/\*Team :

Murali - Member 1: Responsible for the Customers and CreditCards tables. Must create these tables.

Mouna - Member 2: Responsible for Product Categories and Products. Must create these tables.

Nikhitha - Member 3: Responsible for Orders. Must create this table.

Sushmitha - Member 4: Responsible for Invoices. Must create this table.

Ramya - Member 5: Responsible for the Reviews and Recommendations.

\*/

set serveroutput on;

create table customers

(

    Customer\_ID integer not null primary key,

    name varchar(30),

    email varchar(40),

    city varchar(30),

    state varchar(20),

    zip varchar(15)

);

create table Credit\_Card

(

    Credit\_Card# int not null primary key,

    expiration\_year int,

    expiration\_month int,

Credit\_Card\_Type varchar(30),

    Customer\_ID int not null,

    foreign key(Customer\_ID) references customers(Customer\_ID));

drop sequence seq;

create sequence seq start with 1;

--1.

create or replace procedure Add\_Customer(s\_name in varchar,s\_email in varchar, s\_city in varchar, s\_state in varchar,s\_zip in varchar)

is

begin

insert into Customers(customer\_id,name,email,city,state,zip) values(seq.nextval,s\_name,s\_email,s\_city,s\_state,s\_zip);

end;

--

begin

 Add\_Customer('John Smith',  'john@smith.com', 'Baltimore', 'MD', '21250');

 Add\_Customer('Mary Smith'  ,'mary@smith.com', 'Baltimore', 'MD', '21250');

 Add\_Customer('Pat Wagner' , 'pat@smith.com', 'Baltimore', 'MD', '21250');

 Add\_customer('Rajeev Kumar','rajeev@kumar.org','Columbia','SC','44250');

 end;

--Inserting sample data for credit\_card

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (123456789, '2021', '12', 'Visa', 1);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (987654321, '2023', '5', 'Mastercard', 2);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (123456780, '2026', '2', 'Visa', 3);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (987654323, '2022', '3', 'Mastercard', 2);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (987654322, '2021', '5', 'Mastercard', 4);

 --2.Show\_all\_customers\_in\_state: Input parameter is a state. Print a list of all customers in that state including customer name, email, address, credit card numbers, and type.

CREATE OR REPLACE PROCEDURE Show\_all\_customers\_in\_state(s\_state IN VARCHAR)

IS

  CURSOR c1 IS

    SELECT distinct c.name, c.email, (c.city|| ' '||c.state||' ' ||c.zip) AS Address, cc.credit\_card#, cc.credit\_card\_type

    FROM customers c, credit\_card cc

    WHERE c.customer\_id = cc.customer\_id AND c.state = s\_state ;

  cust\_cnt NUMBER;

BEGIN

  SELECT COUNT(\*) INTO cust\_cnt FROM customers WHERE state = s\_state;

  IF cust\_cnt = 0 THEN

    RAISE no\_data\_found;

  END IF;

  FOR item IN c1 LOOP

    dbms\_output.put\_line(item.name||' '||item.email|| ' '||item.Address||' '||item.credit\_card#||' '||item.credit\_card\_type);

  END LOOP;

EXCEPTION

  WHEN no\_data\_found THEN

    dbms\_output.put\_line('No data for customers in the state');

END;

begin

Show\_all\_customers\_in\_state('MD');

end;

set serveroutput on;

drop table reviews;

drop table Recommendation;

drop table invoice;

drop table Credit\_Card;

drop table orders;

drop table Products;

drop table product\_categories;

drop table customers;

drop SEQUENCE Id\_seq;-->(Please use the same sequence name  for every table )

CREATE SEQUENCE Id\_seq

INCREMENT BY 1

START WITH 1;

Member 1:

Member 2 -Mouna :

set serveroutput on;

drop table reviews;

drop table Recommendation;

drop table invoice;

drop table Credit\_Card;

drop table orders;

drop table Products;

drop table product\_categories;

drop table customers;

drop SEQUENCE Id\_seq;

CREATE SEQUENCE Id\_seq

START WITH 1

INCREMENT BY 1;

create table product\_categories

(

    category\_id int,

    category\_name varchar(100),

    description varchar(150),

    primary key(category\_id));

Create  table Products

(

    Product\_ID     int not null primary key ,

    product\_name    varchar(30),

    quantity        number (30)CONSTRAINT Valid\_Quantity\_Check CHECK (quantity > 0),

    unit\_price      float  (30),

    category\_id     int not null,

    FOREIGN KEY (category\_ID) REFERENCES product\_categories(category\_id)

);

--Add\_Category: This procedure takes as input all necessary information to  insert a new category to the Category table.

Create or replace PROCEDURE Add\_Category(

    p\_category\_name  in VARCHAR,

    P\_description in VARCHAR)

AS

BEGIN

    INSERT INTO product\_categories (category\_id,category\_name, description)

    VALUES (Id\_seq.NEXTVAL,p\_category\_name, P\_description);

    dbms\_output.put\_line ('Category Table updated');

EXCEPTION

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

END;

-- helper function to find the category ID that is needed in adding a product.

create or replace function  FIND\_PRODUCT\_CATEGORY\_ID(p\_category\_name in varchar)

return number

IS

v\_category\_id number ;

Begin

select category\_id  into v\_category\_id  from  product\_categories where

category\_name = p\_category\_name;

return v\_category\_id;

dbms\_output.put\_line ('v\_category\_id id fetched');

EXCEPTION

when no\_data\_found then

dbms\_output.put\_line('no such Category');

        return -1;

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

end;

--Add\_Product: This procedure takes as input all necessary information to insert a new product to the Products table.

CREATE OR REPLACE PROCEDURE Add\_Product (

    p\_product\_name IN VARCHAR2,

    p\_category\_name IN VARCHAR2,

    p\_price IN NUMBER,

    p\_quantity IN NUMBER

) AS

    v\_category\_id NUMBER;

BEGIN

    -- Find the category ID for the given category name

    v\_category\_id := FIND\_PRODUCT\_CATEGORY\_ID(p\_category\_name);

    -- Insert the new product into the Products table

    INSERT INTO Products (Product\_ID,product\_name,quantity,unit\_price,category\_id )

    VALUES (Id\_seq.nextval,p\_product\_name,p\_quantity,p\_price,v\_category\_id);

    dbms\_output.put\_line ('Products Table Updated');

EXCEPTION

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

END;

--Inserting data into Categories table

exec Add\_Category('Music', 'vocal and instrumental music');

exec Add\_Category('Electronics','Smartphone,tablets,Computers and laptops');

--Inserting data into product  table

exec Add\_Product('40inchTV','Electronics', 100, 200);

exec Add\_Product('50inchTV','Electronics', 100, 300);

exec Add\_Product('80inchTV','Electronics', 100, 1000);

exec Add\_Product('BestOfBeyonce','Music', 100, 20);

exec Add\_Product('BestOfTaylorSwift','Music', 100, 20);

exec Add\_Product('BestOfEmianem','Music', 100, 20);

exec Add\_Product('BestOfWeeknd','Music', 100, 20);

select \* from  Products;

select \* from  product\_categories;

--Update\_Inventory: When an order is placed, the Products table must be updated with the new remaining quantity.

--Create a procedure that will take as input the productID and the units (quantity) of

--a product in an order, and will update the inventory in the Products table to reflect the new quantity after the order.

CREATE OR REPLACE PROCEDURE Update\_Inventory (

    p\_Product\_ID  IN int ,

    p\_quantity IN number

) AS

v\_Product\_ID int;

Invalid\_Quantity exception;

BEGIN

if p\_quantity<0 THEN RAISE Invalid\_Quantity ;

end if ;

  select Product\_ID into v\_Product\_ID from Products where Product\_ID=p\_Product\_ID

  and quantity >= p\_quantity;

    -- Update  the new remaining  quantity  into the Products table

    Update Products

    set quantity = quantity - p\_quantity

    where Product\_ID = p\_Product\_ID;

    dbms\_output.put\_line ('Products Table Updated  the new remaining  quantity');

EXCEPTION

when no\_data\_found then

  dbms\_output.put\_line('Product  or Quantity is not available ');

when Invalid\_Quantity then

       DBMS\_OUTPUT.PUT\_LINE ('INVALID INPUT FROM USER');

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

END;

exec Update\_Inventory(4,60);

--Report\_Inventory: This procedure will display a report of available inventory.

--For each product category print a list with the name of the category and the quantity of all available products in that category.

CREATE OR REPLACE PROCEDURE Report\_Inventory

AS   cursor c1 is  SELECT   category\_name, sum (quantity) as TotalQuantity

from products p, product\_categories pc

where  p.category\_id=pc.category\_id group by  category\_name ;

rec1    c1%rowtype;

BEGIN

open c1 ;

loop

fetch c1 into rec1 ;

exit when c1%notfound;

dbms\_output.put\_line ('category : '||rec1.category\_name ||' ,TotalQuantity - '||rec1.TotalQuantity);

end loop;

close c1;

EXCEPTION

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

END;

exec Report\_Inventory;

----Helper function to find product ID

create or replace function  FIND\_PRODUCT\_ID(p\_productname in varchar)

return number

IS

v\_Product\_ID number ;

Begin

select Product\_ID  into v\_Product\_ID  from  products where

product\_name = p\_productname;

return v\_Product\_ID;

dbms\_output.put\_line ('v\_Product\_ID id fetched');

EXCEPTION

when no\_data\_found then

dbms\_output.put\_line('no such product');

        return -1;

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

end;

--execute block for getting product

declare

Product\_ID int ;

begin

Product\_ID :=FIND\_PRODUCT\_ID('50inchTV');

dbms\_output.put\_line('Product\_ID :'||Product\_ID);

end;

Member 3 Nikhitha

\* Nikhitha Vikram Melige

set serveroutput on;

-- procedure to place an order

create or replace function find\_customer\_id (p\_email in varchar) return number is v\_customerid number;

begin

select customer\_id into v\_customerid from customers where email = p\_email;

return v\_customerid;

exception

when no\_data\_found then dbms\_output.put\_line('Customer not found');

end;

/

create or replace function find\_product\_id(p\_productname in varchar) return number is v\_productid number;

begin

select product\_id into v\_productid from products where product\_name = p\_productname;

return v\_productid;

exception

when no\_data\_found then

dbms\_output.put\_line('Product not found');

end;

/

CREATE OR REPLACE PROCEDURE Invoice\_customer (

    v\_customerid NUMBER,

    v\_productid NUMBER,

    p\_quantity NUMBER,

    p\_creditcard VARCHAR2

) AS

    I\_customerid NUMBER;

    I\_productid NUMBER;

BEGIN

    dbms\_output.put\_line('--');

END;

/

create or replace procedure Place\_Order (

p\_email in varchar,

p\_productname in varchar,

p\_quantity in number,

p\_creditcard in varchar)  AS

v\_customerid Number;

v\_productid Number;

begin

-- to get customer id

v\_customerid := find\_customer\_id(p\_email);

-- to get product id

v\_productid := find\_product\_id(p\_productname);

-- update inventory

Update\_Inventory(v\_productid, p\_quantity);

-- invoice customer

Invoice\_Customer(v\_customerid, v\_productid, p\_quantity, p\_creditcard);

-- insert order into database

Insert into Orders (customer\_id, productid, quantity) values (v\_customerid, v\_productid, p\_quantity);

commit;

end;

/

-- procedure to show all orders

create or replace procedure Show\_Orders as

v\_totalorders Number := 0;

v\_totalamount Number := 0;

begin

for order\_rec in (select c.name  as customer\_name,

                  p.product\_name,

o.quantity,

p.unit\_price \* o.quantity as amount\_charged

from orders o

join customers c on o.customer\_id = c.customer\_id

join products p on o.productid = p.product\_id)

loop

dbms\_output.put\_line(order\_rec.customer\_name || ' , ' || order\_rec.product\_name || ' , ' || order\_rec.quantity || ', ' || order\_rec.amount\_charged);

v\_totalorders := v\_totalorders + 1;

v\_totalamount := v\_totalamount + order\_rec.amount\_charged;

end loop;

dbms\_output.put\_line('Grand total:  ' || v\_totalorders || ' orders, $' || v\_totalamount);

end;

—-------------------

—-------------------------------------------------

Ramya Jyotsna Neelakantrao  
  
Set serveroutput on;

create or replace function find\_product\_id(p\_productname in varchar) return number is v\_productid number;

begin

select product\_id into v\_productid from products where product\_name = p\_productname;

return v\_productid;

exception

when no\_data\_found then

dbms\_output.put\_line('Product not found');

end;

CREATE OR REPLACE PROCEDURE Add\_Review(

  p\_reviewer\_email IN VARCHAR2,

  p\_stars\_given IN NUMBER,

  p\_product\_name IN VARCHAR2,

  p\_review\_text IN VARCHAR2

)

IS

  v\_product\_id NUMBER;

BEGIN

  -- Find the product ID using a helper function

  v\_product\_id := find\_product\_id(p\_product\_name);

  -- Insert the review into the Reviews table

  INSERT INTO Reviews (reviewer\_email, stars\_given, product\_id, review\_text)

  VALUES (p\_reviewer\_email, p\_stars\_given, v\_product\_id, p\_review\_text);

  COMMIT;

END;

/

Begin

Add\_Review('john@smith.com',2,'50inchTV', 'not a good TV. It could be better');

Add\_Review('barney@abc.com',1,'50inchTV','do not buy');

Add\_Review('barney@abc.com', 5 ,'80inchTV', 'Excellent. This is the best TV.');

Add\_Review('mary@smith.com',5,'80inchTV','Excellent. This is the best TV.');

Add\_Review('mary@smith.com', 5, 'BestOfBeyonce', 'Excellent. Best ever.');

Add\_Review('z@abc.com',4, 'BestOfBeyonce', 'Enjoyed it. Excellent sound.');

End;

EXCUITED CODE

drop table reviews;

drop table Recommendation;

drop table invoice;

drop table Credit\_Card;

drop table orders;

drop table Products;

drop table product\_categories;

drop table customers;

create table customers

(

    Customer\_ID integer not null primary key,

    name varchar(30),

    email varchar(40),

    city varchar(30),

    state varchar(20),

    zip varchar(15)

);

create table product\_categories

(

    category\_id int,

    category\_name varchar(100),

    description varchar(150),

    primary key(category\_id));

Create table Products

(

    Product\_ID     int not null primary key ,

    product\_name    varchar(30),

    quantity        number (30),

    unit\_price      float  (30),

    category\_id     int not null,

    FOREIGN KEY (category\_ID) REFERENCES product\_categories(category\_id)

);

create table orders

(

    orderID int,

    Customer\_ID int,

    productID int,

    quantity int,

    primary key(orderID),

    foreign key(Customer\_ID) references customers(Customer\_ID),

    foreign key(productID) references products(product\_ID)

);

create table Credit\_Card

(

    Credit\_Card# int not null primary key,

    expiration\_year int,

    expiration\_month int,

Credit\_Card\_Type varchar(30),

    Customer\_ID int not null,

    foreign key(Customer\_ID) references customers(Customer\_ID));

create table invoice

(

    invoiceID int,

    orderID int,

    Customer\_ID int,

    creditcard# number,

    amount number,

    primary key(invoiceID),

    foreign key(orderID) references orders(orderID),

    foreign key(Customer\_ID) references customers(Customer\_ID),

    foreign key (creditcard#) references credit\_card(credit\_card#)

);

--

Create table Recommendation

(

    Recommendation\_ID         int not null primary key ,

    Customer\_ID               int,

    Recommended\_Product\_ID    int,

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

    FOREIGN KEY (Recommended\_Product\_ID) REFERENCES Products(Product\_ID)

);

create table reviews

(

    Review\_ID int not null primary key,

    Product\_ID int,

    Reviewer\_email varchar(30),

    Stars\_Given varchar(5),

    Review\_text varchar(10),

    foreign key (Product\_ID) references Products(Product\_ID)

);

--muralidhar reddy

set serveroutput on;

drop sequence seq;

create sequence seq start with 1;

--1.

create or replace procedure Add\_Customer(s\_name in varchar,s\_email in varchar, s\_city in varchar, s\_state in varchar,s\_zip in varchar)

is

begin

insert into Customers(customer\_id,name,email,city,state,zip) values(seq.nextval,s\_name,s\_email,s\_city,s\_state,s\_zip);

end;

--

begin

 Add\_Customer('John Smith',  'john@smith.com', 'Baltimore', 'MD', '21250');

 Add\_Customer('Mary Smith'  ,'mary@smith.com', 'Baltimore', 'MD', '21250');

 Add\_Customer('Pat Wagner' , 'pat@smith.com', 'Baltimore', 'MD', '21250');

 Add\_customer('Rajeev Kumar','rajeev@kumar.org','Columbia','SC','44250');

 end;

--Inserting sample data for credit\_card

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (123456789, '2021', '12', 'Visa', 1);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (987654321, '2023', '5', 'Mastercard', 2);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (123456780, '2026', '2', 'Visa', 3);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (987654323, '2022', '3', 'Mastercard', 4);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (987654322, '2021', '5', 'Mastercard', 1);

--2.Show\_all\_customers\_in\_state: Input parameter is a state. Print a list of all customers in that state including customer name, email, address, credit card numbers, and type.

CREATE OR REPLACE PROCEDURE Show\_all\_customers\_in\_state(s\_state IN VARCHAR)

IS

  CURSOR c1 IS

    SELECT distinct c.name, c.email, (c.city|| ' '||c.state||' ' ||c.zip) AS Address, cc.credit\_card#, cc.credit\_card\_type

    FROM customers c, credit\_card cc

    WHERE c.customer\_id = cc.customer\_id AND c.state = s\_state ;

  cust\_cnt NUMBER;

BEGIN

  SELECT COUNT(\*) INTO cust\_cnt FROM customers WHERE state = s\_state;

  IF cust\_cnt = 0 THEN

    RAISE no\_data\_found;

  END IF;

  FOR item IN c1 LOOP

    dbms\_output.put\_line(item.name||' '||item.email|| ' '||item.Address||' '||item.credit\_card#||' '||item.credit\_card\_type);

  END LOOP;

EXCEPTION

  WHEN no\_data\_found THEN

    dbms\_output.put\_line('No data for customers in the state');

END;

begin

Show\_all\_customers\_in\_state('MD');

end;

--mouna

drop SEQUENCE Id\_seq;-->(Please use the same sequence name  for every table )

CREATE SEQUENCE Id\_seq

START WITH 1

INCREMENT BY 1;

--Add\_Category: This procedure takes as input all necessary information to  insert a new category to the Category table.

Create or replace PROCEDURE Add\_Category(

    p\_category\_name  in VARCHAR,

    P\_description in VARCHAR)

AS

BEGIN

    INSERT INTO product\_categories (category\_id,category\_name, description)

    VALUES (Id\_seq.NEXTVAL,p\_category\_name, P\_description);

    dbms\_output.put\_line ('Cy Table updated');

EXCEPTION

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

END;

-- helper function to find the category ID that is needed in adding a product.

create or replace function  FIND\_PRODUCT\_CATEGORY\_ID(p\_category\_name in varchar)

return number

IS

v\_category\_id number ;

Begin

select category\_id  into v\_category\_id  from  product\_categories where

category\_name = p\_category\_name;

return v\_category\_id;

dbms\_output.put\_line ('v\_category\_id id fetched');

EXCEPTION

when no\_data\_found then

dbms\_output.put\_line('no such Category');

        return -1;

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

end;

--Inserting data into Categoeries table

exec Add\_Category('Music', 'vocal and instrumental music');

exec Add\_Category('Electronics','Smartphone,tablets,Computers and laptops');

--Inserting data into product  table

exec Add\_Product('40inchTV','Electronics', 100, 200);

exec Add\_Product('50inchTV','Electronics', 100, 300);

exec Add\_Product('80inchTV','Electronics', 100, 1000);

exec Add\_Product('BestOfBeyonce','Music', 100, 20);

exec Add\_Product('BestOfTaylorSwift','Music', 100, 20);

exec Add\_Product('BestOfEmianem','Music', 100, 20);

exec Add\_Product('BestOfWeeknd','Music', 100, 20);ategor

--Update\_Inventory: When an order is placed, the Products table must be updated with the new remaining quantity.

--Create a procedure that will take as input the productID and the units (quantity) of

--a product in an order, and will update the inventory in the Products table to reflect the new quantity after the order.

CREATE OR REPLACE PROCEDURE Update\_Inventory (

    p\_Product\_ID  IN int ,

    p\_quantity IN number

) AS

v\_Product\_ID int;

Invalid\_Quantity exception;

BEGIN

if p\_quantity<0 THEN RAISE Invalid\_Quantity ;

end if ;

  select ProductID into v\_Product\_ID from Products where ProductID=p\_Product\_ID

  and quantity >= p\_quantity;

    -- Update  the new remaining  quantity  into the Products table

    Update Products

    set quantity = quantity - p\_quantity

    where ProductID = p\_Product\_ID;

    dbms\_output.put\_line ('Products Table Updated  the new remaining  quantity');

EXCEPTION

when no\_data\_found then

  dbms\_output.put\_line('Product  or Quantity is not available ');

when Invalid\_Quantity then

       DBMS\_OUTPUT.PUT\_LINE ('INVALID INPUT FROM USER');

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

END;

exec Update\_Inventory(4,60);

--nikhitha

-- procedure to place an order

create or replace function find\_customer\_id (p\_email in varchar) return number is v\_customerid number;

begin

select customer\_id into v\_customerid from customers where email = p\_email;

return v\_customerid;

exception

when no\_data\_found then dbms\_output.put\_line('Customer not found');

end;

/

create or replace function find\_product\_id(p\_productname in varchar) return number is v\_productid number;

begin

select product\_id into v\_productid from products where product\_name = p\_productname;

return v\_productid;

exception

when no\_data\_found then

dbms\_output.put\_line('Product not found');

end;

/

CREATE OR REPLACE PROCEDURE Invoice\_customer (

    v\_customerid NUMBER,

    v\_productid NUMBER,

    p\_quantity NUMBER,

    p\_creditcard VARCHAR2

) AS

    I\_customerid NUMBER;

    I\_productid NUMBER;

BEGIN

    dbms\_output.put\_line('--');

END;

/

drop sequence seq;

create sequence seq start with 1;

create or replace procedure Place\_Order (

p\_email in varchar,

p\_productname in varchar,

p\_quantity in number,

p\_creditcard in number)  AS

v\_customerid Number;

v\_productid Number;

begin

-- to get customer id

v\_customerid := find\_customer\_id(p\_email);

-- to get product id

v\_productid := find\_product\_id(p\_productname);

-- update inventory

Update\_Inventory(v\_productid, p\_quantity);

-- invoice customer

Invoice\_Customer(v\_customerid, v\_productid, p\_quantity, p\_creditcard);

-- insert order into database

Insert into Orders (orderid,customer\_id, productid, quantity) values (seq.nextval,v\_customerid, v\_productid, p\_quantity);

commit;

end;

/

exec Place\_Order('pat@smith.com','40inchTV',200,123456780);

-- procedure to show all orders

create or replace procedure Show\_Orders as

v\_totalorders Number := 0;

v\_totalamount Number := 0;

begin

for order\_rec in (select c.name  as customer\_name,

                  p.product\_name,

o.quantity,

p.unit\_price \* o.quantity as amount\_charged

from orders o

join customers c on o.customer\_id = c.customer\_id

join products p on o.productid = p.product\_id)

loop

dbms\_output.put\_line(order\_rec.customer\_name || ' , ' || order\_rec.product\_name || ' , ' || order\_rec.quantity || ', ' || order\_rec.amount\_charged);

v\_totalorders := v\_totalorders + 1;

v\_totalamount := v\_totalamount + order\_rec.amount\_charged;

end loop;

dbms\_output.put\_line('Grand total:  ' || v\_totalorders || ' orders, $' || v\_totalamount);

end;

exec Show\_Orders();

**Member 4 -Sushmitha :**

–Sushmitha Kanapuram

set serveroutput on;

CREATE OR REPLACE PROCEDURE Invoice\_Customer (

  p\_order\_name IN  VARCHAR2,

  p\_customer\_name IN  VARCHAR2,

  p\_cc\_num IN  VARCHAR2,

  p\_amount IN  NUMBER

) AS

  v\_order\_id NUMBER;

  v\_customer\_id NUMBER;

  v\_invoice\_id  NUMBER;

  v\_date        DATE;

BEGIN

  -- find the order ID

  SELECT order\_id INTO v\_order\_id FROM orders WHERE order\_name = p\_order\_name;

  -- find the customer ID

  SELECT customer\_id INTO v\_customer\_id FROM customers WHERE customer\_name = p\_customer\_name;

  -- find the invoice ID

  SELECT invoice\_seq.NEXTVAL INTO v\_invoice\_id FROM DUAL;

  -- get the current date

  v\_date := SYSDATE;

  -- insert the invoice record

  INSERT INTO invoice (

    invoice\_id,

    order\_id,

    customer\_id,

    cc\_num,

    invoice\_date,

    amount

  ) VALUES (

    v\_invoice\_id,

    v\_order\_id,

    v\_customer\_id,

    p\_cc\_num,

    v\_date,

    p\_amount

  );

  -- commit the transaction

  COMMIT;

  -- display a success message

  DBMS\_OUTPUT.PUT\_LINE('Invoice created successfully. Invoice ID: ' || v\_invoice\_id);

END;

EXEC Invoice\_Customer('2 x 50inchTVs', 'John Smith', '12345', 2000);

EXEC Invoice\_Customer('30 x BestOfBeyonce', 'John Smith', '12345', 300);

EXEC Invoice\_Customer('1 x 80inchTV', 'Mary Smith', '12346', 1500);

CREATE OR REPLACE PROCEDURE Report\_Best\_Customers(cutoff\_amount in number) IS

BEGIN

  FOR c IN (SELECT c.customer\_ID, c.name, SUM(Orders.quantity \* Products.unit\_Price) AS total\_spent

            FROM Orders

            JOIN Customers c ON Orders.customer\_ID = c.customer\_ID

            JOIN Products ON Orders.productID = Products.product\_ID

            GROUP BY c.customer\_ID, c.name)

  LOOP

    if c.total\_spent > cutoff\_amount then

        DBMS\_OUTPUT.PUT\_LINE(c.name || ' spent ' || c.total\_spent || ' in total.');

    end if;

  END LOOP;

END;

/

begin

Report\_Best\_Customers(100);

end;

Finall excecution

set serveroutput on;

drop table reviews;

drop table Recommendation;

drop table invoice;

drop table Credit\_Card;

drop table orders;

drop table Products;

drop table product\_categories;

drop table customers;

create table customers

(

    Customer\_ID integer not null primary key,

    name varchar(30),

    email varchar(40),

    city varchar(30),

    state varchar(20),

    zip varchar(15)

);

create table product\_categories

(

    category\_id int,

    category\_name varchar(100),

    description varchar(150),

    primary key(category\_id));

Create table Products

(

    Product\_ID     int not null primary key ,

    product\_name    varchar(30),

    quantity        number (30),

    unit\_price      float  (30),

    category\_id     int not null,

    FOREIGN KEY (category\_ID) REFERENCES product\_categories(category\_id)

);

create table orders

(

    orderID int,

    Customer\_ID int,

    productID int,

    quantity int,

    primary key(orderID),

    foreign key(Customer\_ID) references customers(Customer\_ID),

    foreign key(productID) references products(product\_ID)

);

create table Credit\_Card

(

    Credit\_Card# int not null primary key,

    expiration\_year int,

    expiration\_month int,

Credit\_Card\_Type varchar(30),

    Customer\_ID int not null,

    foreign key(Customer\_ID) references customers(Customer\_ID));

create table invoice

(

    invoiceID int,

    orderID int,

    Customer\_ID int,

    creditcard# number,

    amount number,

    primary key(invoiceID),

    foreign key(orderID) references orders(orderID),

    foreign key(Customer\_ID) references customers(Customer\_ID),

    foreign key (creditcard#) references credit\_card(credit\_card#)

);

--

Create table Recommendation

(

    Recommendation\_ID         int not null primary key ,

    Customer\_ID               int,

    Recommended\_Product\_ID    int,

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

    FOREIGN KEY (Recommended\_Product\_ID) REFERENCES Products(Product\_ID)

);

create table reviews

(

    Review\_ID int not null primary key,

    Product\_ID int,

    Reviewer\_email varchar(30),

    Stars\_Given varchar(5),

    Review\_text varchar(10),

    foreign key (Product\_ID) references Products(Product\_ID)

);

drop sequence seq;

create sequence seq start with 1;

--1.

create or replace procedure Add\_Customer(s\_name in varchar,s\_email in varchar, s\_city in varchar, s\_state in varchar,s\_zip in varchar)

is

begin

insert into Customers(customer\_id,name,email,city,state,zip) values(seq.nextval,s\_name,s\_email,s\_city,s\_state,s\_zip);

end;

--

begin

 Add\_Customer('John Smith',  'john@smith.com', 'Baltimore', 'MD', '21250');

 Add\_Customer('Mary Smith'  ,'mary@smith.com', 'Baltimore', 'MD', '21250');

 Add\_Customer('Pat Wagner' , 'pat@smith.com', 'Baltimore', 'MD', '21250');

 Add\_customer('Rajeev Kumar','rajeev@kumar.org','Columbia','SC','44250');

 end;

--Inserting sample data for credit\_card

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (123456789, '2021', '12', 'Visa', 1);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (987654321, '2023', '5', 'Mastercard', 2);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (123456780, '2026', '2', 'Visa', 3);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (987654323, '2022', '3', 'Mastercard', 4);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (987654322, '2021', '5', 'Mastercard', 1);

--2.Show\_all\_customers\_in\_state: Input parameter is a state. Print a list of all customers in that state including customer name, email, address, credit card numbers, and type.

CREATE OR REPLACE PROCEDURE Show\_all\_customers\_in\_state(s\_state IN VARCHAR)

IS

  CURSOR c1 IS

    SELECT distinct c.name, c.email, (c.city|| ' '||c.state||' ' ||c.zip) AS Address, cc.credit\_card#, cc.credit\_card\_type

    FROM customers c, credit\_card cc

    WHERE c.customer\_id = cc.customer\_id AND c.state = s\_state ;

  cust\_cnt NUMBER;

BEGIN

  SELECT COUNT(\*) INTO cust\_cnt FROM customers WHERE state = s\_state;

  IF cust\_cnt = 0 THEN

    RAISE no\_data\_found;

  END IF;

  FOR item IN c1 LOOP

    dbms\_output.put\_line(item.name||' '||item.email|| ' '||item.Address||' '||item.credit\_card#||' '||item.credit\_card\_type);

  END LOOP;

EXCEPTION

  WHEN no\_data\_found THEN

    dbms\_output.put\_line('No data for customers in the state');

END;

begin

Show\_all\_customers\_in\_state('MD');

end;

--mouna

drop SEQUENCE Id\_seq;-->(Please use the same sequence name  for every table )

CREATE SEQUENCE Id\_seq

START WITH 1

INCREMENT BY 1;

--Add\_Category: This procedure takes as input all necessary information to  insert a new category to the Category table.

Create or replace PROCEDURE Add\_Category(

    p\_category\_name  in VARCHAR,

    P\_description in VARCHAR)

AS

BEGIN

    INSERT INTO product\_categories (category\_id,category\_name, description)

    VALUES (Id\_seq.NEXTVAL,p\_category\_name, P\_description);

    dbms\_output.put\_line ('Cy Table updated');

EXCEPTION

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

END;

-- helper function to find the category ID that is needed in adding a product.

create or replace function  FIND\_PRODUCT\_CATEGORY\_ID(p\_category\_name in varchar)

return number

IS

v\_category\_id number ;

Begin

        select category\_id  into v\_category\_id  from

        product\_categories where

        category\_name = p\_category\_name;

return v\_category\_id;

dbms\_output.put\_line ('v\_category\_id id fetched');

EXCEPTION

when no\_data\_found then

         dbms\_output.put\_line('no such Category');

            return -1;

when others then

          dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

          dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

END;

--Inserting data into Categories table

Begin

Add\_Category('Music', 'vocal and instrumental music');

Add\_Category('Electronics','Smartphone,tablets,Computers and laptops');

End ;

--Inserting data into product  table

Begin

Add\_Product('40inchTV','Electronics', 100, 200);

Add\_Product('50inchTV','Electronics', 100, 300);

Add\_Product('80inchTV','Electronics', 100, 1000);

Add\_Product('BestOfBeyonce','Music', 100, 20);

Add\_Product('BestOfTaylorSwift','Music', 100, 20);

Add\_Product('BestOfEmianem','Music', 100, 20);

Add\_Product('BestOfWeeknd','Music', 100, 20);

end;

--Update\_Inventory: When an order is placed, the Products table must be updated with the new remaining quantity.

--Create a procedure that will take as input the productID and the units (quantity) of

--a product in an order, and will update the inventory in the Products table to reflect the new quantity after the order.

CREATE OR REPLACE PROCEDURE Update\_Inventory ( p\_Product\_ID  IN int ,

                                                                                               p\_quantity IN number )

AS

v\_Product\_ID int;

Invalid\_Quantity exception;

BEGIN

If     p\_quantity<0 THEN RAISE Invalid\_Quantity ;

end if ;

             select Product\_ID into v\_Product\_ID from Products

             where   Product\_ID=p\_Product\_ID

              and quantity >= p\_quantity;

    -- Update  the new remaining  quantity  into the Products table

              Update Products

              set quantity = quantity - p\_quantity

              where Product\_ID = p\_Product\_ID;

    dbms\_output.put\_line ('Products Table Updated  the new remaining  quantity');

EXCEPTION

when no\_data\_found then

         dbms\_output.put\_line('Product  or Quantity is not available ');

when Invalid\_Quantity then

         dbms\_output.put\_line ('Invalid input from user');

when others then

          dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

          dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

END;

Begin

Update\_Inventory(4,60);

End;

--nikhitha

-- procedure to place an order

create or replace function find\_customer\_id (p\_email in varchar) return number is v\_customerid number;

begin

select customer\_id into v\_customerid from customers where email = p\_email;

return v\_customerid;

exception

when no\_data\_found then dbms\_output.put\_line('Customer not found');

end;

/

create or replace function find\_product\_id(p\_productname in varchar) return number is v\_productid number;

begin

select product\_id into v\_productid from products where product\_name = p\_productname;

return v\_productid;

exception

when no\_data\_found then

dbms\_output.put\_line('Product not found');

end;

/

CREATE OR REPLACE PROCEDURE Invoice\_customer (

    v\_customerid NUMBER,

    v\_productid NUMBER,

    p\_quantity NUMBER,

    p\_creditcard VARCHAR2

) AS

    I\_customerid NUMBER;

    I\_productid NUMBER;

BEGIN

    dbms\_output.put\_line('--');

END;

/

drop sequence seq;

create sequence seq start with 1;

create or replace procedure Place\_Order (

p\_email in varchar,

p\_productname in varchar,

p\_quantity in number,

p\_creditcard in number)  AS

v\_customerid Number;

v\_productid Number;

begin

-- to get customer id

v\_customerid := find\_customer\_id(p\_email);

-- to get product id

v\_productid := find\_product\_id(p\_productname);

-- update inventory

Update\_Inventory(v\_productid, p\_quantity);

-- invoice customer

Invoice\_Customer(v\_customerid, v\_productid, p\_quantity, p\_creditcard);

-- insert order into database

Insert into Orders (orderid,customer\_id, productid, quantity) values (seq.nextval,v\_customerid, v\_productid, p\_quantity);

commit;

end;

/

exec Place\_Order('pat@smith.com','40inchTV',200,123456780);

-- procedure to show all orders

create or replace procedure Show\_Orders as

v\_totalorders Number := 0;

v\_totalamount Number := 0;

begin

for order\_rec in (select c.name  as customer\_name,

                  p.product\_name,

o.quantity,

p.unit\_price \* o.quantity as amount\_charged

from orders o

join customers c on o.customer\_id = c.customer\_id

join products p on o.productid = p.product\_id)

loop

dbms\_output.put\_line(order\_rec.customer\_name || ' , ' || order\_rec.product\_name || ' , ' || order\_rec.quantity || ', ' || order\_rec.amount\_charged);

v\_totalorders := v\_totalorders + 1;

v\_totalamount := v\_totalamount + order\_rec.amount\_charged;

end loop;

dbms\_output.put\_line('Grand total:  ' || v\_totalorders || ' orders, $' || v\_totalamount);

end;

exec Show\_Orders();

—----mem 4

CREATE OR REPLACE PROCEDURE Invoice\_Customer (

  p\_orderid IN  number,

  p\_customerid IN  number,

  p\_credit\_card IN  number,

  p\_amount IN  NUMBER

) AS

 d invoice %rowtype;

 begin

 d.invoiceid:=seq.nextval;

 d.orderid:=p\_orderid;

 d.customer\_id:=p\_customerid;

 d.creditcard#:=p\_credit\_card;

 d.amount:=p\_amount;

  -- insert the invoice record

  INSERT INTO invoice values d;

   END;

   /

–need to write calling function

CREATE OR REPLACE PROCEDURE Report\_Best\_Customers(cutoff\_amount in number) IS

BEGIN

  FOR c IN (SELECT c.customer\_ID, c.name, SUM(Orders.quantity \* Products.unit\_Price) AS total\_spent

            FROM Orders

            JOIN Customers c ON Orders.customer\_ID = c.customer\_ID

            JOIN Products ON Orders.productID = Products.product\_ID

            GROUP BY c.customer\_ID, c.name)

  LOOP

    if c.total\_spent > cutoff\_amount then

        DBMS\_OUTPUT.PUT\_LINE(c.name || ' spent ' || c.total\_spent || ' in total.');

    end if;

  END LOOP;

END;

/

begin

Report\_Best\_Customers(100);

End;

—--mem 5

create or replace function FIND\_PRODUCT\_ID (productname in products.product\_Name%type)

return int

is

productID int;

begin

    select product\_ID into productID from products where product\_Name = productname;

return productID;

end;

create or replace procedure Add\_Review(Reviewer\_Email in reviews.Reviewer\_Email%type,

stars in reviews.Stars\_Given%type,

product\_ID in reviews.product\_ID%type,

Review\_Text in reviews.Review\_Text%type) is

 x reviews%rowtype;

Begin

 x.Review\_ID := seq.NEXTVAL ;

 x.Reviewer\_Email := Reviewer\_Email;

 x.Stars\_Given := stars;

 x.product\_ID := product\_ID;

 x.Review\_Text := Review\_Text;

 insert into reviews values x;

end;

/

--- calling add review procedure

declare

productID int;

begin

productID := FIND\_PRODUCT\_ID('50inchTV');

Add\_Review( 'john@smith.com', 2 , productID,'not a good TV. It could be better');

Add\_Review( 'barney@abc.com', 1 ,productID, 'do not buy');

productID := FIND\_PRODUCT\_ID('80inchTV');

Add\_Review( 'barney@abc.com', 5 ,productID, 'Excellent. This is the best TV');

Add\_Review( 'mary@smith.com', 5 ,productID, 'Excellent. Best ever.');

productID := FIND\_PRODUCT\_ID('BestOfBeyonce');

Add\_Review( 'mary@smith.com', 5 ,productID, 'Excellent. Best ever.');

Add\_Review( 'z@abc.com', 4 ,productID, 'Enjoyed it. Excellent sound');

end;

--operation 2

CREATE OR REPLACE PROCEDURE Buy\_Or\_Beware(limit IN INT) IS

var\_product\_name varchar(50);

limit\_variable\_1 int := limit;

limit\_variable\_2 int := limit;

BEGIN

  DBMS\_OUTPUT.PUT\_LINE('Top rated products:');

  FOR c IN (SELECT AVG(Stars\_Given) AS Average\_stars, STDDEV(Stars\_Given) as std\_dev, product\_ID  FROM Reviews GROUP BY product\_ID ORDER BY Average\_stars DESC)

  LOOP

    SELECT Product\_Name INTO var\_product\_name FROM Products WHERE product\_ID = c.product\_ID;

    DBMS\_OUTPUT.PUT\_LINE(c.Average\_stars || ' ' || c.product\_ID || ' ' || var\_product\_name|| ' ' || c.std\_dev);

    limit\_variable\_1 := limit\_variable\_1 - 1;

    IF limit\_variable\_1 = 0 THEN

      EXIT;

    END IF;

  END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Buyer Beware: Stay Away from…');

  FOR b IN (SELECT AVG(Stars\_Given) AS Average\_stars, STDDEV(Stars\_Given) as std\_dev, product\_ID  FROM Reviews GROUP BY product\_ID ORDER BY Average\_stars )

  LOOP

    SELECT Product\_Name INTO var\_product\_name FROM Products WHERE product\_ID = b.product\_ID;

    DBMS\_OUTPUT.PUT\_LINE(b.Average\_stars || ' ' || b.product\_ID || ' ' || var\_product\_name|| ' ' || b.std\_dev);

    limit\_variable\_2 := limit\_variable\_2 - 1;

    IF limit\_variable\_2 = 0 THEN

      EXIT;

    END IF;

  END LOOP;

END;

begin

Buy\_Or\_Beware(2);

end;

Finally Executed 🙂

set serveroutput on;

drop table reviews;

drop table Recommendation;

drop table invoice;

drop table Credit\_Card;

drop table orders;

drop table Products;

drop table product\_categories;

drop table customers;

create table customers

(

    Customer\_ID integer  primary key,

    name varchar(30),

    email varchar(40),

    city varchar(30),

    state varchar(20),

    zip varchar(15)

);

create table product\_categories

(

    category\_id int,

    category\_name varchar(100),

    description varchar(150),

    primary key(category\_id));

Create  table Products

(

    Product\_ID     int primary key ,

    product\_name    varchar(30),

    quantity        number (30),

    unit\_price      float  (30),

    category\_id     int ,

    FOREIGN KEY (category\_ID) REFERENCES product\_categories(category\_id)

);

create table orders

(

    orderID int,

    Customer\_ID int,

    productID int,

    quantity int,

    primary key(orderID),

    foreign key(Customer\_ID) references customers(Customer\_ID),

    foreign key(productID) references products(product\_ID)

);

create table Credit\_Card

(

    Credit\_Card# int primary key,

    expiration\_year int,

    expiration\_month int,

Credit\_Card\_Type varchar(30),

    Customer\_ID int ,

    foreign key(Customer\_ID) references customers(Customer\_ID));

create table invoice

(

    invoiceID int,

    orderID int,

    Customer\_ID int,

    creditcard# number,

    amount number,

    primary key(invoiceID),

    foreign key(orderID) references orders(orderID),

    foreign key(Customer\_ID) references customers(Customer\_ID),

    foreign key (creditcard#) references credit\_card(credit\_card#)

);

--

Create table Recommendation

(

    Recommendation\_ID         int  primary key ,

    Customer\_ID               int,

    Recommended\_Product\_ID    int,

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

    FOREIGN KEY (Recommended\_Product\_ID) REFERENCES Products(Product\_ID)

);

create table reviews

(

    Review\_ID int  primary key,

    Product\_ID int,

    Reviewer\_email varchar(30),

    Stars\_Given number,

    Review\_text varchar(50),

    foreign key (Product\_ID) references Products(Product\_ID)

);

---mem1

drop sequence seq;

create sequence seq start with 1;

--1.

create or replace procedure Add\_Customer(s\_name in varchar,s\_email in varchar, s\_city in varchar, s\_state in varchar,s\_zip in varchar)

is

begin

insert into Customers(customer\_id,name,email,city,state,zip) values(seq.nextval,s\_name,s\_email,s\_city,s\_state,s\_zip);

end;

/

--

begin

 Add\_Customer('John Smith',  'john@smith.com', 'Baltimore', 'MD', '21250');

 Add\_Customer('Mary Smith'  ,'mary@smith.com', 'Baltimore', 'MD', '21250');

 Add\_Customer('Pat Wagner' , 'pat@smith.com', 'Baltimore', 'MD', '21250');

 Add\_customer('Rajeev Kumar','rajeev@kumar.org','Columbia','SC','44250');

 end;

/

--Inserting sample data for credit\_card

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (123456789, '2021', '12', 'Visa', 1);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (987654321, '2023', '5', 'Mastercard', 2);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (123456780, '2026', '2', 'Visa', 3);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (987654323, '2022', '3', 'Mastercard', 4);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (987654322, '2021', '5', 'Mastercard', 1);

--2.Show\_all\_customers\_in\_state: Input parameter is a state. Print a list of all customers in that state including customer name, email, address, credit card numbers, and type.

CREATE OR REPLACE PROCEDURE Show\_all\_customers\_in\_state(s\_state IN VARCHAR)

IS

  CURSOR c1 IS

    SELECT distinct c.name, c.email, (c.city|| ' '||c.state||' ' ||c.zip) AS Address, cc.credit\_card#, cc.credit\_card\_type

    FROM customers c, credit\_card cc

    WHERE c.customer\_id = cc.customer\_id AND c.state = s\_state ;

  cust\_cnt NUMBER;

BEGIN

  SELECT COUNT(\*) INTO cust\_cnt FROM customers WHERE state = s\_state;

  IF cust\_cnt = 0 THEN

    RAISE no\_data\_found;

  END IF;

  FOR item IN c1 LOOP

    dbms\_output.put\_line(item.name||' '||item.email|| ' '||item.Address||' '||item.credit\_card#||' '||item.credit\_card\_type);

  END LOOP;

EXCEPTION

  WHEN no\_data\_found THEN

    dbms\_output.put\_line('No data for customers in the state');

END;

/

begin

Show\_all\_customers\_in\_state('MD');

end;

/

--mem 2

drop SEQUENCE Id\_seq;-->(Please use the same sequence name  for every table )

CREATE SEQUENCE Id\_seq

START WITH 1

INCREMENT BY 1;

--Add\_Category: This procedure takes as input all necessary information to  insert a new category to the Category table.

Create or replace PROCEDURE Add\_Category(

    p\_category\_name  in VARCHAR,

    P\_description in VARCHAR)

AS

BEGIN

    INSERT INTO product\_categories (category\_id,category\_name, description)

    VALUES (Id\_seq.NEXTVAL,p\_category\_name, P\_description);

    dbms\_output.put\_line ('Cy Table updated');

EXCEPTION

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

END;

/

-- helper function to find the category ID that is needed in adding a product.

create or replace function  FIND\_PRODUCT\_CATEGORY\_ID(p\_category\_name in varchar)

return number

IS

v\_category\_id number ;

Begin

select category\_id  into v\_category\_id  from  product\_categories where

category\_name = p\_category\_name;

return v\_category\_id;

dbms\_output.put\_line ('v\_category\_id id fetched');

EXCEPTION

when no\_data\_found then

dbms\_output.put\_line('no such Category');

        return -1;

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

end;

/

--Inserting data into Categoeries table

begin

 Add\_Category('Music', 'vocal and instrumental music');

 Add\_Category('Electronics','Smartphone,tablets,Computers and laptops');

end;

/

--Inserting data into product  table

begin

Add\_Product('40inchTV','Electronics', 100, 200);

 Add\_Product('50inchTV','Electronics', 100, 300);

Add\_Product('80inchTV','Electronics', 100, 1000);

Add\_Product('BestOfBeyonce','Music', 100, 20);

Add\_Product('BestOfTaylorSwift','Music', 100, 20);

 Add\_Product('BestOfEmianem','Music', 100, 20);

 Add\_Product('BestOfWeeknd','Music', 100, 20);

end;

/

--Update\_Inventory: When an order is placed, the Products table must be updated with the new remaining quantity.

--Create a procedure that will take as input the productID and the units (quantity) of

--a product in an order, and will update the inventory in the Products table to reflect the new quantity after the order.

CREATE OR REPLACE PROCEDURE Update\_Inventory (

    p\_Product\_ID  IN int ,

    p\_quantity IN number

) AS

v\_Product\_ID int;

Invalid\_Quantity exception;

BEGIN

if p\_quantity<0 THEN RAISE Invalid\_Quantity ;

end if ;

  select Product\_ID into v\_Product\_ID from Products where Product\_ID=p\_Product\_ID

  and quantity >= p\_quantity;

    -- Update  the new remaining  quantity  into the Products table

    Update Products

    set quantity = quantity - p\_quantity

    where Product\_ID = p\_Product\_ID;

    dbms\_output.put\_line ('Products Table Updated  the new remaining  quantity');

EXCEPTION

when no\_data\_found then

  dbms\_output.put\_line('Product  or Quantity is not available ');

when Invalid\_Quantity then

       DBMS\_OUTPUT.PUT\_LINE ('INVALID INPUT FROM USER');

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

END;

/

exec Update\_Inventory(4,60);

--nikhitha

-- procedure to place an order

create or replace function find\_customer\_id (p\_email in varchar) return number is v\_customerid number;

begin

select customer\_id into v\_customerid from customers where email = p\_email;

return v\_customerid;

exception

when no\_data\_found then dbms\_output.put\_line('Customer not found');

end;

/

create or replace function find\_product\_id(p\_productname in varchar) return number is v\_productid number;

begin

select product\_id into v\_productid from products where product\_name = p\_productname;

return v\_productid;

exception

when no\_data\_found then

dbms\_output.put\_line('Product not found');

end;

/

CREATE OR REPLACE PROCEDURE Invoice\_customer (

    v\_customerid NUMBER,

    v\_productid NUMBER,

    p\_quantity NUMBER,

    p\_creditcard VARCHAR2

) AS

    I\_customerid NUMBER;

    I\_productid NUMBER;

BEGIN

    dbms\_output.put\_line('--');

END;

/

drop sequence seq;

create sequence seq start with 1;

create or replace procedure Place\_Order (

p\_email in varchar,

p\_productname in varchar,

p\_quantity in number,

p\_creditcard in number)  AS

v\_customerid Number;

v\_productid Number;

begin

-- to get customer id

v\_customerid := find\_customer\_id(p\_email);

-- to get product id

v\_productid := find\_product\_id(p\_productname);

-- update inventory

Update\_Inventory(v\_productid, p\_quantity);

-- invoice customer

Invoice\_Customer(v\_customerid, v\_productid, p\_quantity, p\_creditcard);

-- insert order into database

Insert into Orders (orderid,customer\_id, productid, quantity) values (seq.nextval,v\_customerid, v\_productid, p\_quantity);

commit;

end;

/

exec Place\_Order('pat@smith.com','40inchTV',200,123456780);

-- procedure to show all orders

create or replace procedure Show\_Orders as

v\_totalorders Number := 0;

v\_totalamount Number := 0;

begin

for order\_rec in (select c.name  as customer\_name,

                  p.product\_name,

o.quantity,

p.unit\_price \* o.quantity as amount\_charged

from orders o

join customers c on o.customer\_id = c.customer\_id

join products p on o.productid = p.product\_id)

loop

dbms\_output.put\_line(order\_rec.customer\_name || ' , ' || order\_rec.product\_name || ' , ' || order\_rec.quantity || ', ' || order\_rec.amount\_charged);

v\_totalorders := v\_totalorders + 1;

v\_totalamount := v\_totalamount + order\_rec.amount\_charged;

end loop;

dbms\_output.put\_line('Grand total:  ' || v\_totalorders || ' orders, $' || v\_totalamount);

end;

/

exec Show\_Orders();

---mem 4

CREATE OR REPLACE PROCEDURE Invoice\_Customer (

  p\_orderid IN  number,

  p\_customerid IN  number,

  p\_credit\_card IN  number,

  p\_amount IN  NUMBER

) AS

 d invoice %rowtype;

 begin

 d.invoiceid:=seq.nextval;

 d.orderid:=p\_orderid;

 d.customer\_id:=p\_customerid;

 d.creditcard#:=p\_credit\_card;

 d.amount:=p\_amount;

  -- insert the invoice record

  INSERT INTO invoice values d;

   END;

   /

 select \* from orders

exec Invoice\_Customer(1, 3, 123456780,1000);

CREATE OR REPLACE PROCEDURE Report\_Best\_Customers(cutoff\_amount in number) IS

BEGIN

  FOR c IN (SELECT c.customer\_ID, c.name, SUM(Orders.quantity \* Products.unit\_Price) AS total\_spent

            FROM Orders

            JOIN Customers c ON Orders.customer\_ID = c.customer\_ID

            JOIN Products ON Orders.productID = Products.product\_ID

            GROUP BY c.customer\_ID, c.name)

  LOOP

    if c.total\_spent > cutoff\_amount then

        DBMS\_OUTPUT.PUT\_LINE(c.name || ' spent ' || c.total\_spent || ' in total.');

    end if;

  END LOOP;

END;

/

begin

Report\_Best\_Customers(100);

End;

/

--mem 5

create or replace function  FIND\_PRODUCT\_ID(p\_productname in varchar)

return number

IS

v\_Product\_ID int ;

Begin

select Product\_ID  into v\_Product\_ID  from  products where

product\_name = p\_productname;

return v\_Product\_ID;

dbms\_output.put\_line ('v\_Product\_ID id fetched');

EXCEPTION

when no\_data\_found then

dbms\_output.put\_line('no such product');

        return -1;

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

end;

/

create or replace procedure Add\_Review(Reviewer\_Email in reviews.Reviewer\_Email%type,

stars in reviews.Stars\_Given%type,

product\_ID in reviews.product\_ID%type,

Review\_Text in reviews.Review\_Text%type) is

 x reviews%rowtype;

 Begin

 x.Review\_ID := seq.NEXTVAL ;

 x.Reviewer\_Email := Reviewer\_Email;

 x.Stars\_Given := stars;

 x.product\_ID := product\_ID;

 x.Review\_Text := Review\_Text;

 insert into reviews values x;

end;

/

--- calling add review procedure

declare

productID int;

begin

productID := FIND\_PRODUCT\_ID('40inchTV');

Add\_Review( 'john@smith.com', 2 , productID,'not a good TV. It could be better');

Add\_Review( 'barney@abc.com', 1 ,productID, 'do not buy');

productID := FIND\_PRODUCT\_ID('80inchTV');

Add\_Review( 'barney@abc.com', 5 ,productID, 'Excellent. This is the best TV');

Add\_Review( 'mary@smith.com', 5 ,productID, 'Excellent. Best ever.');

productID := FIND\_PRODUCT\_ID('BestOfBeyonce');

Add\_Review( 'mary@smith.com', 5 ,productID, 'Excellent. Best ever.');

Add\_Review( 'z@abc.com', 4 ,productID, 'Enjoyed it. Excellent sound');

 dbms\_output.put\_line ('added reviewes');

end;

/

--operation 2

CREATE OR REPLACE PROCEDURE Buy\_Or\_Beware(limit IN INT) IS

var\_product\_name varchar(50);

limit\_variable\_1 int := limit;

limit\_variable\_2 int := limit;

BEGIN

  DBMS\_OUTPUT.PUT\_LINE('Top rated products:');

  FOR c IN (SELECT AVG(Stars\_Given) AS Average\_stars, STDDEV(Stars\_Given) as std\_dev, product\_ID  FROM Reviews GROUP BY product\_ID ORDER BY Average\_stars DESC)

  LOOP

    SELECT Product\_Name INTO var\_product\_name FROM Products WHERE product\_ID = c.product\_ID;

    DBMS\_OUTPUT.PUT\_LINE(c.Average\_stars || ' ' || c.product\_ID || ' ' || var\_product\_name|| ' ' || c.std\_dev);

    limit\_variable\_1 := limit\_variable\_1 - 1;

    IF limit\_variable\_1 = 0 THEN

      EXIT;

    END IF;

  END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Buyer Beware: Stay Away from…');

  FOR b IN (SELECT AVG(Stars\_Given) AS Average\_stars, STDDEV(Stars\_Given) as std\_dev, product\_ID

  FROM Reviews GROUP BY product\_ID ORDER BY Average\_stars )

  LOOP

    SELECT Product\_Name INTO var\_product\_name FROM Products WHERE product\_ID = b.product\_ID;

    DBMS\_OUTPUT.PUT\_LINE(b.Average\_stars || ' ' || b.product\_ID || ' ' || var\_product\_name|| ' ' || b.std\_dev);

    limit\_variable\_2 := limit\_variable\_2 - 1;

    IF limit\_variable\_2 = 0 THEN

      EXIT;

    END IF;

  END LOOP;

END;

/

begin

Buy\_Or\_Beware(2);

end;

/

FInal

/\*Team :

Murali - Member 1: Responsible for the Customers and CreditCards tables. Must create these tables.

Mouna - Member 2: Responsible for Product Categories and Products. Must create these tables.

Nikhitha - Member 3: Responsible for Orders. Must create this table.

Sushmitha - Member 4: Responsible for Invoices. Must create this table.

Ramya - Member 5: Responsible for the Reviews and Recommendations.

\*/

set serveroutput on;

drop table reviews;

drop table Recommendation;

drop table invoice;

drop table Credit\_Card;

drop table orders;

drop table Products;

drop table product\_categories;

drop table customers;

create table customers

(

    Customer\_ID integer  primary key,

    name varchar(30),

    email varchar(40),

    city varchar(30),

    state varchar(20),

    zip varchar(15)

);

create table product\_categories

(

    category\_id int,

    category\_name varchar(100),

    description varchar(150),

    primary key(category\_id));

Create  table Products

(

    Product\_ID     int primary key ,

    product\_name    varchar(30),

    quantity        number (30),

    unit\_price      float  (30),

    category\_id     int ,

    FOREIGN KEY (category\_ID) REFERENCES product\_categories(category\_id)

);

create table orders

(

    orderID int,

    Customer\_ID int,

    productID int,

    quantity int,

    primary key(orderID),

    foreign key(Customer\_ID) references customers(Customer\_ID),

    foreign key(productID) references products(product\_ID)

);

create table Credit\_Card

(

    Credit\_Card# int primary key,

    expiration\_year int,

    expiration\_month int,

Credit\_Card\_Type varchar(30),

    Customer\_ID int ,

    foreign key(Customer\_ID) references customers(Customer\_ID));

create table invoice

(

    invoiceID int,

    orderID int,

    Customer\_ID int,

    creditcard# number,

    amount number,

    primary key(invoiceID),

    foreign key(orderID) references orders(orderID),

    foreign key(Customer\_ID) references customers(Customer\_ID),

    foreign key (creditcard#) references credit\_card(credit\_card#)

);

Create table Recommendation

(

    Recommendation\_ID         int  primary key ,

    Customer\_ID               int,

    Recommended\_Product\_ID    int,

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

    FOREIGN KEY (Recommended\_Product\_ID) REFERENCES Products(Product\_ID)

);

create table reviews

(

    Review\_ID int  primary key,

    Product\_ID int,

    Reviewer\_email varchar(30),

    Stars\_Given number,

    Review\_text varchar(50),

    foreign key (Product\_ID) references Products(Product\_ID)

);

drop sequence seq;

create sequence seq start with 1;

-- Procedure to insert new customer to customer table. It takes name, email, city name, state name and zip as input.

create or replace procedure Add\_Customer(s\_name in varchar,s\_email in varchar, s\_city in varchar, s\_state in varchar,s\_zip in varchar)

is

begin

insert into Customers(customer\_id,name,email,city,state,zip) values(seq.nextval,s\_name,s\_email,s\_city,s\_state,s\_zip);

end;

/

begin

 Add\_Customer('John Smith',  'john@smith.com', 'Baltimore', 'MD', '21250');

 Add\_Customer('Mary Smith'  ,'mary@smith.com', 'Baltimore', 'MD', '21250');

 Add\_Customer('Pat Wagner' , 'pat@smith.com', 'Baltimore', 'MD', '21250');

 Add\_customer('Rajeev Kumar','rajeev@kumar.org','Columbia','SC','44250');

 end;

/

--Inserting sample data for credit\_card

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (123456789, '2021', '12', 'Visa', 1);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (987654321, '2023', '5', 'Mastercard', 2);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (123456780, '2026', '2', 'Visa', 3);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (987654323, '2022', '3', 'Mastercard', 4);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (987654322, '2021', '5', 'Mastercard', 1);

-- Procedure to print list of customers in state. Input parameter is a state. Output parameter is name, email, address, credit card number and its type.

CREATE OR REPLACE PROCEDURE Show\_all\_customers\_in\_state(s\_state IN VARCHAR)

IS

  CURSOR c1 IS

    SELECT distinct c.name, c.email, (c.city|| ' '||c.state||' ' ||c.zip) AS Address, cc.credit\_card#, cc.credit\_card\_type

    FROM customers c, credit\_card cc

    WHERE c.customer\_id = cc.customer\_id AND c.state = s\_state ;

  cust\_cnt NUMBER;

BEGIN

  SELECT COUNT(\*) INTO cust\_cnt FROM customers WHERE state = s\_state;

  IF cust\_cnt = 0 THEN

    RAISE no\_data\_found;

  END IF;

  FOR item IN c1 LOOP

    dbms\_output.put\_line(item.name||' '||item.email|| ' '||item.Address||' '||item.credit\_card#||' '||item.credit\_card\_type);

  END LOOP;

EXCEPTION

  WHEN no\_data\_found THEN

    dbms\_output.put\_line('No data for customers in the state');

END;

/

begin

Show\_all\_customers\_in\_state('MD');

end;

/

drop SEQUENCE Id\_seq;-->(Please use the same sequence name  for every table )

CREATE SEQUENCE Id\_seq

START WITH 1

INCREMENT BY 1;

-- This procedure inserts a new category to the Category table.  Input is category name and its description.

Create or replace PROCEDURE Add\_Category(

    p\_category\_name  in VARCHAR,

    P\_description in VARCHAR)

AS

BEGIN

    INSERT INTO product\_categories (category\_id,category\_name, description)

    VALUES (Id\_seq.NEXTVAL,p\_category\_name, P\_description);

    dbms\_output.put\_line ('Cy Table updated');

EXCEPTION

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

END;

/

-- helper function to find the category ID that is needed in adding a product.

create or replace function  FIND\_PRODUCT\_CATEGORY\_ID(p\_category\_name in varchar)

return number

IS

v\_category\_id number ;

Begin

select category\_id  into v\_category\_id  from  product\_categories where

category\_name = p\_category\_name;

return v\_category\_id;

dbms\_output.put\_line ('v\_category\_id id fetched');

EXCEPTION

when no\_data\_found then

dbms\_output.put\_line('no such Category');

        return -1;

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

end;

/

--Inserting data into Categoeries table

begin

 Add\_Category('Music', 'vocal and instrumental music');

 Add\_Category('Electronics','Smartphone,tablets,Computers and laptops');

end;

/

--Inserting data into product  table

begin

Add\_Product('40inchTV','Electronics', 100, 200);

 Add\_Product('50inchTV','Electronics', 100, 300);

Add\_Product('80inchTV','Electronics', 100, 1000);

Add\_Product('BestOfBeyonce','Music', 100, 20);

Add\_Product('BestOfTaylorSwift','Music', 100, 20);

 Add\_Product('BestOfEmianem','Music', 100, 20);

 Add\_Product('BestOfWeeknd','Music', 100, 20);

end;

/

--Update\_Inventory: When an order is placed, the Products table must be updated with the new remaining quantity.

--Create a procedure that will take as input the productID and the units (quantity) of

--a product in an order, and will update the inventory in the Products table to reflect the new quantity after the order.

-- Procedure to update inventory, which takes product id and quantity as input.

CREATE OR REPLACE PROCEDURE Update\_Inventory (

    p\_Product\_ID  IN int ,

    p\_quantity IN number

) AS

v\_Product\_ID int;

Invalid\_Quantity exception;

BEGIN

if p\_quantity<0 THEN RAISE Invalid\_Quantity ;

end if ;

  select Product\_ID into v\_Product\_ID from Products where Product\_ID=p\_Product\_ID

  and quantity >= p\_quantity;

    -- Update  the new remaining  quantity  into the Products table

    Update Products

    set quantity = quantity - p\_quantity

    where Product\_ID = p\_Product\_ID;

    dbms\_output.put\_line ('Products Table Updated  the new remaining  quantity');

EXCEPTION

when no\_data\_found then

  dbms\_output.put\_line('Product  or Quantity is not available ');

when Invalid\_Quantity then

       DBMS\_OUTPUT.PUT\_LINE ('INVALID INPUT FROM USER');

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

END;

/

exec Update\_Inventory(4,60);

create or replace function find\_customer\_id (p\_email in varchar) return number is v\_customerid number;

begin

select customer\_id into v\_customerid from customers where email = p\_email;

return v\_customerid;

exception

when no\_data\_found then dbms\_output.put\_line('Customer not found');

end;

/

create or replace function find\_product\_id(p\_productname in varchar) return number is v\_productid number;

begin

select product\_id into v\_productid from products where product\_name = p\_productname;

return v\_productid;

exception

when no\_data\_found then

dbms\_output.put\_line('Product not found');

end;

/

CREATE OR REPLACE PROCEDURE Invoice\_customer (

    v\_customerid NUMBER,

    v\_productid NUMBER,

    p\_quantity NUMBER,

    p\_creditcard VARCHAR2

) AS

    I\_customerid NUMBER;

    I\_productid NUMBER;

BEGIN

    dbms\_output.put\_line('--');

END;

/

drop sequence seq;

create sequence seq start with 1;

-- procedure to place an order. It takes email, product name, quantity, creditcard as input.

create or replace procedure Place\_Order (

p\_email in varchar,

p\_productname in varchar,

p\_quantity in number,

p\_creditcard in number)  AS

v\_customerid Number;

v\_productid Number;

begin

-- to get customer id

v\_customerid := find\_customer\_id(p\_email);

-- to get product id

v\_productid := find\_product\_id(p\_productname);

-- update inventory

Update\_Inventory(v\_productid, p\_quantity);

-- invoice customer

Invoice\_Customer(v\_customerid, v\_productid, p\_quantity, p\_creditcard);

-- insert order into database

Insert into Orders (orderid,customer\_id, productid, quantity) values (seq.nextval,v\_customerid, v\_productid, p\_quantity);

commit;

end;

/

exec Place\_Order('pat@smith.com','40inchTV',200,123456780);

-- procedure to display all the orders. Output is Customer name, product, quantity ordered, amount charged. Also grand total of the number of all orders placed in the system.

create or replace procedure Show\_Orders as

v\_totalorders Number := 0;

v\_totalamount Number := 0;

begin

for order\_rec in (select c.name  as customer\_name,

                  p.product\_name,

o.quantity,

p.unit\_price \* o.quantity as amount\_charged

from orders o

join customers c on o.customer\_id = c.customer\_id

join products p on o.productid = p.product\_id)

loop

dbms\_output.put\_line(order\_rec.customer\_name || ' , ' || order\_rec.product\_name || ' , ' || order\_rec.quantity || ', ' || order\_rec.amount\_charged);

v\_totalorders := v\_totalorders + 1;

v\_totalamount := v\_totalamount + order\_rec.amount\_charged;

end loop;

dbms\_output.put\_line('Grand total:  ' || v\_totalorders || ' orders, $' || v\_totalamount);

end;

/

exec Show\_Orders();

-- Procedure to enter record in invoice table. Input is order ID, Customer ID, Credit Card #, Amount.

CREATE OR REPLACE PROCEDURE Invoice\_Customer (

  p\_orderid IN  number,

  p\_customerid IN  number,

  p\_credit\_card IN  number,

  p\_amount IN  NUMBER

) AS

 d invoice %rowtype;

 begin

 d.invoiceid:=seq.nextval;

 d.orderid:=p\_orderid;

 d.customer\_id:=p\_customerid;

 d.creditcard#:=p\_credit\_card;

 d.amount:=p\_amount;

  -- insert the invoice record

  INSERT INTO invoice values d;

   END;

   /

 select \* from orders

exec Invoice\_Customer(1, 3, 123456780,1000);

-- Procedure to show best customer. Input is dollar amount and will produce a report of all customers (names, and total amount spent) who have spent more than the input amount in the online store.

CREATE OR REPLACE PROCEDURE Report\_Best\_Customers(cutoff\_amount in number) IS

BEGIN

  FOR c IN (SELECT c.customer\_ID, c.name, SUM(Orders.quantity \* Products.unit\_Price) AS total\_spent

            FROM Orders

            JOIN Customers c ON Orders.customer\_ID = c.customer\_ID

            JOIN Products ON Orders.productID = Products.product\_ID

            GROUP BY c.customer\_ID, c.name)

  LOOP

    if c.total\_spent > cutoff\_amount then

        DBMS\_OUTPUT.PUT\_LINE(c.name || ' spent ' || c.total\_spent || ' in total.');

    end if;

  END LOOP;

END;

/

begin

Report\_Best\_Customers(100);

End;

/

create or replace function  FIND\_PRODUCT\_ID(p\_productname in varchar)

return number

IS

v\_Product\_ID int ;

Begin

select Product\_ID  into v\_Product\_ID  from  products where

product\_name = p\_productname;

return v\_Product\_ID;

dbms\_output.put\_line ('v\_Product\_ID id fetched');

EXCEPTION

when no\_data\_found then

dbms\_output.put\_line('no such product');

        return -1;

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

end;

/

-- Procedure to add review. Input to this procedure is the reviewer email, stars given (a number from 1-5), product ID, and review text.

create or replace procedure Add\_Review(Reviewer\_Email in reviews.Reviewer\_Email%type,

stars in reviews.Stars\_Given%type,

product\_ID in reviews.product\_ID%type,

Review\_Text in reviews.Review\_Text%type) is

 x reviews%rowtype;

 Begin

 x.Review\_ID := seq.NEXTVAL ;

 x.Reviewer\_Email := Reviewer\_Email;

 x.Stars\_Given := stars;

 x.product\_ID := product\_ID;

 x.Review\_Text := Review\_Text;

 insert into reviews values x;

end;

/

--- calling add review procedure

declare

productID int;

begin

productID := FIND\_PRODUCT\_ID('40inchTV');

Add\_Review( 'john@smith.com', 2 , productID,'not a good TV. It could be better');

Add\_Review( 'barney@abc.com', 1 ,productID, 'do not buy');

productID := FIND\_PRODUCT\_ID('80inchTV');

Add\_Review( 'barney@abc.com', 5 ,productID, 'Excellent. This is the best TV');

Add\_Review( 'mary@smith.com', 5 ,productID, 'Excellent. Best ever.');

productID := FIND\_PRODUCT\_ID('BestOfBeyonce');

Add\_Review( 'mary@smith.com', 5 ,productID, 'Excellent. Best ever.');

Add\_Review( 'z@abc.com', 4 ,productID, 'Enjoyed it. Excellent sound');

 dbms\_output.put\_line ('added reviewes');

end;

/

-- Procedure to show if an item could be purchased or not.This procedure takes as input a number X and prints the best X products and the worst X products based on the average star ratings of customers who wrote a review for each product.  It will display ‘Top rated products’ then for each line showing the average number of stars (in descending order), product ID, product name, and the standard deviation. Then print ‘Buyer Beware: Stay Away from…” then each consecutive line showing the average number of stars (in ascending order), product ID, product name, and the standard deviation.

CREATE OR REPLACE PROCEDURE Buy\_Or\_Beware(limit IN INT) IS

var\_product\_name varchar(50);

limit\_variable\_1 int := limit;

limit\_variable\_2 int := limit;

BEGIN

  DBMS\_OUTPUT.PUT\_LINE('Top rated products:');

  FOR c IN (SELECT AVG(Stars\_Given) AS Average\_stars, STDDEV(Stars\_Given) as std\_dev, product\_ID  FROM Reviews GROUP BY product\_ID ORDER BY Average\_stars DESC)

  LOOP

    SELECT Product\_Name INTO var\_product\_name FROM Products WHERE product\_ID = c.product\_ID;

    DBMS\_OUTPUT.PUT\_LINE(c.Average\_stars || ' ' || c.product\_ID || ' ' || var\_product\_name|| ' ' || c.std\_dev);

    limit\_variable\_1 := limit\_variable\_1 - 1;

    IF limit\_variable\_1 = 0 THEN

      EXIT;

    END IF;

  END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Buyer Beware: Stay Away from…');

  FOR b IN (SELECT AVG(Stars\_Given) AS Average\_stars, STDDEV(Stars\_Given) as std\_dev, product\_ID

  FROM Reviews GROUP BY product\_ID ORDER BY Average\_stars )

  LOOP

    SELECT Product\_Name INTO var\_product\_name FROM Products WHERE product\_ID = b.product\_ID;

    DBMS\_OUTPUT.PUT\_LINE(b.Average\_stars || ' ' || b.product\_ID || ' ' || var\_product\_name|| ' ' || b.std\_dev);

    limit\_variable\_2 := limit\_variable\_2 - 1;

    IF limit\_variable\_2 = 0 THEN

      EXIT;

    END IF;

  END LOOP;

END;

/

begin

Buy\_Or\_Beware(2);

end;

/

select \* from customers;

select \* from product\_categories;

select \* from Products;

select \* from orders;

select \* from invoice;

select \* from reviews;

—----------22

set serveroutput on;

drop table reviews;

drop table Recommendation;

drop table invoice;

drop table Credit\_Card;

drop table orders;

drop table Products;

drop table product\_categories;

drop table customers;

create table customers

(

    Customer\_ID integer  primary key,

    name varchar(30),

    email varchar(40),

    city varchar(30),

    state varchar(20),

    zip varchar(15)

);

create table product\_categories

(

    category\_id int,

    category\_name varchar(100),

    description varchar(150),

    primary key(category\_id));

Create  table Products

(

    Product\_ID     int primary key ,

    product\_name    varchar(30),

    quantity        number (30),

    unit\_price      float  (30),

    category\_id     int ,

    FOREIGN KEY (category\_ID) REFERENCES product\_categories(category\_id)

);

create table orders

(

    orderID int,

    Customer\_ID int,

    productID int,

    quantity int,

    primary key(orderID),

    foreign key(Customer\_ID) references customers(Customer\_ID),

    foreign key(productID) references products(product\_ID)

);

create table Credit\_Card

(

    Credit\_Card# int primary key,

    expiration\_year int,

    expiration\_month int,

Credit\_Card\_Type varchar(30),

    Customer\_ID int ,

    foreign key(Customer\_ID) references customers(Customer\_ID));

create table invoice

(

    invoiceID int,

    orderID int,

    Customer\_ID int,

    creditcard# number,

    amount number,

    primary key(invoiceID),

    foreign key(orderID) references orders(orderID),

    foreign key(Customer\_ID) references customers(Customer\_ID),

    foreign key (creditcard#) references credit\_card(credit\_card#)

);

--

Create table Recommendation

(

    Recommendation\_ID         int  primary key ,

    Customer\_ID               int,

    Recommended\_Product\_ID    int,

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

    FOREIGN KEY (Recommended\_Product\_ID) REFERENCES Products(Product\_ID)

);

create table reviews

(

    Review\_ID int  primary key,

    Product\_ID int,

    Reviewer\_email varchar(30),

    Stars\_Given number,

    Review\_text varchar(50),

    foreign key (Product\_ID) references Products(Product\_ID)

);

---mem1

drop sequence seq;

create sequence seq start with 1;

--1.

create or replace procedure Add\_Customer(s\_name in varchar,s\_email in varchar, s\_city in varchar, s\_state in varchar,s\_zip in varchar)

is

begin

insert into Customers(customer\_id,name,email,city,state,zip) values(seq.nextval,s\_name,s\_email,s\_city,s\_state,s\_zip);

end;

/

--

begin

 Add\_Customer('John Smith',  'john@smith.com', 'Baltimore', 'MD', '21250');

 Add\_Customer('Mary Smith'  ,'mary@smith.com', 'Baltimore', 'MD', '21250');

 Add\_Customer('Pat Wagner' , 'pat@smith.com', 'Baltimore', 'MD', '21250');

 Add\_customer('Rajeev Kumar','rajeev@kumar.org','Columbia','SC','44250');

 end;

/

--Inserting sample data for credit\_card

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (123456789, '2021', '12', 'Visa', 1);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (987654321, '2023', '5', 'Mastercard', 2);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (123456780, '2026', '2', 'Visa', 3);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (987654323, '2022', '3', 'Mastercard', 4);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (987654322, '2021', '5', 'Mastercard', 1);

--2.Show\_all\_customers\_in\_state: Input parameter is a state. Print a list of all customers in that state including customer name, email, address, credit card numbers, and type.

CREATE OR REPLACE PROCEDURE Show\_all\_customers\_in\_state(s\_state IN VARCHAR)

IS

  CURSOR c1 IS

    SELECT distinct c.name, c.email, (c.city|| ' '||c.state||' ' ||c.zip) AS Address, cc.credit\_card#, cc.credit\_card\_type

    FROM customers c, credit\_card cc

    WHERE c.customer\_id = cc.customer\_id AND c.state = s\_state ;

  cust\_cnt NUMBER;

BEGIN

  SELECT COUNT(\*) INTO cust\_cnt FROM customers WHERE state = s\_state;

  IF cust\_cnt = 0 THEN

    RAISE no\_data\_found;

  END IF;

  FOR item IN c1 LOOP

    dbms\_output.put\_line(item.name||' '||item.email|| ' '||item.Address||' '||item.credit\_card#||' '||item.credit\_card\_type);

  END LOOP;

EXCEPTION

  WHEN no\_data\_found THEN

    dbms\_output.put\_line('No data for customers in the state');

END;

/

begin

Show\_all\_customers\_in\_state('MD');

end;

/

--mem 2

drop SEQUENCE Id\_seq;-->(Please use the same sequence name  for every table )

CREATE SEQUENCE Id\_seq

START WITH 1

INCREMENT BY 1;

--Add\_Category: This procedure takes as input all necessary information to  insert a new category to the Category table.

Create or replace PROCEDURE Add\_Category(

    p\_category\_name  in VARCHAR,

    P\_description in VARCHAR)

AS

BEGIN

    INSERT INTO product\_categories (category\_id,category\_name, description)

    VALUES (Id\_seq.NEXTVAL,p\_category\_name, P\_description);

    dbms\_output.put\_line ('Cy Table updated');

EXCEPTION

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

END;

/

-- helper function to find the category ID that is needed in adding a product.

create or replace function  FIND\_PRODUCT\_CATEGORY\_ID(p\_category\_name in varchar)

return number

IS

v\_category\_id number ;

Begin

select category\_id  into v\_category\_id  from  product\_categories where

category\_name = p\_category\_name;

return v\_category\_id;

dbms\_output.put\_line ('v\_category\_id id fetched');

EXCEPTION

when no\_data\_found then

dbms\_output.put\_line('no such Category');

        return -1;

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

end;

/

--Inserting data into Categoeries table

begin

 Add\_Category('Music', 'vocal and instrumental music');

 Add\_Category('Electronics','Smartphone,tablets,Computers and laptops');

end;

/

--Inserting data into product  table

begin

Add\_Product('40inchTV','Electronics', 100, 200);

 Add\_Product('50inchTV','Electronics', 100, 300);

Add\_Product('80inchTV','Electronics', 100, 1000);

Add\_Product('BestOfBeyonce','Music', 100, 20);

Add\_Product('BestOfTaylorSwift','Music', 100, 20);

 Add\_Product('BestOfEmianem','Music', 100, 20);

 Add\_Product('BestOfWeeknd','Music', 100, 20);

end;

/

--Update\_Inventory: When an order is placed, the Products table must be updated with the new remaining quantity.

--Create a procedure that will take as input the productID and the units (quantity) of

--a product in an order, and will update the inventory in the Products table to reflect the new quantity after the order.

CREATE OR REPLACE PROCEDURE Update\_Inventory (

    p\_Product\_ID  IN int ,

    p\_quantity IN number

) AS

v\_Product\_ID int;

Invalid\_Quantity exception;

BEGIN

if p\_quantity<0 THEN RAISE Invalid\_Quantity ;

end if ;

  select Product\_ID into v\_Product\_ID from Products where Product\_ID=p\_Product\_ID

  and quantity >= p\_quantity;

    -- Update  the new remaining  quantity  into the Products table

    Update Products

    set quantity = quantity - p\_quantity

    where Product\_ID = p\_Product\_ID;

    dbms\_output.put\_line ('Products Table Updated  the new remaining  quantity');

EXCEPTION

when no\_data\_found then

  dbms\_output.put\_line('Product  or Quantity is not available ');

when Invalid\_Quantity then

       DBMS\_OUTPUT.PUT\_LINE ('INVALID INPUT FROM USER');

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

END;

/

exec Update\_Inventory(4,60);

--nikhitha

-- procedure to place an order

create or replace function find\_customer\_id (p\_email in varchar) return number is v\_customerid number;

begin

select customer\_id into v\_customerid from customers where email = p\_email;

return v\_customerid;

exception

when no\_data\_found then dbms\_output.put\_line('Customer not found');

end;

/

create or replace function find\_product\_id(p\_productname in varchar) return number is v\_productid number;

begin

select product\_id into v\_productid from products where product\_name = p\_productname;

return v\_productid;

exception

when no\_data\_found then

dbms\_output.put\_line('Product not found');

end;

/

CREATE OR REPLACE PROCEDURE Invoice\_customer (

    v\_customerid NUMBER,

    v\_productid NUMBER,

    p\_quantity NUMBER,

    p\_creditcard VARCHAR2

) AS

    I\_customerid NUMBER;

    I\_productid NUMBER;

BEGIN

    dbms\_output.put\_line('--');

END;

/

drop sequence seq;

create sequence seq start with 1;

create or replace procedure Place\_Order (

p\_email in varchar,

p\_productname in varchar,

p\_quantity in number,

p\_creditcard in number)  AS

v\_customerid Number;

v\_productid Number;

begin

-- to get customer id

v\_customerid := find\_customer\_id(p\_email);

-- to get product id

v\_productid := find\_product\_id(p\_productname);

-- update inventory

Update\_Inventory(v\_productid, p\_quantity);

-- invoice customer

Invoice\_Customer(v\_customerid, v\_productid, p\_quantity, p\_creditcard);

-- insert order into database

Insert into Orders (orderid,customer\_id, productid, quantity) values (seq.nextval,v\_customerid, v\_productid, p\_quantity);

commit;

end;

/

exec Place\_Order('pat@smith.com','40inchTV',200,123456780);

-- procedure to show all orders

create or replace procedure Show\_Orders as

v\_totalorders Number := 0;

v\_totalamount Number := 0;

begin

for order\_rec in (select c.name  as customer\_name,

                  p.product\_name,

o.quantity,

p.unit\_price \* o.quantity as amount\_charged

from orders o

join customers c on o.customer\_id = c.customer\_id

join products p on o.productid = p.product\_id)

loop

dbms\_output.put\_line(order\_rec.customer\_name || ' , ' || order\_rec.product\_name || ' , ' || order\_rec.quantity || ', ' || order\_rec.amount\_charged);

v\_totalorders := v\_totalorders + 1;

v\_totalamount := v\_totalamount + order\_rec.amount\_charged;

end loop;

dbms\_output.put\_line('Grand total:  ' || v\_totalorders || ' orders, $' || v\_totalamount);

end;

/

exec Show\_Orders();

---mem 4

CREATE OR REPLACE PROCEDURE Invoice\_Customer (

  p\_orderid IN  number,

  p\_customerid IN  number,

  p\_credit\_card IN  number,

  p\_amount IN  NUMBER

) AS

 d invoice %rowtype;

 begin

 d.invoiceid:=seq.nextval;

 d.orderid:=p\_orderid;

 d.customer\_id:=p\_customerid;

 d.creditcard#:=p\_credit\_card;

 d.amount:=p\_amount;

  -- insert the invoice record

  INSERT INTO invoice values d;

   END;

   /

--need to write calling function

CREATE OR REPLACE PROCEDURE Report\_Best\_Customers(cutoff\_amount in number) IS

BEGIN

  FOR c IN (SELECT c.customer\_ID, c.name, SUM(Orders.quantity \* Products.unit\_Price) AS total\_spent

            FROM Orders

            JOIN Customers c ON Orders.customer\_ID = c.customer\_ID

            JOIN Products ON Orders.productID = Products.product\_ID

            GROUP BY c.customer\_ID, c.name)

  LOOP

    if c.total\_spent > cutoff\_amount then

        DBMS\_OUTPUT.PUT\_LINE(c.name || ' spent ' || c.total\_spent || ' in total.');

    end if;

  END LOOP;

END;

/

begin

Report\_Best\_Customers(100);

End;

/

--mem 5

create or replace function  FIND\_PRODUCT\_ID(p\_productname in varchar)

return number

IS

v\_Product\_ID int ;

Begin

select Product\_ID  into v\_Product\_ID  from  products where

product\_name = p\_productname;

return v\_Product\_ID;

dbms\_output.put\_line ('v\_Product\_ID id fetched');

EXCEPTION

when no\_data\_found then

dbms\_output.put\_line('no such product');

        return -1;

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

end;

/

create or replace procedure Add\_Review(Reviewer\_Email in reviews.Reviewer\_Email%type,

stars in reviews.Stars\_Given%type,

product\_ID in reviews.product\_ID%type,

Review\_Text in reviews.Review\_Text%type) is

 x reviews%rowtype;

 Begin

 x.Review\_ID := seq.NEXTVAL ;

 x.Reviewer\_Email := Reviewer\_Email;

 x.Stars\_Given := stars;

 x.product\_ID := product\_ID;

 x.Review\_Text := Review\_Text;

 insert into reviews values x;

end;

/

--- calling add review procedure

declare

productID int;

begin

productID := FIND\_PRODUCT\_ID('40inchTV');

Add\_Review( 'john@smith.com', 2 , productID,'not a good TV. It could be better');

Add\_Review( 'barney@abc.com', 1 ,productID, 'do not buy');

productID := FIND\_PRODUCT\_ID('80inchTV');

Add\_Review( 'barney@abc.com', 5 ,productID, 'Excellent. This is the best TV');

Add\_Review( 'mary@smith.com', 5 ,productID, 'Excellent. Best ever.');

productID := FIND\_PRODUCT\_ID('BestOfBeyonce');

Add\_Review( 'mary@smith.com', 5 ,productID, 'Excellent. Best ever.');

Add\_Review( 'z@abc.com', 4 ,productID, 'Enjoyed it. Excellent sound');

 dbms\_output.put\_line ('added reviewes');

end;

/

--operation 2

CREATE OR REPLACE PROCEDURE Buy\_Or\_Beware(limit IN INT) IS

var\_product\_name varchar(50);

limit\_variable\_1 int := limit;

limit\_variable\_2 int := limit;

BEGIN

  DBMS\_OUTPUT.PUT\_LINE('Top rated products:');

  FOR c IN (SELECT AVG(Stars\_Given) AS Average\_stars, STDDEV(Stars\_Given) as std\_dev, product\_ID  FROM Reviews GROUP BY product\_ID ORDER BY Average\_stars DESC)

  LOOP

    SELECT Product\_Name INTO var\_product\_name FROM Products WHERE product\_ID = c.product\_ID;

    DBMS\_OUTPUT.PUT\_LINE(c.Average\_stars || ' ' || c.product\_ID || ' ' || var\_product\_name|| ' ' || c.std\_dev);

    limit\_variable\_1 := limit\_variable\_1 - 1;

    IF limit\_variable\_1 = 0 THEN

      EXIT;

    END IF;

  END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Buyer Beware: Stay Away from…');

  FOR b IN (SELECT AVG(Stars\_Given) AS Average\_stars, STDDEV(Stars\_Given) as std\_dev, product\_ID

  FROM Reviews GROUP BY product\_ID ORDER BY Average\_stars )

  LOOP

    SELECT Product\_Name INTO var\_product\_name FROM Products WHERE product\_ID = b.product\_ID;

    DBMS\_OUTPUT.PUT\_LINE(b.Average\_stars || ' ' || b.product\_ID || ' ' || var\_product\_name|| ' ' || b.std\_dev);

    limit\_variable\_2 := limit\_variable\_2 - 1;

    IF limit\_variable\_2 = 0 THEN

      EXIT;

    END IF;

  END LOOP;

END;

/

begin

Buy\_Or\_Beware(2);

end;

/

set serveroutput on;

drop table reviews;

drop table Recommendation;

drop table invoice;

drop table Credit\_Card;

drop table orders;

drop table Products;

drop table product\_categories;

drop table customers;

create table customers

(

    Customer\_ID integer  primary key,

    name varchar(30),

    email varchar(40),

    city varchar(30),

    state varchar(20),

    zip varchar(15)

);

create table product\_categories

(

    category\_id int,

    category\_name varchar(100),

    description varchar(150),

    primary key(category\_id));

Create  table Products

(

    Product\_ID     int primary key ,

    product\_name    varchar(30),

    quantity        number (30),

    unit\_price      float  (30),

    category\_id     int ,

    FOREIGN KEY (category\_ID) REFERENCES product\_categories(category\_id)

);

create table orders

(

    orderID int,

    Customer\_ID int,

    productID int,

    quantity int,

    primary key(orderID),

    foreign key(Customer\_ID) references customers(Customer\_ID),

    foreign key(productID) references products(product\_ID)

);

create table Credit\_Card

(

    Credit\_Card# int primary key,

    expiration\_year int,

    expiration\_month int,

Credit\_Card\_Type varchar(30),

    Customer\_ID int ,

    foreign key(Customer\_ID) references customers(Customer\_ID));

create table invoice

(

    invoiceID int,

    orderID int,

    Customer\_ID int,

    creditcard# number,

    amount number,

    primary key(invoiceID),

    foreign key(orderID) references orders(orderID),

    foreign key(Customer\_ID) references customers(Customer\_ID),

    foreign key (creditcard#) references credit\_card(credit\_card#)

);

--

Create table Recommendation

(

    Recommendation\_ID         int  primary key ,

    Customer\_ID               int,

    Recommended\_Product\_ID    int,

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

    FOREIGN KEY (Recommended\_Product\_ID) REFERENCES Products(Product\_ID)

);

create table reviews

(

    Review\_ID int  primary key,

    Product\_ID int,

    Reviewer\_email varchar(30),

    Stars\_Given number,

    Review\_text varchar(50),

    foreign key (Product\_ID) references Products(Product\_ID)

);

---mem1

drop sequence seq;

create sequence seq start with 1;

--1.

create or replace procedure Add\_Customer(s\_name in varchar,s\_email in varchar, s\_city in varchar, s\_state in varchar,s\_zip in varchar)

is

begin

insert into Customers(customer\_id,name,email,city,state,zip) values(seq.nextval,s\_name,s\_email,s\_city,s\_state,s\_zip);

end;

/

--

begin

 Add\_Customer('John Smith',  'john@smith.com', 'Baltimore', 'MD', '21250');

 Add\_Customer('Mary Smith'  ,'mary@smith.com', 'Baltimore', 'MD', '21250');

 Add\_Customer('Pat Wagner' , 'pat@smith.com', 'Baltimore', 'MD', '21250');

 Add\_customer('Rajeev Kumar','rajeev@kumar.org','Columbia','SC','44250');

 end;

/

--Inserting sample data for credit\_card

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (123456789, '2021', '12', 'Visa', 1);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (987654321, '2023', '5', 'Mastercard', 2);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (123456780, '2026', '2', 'Visa', 3);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (987654323, '2022', '3', 'Mastercard', 4);

insert into Credit\_Card (Credit\_Card#, expiration\_year, expiration\_month, Credit\_Card\_Type, Customer\_ID)

values (987654322, '2021', '5', 'Mastercard', 1);

--2.Show\_all\_customers\_in\_state: Input parameter is a state. Print a list of all customers in that state including customer name, email, address, credit card numbers, and type.

CREATE OR REPLACE PROCEDURE Show\_all\_customers\_in\_state(s\_state IN VARCHAR)

IS

  CURSOR c1 IS

    SELECT distinct c.name, c.email, (c.city|| ' '||c.state||' ' ||c.zip) AS Address, cc.credit\_card#, cc.credit\_card\_type

    FROM customers c, credit\_card cc

    WHERE c.customer\_id = cc.customer\_id AND c.state = s\_state ;

  cust\_cnt NUMBER;

BEGIN

  SELECT COUNT(\*) INTO cust\_cnt FROM customers WHERE state = s\_state;

  IF cust\_cnt = 0 THEN

    RAISE no\_data\_found;

  END IF;

  FOR item IN c1 LOOP

    dbms\_output.put\_line(item.name||' '||item.email|| ' '||item.Address||' '||item.credit\_card#||' '||item.credit\_card\_type);

  END LOOP;

EXCEPTION

  WHEN no\_data\_found THEN

    dbms\_output.put\_line('No data for customers in the state');

END;

/

begin

Show\_all\_customers\_in\_state('MD');

end;

/

--mem 2

drop SEQUENCE Id\_seq;-->(Please use the same sequence name  for every table )

CREATE SEQUENCE Id\_seq

START WITH 1

INCREMENT BY 1;

--Add\_Category: This procedure takes as input all necessary information to  insert a new category to the Category table.

Create or replace PROCEDURE Add\_Category(

    p\_category\_name  in VARCHAR,

    P\_description in VARCHAR)

AS

BEGIN

    INSERT INTO product\_categories (category\_id,category\_name, description)

    VALUES (Id\_seq.NEXTVAL,p\_category\_name, P\_description);

    dbms\_output.put\_line ('Cy Table updated');

EXCEPTION

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

END;

/

-- helper function to find the category ID that is needed in adding a product.

create or replace function  FIND\_PRODUCT\_CATEGORY\_ID(p\_category\_name in varchar)

return number

IS

v\_category\_id number ;

Begin

select category\_id  into v\_category\_id  from  product\_categories where

category\_name = p\_category\_name;

return v\_category\_id;

dbms\_output.put\_line ('v\_category\_id id fetched');

EXCEPTION

when no\_data\_found then

dbms\_output.put\_line('no such Category');

        return -1;

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

end;

/

--Inserting data into Categoeries table

begin

 Add\_Category('Music', 'vocal and instrumental music');

 Add\_Category('Electronics','Smartphone,tablets,Computers and laptops');

end;

/

--Inserting data into product  table

begin

Add\_Product('40inchTV','Electronics', 100, 200);

 Add\_Product('50inchTV','Electronics', 100, 300);

Add\_Product('80inchTV','Electronics', 100, 1000);

Add\_Product('BestOfBeyonce','Music', 100, 20);

Add\_Product('BestOfTaylorSwift','Music', 100, 20);

 Add\_Product('BestOfEmianem','Music', 100, 20);

 Add\_Product('BestOfWeeknd','Music', 100, 20);

end;

/

--Update\_Inventory: When an order is placed, the Products table must be updated with the new remaining quantity.

--Create a procedure that will take as input the productID and the units (quantity) of

--a product in an order, and will update the inventory in the Products table to reflect the new quantity after the order.

CREATE OR REPLACE PROCEDURE Update\_Inventory (

    p\_Product\_ID  IN int ,

    p\_quantity IN number

) AS

v\_Product\_ID int;

Invalid\_Quantity exception;

BEGIN

if p\_quantity<0 THEN RAISE Invalid\_Quantity ;

end if ;

  select Product\_ID into v\_Product\_ID from Products where Product\_ID=p\_Product\_ID

  and quantity >= p\_quantity;

    -- Update  the new remaining  quantity  into the Products table

    Update Products

    set quantity = quantity - p\_quantity

    where Product\_ID = p\_Product\_ID;

    dbms\_output.put\_line ('Products Table Updated  the new remaining  quantity');

EXCEPTION

when no\_data\_found then

  dbms\_output.put\_line('Product  or Quantity is not available ');

when Invalid\_Quantity then

       DBMS\_OUTPUT.PUT\_LINE ('INVALID INPUT FROM USER');

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

END;

/

exec Update\_Inventory(4,60);

--nikhitha

-- procedure to place an order

create or replace function find\_customer\_id (p\_email in varchar) return number is v\_customerid number;

begin

select customer\_id into v\_customerid from customers where email = p\_email;

return v\_customerid;

exception

when no\_data\_found then dbms\_output.put\_line('Customer not found');

end;

/

create or replace function find\_product\_id(p\_productname in varchar) return number is v\_productid number;

begin

select product\_id into v\_productid from products where product\_name = p\_productname;

return v\_productid;

exception

when no\_data\_found then

dbms\_output.put\_line('Product not found');

end;

/

CREATE OR REPLACE PROCEDURE Invoice\_customer (

    v\_customerid NUMBER,

    v\_productid NUMBER,

    p\_quantity NUMBER,

    p\_creditcard VARCHAR2

) AS

    I\_customerid NUMBER;

    I\_productid NUMBER;

BEGIN

    dbms\_output.put\_line('--');

END;

/

drop sequence seq;

create sequence seq start with 1;

create or replace procedure Place\_Order (

p\_email in varchar,

p\_productname in varchar,

p\_quantity in number,

p\_creditcard in number)  AS

v\_customerid Number;

v\_productid Number;

begin

-- to get customer id

v\_customerid := find\_customer\_id(p\_email);

-- to get product id

v\_productid := find\_product\_id(p\_productname);

-- update inventory

Update\_Inventory(v\_productid, p\_quantity);

-- invoice customer

Invoice\_Customer(v\_customerid, v\_productid, p\_quantity, p\_creditcard);

-- insert order into database

Insert into Orders (orderid,customer\_id, productid, quantity) values (seq.nextval,v\_customerid, v\_productid, p\_quantity);

commit;

end;

/

exec Place\_Order('pat@smith.com','40inchTV',200,123456780);

-- procedure to show all orders

create or replace procedure Show\_Orders as

v\_totalorders Number := 0;

v\_totalamount Number := 0;

begin

for order\_rec in (select c.name  as customer\_name,

                  p.product\_name,

o.quantity,

p.unit\_price \* o.quantity as amount\_charged

from orders o

join customers c on o.customer\_id = c.customer\_id

join products p on o.productid = p.product\_id)

loop

dbms\_output.put\_line(order\_rec.customer\_name || ' , ' || order\_rec.product\_name || ' , ' || order\_rec.quantity || ', ' || order\_rec.amount\_charged);

v\_totalorders := v\_totalorders + 1;

v\_totalamount := v\_totalamount + order\_rec.amount\_charged;

end loop;

dbms\_output.put\_line('Grand total:  ' || v\_totalorders || ' orders, $' || v\_totalamount);

end;

/

exec Show\_Orders();

---mem 4

CREATE OR REPLACE PROCEDURE Invoice\_Customer (

  p\_orderid IN  number,

  p\_customerid IN  number,

  p\_credit\_card IN  number,

  p\_amount IN  NUMBER

) AS

 d invoice %rowtype;

 begin

 d.invoiceid:=seq.nextval;

 d.orderid:=p\_orderid;

 d.customer\_id:=p\_customerid;

 d.creditcard#:=p\_credit\_card;

 d.amount:=p\_amount;

  -- insert the invoice record

  INSERT INTO invoice values d;

   END;

   /

--need to write calling function

CREATE OR REPLACE PROCEDURE Report\_Best\_Customers(cutoff\_amount in number) IS

BEGIN

  FOR c IN (SELECT c.customer\_ID, c.name, SUM(Orders.quantity \* Products.unit\_Price) AS total\_spent

            FROM Orders

            JOIN Customers c ON Orders.customer\_ID = c.customer\_ID

            JOIN Products ON Orders.productID = Products.product\_ID

            GROUP BY c.customer\_ID, c.name)

  LOOP

    if c.total\_spent > cutoff\_amount then

        DBMS\_OUTPUT.PUT\_LINE(c.name || ' spent ' || c.total\_spent || ' in total.');

    end if;

  END LOOP;

END;

/

begin

Report\_Best\_Customers(100);

End;

/

--mem 5

create or replace function  FIND\_PRODUCT\_ID(p\_productname in varchar)

return number

IS

v\_Product\_ID int ;

Begin

select Product\_ID  into v\_Product\_ID  from  products where

product\_name = p\_productname;

return v\_Product\_ID;

dbms\_output.put\_line ('v\_Product\_ID id fetched');

EXCEPTION

when no\_data\_found then

dbms\_output.put\_line('no such product');

        return -1;

when others then

  dbms\_output.put\_line ('SQLCODE: ' || SQLCODE);

  dbms\_output.put\_line ('SQLERRM: ' || SQLERRM);

end;

/

create or replace procedure Add\_Review(Reviewer\_Email in reviews.Reviewer\_Email%type,

stars in reviews.Stars\_Given%type,

product\_ID in reviews.product\_ID%type,

Review\_Text in reviews.Review\_Text%type) is

 x reviews%rowtype;

 Begin

 x.Review\_ID := seq.NEXTVAL ;

 x.Reviewer\_Email := Reviewer\_Email;

 x.Stars\_Given := stars;

 x.product\_ID := product\_ID;

 x.Review\_Text := Review\_Text;

 insert into reviews values x;

end;

/

--- calling add review procedure

declare

productID int;

begin

productID := FIND\_PRODUCT\_ID('40inchTV');

Add\_Review( 'john@smith.com', 2 , productID,'not a good TV. It could be better');

Add\_Review( 'barney@abc.com', 1 ,productID, 'do not buy');

productID := FIND\_PRODUCT\_ID('80inchTV');

Add\_Review( 'barney@abc.com', 5 ,productID, 'Excellent. This is the best TV');

Add\_Review( 'mary@smith.com', 5 ,productID, 'Excellent. Best ever.');

productID := FIND\_PRODUCT\_ID('BestOfBeyonce');

Add\_Review( 'mary@smith.com', 5 ,productID, 'Excellent. Best ever.');

Add\_Review( 'z@abc.com', 4 ,productID, 'Enjoyed it. Excellent sound');

 dbms\_output.put\_line ('added reviewes');

end;

/

--operation 2

CREATE OR REPLACE PROCEDURE Buy\_Or\_Beware(limit IN INT) IS

var\_product\_name varchar(50);

limit\_variable\_1 int := limit;

limit\_variable\_2 int := limit;

BEGIN

  DBMS\_OUTPUT.PUT\_LINE('Top rated products:');

  FOR c IN (SELECT AVG(Stars\_Given) AS Average\_stars, STDDEV(Stars\_Given) as std\_dev, product\_ID  FROM Reviews GROUP BY product\_ID ORDER BY Average\_stars DESC)

  LOOP

    SELECT Product\_Name INTO var\_product\_name FROM Products WHERE product\_ID = c.product\_ID;

    DBMS\_OUTPUT.PUT\_LINE(c.Average\_stars || ' ' || c.product\_ID || ' ' || var\_product\_name|| ' ' || c.std\_dev);

    limit\_variable\_1 := limit\_variable\_1 - 1;

    IF limit\_variable\_1 = 0 THEN

      EXIT;

    END IF;

  END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Buyer Beware: Stay Away from…');

  FOR b IN (SELECT AVG(Stars\_Given) AS Average\_stars, STDDEV(Stars\_Given) as std\_dev, product\_ID

  FROM Reviews GROUP BY product\_ID ORDER BY Average\_stars )

  LOOP

    SELECT Product\_Name INTO var\_product\_name FROM Products WHERE product\_ID = b.product\_ID;

    DBMS\_OUTPUT.PUT\_LINE(b.Average\_stars || ' ' || b.product\_ID || ' ' || var\_product\_name|| ' ' || b.std\_dev);

    limit\_variable\_2 := limit\_variable\_2 - 1;

    IF limit\_variable\_2 = 0 THEN

      EXIT;

    END IF;

  END LOOP;

END;

/

begin

Buy\_Or\_Beware(2);

end;

/

Deliverable 3

Member 1

 create or replace function get\_customerid(s\_customer\_id number) return number

 IS

 v\_res number;

 begin

    select customer\_id into v\_res from Customers where customer\_id=s\_customer\_id;

    return v\_res;

 exception

    when no\_data\_found then

    dbms\_output.put\_line('no customer with this ID');

    return 1;

 end;

 /

create or replace procedure Add\_CreditCard(s\_customer\_id in number,s\_credit\_card in number,s\_EXPIRATION\_YEAR in number,s\_EXPIRATION\_MONTH in number,s\_CREDIT\_CARD\_TYPE in varchar)

IS

v\_creditcardid number;

v\_customerid number;

 function get\_customerid(s\_customer\_id number) return number

 IS

    v\_res number;

 begin

    select customer\_id into v\_res from customers where customer\_id=s\_customer\_id;

    return v\_res;

 exception

    when no\_data\_found then

    dbms\_output.put\_line('no customer with this ID');

    return NULL;

 end get\_customerid;

--

begin

 v\_customerid:=get\_customerid(s\_customer\_id);

 insert into credit\_card#s(credit\_card#, expiration\_year, expiration\_month, credit\_card#type, customer\_id)

 values(s\_credit\_card,s\_EXPIRATION\_YEAR,s\_EXPIRATION\_MONTH,s\_CREDIT\_CARD\_TYPE,s\_customer\_id);

end;

/

begin

 Add\_CreditCard(4,765432,2023,6,'Mastercard');

 end;

/

 -----

CREATE OR REPLACE PROCEDURE Report\_Cards\_Expire(given\_date IN varchar) IS

  days\_until\_expire number;

BEGIN

  days\_until\_expire := 0;

  FOR rec IN (

    SELECT c.name, cc.credit\_card#, cc.credit\_card#type, cc.expiration\_year, cc.expiration\_month,

      ROUND((ADD\_MONTHS(TO\_DATE(cc.expiration\_year || '-' || cc.expiration\_month || '-01', 'YYYY-MM-DD'), -0) - to\_date(given\_date,'yyyy-mm-dd'))) as days\_until\_expire

    FROM customers c

    JOIN credit\_card#s cc ON c.customer\_id = cc.customer\_id

    ORDER BY c.name

  ) LOOP

    if(rec.days\_until\_expire = 61) then

      DBMS\_OUTPUT.PUT\_LINE(rec.name || ', ' || rec.credit\_card# || ', ' || rec.credit\_card#type || ', ' || rec.expiration\_year || ', ' || rec.expiration\_month|| '-' ||rec.days\_until\_expire);

    end if;

  END LOOP;

EXCEPTION

  WHEN no\_data\_found THEN

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

  WHEN others THEN

    dbms\_output.put\_line('error: '||SQLERRM);

END;

/

BEGIN

  Report\_Cards\_Expire('2023-09-01');

END;

/

Member 2

Member 3

-- procedure to report orders by state

create or replace procedure Report\_Orders\_by\_State ( p\_state in varchar)

as

v\_total\_amount Number := 0;

begin

for cust\_rec in (select c.customer\_id,

                c.name as customer\_name,

c.email,

count(\*) as total\_orders,

sum(p.unit\_price\*o.quantity) as total\_amount

      from Orders o

      join Customers c on o.customer\_id = c.customer\_id

      join Products p on o.productid = p.productid

      where c.state = p\_state

      group by c.customer\_id, c.name, c.email)

loop

dbms\_output.put\_line(cust\_rec.customer\_name || ', ' || cust\_rec.email || ', ' || cust\_rec.total\_orders || ' , $' || cust\_rec.total\_amount);

v\_total\_amount := v\_total\_amount + cust\_rec.total\_amount;

end loop;

dbms\_output.put\_line('Grand total: $' || v\_total\_amount);

end;

/

begin

Report\_Orders\_by\_State('MD');

end;

/

-- Trigger to report low inventory

CREATE OR REPLACE TRIGGER Low\_Inventory

AFTER INSERT OR UPDATE ON Products

FOR EACH ROW

BEGIN

  IF :New.quantity < 50 THEN

    DBMS\_OUTPUT.PUT\_LINE('Low inventory: ' || :New.productid || ', ' || :New.product\_name || ', ' || :New.quantity);

  END IF;

END;

/

create or replace procedure Restock\_Product (

p\_productname in products.product\_name%TYPE,

p\_units in products.quantity%TYPE

) as

v\_productid products.productid%TYPE;

begin

select productid into v\_productid from products where product\_name = p\_productname;

Update\_Inventory(v\_productid, p\_units);

end;

/

**Member 4**

**--Problem 15**

CREATE OR REPLACE PROCEDURE Payments\_to\_CC

( INP IN VARCHAR)

IS

    v\_cardType  VARCHAR(50);

    v\_amount  NUMBER;

    v\_fee     NUMBER;

    CURSOR C1 IS

select upper(cc.credit\_card\_type), sum(amount) from invoice i, credit\_card cc

where i.CREDITCARD#=cc.CREDIT\_CARD#

group by cc.credit\_card\_type;

BEGIN

        OPEN C1;

        LOOP

FETCH C1 INTO v\_cardType, v\_amount;

EXIT WHEN C1%NOTFOUND;

IF (v\_cardType = 'VISA')

THEN

v\_fee := v\_amount\*0.03;

ELSIF (v\_cardType = 'MASTERCARD')

THEN

v\_fee := v\_amount\*0.03;

ELSIF (v\_cardType = 'AMEX')

THEN

v\_fee := v\_amount\*0.05;

ELSIF (v\_cardType = 'DISCOVER')

THEN

v\_fee := v\_amount\*0.02;

END IF;

dbms\_output.put\_line('Credit Card fee for ' || v\_cardType || ' card is : ' || v\_fee);

        END LOOP;

        CLOSE C1;

END;

/

exec Payments\_to\_CC(NULL);

**--Problem 16**

CREATE OR REPLACE PROCEDURE Thrifty\_Customer

( INP\_NUM IN NUMBER)

IS

    v\_custId  INT;

    v\_custName  VARCHAR(50);

    v\_amount     NUMBER;

    CURSOR C1 IS

select i.customerid, c.name, sum(amount) from invoice i, customers  c

where

i.customerID=c.CustomerID and

rownum<=INP\_NUM

group by i.customerid,c.name order by sum(amount) ;

BEGIN

        OPEN C1;

        LOOP

FETCH C1 INTO v\_custId, v\_custName, v\_amount;

EXIT WHEN C1%NOTFOUND;

dbms\_output.put\_line('Customer Id : ' || v\_custId || ' Name : ' || v\_custName || ' Amount Spent : ' || v\_amount );

        END LOOP;

        CLOSE C1;

END;

/

exec Thrifty\_Customer(4);

**Member 5:**

**--Problem 19**

DROP SEQUENCE SEQ\_RECOM\_ID;

CREATE SEQUENCE SEQ\_RECOM\_ID

 START WITH     100

 INCREMENT BY   1;

CREATE OR REPLACE PROCEDURE Recommend\_To\_Customer

( INP\_CUSTID IN INT)

IS

    v\_categoryId        INT;

    v\_recom\_prodId      INT;

    v\_recom\_seqId       INT;

    v\_rec\_cnt           INT;

BEGIN

        -- get the top rated category for the given customer

        select p.category\_id into v\_categoryId

        from

        orders o, reviews r, Products p

        where

        o.productID=r.Product\_ID

        and p.Product\_ID=o.productID

        and o.customerid=INP\_CUSTID

        and rownum<=1

        order by r.stars\_given desc;

        if ( v\_categoryId is null )

        then

                dbms\_output.put\_line('Purchased product is not found for the given customer');

                return;

        end if;

        -- get the recommonded product in the same product category

        select p.Product\_ID into v\_recom\_prodId

        from

        products p, reviews r

        where p.Product\_ID = r.Product\_ID

        and p.category\_id=v\_categoryId

        and p.product\_ID not in (select productID from orders where CustomerID=INP\_CUSTID)

        and rownum<=1

        order by r.stars\_given desc;

        if ( v\_recom\_prodId is null )

        then

                dbms\_output.put\_line('Recommonded product is not found for the given customer');

                return;

        end if;

        dbms\_output.put\_line('Recommonded product Id: ' || v\_recom\_prodId || ' for Customer Id : ' || INP\_CUSTID );

--Check if the Recommonded product already in the table, otherwise, insert into Recommendation Table

 select count(1) into v\_rec\_cnt

        from Recommendation

        where customer\_ID=INP\_CUSTID and Recommended\_Product\_ID=v\_recom\_prodId;

        if ( v\_rec\_cnt > 0 )

        then

                dbms\_output.put\_line('Recommonded product is already exists in Recommendation table. so, skipping the insert');

                return;

        else

                v\_recom\_seqId := SEQ\_RECOM\_ID.nextval;

                insert into Recommendation (Recommendation\_ID, customer\_ID, Recommended\_Product\_ID)

                values (v\_recom\_seqId, INP\_CUSTID, v\_recom\_prodId);

                commit;

        end if;

END;

**/**

exec Recommend\_To\_Customer(1);

**--Problem 20**

CREATE OR REPLACE PROCEDURE List\_Recommendations

( INPUT IN NUMBER)

IS

    v\_custId            INT;

    v\_categoryId        INT;

    v\_recom\_prodId      INT;

    v\_avg\_rating        INT;

    CURSOR C1 IS

        select o.customerid, p.category\_id

        from

        orders o, Products p

        where

        p.Product\_ID=o.productID

        group by o.customerid, p.category\_id;

BEGIN

        --Get the customers and their purchased product category

        OPEN C1;

        LOOP

                FETCH C1 INTO v\_custId, v\_categoryId;

                EXIT WHEN C1%NOTFOUND;

                -- For each customer, get the recommended product in that category

                select p.Product\_ID into v\_recom\_prodId

                from

                products p, reviews r

                where p.Product\_ID = r.Product\_ID

                and p.category\_id=v\_categoryId

                and p.product\_ID not in (select productID from orders where CustomerID=v\_custId)

                and rownum<=1

                order by r.stars\_given desc;

                -- Get the average stars given for the recommended product

                select avg(stars\_given) into v\_avg\_rating from reviews where product\_id=v\_recom\_prodId;

                dbms\_output.put\_line('For Customer : ' || v\_custId || ', the Recommonded product Id: ' || v\_recom\_prodId || ' Average Stars given : ' || v\_avg\_rating );

        END LOOP;

        CLOSE C1;

END;

/

exec List\_Recommendations();