Set Serveroutput On;

/\*

DROP TABLE Orders; --Member4--

DROP TABLE Customers; --Member5--

DROP TABLE Restaurant\_Inventory; --Member3--

DROP TABLE Menu\_Items; --Member3--

DROP TABLE Waiters; --Member2--

DROP TABLE Restaurants; --Member1--

DROP TABLE Cuisine\_Type; --Member1--

--Drop sequence for all the table

Drop SEQUENCE Cuisine\_ID\_seq;

Drop SEQUENCE Restaurant\_ID\_seq;

DROP SEQUENCE Waiter\_ID\_seq; --Member2--

DROP SEQUENCE Menu\_Items\_ID; --Member3--

DROP sequence Customer\_ID\_seq; --Member5--

DROP SEQUENCE ORDER\_ID\_SEQUENCE; --Member4--

\*/

CREATE SEQUENCE Cuisine\_ID\_seq START WITH 101; --Member1--

CREATE SEQUENCE Restaurant\_ID\_seq START WITH 1001; --Member1--

Create SEQUENCE Waiter\_ID\_seq START WITH 201 INCREMENT BY 1; --Member2--

Create SEQUENCE Menu\_Items\_ID START WITH 11 INCREMENT BY 1 NOCACHE; --Member3--

create sequence Customer\_ID\_seq start with 100 increment by 1 nocache; --Member5--

CREATE SEQUENCE ORDER\_ID\_SEQUENCE INCREMENT BY 1 START WITH 1000; --Member4

--Member1--

--Create all the tables--

create table Cuisine\_Type(

Cuisine\_ID number,

Cuisine\_TName varchar(20),

Primary Key(Cuisine\_ID)

);

create table Restaurants(

Restaurant\_ID number,

R\_Name varchar(20),

Rstreet\_Address varchar(20),

R\_City varchar(20),

R\_State varchar(20),

R\_Zip number,

Cuisine\_ID number,

Primary Key(Restaurant\_ID),

foreign key(Cuisine\_ID) references Cuisine\_Type(Cuisine\_ID)

);

--Member2--

Create table Waiters(Waiter\_ID int,

Waiter\_Name varchar(30) not null,

Restaurant\_ID number not null,

PRIMARY Key(Waiter\_ID),

FOREIGN KEY(Restaurant\_ID) references Restaurants(Restaurant\_ID)

);

--Member3--

create table MENU\_ITEMS (

Menu\_Items\_Id INT NOT NULL,

Cuisine\_type\_Id Int NOT NULL,

Menu\_Item\_Name Varchar (100) Not Null,

Price NUMBER Not Null,

Primary KEY (Menu\_Items\_Id),

Foreign Key (Cuisine\_type\_Id) references CUISINE\_TYPE

);

create table RESTAURANT\_INVENTORY (

Menu\_Items\_Id INT NOT NULL,

Restaurant\_ID INT NOT NULL,

Menu\_Item\_Name Varchar (100) Not Null,

Quantity Number Not Null,

Foreign Key (Menu\_Items\_Id) references MENU\_ITEMS,

Foreign Key (Restaurant\_ID) references RESTAURANTS

);

--Member5--

create table Customers

(

Customer\_ID int Primary Key,

Customer\_Name Varchar(30) NOT NULL,

Email varchar(20) NOT NULL,

Street\_Address varchar(20) NOT NULL,

City varchar(20) NOT NULL,

State Varchar(20) NOT NULL,

Zip int NOT NULL,

Credit\_Card char(20) NOT NULL UNIQUE

);

--Member4--

CREATE TABLE Orders(

Order\_ID INT NOT NULL,

Restaurant\_ID INT,

Customer\_ID INT,

Order\_Date DATE,

Menu\_Items\_ID INT,

Waiter\_ID INT,

Amount FLOAT,

Tip FLOAT,

PRIMARY KEY(Order\_ID),

FOREIGN KEY(Restaurant\_ID) REFERENCES Restaurants(Restaurant\_ID),

FOREIGN KEY(Customer\_ID) REFERENCES Customers(Customer\_ID),

FOREIGN KEY(Menu\_Items\_ID) REFERENCES Menu\_Items(Menu\_Items\_ID),

FOREIGN KEY(Waiter\_ID) REFERENCES Waiters(Waiter\_ID)

);

--Member 1 function creation

/

drop function FIND\_CUISINE\_TYPE\_ID;

create or replace function FIND\_CUISINE\_TYPE\_ID(

C\_TName IN Cuisine\_Type.Cuisine\_TName%type)

return number

IS

x varchar(30);

y number;

Begin

select Cuisine\_ID into y from CUISINE\_TYPE where Cuisine\_TName = C\_TName;

return y;

end;

/

drop function FIND\_RESTAURANT\_ID;

create or replace function FIND\_RESTAURANT\_ID(

Restaurant\_Name IN Restaurants.R\_Name%type)

return number

IS

a varchar(30);

b number;

Begin

select Restaurant\_ID into b from Restaurants where R\_Name = Restaurant\_Name;

return b;

end;

/

--Procedure to insert values in Cuisine\_Type table

Create or replace procedure add\_CuisineType(

C\_ID IN Cuisine\_Type.Cuisine\_ID%type,

C\_TName IN Cuisine\_Type.Cuisine\_TName%type)

IS

Begin

Insert into Cuisine\_Type(Cuisine\_ID, Cuisine\_TName)

Values(C\_ID, C\_TName);

EXCEPTION

WHEN OTHERS THEN

dbms\_output.put\_line('Error adding desired Cuisine Type.');

dbms\_output.put\_line ('An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

End;

/

--Procedure to insert values in Restaurants table

create or replace procedure add\_Restaurants(

R\_ID IN Restaurants.Restaurant\_ID%type,

RName IN Restaurants.R\_Name%type,

RStreet IN Restaurants.Rstreet\_Address%type,

RCity IN Restaurants.R\_City%type,

RState IN Restaurants.R\_State%type,

RZip IN Restaurants.R\_Zip%type ,

CR\_ID IN Restaurants.Cuisine\_ID%type)

IS

Begin

Insert into Restaurants(Restaurant\_ID, R\_Name, Rstreet\_Address, R\_City, R\_State, R\_Zip, Cuisine\_ID)

Values(R\_ID, RName, RStreet, RCity, RState, RZip, CR\_ID);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.Put\_line('Error adding desired Restaurants.');

DBMS\_OUTPUT.Put\_line ('An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

END;

/

Begin

dbms\_output.put\_line('Member 1 adding cuisine type names: American, Indian, Italian, BBQ, Ethiopian.');

add\_CuisineType(Cuisine\_ID\_seq.nextval,'American');

add\_CuisineType(Cuisine\_ID\_seq.nextval,'Italian');

add\_CuisineType(Cuisine\_ID\_seq.nextval,'BBQ');

add\_CuisineType(Cuisine\_ID\_seq.nextval,'Indian');

add\_CuisineType(Cuisine\_ID\_seq.nextval,'Ethiopian');

End;

/

-----------------Adding records in Restaurants Table----------------

Begin

dbms\_output.put\_line('Member 1 adding Restaurants names with all data');

add\_Restaurants(Restaurant\_ID\_seq.nextval,'Ribs\_R\_US', '1313 Mockingbird', 'CT', 'MD', 21250, 101 );

add\_Restaurants(Restaurant\_ID\_seq.nextval,'Bella\_Italia', 'Paterson Str', 'AT', 'MD', 21043, 102 );

add\_Restaurants(Restaurant\_ID\_seq.nextval,'Roma', 'Queens', 'AT', 'MD', 21043, 102 );

add\_Restaurants(Restaurant\_ID\_seq.nextval,'Bull Roast', '14 Brooklyn', 'NYC', 'NY', 10013, 103 );

add\_Restaurants(Restaurant\_ID\_seq.nextval,'Taj Mahal', '21 strt Manhattan', 'NYC', 'NY', 10013, 104 );

add\_Restaurants(Restaurant\_ID\_seq.nextval,'Selasie', 'Jersey strt', 'philadelphia', 'PA', 16822, 105 );

add\_Restaurants(Restaurant\_ID\_seq.nextval,'Ethiop', 'Jersey strt', 'philadelphia', 'PA', 16822, 105 );

End;

---------MEMBER 2 Procedure to insert values in Waiters table

/

Begin

DBMS\_OUTPUT.Put\_line('Member 2 creating procedure to insert data to Waiters table');

end;

/

Create or replace procedure

hire\_waiters(W\_name IN Waiters.Waiter\_Name%type,

Restaurant\_name IN Restaurants.R\_Name%type)

IS

Restaurant\_ID number;

Begin

Restaurant\_ID:=FIND\_RESTAURANT\_ID(Restaurant\_name);

Insert into Waiters(Waiter\_ID,Waiter\_Name,Restaurant\_ID)

Values(Waiter\_ID\_seq.nextval,W\_name,Restaurant\_ID);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.Put\_line('Error hiring desired waiter');

DBMS\_OUTPUT.Put\_line ('An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

End;

/

-----------------MEMBER 2 Adding records in waiters Table----------------

Begin

DBMS\_OUTPUT.Put\_line('Member 2 inserting data into Waiters table');

end;

/

Exec hire\_waiters('Jack','Ribs\_R\_US');

Exec hire\_waiters('Jill','Ribs\_R\_US');

Exec hire\_waiters('Wendy','Ribs\_R\_US');

Exec hire\_waiters('Hailey','Ribs\_R\_US');

Exec hire\_waiters('Mary','Bella\_Italia');

Exec hire\_waiters('Pat','Bella\_Italia');

Exec hire\_waiters('Michael','Bella\_Italia');

Exec hire\_waiters('Rakesh','Bella\_Italia');

Exec hire\_waiters('Verma','Bella\_Italia');

Exec hire\_waiters('Mike','Roma');

Exec hire\_waiters('Judy','Roma');

Exec hire\_waiters('Trevor','Selasie');

Exec hire\_waiters('Trudy','Ethiop');

Exec hire\_waiters('Trisha','Ethiop');

Exec hire\_waiters('Tariq','Ethiop');

Exec hire\_waiters('Chap','Taj Mahal');

Exec hire\_waiters('Hannah','Bull Roast');

----------Member 2 creating function FIND\_WAITER\_ID

/

Begin

DBMS\_OUTPUT.Put\_line('Member 2 creating function FIND\_WAITER\_ID');

end;

/

DROP function FIND\_WAITER\_ID;

Create or replace function FIND\_WAITER\_ID(

W\_Name IN Waiters.Waiter\_Name%type)

return number

IS

b number;

Begin

select WAITER\_ID into b from Waiters where Waiter\_Name = W\_Name;

return b;

exception

when no\_data\_found then

dbms\_output.put\_line('no waiter found');

return -1;

End;

--Member3--

-- Creating funtion to find the Menu\_Item\_ID

/

drop function FIND\_MENU\_ITEM\_ID;

/

Create or replace function FIND\_MENU\_ITEM\_ID (Menu\_Items\_Name in Varchar)

return Number

Is

menu\_item\_id Number;

Begin

select Menu\_Items\_Id into menu\_item\_id from Menu\_ITEMS where Menu\_Items\_Name=Menu\_Item\_Name;

return menu\_item\_id;

-- If no Items is found the exception block is executed

exception

when no\_data\_found then

dbms\_output.put\_line('no items found');

return -1;

End;

/

-- Procedure to insert the Menu data in Menu\_Items table

create or replace procedure Menu (C\_TNAME IN Varchar, Menu\_Item\_Name In Varchar, Price IN Number)

IS

Cuisine\_ID INT;

-- Calling a function to get a ID of Cuisine Type

Begin

Cuisine\_ID:=FIND\_CUISINE\_TYPE\_ID(C\_TNAME);

Insert INTO MENU\_ITEMS(Menu\_Items\_Id,Cuisine\_type\_Id,Menu\_Item\_Name,Price) Values

(Menu\_Items\_ID.nextval,Cuisine\_ID,Menu\_Item\_Name,Price);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.Put\_line('Error adding desired Menu.');

End;

/

-- Inserting the data into Menu\_Items table

exec Menu('American','Burger',10);

exec Menu('American','Fries',5);

exec Menu('American','Pasta',15);

exec Menu('American','Salad',10);

exec Menu('American','Salmon',20);

exec Menu('Italian','Lasanga',15);

exec Menu('Italian','Meatballs',10);

exec Menu('Italian','Spaghetti',15);

exec Menu('Italian','Pizza',20);

exec Menu('BBQ','Steak',25);

exec Menu('BBQ','burger',10);

exec Menu('BBQ','pork loin',15);

exec Menu('BBQ','fillet mignon',30);

exec Menu('Indian','dal soup',10);

exec Menu('Indian','rice',5);

exec Menu('Indian','tandoori chicken',10);

exec Menu('Indian','samosa',8);

exec Menu('Ethiopian','meat chunks',12);

exec Menu('Ethiopian','legume stew',10);

exec Menu('Ethiopian','flatbread',3);

/

-- Procedure to insert the inventory data in restaurant\_inventory table

create or replace procedure Restaurant\_Invent (R\_NAME IN Varchar, MENU\_ITEMS\_NAME IN VARCHAR, Menu\_Item\_Name In Varchar, Quantity In Number)

IS

RESTAURANT\_ID INT;

Menu\_Itemm\_Id INT;

-- Calling a function to get a Restaurant ID and Menu Item ID

Begin

RESTAURANT\_ID:=FIND\_RESTAURANT\_ID(R\_NAME);

Menu\_Itemm\_Id:=FIND\_MENU\_ITEM\_ID(Menu\_Items\_Name);

Insert INTO RESTAURANT\_INVENTORY(Menu\_Items\_Id,RESTAURANT\_ID,Menu\_Item\_Name,QUANTITY) Values

(Menu\_Itemm\_ID,RESTAURANT\_ID,Menu\_Item\_Name,Quantity);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.Put\_line('Error adding desired Restaurants menu.');

End;

/

exec Restaurant\_Invent('Ribs\_R\_US','Burger','Burger',50);

exec Restaurant\_Invent('Ribs\_R\_US','Fries','Fries',150);

exec Restaurant\_Invent('Bella\_Italia','Lasanga','lasanga',10);

exec Restaurant\_Invent('Bull Roast','Steak','Steak',15);

exec Restaurant\_Invent('Bull Roast','pork loin','pork loin',50);

exec Restaurant\_Invent('Bull Roast','fillet mignon','fillet mignon',5);

exec Restaurant\_Invent('Taj Mahal','dal soup','dal soup',50);

exec Restaurant\_Invent('Taj Mahal','rice','rice',500);

exec Restaurant\_Invent('Taj Mahal','samosa','samosa',150);

exec Restaurant\_Invent('Selasie','meat chunks','meat chunks',150);

exec Restaurant\_Invent('Selasie','legume stew','legume stew',150);

exec Restaurant\_Invent('Selasie','flatbread','flatbread',500);

exec Restaurant\_Invent('Ethiop','meat chunks','meat chunks',150);

exec Restaurant\_Invent('Ethiop','legume stew','legume stew',150);

exec Restaurant\_Invent('Ethiop','flatbread','flatbread',500);

/

-- Creating Cursor to show the initial inventory from Ethiop restaurant before update

Declare

cursor c1 is select Menu\_ITEM\_NAME, Quantity from restaurant\_inventory where restaurant\_id in (select restaurant\_ID from restaurants where r\_name='Ethiop');

MENU\_NAME varchar(50);

Quantity\_Number int;

NoDataFound Exception;

Begin

DBMS\_OUTPUT.Put\_line('--------------- Initial Inventory for Ethiop restaurant from Member-3 -------------------');

Open c1;

loop

fetch c1 into Menu\_Name,Quantity\_Number;

exit when c1%notfound;

dbms\_output.put\_line('Menu\_Name: ' || Menu\_Name || ' Quantity' || Quantity\_Number);

End loop;

if c1%rowcount=0 then

raise NoDataFound;

end if;

EXCEPTION

WHEN NoDataFound THEN

DBMS\_OUTPUT.Put\_line (' No Data Found with respective restaurant');

End;

/

-- Creating procedure to update the inventory for restaurant

create or replace procedure Update\_Invent (R\_NAME IN Varchar, MENU\_ITEMS\_NAME IN VARCHAR, Num\_Quantity In Number)

IS

REST\_ID INT;

Menu\_Id INT;

Numm\_Quantity INT;

-- Calling a function to get a Restauran ID and Menu Item ID

Begin

REST\_ID:=FIND\_RESTAURANT\_ID(R\_NAME);

Menu\_Id:=FIND\_MENU\_ITEM\_ID(Menu\_Items\_Name);

Update RESTAURANT\_INVENTORY set quantity = quantity-Num\_Quantity where menu\_items\_id = menu\_id and restaurant\_id=rest\_ID;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.Put\_line('Error in updating the restaurant\_inventory');

End;

/

exec Update\_Invent('Taj Mahal','rice',25);

exec Update\_Invent('Selasie','meat chunks',50);

exec Update\_Invent('Bull Roast','fillet mignon',2);

exec Update\_Invent('Bull Roast','fillet mignon',2);

exec Update\_Invent('Ethiop','meat chunks',30);

exec Update\_Invent('Ethiop','meat chunks',30);

exec Update\_Invent('Ethiop','legume stew',20);

-- Creating Cursor to show the final inventory from Ethiop restaurant after update

/

Declare

cursor c1 is select Menu\_ITEM\_NAME, Quantity from restaurant\_inventory where restaurant\_id in (select restaurant\_ID from restaurants where r\_name='Ethiop');

MENU\_NAME varchar(50);

Quantity\_Number int;

NoDataFound Exception;

Begin

DBMS\_OUTPUT.Put\_line('--------------- Final Inventory for Ethiop restaurant from Member-3 -------------------');

Open c1;

loop

fetch c1 into Menu\_Name,Quantity\_Number;

exit when c1%notfound;

dbms\_output.put\_line('Menu\_Name: ' || Menu\_Name || ' Quantity' || Quantity\_Number);

End loop;

if c1%rowcount=0 then

raise NoDataFound;

end if;

EXCEPTION

WHEN NoDataFound THEN

DBMS\_OUTPUT.Put\_line (' No Data Found with respective restaurant');

End;

/

--Member5--

-- Procedure to insert data in Customers table by Member-5

/

create or replace procedure insertCust

(

c\_custid IN customers.Customer\_ID%type,

c\_custname IN customers.Customer\_Name%type,

c\_custmail IN customers.Email%type,

c\_custadd IN customers.Street\_Address%type,

c\_custcity IN customers.City%type,

c\_custstate IN customers.State%type,

c\_custzip IN customers.Zip%type,

c\_custcc IN customers.Credit\_Card%type

)

IS

Begin

insert into customers (Customer\_ID,Customer\_Name,Email,Street\_Address,City,State,Zip,Credit\_Card)

Values

(

c\_custid,

c\_custname,

c\_custmail,

c\_custadd,

c\_custcity,

c\_custstate,

c\_custzip,

c\_custcc

);

commit;

end;

/

Begin

insertCust(Customer\_ID\_seq.nextval,'cust1','cust1@gmail.com','Belwood Green','Baltimore','Maryland',21045,858585858585);

insertCust(Customer\_ID\_seq.nextval,'cust11','cust11@gmail.com','Courtney Road','Baltimore','Maryland',21045,858585858586);

insertCust(Customer\_ID\_seq.nextval,'cust3','cust3@gmail.com','Howland Square','Baltimore','Maryland',21046,858585858587);

insertCust(Customer\_ID\_seq.nextval,'cust111','cust111@gmail.com','Avenue Road','Baltimore','Maryland',21045,858585858588);

insertCust(Customer\_ID\_seq.nextval,'custNY1','custNY1@gmail.com','Liberty St','New York City','New York',10045,858585858591);

insertCust(Customer\_ID\_seq.nextval,'custNY2','custNY2@gmail.com','Maiden Ln','New York City','New York',10045,858585858592);

insertCust(Customer\_ID\_seq.nextval,'custNY3','custNY3@gmail.com','Pine St','New York City','New York',10045,858585858593);

insertCust(Customer\_ID\_seq.nextval,'custPA1','custPA1@gmail.com','Haagen Ln','Beech Creek','Pennsylvania',16822,858585858594);

insertCust(Customer\_ID\_seq.nextval,'custPA2','custPA2@gmail.com','Telephone Ln','Beech Creek','Pennsylvania',16822,858585858595);

insertCust(Customer\_ID\_seq.nextval,'custPA3','custPA3@gmail.com','Eagle Valley Rd','Beech Creek','Pennsylvania',16822,858585858596);

insertCust(Customer\_ID\_seq.nextval,'custPA4','custPA4@gmail.com','Smith Rd','Beech Creek','Pennsylvania',16822,858585858597);

insertCust(Customer\_ID\_seq.nextval,'custPA5','custPA5@gmail.com','Sand Spring Rd','Beech Creek','Pennsylvania',16822,858585858598);

insertCust(Customer\_ID\_seq.nextval,'custPA6','custPA6@gmail.com','Tank Ln','Beech Creek','Pennsylvania',16822,858585858599);

end;

/

--Function to find Customer ID by Member-5

create or replace function FIND\_CUSTOMER\_ID(Custname varchar2)

return int

IS

x int;

begin

select Customer\_ID into x from customers where Customer\_Name=Custname;

dbms\_output.put\_line(x);

return x;

exception

when no\_data\_found then

dbms\_output.put\_line('Invalid Customer name');

when others then

DBMS\_output.put\_line('Too many rows');

end;

/

--Member4--

/

CREATE OR REPLACE PROCEDURE PROC\_PLACE\_ORDER(RESTAURANT\_NAME VARCHAR2, CUSTOMER\_NAME VARCHAR2, ORDER\_DATE DATE, MENU\_ITEM\_NAME VARCHAR2, WAITER\_NAME VARCHAR2, AMOUNT NUMBER)

AS

RESTAURANT\_ID NUMBER;

CUSTOMER\_ID NUMBER;

MENU\_ITEMS\_ID NUMBER;

WAITER\_ID NUMBER;

TIP NUMBER;

BEGIN

RESTAURANT\_ID:=FIND\_RESTAURANT\_ID(RESTAURANT\_NAME);

CUSTOMER\_ID:=FIND\_CUSTOMER\_ID(CUSTOMER\_NAME);

--MENU\_ITEMS\_ID:=FIND\_MENU\_ITEM\_ID(MENU\_ITEM\_NAME);

Menu\_Items\_Id:=FIND\_MENU\_ITEM\_ID(Menu\_Item\_Name);

WAITER\_ID:=FIND\_WAITER\_ID(WAITER\_NAME);

TIP:=AMOUNT\*0.2;

INSERT INTO ORDERS(ORDER\_ID, RESTAURANT\_ID, CUSTOMER\_ID, ORDER\_DATE, MENU\_ITEMS\_ID, WAITER\_ID, AMOUNT, TIP)

VALUES(ORDER\_ID\_SEQUENCE.NEXTVAL, RESTAURANT\_ID, CUSTOMER\_ID, ORDER\_DATE, MENU\_ITEMS\_ID, WAITER\_ID, AMOUNT, TIP);

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Encountered error ....try again');

RAISE\_APPLICATION\_ERROR(-20001,'AN ERROR WAS ENCOUNTERED WHEN PLACING ORDER- '||SQLCODE||' -ERROR- '||SQLERRM);

END PROC\_PLACE\_ORDER;

/

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

-----PROCEDURE : PROC\_LIST\_ORDER

-----This procedure retrieves all the orders based on given filter condition

/

CREATE OR REPLACE PROCEDURE PROC\_LIST\_ORDERS(RESTAURANT\_NAME VARCHAR2, ORDER\_DATE DATE)

AS

RESTAURANT\_ID NUMBER;

BEGIN

RESTAURANT\_ID:=FIND\_RESTAURANT\_ID(RESTAURANT\_NAME);

DBMS\_OUTPUT.PUT\_LINE('The Orders for Restaurant : '||RESTAURANT\_NAME||' On '||TO\_CHAR(ORDER\_DATE,'DD-MON-YYYY')||':');

DBMS\_OUTPUT.PUT\_LINE('ORDER\_ID,RESTAURANT\_ID,CUSTOMER\_ID,ORDER\_DATE,MENU\_ITEMS\_ID,WAITER\_ID,AMOUNT,TIP');

FOR RECORD IN (SELECT ORDER\_ID, RESTAURANT\_ID, CUSTOMER\_ID, ORDER\_DATE, MENU\_ITEMS\_ID, WAITER\_ID, AMOUNT, TIP

FROM ORDERS

WHERE RESTAURANT\_ID=RESTAURANT\_ID AND ORDER\_DATE=ORDER\_DATE)

LOOP

DBMS\_OUTPUT.PUT\_LINE(RECORD.ORDER\_ID||','||RECORD.RESTAURANT\_ID||','||RECORD.CUSTOMER\_ID||','||RECORD.ORDER\_DATE||','||RECORD.MENU\_ITEMS\_ID||','||RECORD.WAITER\_ID||','||RECORD.AMOUNT||','||RECORD.TIP);

END LOOP;

EXCEPTION

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20001,'AN ERROR ENCOUNTERED- '||SQLCODE||' -ERROR- '||SQLERRM);

END PROC\_LIST\_ORDERS;

/

/

exec PROC\_PLACE\_ORDER('Bella\_Italia', 'cust1', TO\_DATE('10-MAR-2022','DD-MON-YYYY'), 'Pizza', 'Mary', 20);

exec PROC\_PLACE\_ORDER('Bella\_Italia', 'cust11', TO\_DATE('15-MAR-2022','DD-MON-YYYY'), 'Spaghetti', 'Mary', 30);

exec PROC\_PLACE\_ORDER('Bella\_Italia', 'cust11', TO\_DATE('15-MAR-2022','DD-MON-YYYY'), 'Pizza', 'Mary', 20);

-----PLSQL BLOCK TO PLACE ORDER

/

BEGIN

PROC\_PLACE\_ORDER('Bull Roast', 'custNY1', TO\_DATE('01-APR-2022','DD-MON-YYYY'), 'fillet mignon', 'Hannah', 60);

END;

/

-----PLSQL BLOCK TO PLACE ORDER

/

BEGIN

PROC\_PLACE\_ORDER('Bull Roast', 'custNY1', TO\_DATE('2-APR-2022','DD-MON-YYYY'), 'fillet mignon', 'Hannah', 60);

END;

/

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

-----PLSQL BLOCK TO PLACE ORDER

/

BEGIN

PROC\_PLACE\_ORDER('Bull Roast', 'custNY2', TO\_DATE('1-APR-2022','DD-MON-YYYY'), 'pork loin', 'Hannah', 15);

END;

/

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

-----PLSQL BLOCK TO PLACE ORDER

/

BEGIN

PROC\_PLACE\_ORDER('Ethiop', 'custPA1', TO\_DATE('1-APR-2022','DD-MON-YYYY'), 'meat chunks', 'Trisha', 120);

END;

/

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

-----PLSQL BLOCK TO PLACE ORDER

/

BEGIN

PROC\_PLACE\_ORDER('Ethiop', 'custPA1', TO\_DATE('1-MAY-2022','DD-MON-YYYY'), 'meat chunks', 'Trisha', 120);

END;

/

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

-----PLSQL BLOCK TO PLACE ORDER

/

BEGIN

PROC\_PLACE\_ORDER('Ethiop', 'custPA1', TO\_DATE('10-MAY-2022','DD-MON-YYYY'), 'meat chunks', 'Trisha', 120);

END;

/

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

-----PLSQL BLOCK TO PLACE ORDER

/

BEGIN

PROC\_PLACE\_ORDER('Ethiop', 'custPA2', TO\_DATE('1-MAY-2022','DD-MON-YYYY'), 'legume stew', 'Trevor', 100);

END;

/

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

-----PLSQL BLOCK TO PLACE ORDER

/

BEGIN

PROC\_PLACE\_ORDER('Ethiop', 'custPA2', TO\_DATE('11-MAY-2022','DD-MON-YYYY'), 'legume stew', 'Trevor', 100);

END;

/

----Member 1: Display restaurant by cuisine--------

create or replace procedure get\_restaurant(cui\_tn varchar) As

restaurant\_name restaurants.R\_Name%type; ----variables to store the values fetched in cursor

restaurant\_address restaurants.RSTREET\_ADDRESS%type;

cursor c1 is select distinct R\_Name, RSTREET\_ADDRESS from restaurants,cuisine\_type where restaurants.cuisine\_id = cuisine\_type.cuisine\_id

and cuisine\_tname = cui\_tn; --Cursor Definition

begin

dbms\_output.put\_line('displaying the restaurants name and address');

open c1; ----Open Cursor

loop

fetch c1 into restaurant\_name, restaurant\_address;

exit when c1% NOTFOUND; --Cursor Exception

dbms\_output.put\_line('Restaurant Name:' || restaurant\_name);

dbms\_output.put\_line('Restaurant Address:' || restaurant\_address);

end loop; ---ending loop

close c1; ---Close Cursor

end;

/

exec get\_restaurant('Italian');

--exec get\_restaurant('BBQ');

--exec get\_restaurant('American');

--exec get\_restaurant('Indian');

exec get\_restaurant('Ethiopian');

----------Report By Member 1:- Report Income by state -----

/

create or replace procedure report\_restaurant\_amount IS

income orders.amount%type;

r\_state1 restaurants.r\_state%type;

c\_t1 cuisine\_type.cuisine\_tname%type;

r\_name1 restaurants.r\_name%type;

cursor c1 is select r\_state, sum(amount), cuisine\_tname, r\_name from restaurants, orders, cuisine\_type where

restaurants.restaurant\_id = orders.restaurant\_id and restaurants.cuisine\_id = cuisine\_type.cuisine\_id

group by r\_state, cuisine\_tname, r\_name;

begin

dbms\_output.put\_line('========================================================================

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

---- R E P O R T S below ----

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

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

');

dbms\_output.put\_line('-------------------------- REPORT BY MEMBER 1 ---------------');

open c1;

loop

fetch c1 into r\_state1,income,c\_t1,r\_name1;

exit when c1%NOTFOUND;

dbms\_output.put\_line('Income is :' || income || ' Restaurant state is:'|| r\_state1 ||' cuisine type is '|| c\_t1 ||' Restaurant name is '||r\_name1);

end loop;

end;

/

exec report\_restaurant\_amount;

/

--------Member 2 Procedure to show list of Waiters information at a given restaurant

/

Create or Replace Procedure List\_Waiters(Restaurant\_name IN Restaurants.R\_Name%type)

IS

v\_Restaurant\_ID number;

Begin

v\_Restaurant\_ID:=FIND\_RESTAURANT\_ID(Restaurant\_name);

dbms\_output.put\_line('Information about waiters working at the ' ||Restaurant\_name|| ' restaurant with '||' Restaurant\_ID:'||v\_Restaurant\_ID);

for i IN (select Waiter\_ID,Waiter\_Name from Waiters where Restaurant\_ID=v\_Restaurant\_ID)

loop

dbms\_output.put\_line('Restaurant\_ID:'||v\_Restaurant\_ID ||' with Waiter\_ID: '||i.Waiter\_ID|| ' and Waiter\_Name: '|| i.Waiter\_Name);

End loop;

Exception

when no\_data\_found then

dbms\_output.put\_line('no data found');

End;

/

-- Exec list\_waiters('Bella\_Italia');

-- Exec list\_waiters('Roma');

/

BEGIN

dbms\_output.put\_line('-------------------------- REPORT BY MEMBER 2 ---------------');

END;

/

Exec list\_waiters('Ethiop');

/

------------------MEMBER 2 Procedure to Show total tips by each waiter

Create or Replace Procedure Report\_tips

IS

cursor c1 is select Waiters.Waiter\_ID,Waiters.Waiter\_Name,sum(orders.tip) as Totaltip from Waiters, orders where orders.waiter\_id=Waiters.waiter\_id group by Waiters.Waiter\_ID,Waiters.Waiter\_Name;

Begin

dbms\_output.put\_line('Report showing total tips by each waiter:');

for t in c1

loop

dbms\_output.put\_line('Waiter ID: '||t.Waiter\_ID ||' with Waiter Name: '||t.Waiter\_Name|| ' earns a Total tip: '|| t.Totaltip);

exit when c1%notfound;

End loop;

Exception

when no\_data\_found then

dbms\_output.put\_line('no data found');

End;

-------------MEMBER 2 calling the procedure report tips

/

Exec Report\_tips;

/

----------- MEMBER 2 procedure to Show total tips earned by waiters per state

Create or Replace Procedure Report\_tips\_bystate

IS

cursor c1 is select distinct restaurants.r\_state,sum(orders.tip) as tipbystate from restaurants,orders where

orders.restaurant\_id=restaurants.restaurant\_id group by restaurants.r\_state;

Begin

dbms\_output.put\_line('Report showing total tips earned by waiters by state:');

for x in c1

loop

dbms\_output.put\_line(' state: '||x.r\_state ||'; Total tips earned by waiters in this state: '|| x.tipbystate);

exit when c1%notfound;

End loop;

Exception

when no\_data\_found then

dbms\_output.put\_line('no data found');

End;

----------MEMBER 2 calling procedure report tips bystate

/

Exec Report\_tips\_bystate;

--Member 3--

/

-- Creating a procedure to generate a report from menu table -Member-3

create or replace procedure report

IS

cursor c1 is select distinct M.Menu\_item\_name,sum(R.quantity), C.cuisine\_tname from Menu\_Items M,CUISINE\_TYPE C,Restaurant\_Inventory R where C.Cuisine\_ID=M.Cuisine\_Type\_ID AND

M.Menu\_Items\_ID=R.Menu\_Items\_ID GROUP BY C.CUISINE\_TNAME,M.Menu\_Item\_name;

R\_quantity int;

M\_menu varchar (50);

C\_Tname varchar (50);

NoDataFound Exception;

Begin

dbms\_output.put\_line('-------------------------- REPORT BY MEMBER 3 ---------------');

dbms\_output.put\_line('Report for Menu is:');

Open c1;

loop

fetch c1 into M\_menu,R\_quantity,C\_Tname;

exit when c1%notfound;

dbms\_output.put\_line('Menu\_Name: ' || M\_menu || ', belong to Cusisine\_Type: ' || C\_Tname || ', have quantity: ' || R\_quantity );

End loop;

if c1%rowcount=0 then

raise NoDataFound;

end if;

EXCEPTION

WHEN NoDataFound THEN

DBMS\_OUTPUT.Put\_line (' Report for the menu item is not exists' || '');

End;

/

exec Report;

/

--dbms\_output.put\_line('--------------------------------REPORTS BY MEMBER 4----------------------------------');

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

-----PROCEDURE : PROC\_TOP\_ORDERED\_ITEMS

-----This procedure lists the most popular menu item ordered for each cuisine type

/

CREATE OR REPLACE PROCEDURE PROC\_TOP\_ORDERED\_ITEMS

AS

BEGIN

dbms\_output.put\_line('-------------------------- REPORT BY MEMBER 4 ---------------');

FOR RECORD IN (SELECT CUISINE\_TNAME,MENU\_ITEM\_NAME

FROM (

SELECT CUISINE\_TNAME,MENU\_ITEM\_NAME,TOTAL\_ITEMS,DENSE\_RANK() OVER (PARTITION BY CUISINE\_TNAME ORDER BY TOTAL\_ITEMS DESC) AS TOTAL\_RANK

FROM (SELECT CT.CUISINE\_TNAME,MI.MENU\_ITEM\_NAME,COUNT(OS.ORDER\_ID) AS TOTAL\_ITEMS

FROM ORDERS OS,MENU\_ITEMS MI,CUISINE\_TYPE CT

WHERE MI.CUISINE\_TYPE\_ID=CT.CUISINE\_ID AND OS.MENU\_ITEMS\_ID=MI.MENU\_ITEMS\_ID

GROUP BY (CT.CUISINE\_TNAME,MI.MENU\_ITEM\_NAME)))

WHERE TOTAL\_RANK=1)

LOOP

DBMS\_OUTPUT.PUT\_LINE(RECORD.CUISINE\_TNAME||','||RECORD.MENU\_ITEM\_NAME);

END LOOP;

EXCEPTION

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20001,'AN ERROR ENCOUNTERED WHILE FETCHING ITEMS- '||SQLCODE||' -ERROR- '||SQLERRM);

END PROC\_TOP\_ORDERED\_ITEMS;

/

---Calling a Procedure

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

-----PLSQL BLOCK TO CALL THE PROCEDURE PROC\_TOP\_ORDERED\_ITEMS

/

BEGIN

PROC\_TOP\_ORDERED\_ITEMS();

END;

/

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

-----PROCEDURE : PROC\_TOP\_THREE\_RESTAURANTS

/

CREATE OR REPLACE PROCEDURE PROC\_TOP\_THREE\_RESTAURANTS

AS

BEGIN

DBMS\_OUTPUT.PUT\_LINE('DISPLAYING TOP THREE RESTAURENTS ON EACH STATE ');

DBMS\_OUTPUT.PUT\_LINE('R\_STATE,R\_NAME,TOTAL\_REVENUE');

FOR RECORD IN (SELECT R\_STATE,R\_NAME,TOTAL\_REVENUE

FROM (

SELECT R\_STATE,R\_NAME,TOTAL\_REVENUE, DENSE\_RANK() OVER

(PARTITION BY R\_STATE ORDER BY TOTAL\_REVENUE DESC) AS RANKING

FROM (

SELECT RS.R\_STATE,RS.R\_NAME,SUM(OS.AMOUNT) AS TOTAL\_REVENUE

FROM RESTAURANTS RS,ORDERS OS

WHERE OS.RESTAURANT\_ID=RS.RESTAURANT\_ID

GROUP BY (RS.R\_STATE,RS.R\_NAME)))

WHERE RANKING<=3

ORDER BY R\_STATE,TOTAL\_REVENUE DESC)

LOOP

DBMS\_OUTPUT.PUT\_LINE(RECORD.R\_STATE||','||RECORD.R\_NAME||','||RECORD.TOTAL\_REVENUE);

END LOOP;

EXCEPTION

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20001,'ERROR ENCOUNTERED WHILE FETCHING REVENUE');

END PROC\_TOP\_THREE\_RESTAURANTS;

/

---Calling a Procedure

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

-----PLSQL BLOCK TO CALL THE PROCEDURE

/

BEGIN

PROC\_TOP\_THREE\_RESTAURANTS();

END;

/

----member 5------

-- Procedure to retrieve the generous customers who spent least money------

/

create or replace procedure Custdesc

IS

x varchar(20);

y number;

cursor c1 is select customer\_name, sum(amount) as Amount from customers c, orders o where c.customer\_ID=o.customer\_ID group by customer\_name order by amount desc;

begin

dbms\_output.put\_line('-------------------------- REPORT BY MEMBER 5 ---------------');

open c1;

dbms\_output.put\_line('The Most generous customers who would get discount coupon :');

loop

fetch c1 into x, y;

exit when c1%rowcount=4;

dbms\_output.put\_line('Customer\_Name = ' || x || ' Tips = ' || y );

end loop;

close c1;

end;

/

exec custdesc;

/

--Procedure to retrieve the frugal customers who spent least money------

/

create or replace procedure CustAsc

IS

x varchar(20);

y number;

cursor c1 is select customer\_name, sum(amount) as amount from customers c, orders o where c.customer\_ID=o.customer\_ID group by customer\_name order by amount asc; --

begin

open c1;

loop

fetch c1 into x,y;

exit when c1%rowcount=4;

dbms\_output.put\_line('Customer : ' || x || ' tip : ' || y);

end loop;

close c1;

end;

/

exec custasc;

/

--procedure to retrieve states who tipped generously

/

create or replace procedure Custtot

IS

x varchar(20);

y number;

cursor c1 is select state, sum(tip) as tip from customers c, orders o where c.customer\_ID=o.customer\_ID group by state; --

begin

open c1;

loop

fetch c1 into x,y;

exit when c1%notfound;

dbms\_output.put\_line('State is : ' || x || ' tip :' || y);

end loop;

close c1;

end;

/

exec custtot;

/

/

begin

dbms\_output.put\_line(' Displaying the data from all the tables');

end;

/

select \* from Orders;

select \* from Customers;

select \* from Restaurant\_Inventory;

select \* from Menu\_Items;

select \* from Restaurants;

select \* from Cuisine\_Type;

select \* from Waiters;