAY ITEMNO,ITEMNAME,PRICE AND TAX FOR ITEMS THAT ARE TAXABLE.

```
SELECT ITEMNO, ITEMNAME, PRICE , PRICE * TAX /100 "TAX"
FROM ITEMS
WHERE TAXRATE IS NOT NULL;
```

DISPLAY ORDERNO,CUSTMERNO,ORDERDATE,NO. OF DAYS BETWEEN DAYS ORDERDATE AND SYSTEM DATE AND DATE ON WHICH THE AMOUNT SHOULD BE COLLECTED, WHICH IS 5TH OF NEXT MONTH OF THE MONTH IN WHICH ITEMS ARE DELIVERED.

```
  SELECT ORDNO, CUSTNO, ORDDATE, SYSDATE - ORDDATE "NODAYS" ,
LAST_DAY(SHIPDATE) + 5 "COLLDATE"
FROM ORDERS
WHERE SHIPDATE IS NOT NULL;
```

DISPLAY THE DETAILS OF ORDERS THAT PLACED IN THE LAST 20 DAYS AND DELIVERED.

```
SELECT * FROM ORDERS
WHERE  SYSDATE - ORDDATE <= 20 AND SHIPDATE IS NOT NULL;
```

CHANGE THE RATE OF ITEMS IN ORDER 1003 SO THAT 10% DISCOUNT IS GIVEN TO ALL ITEMS.

```
UPDATE LINEITEMS SET  PRICE = PRICE * 0.90
WHERE ORDNO = 1003;
```

DISPLAY THE ITEMS WHERE ITEMNAME CONTAINS MORE THAN 10 CHARACTERS.

```
SELECT * FROM ITEMS
WHERE  LENGTH(ITEMNAME) > 10;
```

DISPLAY ITEMS WHERE ITEMNAME CONTAINS LETTER ‘O’ AFTER 5TH POSITION.

```
SELECT * FROM ITEMS
WHERE INSTR(ITEMNAME,'O') > 5;
```

DISPLAY FIRST NAME OF THE CUSTOMER.

```
SELECT  SUBSTR(ITEMNAME,1, INSTR(ITEMNAME,' ') -1 ) "FIRST NAME"
FROM CUSTOMERS;
```

DISPLAY ITEMNO,ITEMNAME IN UPPER CASE FOR ALL ITEMS WHERE THE LETTER ‘M’ IS EXISTING IN ANY CASE.

```
SELECT ITEMNO, UPPER(ITEMNAME)
FROM ITEMS
WHERE  UPPER(ITEMNAME) LIKE '%M%';
```

DISPLAY THE ORDERS THAT ARE PLACED IN THE CURRENT MONTH.

```
SELECT * FROM ORDERS
WHERE  TO_CHAR(ORDDATE,'YYMM') = TO_CHAR(SYSDATE,'YYMM');
```

INSERT INTO A NEW ORDER WITH THE FOLLOWING: ORDERNO-1010,CUSTOMERNO-105,ORDERDATE-13-JULY-2001 AT 4:45 PM,SHIPDATE-NULL, SHIPADDRESS-NULL.

```
INSERT INTO ORDERS VALUES(1010,TO_DATE('13-07-2001 16:45','DD-MM-YYYY HH24:MI'),NULL,105,
                            NULL,NULL,NULL,NULL,NULL,NULL);
```

DISPLAY ORDERNO,CUSTOMERNO,THE NO. OF DAYS BETWEEN SHIPDATE AND ORDERDATE.IF SHIPDATE IS-NOT AVAILABLE, TAKE IT AS SYSTEM DATE.

```
SELECT  ORDNO,CUSTNO, NVL(SHIPDATE,SYSDATE)-ORDDATE
FROM ORDERS;
```

DISPLAY ITEMNO,PRICE,QUANTITY,DISCOUNT RATE FOR ITEMS WHERE THE DISCOUNT RATE IS NON-ZERO. DISCOUNT-RATE IS CALUCULATED AS 10% FOR ITEM 1,7% FOR ITEM 6 AND 8% FOR REMAINING.

```
SELECT ITEMNO, PRICE, QTY, DECODE(ITEMNO,1,10,6,7,10) "DISRATE"
FROM LINEITEMS
WHERE DISRATE <> 0
```

DISPLAY TOTAL AMOUNT OF ORDERS WE RECEIVED SO FAR.

```
SELECT  SUM(QTY*PRICE)
FROM LINEITEMS;
```

DISPLAY CUSTOMERNO,MONTH-NAME,NO. OF ORDERS OF THE CURRENT YEAR.

```
SELECT  CUSTNO, TO_CHAR(ORDDATE,'MONTH'), COUNT(*)
FROM ORDERS
GROUP BY CUSTNO, TO_CHAR(ORDDATE,'MONTH');
```

DISPLAY DIFFERENCE BETWEEN HIGHEST PRICE AND LOWEST PRICE AT WHICH THE ITEM WAS SOLD.

```
SELECT MAX(PRICE) - MIN(PRICE)
FROM LINEITEMS
GROUP BY ITEMNO;
```

DISPLAY HOW MANY ORDERS ARE STILL PENDING.

```
SELECT COUNT(*)
FROM ORDERS
WHERE SHIPDATE IS NULL;
```

DISPLAY ORDERNO,AVERAGE OF PRICE BY TAKING INTO ORDERS THAT WERE PLACED IN THE LAST 15 DAYS.

```
SELECT O.ORDNO, AVG(PRICE)
FROM  ORDERS O, LINEITEMS L
WHERE  O.ORDNO = L.ORDNO AND  SYSDATE - ORDDATE <= 15
GROUP BY O.ORDNO;
```

DISPLAY YEAR,NO.OF ORDERS IN WHICH THE DIFFERENCE BETWEEN SHIPDATE AND ORDERDATE IS LESS THAN 10 DAYS.

```
SELECT  TO_CHAR(ORDDATE,'YYYY'), COUNT(*)
FROM  ORDERS
WHERE  SHIPDATE  - ORDDATE <=10
GROUP BY TO_CHAR(ORDDATE,'YYYY');
```

DISPLAY STATE,NO.OF CUSTOMERS IN THE STATE WHERE THE CUSTOMER NAME CONTAINS THE WORD ‘NIKE’.

```
SELECT STATE, COUNT(*)
FROM  CUSTOMERS
WHERE  CUSTNAME LIKE '%NIKE%'
GROUP BY STATE;
```

DISPLAY CUSTOMER WHO HAS PLACED MORE THAN 2 ORDERS IN A SINGLE MONTH.

```
SELECT CUSTNO
FROM ORDERS
GROUP BY  CUSTNO, TO_CHAR(ORDDATE,'MMYY')
HAVING COUNT(*) > 2;
```

DISPLAY HIGHEST NO.OF ORDERS PLACED BY A SINGLE CUSTOMER.

```
SELECT  MAX( COUNT(*))
FROM  ORDERS
GROUP BY  CUSTNO;
```

DISPLAY CUSTOMERNO,NO.OF COMPLETED ORDERS AND NO.OF INCOMPLETE ORDERS.

```
SELECT CUSTNO, SUM( DECODE(SHIPDATE,NULL,1,0) ) "INCOMP ORDERS", SUM(
DECODE(SHIPDATE,NULL,0,1)) "COMP ORDERS"
FROM  ORDERS
GROUP BY CUSTNO;
```

DISPLAY ORDERNO,ITEMNO,ITEMNAME,PRICE AT WHICH ITEM IS SOLD AND CURRENT PRICE OF THE ITEM.

```
SELECT  ORDNO, L.ITEMNO, ITEMNAME, PRICE,RATE
FROM  LINEITEMS L , ITEMS I
WHERE  L.ITEMNO  = I.ITEMNO;
```

DISPLAY ORDERNO,ITEMNO,AMOUNT FOR ITEMS WHERE THE PRICE OF THE ITEM IS MORE THAN THE CURRENT PRICE OF THE ITEM.

```
SELECT ORDNO, L.ITEMNO, QTY  * PRICE
FROM LINEITEMS L, ITEMS I
WHERE PRICE > RATE
   AND  L.ITEMNO = I.ITEMNO;
```

DISPLAY ITEMNO,ITEMNAME,ORDERNO,DIFFERENCE BETWEEN CURRENT PRICE AND SELLING PRICE FOR THE ITEMS WHERE THERE IS A DIFFERENCE BETWEEN CURRENT PRICE AND SELLING PRICE.

```
SELECT L.ITEMNO, ITEMNAME, ORDNO, RATE- PRICE
FROM   ITEMS I, LINEITEMS L
WHERE  I.ITEMNO = L.ITEMNO AND  RATE <>PRICE;
```

DISPLAY CUSTOMERNO,CUTOMER NAME,ORDERNO, ORDERDATE FOR ORDERS WHERE THE SHIPADDRESS AND CUSTOMER ADDRESS ARE SAME.

```
SELECT O.CUSTNO, CUSTNAME, ORDNO, ORDDATE
FROM  ORDERS O, CUSTOMERS C
```

```
WHERE  O.ADDRESS1 = C.ADDRESS1 AND  O.ADDRESS2= C.ADDRESS2  AND C.CITY
= O.CITY
AND C.STATE  = O.STATE AND C.PIN = O.PIN;
```

DISPLAY ITEMNO,ITEMNAME,ORDERNO,QUANTITY REQUIRED FOR ALL ITEMS (THAT ARE NOT EVEN ORDERED FOR).

```
SELECT  I.ITEMNO, ITEMNAME, ORDNO, QTY
FROM    LINEITEMS L , ITEMS I
WHERE   I.ITEMNO =  L.ITEMNO(+);
```

DISPLAY NO.OF ORDERS PLACED BY CUSTOMERS RESIDING IN VIZAG.

```
SELECT O.CUSTNO, COUNT(*)
FROM  ORDERS O, CUSTOMERS C
WHERE  O.CUSTNO = C.CUSTNO AND  C.CITY = 'VIZAG'
GROUP BY  O.CUSTNO;
```

DISPLAY ORDERNO,CUSTOMER NAME,DIFFERENCE BETWEEN SYSTEM DATE AND ORDERDATE FOR ORDERS THAT HAVE NOT BEEN SHIPPED AND OLDER THAN 10 DAYS.

```
SELECT ORDNO, CUSTNAME, SYSDATE - ORDDATE
FROM   ORDERS O, CUSTOMERS C
WHERE  O.CUSTNO = C.CUSTNO AND SYSDATE - ORDDATE > 10 AND SHIPDATE IS
NULL;
```

DISPLAY CUSTOMER NAME AND TOTAL AMOUNT OF ITEMS PURCHASED BY CUSTOMER.

```
SELECT  CUSTNAME, SUM(QTY * PRICE)
FROM    LINEITEMS L, ORDERS O, CUSTOMERS C
WHERE   L.ORDNO = O.ORDNO AND O.CUSTNO = C.CUSTNO
GROUP  BY CUSTNAME;
```

DISPLAY THE DETAILS OF ITEM THAT HAS HIGHEST PRICE.

```
   SELECT * FROM ITEMS
   WHERE  RATE = ( SELECT  MAX(RATE) FROM ITEMS);
```

DISPLAY DETAILS OF CUSTOMERS WHO PLACED MORE THAN 5 ORDERS.

```
SELECT  * FROM CUSTOMERS
WHERE CUSTNO IN ( SELECT CUSTNO FROM ORDERS GROUP BY CUSTNO HAVING
COUNT(*) > 5);
```

DISPLAY DETAILS OF CUTOMERS WHO HAVE NOT PLACED ANY ORDER.

```
SELECT * FROM CUSTOMERS
WHERE CUSTNO NOT IN ( SELECT CUSTNO FROM ORDERS);
```

DISPLAY DETAILS OF CUTOMERS WHO HAVE PLACED AN ORDER IN THE LAST 6 MONTHS.

```
SELECT * FROM CUSTOMERS
WHERE CUSTNO IN ( SELECT CUSTNO FROM ORDERS WHERE
MONTHS_BETWEEN(SYSDATE,ORDDATE) <= 6);
```

DISPLAY THE ITEMS FOR WHICH WE HAVE SOLD MORE THAN 50 UNITS BY TAKING INTO ORDERS WHERE THE PRICE IS MORE THAN 5000.

```
SELECT * FROM ITEMS
WHERE  ITEMNO IN ( SELECT ITEMNO FROM LINEITEMS WHERE  PRICE > 5000
GROUP BY ITEMNO
                   HAVING SUM(QTY) > 50);
```

DISPLAY THE DETAILS OF ORDERS THAT WERE PLACED BY A CUSTOMER WITH PHONE NUMBER STARTING WITH 541 OR THE ORDERS IN WHICH WE HAVE MORE THAN 5 ITEMS.

```
SELECT * FROM ORDERS
WHERE CUSTNO IN (SELECT CUSTNO FROM CUSTOMERS WHERE  PHONE LIKE
'541%')
  OR  ORDNO IN  (SELECT ORDNO FROM LINEITEMS GROUP BY ORDNO HAVING
COUNT(*) > 5);
```

CHANGE THE RATE OF ITEMNO 1 IN ITEMS TABLE TO THE HIGHEST RATE OF LINEITEMS TABLE OF THAT ITEM.

```
UPDATE  ITEMS  SET  RATE  = ( SELECT MAX(PRICE) FROM LINEITEMS WHERE
ITEMNO = 1)
WHERE ITEMNO = 1;
```

DELETE CUSTOMERS WHO HAVE NOT PLACED ANY ORDER.

```
DELETE FROM CUSTOMERS WHERE CUSTNO NOT IN ( SELECT CUSTNO FROM
ORDERS);
```

RENAME COLUMN RATE IN ITEMS TO PRICE

```
STEP1: CREATE TABLE  NEWITEMS AS SELECT ITEMNO, ITEMNAME, RATE PRICE,
TAXRATE
       FROM  ITEMS;

STEP2: DROP TABLE ITEMS;
STEP3: RENAME NEWITEMS TO ITEMS;
```

DISPLAY DETAILS OF CUSTOMERS WHO HAVE PLACED MAXIMUM NUMBER OF ORDERS.

```
SELECT  * FROM CUSTOMERS
WHERE  CUSTNO IN  ( SELECT  CUSTNO FROM ORDERS
                    GROUP BY CUSTNO HAVING COUNT(*) =
                         ( SELECT  MAX(COUNT(*))
                           FROM   ORDERS
                           GROUP BY  CUSTNO));
```

DISPLAY DETAILS OF CUSTOMERS WHO HAVEN’T PLACED ANY ORDER IN THAT CURRENT MONTH.

```
SELECT * FROM CUSTOMERS
WHERE CUSTNO NOT IN ( SELECT  CUSTNO FROM ORDERS WHERE
TO_CHAR(ORDDATE,'MMYY') =
                             TO_CHAR(SYSDATE,'MMYY'));
```

DISPLAY DETAILS OF ITEMS FOR WHICH THERE WAS NO ORDER IN THE CURRENT MONTH BUT THERE WAS AN ORDER IN THE PREVIOUS MONTH.

```
SELECT  *  FROM ITEMS
WHERE  ITEMNO IN ( SELECT ITEMNO FROM LINEITEMS L, ORDERS O
                   WHERE  L.ORDNO = O.ORDNO  AND
                   TO_CHAR( ADD_MONTHS(SYSDATE,-1),'MMYY') =
TO_CHAR(ORDDATE,'MMYY'))
```

```
AND  ITEMNO NOT IN (SELECT ITEMNO FROM LINEITEMS L, ORDERS O
                   WHERE  L.ORDNO = O.ORDNO  AND
                   TO_CHAR(SYSDATE,'MMYY') = TO_CHAR(ORDDATE,'MMYY'));
```

DISPLAY DETAILS OF ITEMS THAT WERE PURCHASED BY CUSTOMER WHO HAS PLACED MORE THAN 3 ORDERS.

```
SELECT * FROM ITEMS
WHERE ITEMNO IN ( SELECT ITEMNO FROM LINEITEMS
                  WHERE  ORDNO IN ( SELECT  ORDNO FROM ORDERS
                                    WHERE  CUSTNO IN (
                                                        SELECT  CUSTNO
                                                        FROM ORDERS
                                                GROUP BY  CUSTNO
                                                HAVING COUNT(*) > 1
                                              )
                                )
              );
```

DISPLAY THE ORDERS IN WHICH THE GAP BETWEEN SHIPDATE AND ORDERDATE IS MORE THAN THE AVERAGE GAP FOR INDIVIDUAL CUSTOMERS.

```
SELECT  * FROM ORDERS  O
WHERE  SHIPDATE - ORDDATE >
       (SELECT  AVG(SHIPDATE - ORDDATE)
        FROM ORDERS
        WHERE CUSTNO = O.CUSTNO);
```

DISPLAY THE DETAILS OF ITEMS IN WHICH THE CURRENT PRICE IS MORE THAN THE MAXIMUM PRICE AT WHICH WE SOLD IT.

```
SELECT * FROM ITEMS I
WHERE  RATE >
     ( SELECT  MAX(PRICE)
       FROM LINEITEMS
       WHERE  ITEMNO = I.ITEMNO);
```

CREATE A NEW TABLE ‘COMPORDERS’ WITH ORDNO, CUSTOMERNAME,ORDERDATE,SHIPDATE,DIFFERENCE BETWEEN SHIPDATE AND ORDERDATE.

```
CREATE TABLE COMPORDERS AS SELECT  ORDNO, CUSTNAME,ORDDATE, SHIPDATE,
SHIPDATE-ORDDATE "NODAYS"
FROM  ORDERS O, CUSTOMERS C
WHERE  O.CUSTNO= C.CUSTNO AND SHIPDATE IS NOT NULL;
```

DISPLAY THE ITEMS THAT HAVE TOP 3 HIGHEST PRICES.

```
SELECT  * FROM ITEMS I
WHERE  2 >= ( SELECT COUNT(*) FROM ITEMS WHERE RATE > I.RATE)
ORDER BY RATE DESC;
```

DISPLAY DETAILS OF ITEM THAT HAS SECOND LOWEST PRICE.

```
SELECT  * FROM ITEMS I
WHERE  1 = ( SELECT  COUNT(*) FROM ITEMS WHERE RATE < I.RATE)
```

<

ADD A NEW ITEM TO THE LAST ORDER PLACED BY CUSTOMER 106 WITH THE FOLLOWING DETAILS- ITEMNO-3,QUANTITY-2,PRICE AS THE CURRENT RATE OF THE ITEM,DISCOUNT-8%.

```
DECLARE
   V_ORDNO  ORDERS.ORDNO%TYPE;
   V_RATE   ITEMS.RATE%TYPE;

BEGIN
    SELECT  MAX(ORDNO) INTO V_ORDNO
    FROM  ORDERS WHERE CUSTNO = 106;

    SELECT RATE INTO V_RATE
    FROM   ITEMS WHERE  ITEMNO = 3;

    INSERT INTO LINEITEMS VALUES (V_ORDNO,3,2,V_RATE,8);

END;
```

CHANGE RATE OF ITEM 5 TO EITHER AVERAGE RATE OF ITEM 5 OR CURRENT RATE WHICHEVER IS HIGHER.

```
DECLARE
   V_APRICE LINEITEMS.PRICE%TYPE;
   V_RATE   ITEMS.RATE%TYPE;

BEGIN
    SELECT  AVG(PRICE) INTO V_APRICE
    FROM   LINEITEMS WHERE ITEMNO = 5;

    SELECT  RATE INTO V_RATE
    FROM  ITEMS WHERE ITEMNO  = 5;

    UPDATE  ITEMS SET RATE = GREATEST( V_APRICE, V_RATE)
    WHERE ITEMNO = 5;
END;
```

INSERT A NEW ROW INTO LINEITEMS WITH THE FOLLOWING DETAILS. ORDERNO IS THE LAST ORDER PLACED BY CUSTOMERNO 102,ITEMNO IS THE ITEM OF P3 PROCESSOR, RATE IS LOWEST RATE OF THAT ITEM,QUANTITY IS 2,DISCOUNT IS 10% IF ITEM'S CURRENT RATE IS MORE THAN THE LEAST RATE OTHERWISE NO DISCOUNT.

```
DECLARE
   V_ORDNO  ORDERS.ORDNO%TYPE;
   V_PRICE  LINEITEMS.PRICE%TYPE;
   V_DIS    NUMBER(2);
   V_RATE   ITEMS.RATE%TYPE;
   V_ITEMNO ITEMS.ITEMNO%TYPE;

BEGIN
   SELECT  MAX(ORDNO) INTO V_ORDNO
   FROM    ORDERS WHERE CUSTNO = 102;

   SELECT ITEMNO, RATE INTO V_ITEMNO , V_RATE
   FROM  ITEMS WHERE  UPPER(ITEMNAME) = 'PIII PROCESSOR';

   -- GET LOWEST RATE OF THE ITEM

   SELECT MIN(PRICE) INTO V_PRICE
   FROM   LINEITEMS
   WHERE  ITEMNO = V_ITEMNO;

   IF  V_RATE > V_PRICE THEN
         V_DIS := 10;
   ELSE
         V_DIS := 0;
   END IF;

   INSERT INTO LINEITEMS VALUES  ( V_ORDNO, V_ITEMNO, 2, V_PRICE,
V_DIS);

END;
```

DISPLAY THE HIGHEST OF THE MISSING ORDERNOS.

```
DECLARE

     V_MAXORDNO ORDERS.ORDNO%TYPE;
     V_MINORDNO ORDERS.ORDNO%TYPE;
     V_CNT      NUMBER(2);

BEGIN
    SELECT  MAX(ORDNO), MIN(ORDNO) INTO  V_MAXORDNO, V_MINORDNO
FROM    ORDERS;

    FOR I IN REVERSE V_MINORDNO..V_MAXORDNO
    LOOP
        SELECT COUNT(*) INTO V_CNT
        FROM   ORDERS WHERE  ORDNO = I;

        IF  V_CNT = 0 THEN
            DBMS_OUTPUT.PUT_LINE(I);
            EXIT;
        END IF;
    END LOOP;

END;
```

DISPLAY CUSTOMER NAMES OF THE CUSTOMERS WHO HAVE PLACED MORE THAN 3 ORDERS WHERE THE TOTAL AMOUNT OF THE ORDER IS MORE THAN 10,000.

```
 SELECT  CUSTNAME
 FROM  CUSTOMERS
 WHERE CUSTNO IN ( SELECT  CUSTNO
                   FROM    ORDERS
```

```
                  WHERE  ORDNO IN ( SELECT ORDNO
                                    FROM   LINEITEMS
                                    GROUP BY ORDNO
                                HAVING  SUM(QTY*PRICE) > 10000)
                  GROUP BY  CUSTNO
                  HAVING  COUNT(*) > 1 );
```

CHANGE THE RATE OF EACH ITEM AS FOLLOWS (1) INCREASE THE RATE BY 10% IF THE ITEM WAS SOLD IN MORE THAN 5 ITEMS. (2) INCREASE THE RATE BY 2% IF AVERAGE PRICE IS GREATER THAN CURRENT PRICE, OTHERWISE DECREASE THE PRICE BY 3%.

```
DECLARE
   CURSOR  CITEMS IS
     SELECT  ITEMNO,COUNT(*) CNT, AVG(PRICE) APRICE  FROM LINEITEMS
     GROUP BY ITEMNO;

   V_PER NUMBER(5,2);
   V_RATE ITEMS.RATE%TYPE;

BEGIN
   FOR REC  IN  CITEMS
   LOOP


       IF  REC.CNT > 5 THEN
           V_PER := 0.90;
       ELSE
           -- GET CURRENT RATE
           SELECT  RATE INTO V_RATE
           FROM  ITEMS WHERE ITEMNO =  REC.ITEMNO;

           IF REC.APRICE > V_RATE THEN
               V_PER := 1.02;
           ELSE
               V_PER := 0.97;
           END IF;
       END IF;

       UPDATE ITEMS SET RATE = RATE * V_PER
       WHERE ITEMNO = REC.ITEMNO;
   END LOOP;

END;
```

CREATE A NEW TABLE CALLED CUSTSUM AND STORE THE FOLLOWING DATA INTO THE TABLE - CUSTOMERNO,CUSTOMER NAME,NO.OF ORDERS PLACED, DATE OF MOST RECENT ORDER AND TOTAL AMOUNT OF ALL THE ORDERS.

```
BEFORE THIS PROGRAM IS RUN, YOU HAVE TO CREATE TABLE AS FOLLOWS:

CREATE TABLE CUSTSUM
(  CUSTNO NUMBER(5),
   CUSTNAME VARCHAR2(20),
   NOORD    NUMBER(5),
   RORDDATE DATE,
   TOTAMT   NUMBER(10)
);


DECLARE
   CURSOR CUSTCUR IS
        SELECT  CUSTNO, CUSTNAME FROM CUSTOMERS;
   V_ORDCNT NUMBER(5);
   V_MORDDATE DATE;
   V_TOTAMT  NUMBER(10);


BEGIN



    FOR REC IN  CUSTCUR
    LOOP
        -- GET DETAILS OF CUSTOMER
        SELECT COUNT(*), MAX(ORDDATE), SUM(QTY*PRICE) INTO V_ORDCNT,
V_MORDDATE, V_TOTAMT
        FROM  ORDERS O, LINEITEMS L         WHERE  O.ORDNO
= L.ORDNO AND  CUSTNO = REC.CUSTNO;
        INSERT INTO CUSTSUM VALUES ( REC.CUSTNO, REC.CUSTNAME,
V_ORDCNT, V_MORDDATE,V_TOTAMT);
```

```
    END LOOP;

END;
```

DISPLAY ITEMNAMES OF ITEMS FOR WHICH THE CURRENT PRICE IS LESS THAN THE AVERAGE PRICE OR TOTAL QUANTITY SOLD IS LESS THAN 10 UNITS.

```
SELECT ITEMNAME
FROM  ITEMS
WHERE ITEMNO  IN
   ( SELECT  ITEMNO
     FROM ITEMS I
     WHERE  RATE < ( SELECT AVG(PRICE) FROM LINEITEMS WHERE ITEMNO =
I.ITEMNO)
   )
OR  ITEMNO IN
   ( SELECT ITEMNO
     FROM LINEITEMS
     GROUP BY ITEMNO
     HAVING SUM(QTY) >  10 );
```

CREATE A PROCEDURE THAT TAKES ORDERNO,ITEMNO AND INSERTS A ROW INTO LINEITEMS, PRICE-RATE OF THE ITEM, QTY-1,DISCOUNT-10%.

```
CREATE OR REPLACE PROCEDURE NEWITEM(P_ORDNO NUMBER, P_ITEMNO NUMBER)
AS
    V_RATE ORDERS.ORDNO%TYPE;
BEGIN
    SELECT RATE INTO  V_RATE
    FROM  ITEMS WHERE ITEMNO = P_ITEMNO;

    INSERT INTO LINEITEMS VALUES ( P_ORDNO, P_ITEMNO, V_RATE, 1, 10);

END;
```

CREATE A FUNCTION THAT RETURNSTHE FIRST MISSING ORDERNO.

```
CREATE OR REPLACE FUNCTION  FIRSTMISORDNO RETURN NUMBER
```

```
AS
     V_MAXORDNO ORDERS.ORDNO%TYPE;
     V_MINORDNO ORDERS.ORDNO%TYPE;
     V_CNT      NUMBER(2);

BEGIN
    SELECT  MAX(ORDNO), MIN(ORDNO) INTO  V_MAXORDNO, V_MINORDNO
    FROM    ORDERS;

    FOR I IN V_MINORDNO..V_MAXORDNO
    LOOP
        SELECT COUNT(*) INTO V_CNT
        FROM   ORDERS WHERE  ORDNO = I;

        IF  V_CNT = 0 THEN
           RETURN  I;
        END IF;
    END LOOP;

    -- NO MISSING ORDNO
    RETURN NULL;
END;
```

CREATE A FUNCTION THAT TAKES ORDERNO AND RETURNS CUSTOMER NAME OF THAT ORDER.

```
CREATE OR REPLACE FUNCTION  GETCUSTNAME ( P_ORDNO NUMBER) RETURN
VARCHAR2
IS
    V_CUSTNAME VARCHAR2(30);
BEGIN
      SELECT  CUSTNAME  INTO V_CUSTNAME
      FROM CUSTOMERS
      WHERE CUSTNO = ( SELECT CUSTNO FROM ORDERS WHERE ORDNO =
P_ORDNO);

      RETURN  V_CUSTNAME;
END;
```

CREATE A PROCEDURE THAT INSERTS A NEW ROW INTO LINEITEMS WITH GIVEN ITEMNO,PRICE,QUANTITY, ORDERNO IS THE MOST RECENT ORDER. CHECK

WHETHER PRICE IS MORE THAN THE CURRENT RATE OF THE ITEM, CHECK WHETHER ITEM IS ALREADY EXISTING IN THE ORDER AND CHECK WHETHER THE TOTAL AMOUNT OF THE ORDER INCLUDING THE NEW ITEM HAS EXCEEDED 50,000.

```
CREATE OR REPLACE PROCEDURE  NEWITEMS(P_ITEMNO NUMBER, P_PRICE NUMBER,
P_QTY NUMBER)
IS
    V_CNT NUMBER(2);
    V_RATE ITEMS.RATE%TYPE;
    V_TOTAMT NUMBER(10);
    V_ORDNO ORDERS.ORDNO%TYPE;
BEGIN
    SELECT MAX(ORDNO) INTO V_ORDNO FROM ORDERS;

    -- CHECK CONDITIONS

    SELECT RATE INTO V_RATE  FROM ITEMS WHERE ITEMNO = P_ITEMNO;

    IF P_PRICE > V_RATE THEN
          RAISE_APPLICATION_ERROR(-20001,'PRICE IS MORE THAN CURRENT
PRICE');
    END IF;

    SELECT COUNT(*) INTO V_CNT
    FROM  LINEITEMS
    WHERE  ORDNO = V_ORDNO AND ITEMNO = P_ITEMNO;

    IF  V_CNT = 1 THEN
         RAISE_APPLICATION_ERROR(-20002,'ITEM IS ALREADY EXISTING');
    END IF;


    -- GET TOTAL AMOUNT

    SELECT  SUM(QTY * PRICE) INTO V_TOTAMT
    FROM LINEITEMS WHERE ORDNO = V_ORDNO;

    IF   V_TOTAMT + P_PRICE * P_QTY > 50000 THEN
         RAISE_APPLICATION_ERROR(-20003,'TOTAL AMOUNT EXCEEDED
50000');
```

```
    END IF;

    INSERT INTO LINEITEMS VALUES (V_ORDNO, P_ITEMNO, P_PRICE,P_QTY,0);

END;
```

MAKE SURE AN ORDER IS NOT CONTAINING MORE THAN 5 ITEMS.

```
CREATE OR REPLACE TRIGGER  CHECKITEMCOUNT
BEFORE INSERT
ON  LINEITEMS
FOR EACH ROW
DECLARE
    V_CNT NUMBER(5);
BEGIN

     SELECT COUNT(*) INTO V_CNT
     FROM   LINEITEMS WHERE  ORDNO = :NEW.ORDNO;

     IF  V_CNT >= 5  THEN
           RAISE_APPLICATION_ERROR(-20010,'CANNOT HAVE MORE THAN 5
ITEMS IN AN ORDER');
     END IF;
END;
```

DO NOT ALLOW ANY CHANGES TO ITEMS TABLE AFTER 9PM BEFORE 9AM.

```
CREATE OR REPLACE TRIGGER CHECKTIME
BEFORE INSERT OR DELETE OR UPDATE
ON  ITEMS
BEGIN
      IF  TO_CHAR(SYSDATE,'HH24') < 9  OR  TO_CHAR(SYSDATE,'HH24') >
21 THEN
            RAISE_APPLICATION_ERROR(-200011,'NO CHANGES CAN BE MADE
BEFORE 9 A.M AND AFTER 9 P.M');
      END IF;
END;
```

DO NOT ALLOW ANY CHANGE TO ITEM RATE IN SUCH A WAY DIFFERENCE IS MORE THAN 25% OF THE EXISTING RATE.

```
CREATE OR REPLACE TRIGGER  TRGDIFFRATE
BEFORE UPDATE
ON ITEMS
FOR EACH ROW
DECLARE
    V_DIFF NUMBER(5);
BEGIN
       V_DIFF := ABS(:NEW.RATE - :OLD.RATE);

       IF  V_DIFF > :OLD.RATE * 0.25 THEN
           RAISE_APPLICATION_ERROR(-20014,'INVALID RATE FOR AMOUNT.
CHANGE IS TOO BIG');
       END IF;

END;
```

PL/SQL Programs

INSERT A NEW ROW INTO ORDERS AND ALSO LINEITEMS WITH THE FOLLOWING DATA

- ORDERNO 1 + HIGHEST ORDER NUMBER
- ORDER DATE IS YESTERDAY
- CUSTNO IS 103
- SHIPDATE IS 15 DAYS FROM THE ORDER.
- SHIPPING ADDRESS IS SAME AS CUSTOMER ADDRESS.
- INSERT LINEITEMS INTO THIS ORDER ITEM 4 WITH LEAST RATE OF THE ORDERS. QUANTITY IS 2 DISCOUNT IS 0

```
DECLARE
   V_ORDNO    ORDERS.ORDNO%TYPE;
   V_ADDRESS1        CUSTOMERS.ADDRESS1%TYPE;
   V_ADDRESS2        CUSTOMERS.ADDRESS2%TYPE;
   V_CITY     CUSTOMERS.CITY%TYPE;
   V_STATE    CUSTOMERS.STATE%TYPE;
   V_PIN      CUSTOMERS.PIN%TYPE;
   V_PHONE    CUSTOMERS.PHONE%TYPE;
   V_PRICE    LINEITEMS.PRICE%TYPE;
```

```
BEGIN
   -- GET HIGHEST ORDER NO.
   SELECT MAX(ORDNO) INTO  V_ORDNO
   FROM ORDERS;

   -- GET CUSTOMER ADDRESS
   SELECT  ADDRESS1, ADDRESS2, CITY,STATE, PIN,PHONE INTO V_ADDRESS1,
V_ADDRESS2, V_CITY,
                        V_STATE, V_PIN, V_PHONE
   FROM  CUSTOMERS WHERE CUSTNO = 103;


   -- INSERT INTO ORDERS TABLE
   INSERT INTO ORDERS VALUES ( V_ORDNO + 1, SYSDATE - 1, SYSDATE +
14,103,
                             V_ADDRESS1, V_ADDRESS2, V_CITY, V_STATE,
V_PIN, V_PHONE);


   -- GET LEAST RATE OF ITEM 4

   SELECT  MIN(PRICE) INTO  V_PRICE
   FROM    LINEITEMS
   WHERE   ITEMNO = 4;

   -- INSERT INTO LINEITEMS TABLE
   INSERT INTO LINEITEMS  VALUES (V_ORDNO + 1, 4, 2, V_PRICE, 0);

   COMMIT;

END;
```

DISPLAY ITEMNAME AND NO. OF UNITS SOLD. IGNORE THE ITEMS FOR WHICH THERE IS NO SALES.

```
DECLARE

   -- DECLARE A CUSOR

   CURSOR  ITEMSCUR IS
      SELECT  ITEMNO, SUM(QTY)  TOTALQTY
      FROM    LINEITEMS
      GROUP   BY ITEMNO;

   V_ITEMNAME  ITEMS.ITEMNAME%TYPE;
BEGIN

    FOR REC  IN  ITEMSCUR
    LOOP

        -- GET ITEMSNAME
        SELECT  ITEMNAME INTO  V_ITEMNAME
        FROM    ITEMS
        WHERE   ITEMNO = REC.ITEMNO;

        DBMS_OUTPUT.PUT_LINE( V_ITEMNAME  || ' - '  ||
TO_CHAR(REC.TOTALQTY));
   END LOOP;

END;
```

IF THE PROGRAM IS TO BE DONE WITHOUT USING CURSOR THEN THE FOLLOWING WILL BE THE CODE FOR THE SAME REQUIREMENT

```
DECLARE

      V_MINITEMNO  ITEMS.ITEMNO%TYPE;
      V_MAXITEMNO  ITEMS.ITEMNO%TYPE;
      V_ITEMNO     ITEMS.ITEMNO%TYPE;
      V_SUMQTY     NUMBER(5);
      V_ITEMNAME   ITEMS.ITEMNAME%TYPE;


BEGIN
     -- GET MINIMUM AND MAXIMUM ITEM NUMBERS
     SELECT MIN(ITEMNO),  MAX(ITEMNO) INTO V_MINITEMNO, V_MAXITEMNO
     FROM   ITEMS;

     FOR V_ITEMNO  IN  V_MINITEMNO.. V_MAXITEMNO
     LOOP

          -- FIND OUT SUM OF QTY FOR THE ITEMNO
          SELECT  SUM(QTY) INTO  V_SUMQTY
          FROM    LINEITEMS
          WHERE   ITEMNO =  V_ITEMNO;

          IF   V_SUMQTY IS NOT NULL THEN
              -- GET ITEM NAME
              SELECT  ITEMNAME  INTO V_ITEMNAME
              FROM    ITEMS
              WHERE   ITEMNO = V_ITEMNO;

              DBMS_OUTPUT.PUT_LINE( V_ITEMNAME || ' - ' || TO_CHAR(
V_SUMQTY) );

         END IF;

     END LOOP;
END;
```

INSERT A NEW ITEMS INTO LINEITEMS INTHE FOLLOWING INFORMATION.

- ORDERNO 1003
- ITEMNO 4
- QTY 1
- RATE CURRENT RATE OF THE ITEM • DISCOUNT 5%.

```
DECLARE

    V_RATE ITEMS.RATE%TYPE;

BEGIN

    -- GET RATE OF ITEM 4
    SELECT RATE INTO  V_RATE
    FROM  ITEMS
    WHERE ITEMNO = 4;

    INSERT INTO LINEITEMS VALUES ( 1003,4,1, V_RATE,5);

    -- COMMIT;

END;
```

FOR THE PREVIOUS INSERT APPLY THE FOLLOWING CONDITION •

CHECK THE ITEM IS NOT ALREADY EXISTING IN LINEITEMS TABLE.

- CHECK THE TOTAL AMOUNT OF ORDER AS OF NOW IS NOT CROSSING THE TOTAL 30,000.(AMOUNT)
- CHECK THE ORDER IS PLACED IN THE LAST 4 DAYS.

```
DECLARE

    V_RATE ITEMS.RATE%TYPE;

BEGIN

    -- CHECK WHETHER ITEM IS ALREADY EXISTING IN LINEITEMS TABLE
    SELECT COUNT(*) INTO V_COUNT
    FROM   LINEITEMS
    WHERE ORDERNO = 1003 AND ITEMNO = 4;

    IF  V_COUNT = 1 THEN
        RAISE_APPLICATION_ERROR(-20001,'ITEM 4 IS ALREADY EXISTING FOR
ORDER 1003');
    END IF;


    --  FIND OUT AMOUNT OF ORDER

    SELECT SUM (QTY*PRICE) INTO  V_ORDERAMT
    FROM   LINEITEMS
    WHERE  ORDERNO = 1003;

    IF  V_ORDERAMT > 30000 THEN
         RAISE_APPLICATION_ERROR( -20002,'TOTAL AMOUNT HAS CROSSED
30000');
    END IF;

    -- CHECK WHETHER ORDER IS PLACED IN THE LAST FOUR DAYS

    SELECT  ORDDATE INTO V_ORDDATE
    FROM    ORDERS
    WHERE   ORDNO = 1003;
    IF   SYSDATE - ORDDATE > 4 THEN
        RAISE_APPLICATION_ERROR(-2003,'ORDER WAS NOT PLACED IN THE
LAST FOUR DAYS');
    END IF

    -- GET RATE OF ITEM 4
    SELECT RATE INTO  V_RATE
    FROM  ITEMS
    WHERE ITEMNO = 4;

    INSERT INTO LINEITEMS VALUES ( 1003,4,1, V_RATE,5);
    -- COMMIT;

END;
```

DISPLAY THE AMOUNT OF ORDER PLACED BY FIRST 5 CUSTOMERS.

```
DECLARE
    CURSOR  CUSTCUR IS
      SELECT CUSTNO,CUSTNAME FROM  CUSTOMERS;

    I NUMBER(2) := 1;

    V_AMT NUMBER(6);

BEGIN
    FOR REC IN  CUSTCUR
    LOOP

       -- GET TOTAL AMOUNT OF ORDERS PLACED BY CUSTOMER
       SELECT SUM(QTY*PRICE) INTO V_AMT
       FROM   LINEITEMS
       WHERE  ORDNO IN
            ( SELECT  ORDNO FROM ORDERS
              WHERE  CUSTNO = REC.CUSTNO);

       DBMS_OUTPUT.PUT_LINE( REC.CUSTNAME || ' - ' || V_AMT);

       I := I + 1;

       EXIT WHEN I > 5;
    END LOOP;

END;
```

CHANGE THE RATE OF EACH ITEMS ACCORDING TO THE FOLLOWING CONDITIONS.

- INCREASE THE RATE BY 10% IF THE ITEMS HAS GOT MORE THAN 5 ORDERS OR MORE THAN 25 UNITS SOLD.
- INCREASE RATE BY 5% IF ITEM WAS SOLD FOR RATE THAT IS MORE THAN THE CURRENT RATE OF THE ITEM.

```
DECLARE
    CURSOR  ITEMSCUR IS
      SELECT ITEMNO,RATE FROM ITEMS;
    V_COUNT  NUMBER(2);
    V_QTY    NUMBER(5);

BEGIN

   FOR REC IN ITEMSCUR
   LOOP

      -- FIND OUT HOW MANY ORDERS ARE THERE FOR THE ITEM
      SELECT COUNT(*) , SUM(QTY) INTO  V_COUNT, V_QTY
      FROM  LINEITEMS
      WHERE  ITEMNO = REC.ITEMNO;

      IF  V_COUNT > 5 OR V_QTY > 25 THEN

          UPDATE ITEMS SET  RATE = RATE * 1.1 WHERE ITEMNO =
REC.ITEMNO;

      ELSE

          -- CHECK WHETHER ANY ITEM WAS SOLD FOR MORE THAN THE CURRENT
RATE
          SELECT COUNT(*) INTO V_COUNT
          FROM   LINEITEMS
          WHERE ITEMNO = REC.ITEMNO AND  PRICE > REC.RATE;
          IF  V_COUNT >= 1 THEN
             UPDATE ITEMS SET RATE = RATE * 1.5 WHERE ITEMNO =
REC.ITEMNO;
          END IF;
     END IF;
  END LOOP;

END;
```

PREVENT A NEW ORDER FROM A CUSTOMER WHO'S PREVIOUS ORDER AS NOT BEING SO FAR.

```
CREATE OR REPLACE TRIGGER  CHECK_PREV_ORDER
BEFORE  INSERT
ON  ORDERS
FOR EACH ROW
DECLARE
   V_COUNT NUMBER(2);
BEGIN
     -- CHECK WHETHER ANY OTHER ORDER IS EXISTING FOR THE CUSTOMER
     SELECT  COUNT(*) INTO V_COUNT
     FROM  ORDERS
     WHERE  SHIPDATE IS NULL  AND CUSTNO = :NEW.CUSTNO;

     IF  V_COUNT >= 1 THEN
       RAISE_APPLICATION_ERROR(-20001,'CUSTOMER IS ALREADY HAVING AN
INCOMPLETE ORDER');
     END IF;
END;
```

PREVENT ANY INCREASE IN THE PRICE OF LINEITEMS.

```
CREATE OR REPLACE TRIGGER  CHECK_PRICE_INCREASE
BEFORE UPDATE
ON  LINEITEMS
FOR EACH ROW
BEGIN

     IF  :OLD.PRICE < :NEW.PRICE THEN
        RAISE_APPLICATION_ERROR(-20001,'IT IS NOT POSSIBLE TO CHANGE
PRICE OF AN ITEM IN LINEITEMS TABLE');
```

```
     END IF;
END;
```

CREATE TRIGGER TO PREVENT USERS FROM MAKING ANY CHANGES TO ORDERS TABLE BETWEEN 9PM. TO 9AM.

```
CREATE OR REPLACE TRIGGER  CHECK_UPDATE_TIME
BEFORE INSERT OR DELETE OR UPDATE
ON ORDERS
DECLARE
   CT  NUMBER(2);
BEGIN

     -- GET CURRENT TIME

     CT := TO_NUMBER( TO_CHAR(SYSDATE,'HH24') );

     IF  CT > 21  OR  CT < 9  THEN
           RAISE_APPLICATION_ERROR(-20010,'NO CHANGES CAN BE MADE
BETWEEN 9P.M AND 9A.M');
     END IF;
END;
```

CREATE A FUNCTION THAT RETURNS THE NEXT ORDER NUMBER

```
CREATE OR REPLACE FUNCTION GETNEXTORDNO  RETURN  NUMBER
IS
  V_ORDNO  ORDERS.ORDNO%TYPE;
BEGIN

  SELECT  MAX(ORDNO) + 1 INTO V_ORDNO
  FROM  ORDERS;

  RETURN   V_ORDNO;
END;
```

CREATE A FUNCTION THAT RETURNS THE FIRST MISSING ORDER NUMBER

```
CREATE OR REPLACE FUNCTION FIRSTMISSINGORDNO  RETURN  NUMBER
IS
  V_MINORDNO  ORDERS.ORDNO%TYPE;
  V_MAXORDNO  ORDERS.ORDNO%TYPE;
  V_ORDNO     ORDERS.ORDNO%TYPE;
  V_COUNT     NUMBER(2);

BEGIN

  SELECT MIN(ORDNO), MAX(ORDNO) INTO  V_MINORDNO, V_MAXORDNO
  FROM   ORDERS;

  FOR  V_ORDNO IN V_MINORDNO .. V_MAXORDNO
  LOOP

      -- FIND OUT WHETHER THERE IS ANY ORDER WITH THE CURRENT NUMBER
      SELECT COUNT(*) INTO V_COUNT
      FROM   ORDERS
      WHERE  ORDNO = V_ORDNO;

      IF  V_COUNT = 0 THEN
         RETURN V_ORDNO;
      END IF;

  END LOOP;


  RETURN  -1;   -- INDICATES THERE IS NO MISSING NUMBER

END;
```