IS_620_ADVANCED DATABASE PROJECT

PL/SQL CODE

```
--Drop, Create and Insert Statements:

drop table Customer cascade constraints;
drop table Customer_Discount cascade constraints;
drop table Discount cascade constraints;
drop table restaurant cascade constraints;
drop table category cascade constraints;
drop table restaurant_category cascade constraints;
drop table dish cascade constraints;
drop table cart cascade constraints;
drop table tax_rate cascade constraints;
drop table dish_cart cascade constraints;
drop table Orders cascade constraints;
drop table Dish_Order cascade constraints;
drop table Payment cascade constraints;
drop table message cascade constraints;
drop table review cascade constraints;


CREATE TABLE Customer
(
Customer_ID INT,
Customer_Name VARCHAR (50),
Customer_Address VARCHAR (50),
Customer_Zip_code NUMBER,
Customer_State VARCHAR (50),
Customer_Email VARCHAR (50),
Customer_Credit NUMBER,
PRIMARY KEY (Customer_ID)
);

insert into Customer values
(1, 'Eric','Arbutus',21227,'MD','eric@gmail.com',0);
insert into Customer values
(2, 'John','Baltimore',21225,'MD','john@gmail.com',110);
```

```
insert into Customer values
(3, 'David','Westland Grdens',21228,'MD','david@gmail.com',40);
insert into Customer values
(4, 'Jack','Catonsville',21250,'MD','jack@gmail.com',900);
insert into Customer values
(5, 'Rose','Rockville',21250,'MD','rose@gmail.com',440);


create table Discount
(
Discount_ID int,
Discount_description varchar(100),
discount_type int,
discount_amount number,
primary key(Discount_id)
);


insert into Discount values (1,'Free Delivery',1,0);
insert into Discount values (2,'Discount of 10 %',2,0.1);
insert into Discount values (3,'Fixed amount off',3,20);
insert into Discount values (4,'Discount of 15%',2,0.15);
insert into Discount values (5,'Fixed Amountoff',3,30);


create table Customer_Discount
(
Discount_ID int,
Customer_ID int,
discount_start_date timestamp,
discount_end_date timestamp,
foreign key(Discount_id)references Discount,
foreign key(Customer_ID) references Customer
);


insert into Customer_Discount values(1,1, timestamp '2022-10-10 09:05:00.00',timestamp '2022-
10-11 09:05:00.00');
insert into Customer_Discount values(2,1, timestamp '2022-10-10 09:05:00.00',timestamp '2022-
10-11 09:05:00.00');
```

```
insert into Customer_Discount values(3,2, timestamp '2022-10-17 09:05:00.00',timestamp '2022-10-23 09:05:00.00');
insert into Customer_Discount values(4,3, timestamp '2022-10-20 09:05:00.00',timestamp '2022-10-24 09:05:00.00');
insert into Customer_Discount values(5,4, timestamp '2022-11-01 09:05:00.00',timestamp '2022-11-08 09:05:00.00');


create table restaurant
(restaurant_id int,
restaurant_name varchar(30),
address varchar(50),
phone_number int,
status int,
state varchar(50),
zipcode int,
average_wait_time interval day to second,
average_review_score number,
primary key(restaurant_id));


insert into restaurant values(1, 'ihop', '5525A Baltimore National Pike,
Catonsville',6678025105,1,'Maryland',21228,interval '30' minute,4.5);
insert into restaurant values(2, 'burger king', '5604 Baltimore National Pike,
Catonsville',4107473898,1,'Maryland',21228,interval '40' minute,4.4);
insert into restaurant values(3, 'pizza hut', '6415 Frederick Rd,
Catonsville',4107449380,1,'Maryland',21228,interval '10' minute,4.3);
insert into restaurant values(4, 'sorrento', '5401 East Dr,
Halethorpe',4102426474,1,'Maryland',21227,interval '15' minute,4.2);
insert into restaurant values(5, 'subway', '602 Frederick Rd,
Catonsville',4107884919,1,'Maryland',21228,interval '25' minute,4.1);
insert into restaurant values(6, 'chick-fil-a', 'University Center, 1000 Hilltop
Cir',4436128390,1,'Maryland',21250,interval '45' minute,4.0);


create table category
(category_id int,
category_name varchar(30),
primary key(category_id));
```

```
Insert into category values(1,'cold beverage');
Insert into category values(2,'vegan sandwich');
Insert into category values(3,'chicken burgers');
Insert into category values(4,'french fries');
Insert into category values(5,'cheese pizza');
Insert into category values(6,'pasta');
Insert into category values(7,'fried chicken');


create table restaurant_category
(restaurant_id int,
category_id int,
foreign key(category_id) references category(category_id),
foreign key(restaurant_id) references restaurant(restaurant_id),
primary key(category_id,restaurant_id));


insert into restaurant_category values(1,1);
insert into restaurant_category values(1,2);
insert into restaurant_category values(2,1);
insert into restaurant_category values(2,3);
insert into restaurant_category values(2,4);
insert into restaurant_category values(3,1);
insert into restaurant_category values(3,5);
insert into restaurant_category values(3,6);
insert into restaurant_category values(4,5);
insert into restaurant_category values(5,1);
insert into restaurant_category values(6,7);


create table cart(
cart_id int,
customer_id int not null references customer(customer_id),
restaurant_id int not null references restaurant(restaurant_id),
primary key(cart_id)
);


Insert into cart values(111, 1, 2);
Insert into cart values(112, 2, 1);
```

```
Insert into cart values(113, 3, 4);
Insert into cart values(114, 4, 3);


create table dish(
dish_id int not null,
dish_name varchar(100) not null,
dish_price number not null,
restaurant_id int not null references restaurant(restaurant_id),
primary key(dish_id));


Insert into dish values(210, 'Coffee and Hot Chocolate', 5.99,1);
Insert into dish values(211, 'Eggs Benedict', 11.79,1);
Insert into dish values(212, 'Turkey Sausage Links', 3.29, 1);
Insert into dish values(213, 'Pot Roast Dinner', 12.49, 1);
Insert into dish values(311, 'Whooper', 11.69, 2);
Insert into dish values(312, 'Impossible Whooper', 11.69, 2);
Insert into dish values(313, 'Double Whooper', 12.73, 2);
Insert into dish values(314, 'Big King XL', 11.18, 2);
Insert into dish values(411, 'Garden PartyTM (Thin N Crispy)', 16.30, 3);
Insert into dish values(412, 'Old Fashioned MeatbrawlTM (Pan Pizza)', 6.50, 3);
Insert into dish values(413, 'Cock-a-doodle BaconTM (Hand-Tossed)', 16.30, 3);
Insert into dish values(414, 'Hot and TwistedTM (Hand-Tossed)', 16.30, 3);
Insert into dish values(511, 'Fried Cheese Sticks', 8, 4);
Insert into dish values(512, 'Fried Cheese Ravioli', 8, 4);
Insert into dish values(513, 'Zuppa di cozze',12.5, 4);
Insert into dish values(514, 'Fried Calamari', 12, 4);


create table dish_cart(
cart_id int not null references cart(cart_id),
dish_id int not null references dish(dish_id),
dish_quantity int not null,
primary key(cart_id,dish_id));


Insert into dish_cart values(111,211,1);
Insert into dish_cart values(111,213,2);
Insert into dish_cart values(112,411,1);
```

```
Insert into dish_cart values(112,412,1);
Insert into dish_cart values(112,413,1);
Insert into dish_cart values(112,414,2);
Insert into dish_cart values(113,511,1);
Insert into dish_cart values(113,514,1);
Insert into dish_cart values(114,313,1);
Insert into dish_cart values(114,312,2);
Insert into dish_cart values(114,314,2);


create table tax_rate
(
state varchar(50),
tax_rate float,
primary key(state)
);

Insert into tax_rate values ( 'Alabama', 0.098);
Insert into tax_rate values ( 'Alaska', 0.046);
Insert into tax_rate values ( 'Arizona', 0.095);
Insert into tax_rate values ( 'Arkansas', 0.102);
Insert into tax_rate values ( 'California', 0.135);
Insert into tax_rate values ( 'Colorado', 0.097);
Insert into tax_rate values ( 'Connecticut', 0.154);
Insert into tax_rate values ( 'Delaware', 0.124);
Insert into tax_rate values ( 'District of Columbia', 0.12);
Insert into tax_rate values ( 'Florida', 0.091);


Insert into tax_rate values ( 'Georgia', 0.089);
Insert into tax_rate values ( 'Hawaii', 0.141);
Insert into tax_rate values ( 'Idaho', 0.107);
Insert into tax_rate values ( 'Illinois', 0.129);
Insert into tax_rate values ( 'Indiana', 0.093);
Insert into tax_rate values ( 'Iowa', 0.112);
Insert into tax_rate values ( 'Kansas', 0.112);
Insert into tax_rate values ( 'Kentucky', 0.096);
Insert into tax_rate values ( 'Louisiana', 0.091);
Insert into tax_rate values ( 'Maine', 0.124);
Insert into tax_rate values ( 'Maryland', 0.113);
```

```
Insert into tax_rate values ( 'Massachusetts', 0.115);
Insert into tax_rate values ( 'Michigan', 0.086);
Insert into tax_rate values ( 'Minnesota', 0.121);
Insert into tax_rate values ( 'Mississippi', 0.098);
Insert into tax_rate values ( 'Missouri', 0.093);
Insert into tax_rate values ( 'Montana', 0.105);
Insert into tax_rate values ( 'Nebraska', 0.115);
Insert into tax_rate values ( 'Nevada', 0.096);
Insert into tax_rate values ( 'New Hampshire', 0.096);
Insert into tax_rate values ( 'New Jersey', 0.132);
Insert into tax_rate values ( 'New Mexico', 0.102);
Insert into tax_rate values ( 'New York', 0.159);
Insert into tax_rate values ( 'North Carolina', 0.099);
Insert into tax_rate values ( 'North Dakota', 0.088);
Insert into tax_rate values ( 'Ohio', 0.1);
Insert into tax_rate values ( 'Oklahoma', 0.09);
Insert into tax_rate values ( 'Oregon', 0.108);
Insert into tax_rate values ( 'Pennsylvania', 0.106);
Insert into tax_rate values ( 'Rhode Island', 0.114);
Insert into tax_rate values ( 'South Carolina', 0.089);
Insert into tax_rate values ( 'South Dakota', 0.084);
Insert into tax_rate values ( 'Tennessee', 0.076);
Insert into tax_rate values ( 'Texas', 0.086);
Insert into tax_rate values ( 'Utah', 0.121);
Insert into tax_rate values ( 'Vermont', 0.136);
Insert into tax_rate values ( 'Virginia', 0.125);
Insert into tax_rate values ( 'Washington', 0.107);
Insert into tax_rate values ( 'West Virginia', 0.098);
Insert into tax_rate values ( 'Wisconsin', 0.109);
Insert into tax_rate values ( 'Wyoming', 0.075);

create table Orders
(order_id int, order_time timestamp, delivery_time timestamp,
estimated_time timestamp, payment_status char(1), status int, flag int,
total number, food_total number,  delivery_fee number, tax number, tip number,
customer_id int not null references Customer(customer_id),
restaurant_id int not null references restaurant(restaurant_id),
primary key(order_id));

insert into orders values (1, timestamp '2022-10-12 11:07:09.00',
```

```
timestamp '2022-10-25 11:07:09.00', timestamp '2022-10-30 11:07:09.00', 'Y',2,1,
600.00,512.39,60.12,3.87,10,2,2);

insert into orders values (2, timestamp '2022-10-12 11:07:09.00',
timestamp '2022-10-25 11:07:09.00', timestamp '2022-10-30 11:07:09.00', 'Y',1,1,
600.00,512.39,60.12,3.87,10,2,1);

insert into orders values (3, timestamp '2022-10-12 11:07:09.00',
timestamp '2022-10-25 11:07:09.00', timestamp '2022-10-30 11:07:09.00', 'Y',2,1,
600.00,512.39,60.12,3.87,10,1,1);

insert into orders values (4, timestamp '2022-10-12 11:07:09.00',
timestamp '2022-10-25 11:07:09.00', timestamp '2022-10-30 11:07:09.00', 'Y',2,1,
600.00,512.39,60.12,3.87,10,1,3);

insert into orders values (5, timestamp '2022-10-12 11:07:09.00',
timestamp '2022-10-25 11:07:09.00', timestamp '2022-10-30 11:07:09.00', 'Y',2,1,
600.00,512.39,60.12,3.87,10,4,4);

insert into orders values (6, timestamp '2022-10-12 11:07:09.00',
timestamp '2022-10-25 11:07:09.00', timestamp '2022-10-30 11:07:09.00', 'Y',2,1,
600.00,512.39,60.12,3.87,10,5,5);

create table Dish_Order (order_id int, dish_id int,
foreign key(order_id) references orders(order_id),
foreign key(dish_id) references dish(dish_id));

insert into dish_order values (1,210);
insert into dish_order values (2,211);
insert into dish_order values (3,212);
insert into dish_order values (4,213);
insert into dish_order values (5,311);

create table Payment (payment_id number, customer_id int,
order_id int, payment_time timestamp, payment_amount float,
payment_method int, primary key(payment_id),
foreign key(customer_id) references customer(customer_id),
foreign key(order_id) references orders(order_id));

insert into payment values (1,1,1, timestamp '2022-10-11 11:07:09.00', 500.00, 1);
```

```
insert into payment values (2,2,2, timestamp '2022-10-12 11:07:09.00', 600.00, 2);
insert into payment values (3,3,3, timestamp '2022-10-13 11:07:09.00', 600.00, 3);
insert into payment values (4,4,4, timestamp '2022-10-14 11:07:09.00', 400.00, 1);
insert into payment values (5,5,5, timestamp '2022-10-15 11:07:09.00', 300.00, 1);

create table Message (message_id number, customer_id int,
message_time timestamp, message_body varchar(1000), primary key (message_id),
foreign key(customer_id) references customer(customer_id));

insert into message values (1, 1,  timestamp '2022-10-11 11:07:09.00',  'Order Completed');
insert into message values (2, 2,  timestamp '2022-10-11 11:07:09.00',  'Order Delivered');

insert into message values (3, 3, timestamp '2022-10-11 11:07:09.00', 'Payment Received.');
insert into message values (4, 4, timestamp '2022-10-11 11:07:09.00', 'Completed');
insert into message values  (5, 5, timestamp '2022-10-11 11:07:09.00', 'Order Delivered
Successfully');

create table Review
(
review_id int,
Customer_ID int,
restaurant_id int,
review_date timestamp,
review_score int,
review_comment varchar(100),
average_score float,
primary key (review_id),
foreign key (Customer_ID) references Customer,
foreign key (restaurant_id) references restaurant
);


insert into Review values (1,1,1, timestamp '2022-10-10 09:05:00.00',09, 'Excellent Food Mama
Mia',5);
insert into Review values (2,2,1, timestamp '2022-10-10 19:05:00.00',08, 'Tasty food',4);
insert into Review values (3,3,2, timestamp '2022-10-10 10:25:00.00',10, 'Very Good Service
and awesome food',5);
insert into Review values (4,4,2, timestamp '2022-10-10 12:00:00.00',05, 'Rude waiter',3);
insert into Review values (5,5,3, timestamp '2022-10-10 14:30:00.00',09, 'Healthy food loved
it',5);
```

```
--SEQUENCES
drop sequence paymentid_seq;
drop sequence messageid_seq;
drop sequence oid_seq;
drop sequence rev_seq;

create sequence paymentid_seq start with 6;
create sequence messageid_seq start with 6;
create sequence oid_seq start with 7;
create sequence rev_seq;

--INDIVIDUAL FEATURES

--Member 1: Asmita Dhananjay Deshpande
--Feature 1:

set serveroutput on;
--creating sequence for customer table
drop sequence cust_seq;
create sequence cust_seq
start with 6;
–creating a procedure to add new customer
create or replace procedure add_customer(c_name in customer.customer_name%type,
c_address in customer.customer_Address%type,
c_state in customer.customer_State%type,
z_code in customer.customer_Zip_code%type,
c_email in customer.customer_Email%type)
is
c_id number;
begin
select count(customer_ID) into c_id from customer where customer_email = c_email;
if c_id != 0 then
   dbms_output.put_line('Client already exists');
else
   insert into customer (customer_ID, customer_name, customer_address, customer_state,
customer_zip_code, customer_email)
   values(cust_seq.nextval, c_name, c_address, c_state, z_code, c_email);
end if;
end;
```

```
/

–To display the data in the customer table
select * from customer;
/
-- Call feature 1 procedure
exec add_customer('Catherine', 'Caton Hills','MD' ,12445,'cat@umbc.edu');
-- displaying the table after insertion
select * from customer;


--Member 1: Asmita Dhananjay Deshpande
--Feature 2:

–create a procedure to check the valid and invalid customer
create or replace procedure show_customer(c_email in Customer.Customer_Email%type)
is
cursor c1 is select Customer_Id, Customer_Name, Customer_Address, Customer_Zip_code,
Customer_Email, Customer_Credit, Customer_State from customer where Customer_Email =
c_email;
r c1%rowtype;
cust_count number;
total_orders number;
total_amount number;
begin
 select count(*) into cust_count from customer where c_email=Customer_Email;
 if cust_count = 0 then
   dbms_output.put_line('No such customer');
 else
    for item in c1
    loop
       select count(*), sum(total) into total_orders, total_amount
        from orders
        where trunc(delivery_time) < sysdate
        and trunc(delivery_time) > trunc(delivery_time) - interval '6' month
        and status = 2
        and customer_id = item.customer_id;
        -- Displaying customer details --
        dbms_output.put_line('The Details of the Customer are:');
        dbms_output.put_line('Customer Name : '|| item.Customer_Name);
```

```
        dbms_output.put_line('Customer Address : ' || item.Customer_Address);
        dbms_output.put_line(' Customer State :' ||item.Customer_State);
        dbms_output.put_line('Customer Zip_Code :' || item.Customer_Zip_code);
        dbms_output.put_line('Customer Email :' || item.Customer_Email);
        dbms_output.put_line('Customer Credit :' || item.Customer_Credit);
        dbms_output.put_line('Total Order : ' || total_orders);
        dbms_output.put_line('Total Amount :' || total_amount);
    end loop;
     end if;
end;
/
–To display the data in the customer table
select * from customer;
–call feature 2 procedure
–to check if customer exists and display its all details
exec show_customer ('eric@gmail.com');
—invalid customer or customer does not exists
exec show_customer ('ravi@gmail.com');
select * from customer;



--Member 2: Sayali  Satish Dhavale
--Feature 3:

--create procedure to find restaurant by category
CREATE OR REPLACE PROCEDURE FIND_RESTAURANT_BY_CATEGORY (X IN
VARCHAR) IS
--cursor to select restaurant details for input category
CURSOR C1 IS SELECT r.restaurant_name, r.average_review_score, r.average_wait_time,
r.zipcode FROM restaurant r, category c, restaurant_category rc WHERE
r.restaurant_id=rc.restaurant_id and c.category_id=rc.category_id and category_name LIKE X;
a number;
counter number;
BEGIN
--check if input category exists
select count(1) into counter from category where category_name like X;
--if input category does not exist
if counter<=0 then
```

```
dbms_output.put_line('No such category');
-if input category exists
else
--call cursor in for loop
for i in c1 loop
--convert average wait time in minutes
a:=((extract(day from i.average_wait_time))*60*24)+((extract(hour from
i.average_wait_time))*60)+(extract(minute from i.average_wait_time))+((extract(second from
i.average_wait_time)/60));
--print restaurant details
dbms_output.put_line('Restaurant Name: ' || i.restaurant_name || ' | Average Review Score: '
||i.average_review_score || ' | Average Wait Time: ' || a ||' minutes | Zipcode: ' || i.zipcode);
end loop;
end if;
END;
/
--This is the scenario when the input (category substring) does not exist or if there is a spelling
error1.
exec find_restaurant_by_category('vegetarian');
/
--This is the scenario when the input exists.
exec find_restaurant_by_category('%vegan%');
/
--This is the scenario when the input exists.
exec find_restaurant_by_category('%chicken%');

--Member 2: Sayali  Satish Dhavale
--Feature 4:

--create procedure to show dishes by restaurant
CREATE OR REPLACE PROCEDURE show_dishes_by_restaurant(X IN number) IS
--cursor to select required details from dish and restaurant table
CURSOR C1 IS SELECT r.restaurant_id, d.dish_name, d.dish_price FROM restaurant r, dish d
WHERE r.restaurant_id=d.restaurant_id and r.restaurant_id = X;
counter number;
BEGIN
--check input restaurant id exists
select count(1) into counter from restaurant where restaurant_id=X;
--if exists
if counter>0 then
```

```
--call cursor in for loop
for i in c1 loop
dbms_output.put_line('Dish Name: ' || i.dish_name || ' at ' || i.dish_price);
end loop;
--if does not exist
else
dbms_output.put_line('No such Restaurant');
end if;
END ;
/
--if restaurant id does exists
begin
show_dishes_by_restaurant(1);
end;
/
--if restaurant id does not exists
begin
show_dishes_by_restaurant(9);
end;
/



--Member 3: Savita Ningappa Navalgi
 --Feature 5 :

create or replace procedure showDishes(cartId in cart.cart_id%type) IS
   Cursor c1 is select dish.dish_name, dish.dish_price, dish_cart.dish_quantity from
cart,dish_cart,dish
   where dish_cart.cart_id=cart.cart_id and cart.cart_id=cartId and
dish.dish_id=dish_cart.dish_id;
   --cursor to fetch cart dishes and its detail from dish_cart table
   counter int;
   count_rows int;
BEGIN
   count_rows:=0;
   counter:=0;
  --to check whether cart id is valid
   select count(*)into count_rows from cart where cart.cart_id = cartId;
   if(count_rows=0) then
```

```
        dbms_output.put_line('Invalid cart Id');
    else
        for item in c1
        loop
        counter:=counter+1;
        dbms_output.put_line(counter||': Dish Name: '||item.dish_name||', Dish Price:
$'||item.dish_price||', Dish Quantity: '||item.dish_quantity);
        end loop;
    end if;
exception
    when no_data_found then
    dbms_output.put_line('Invalid Cart ID');
END;


--Invalid Cart
set serveroutput on;
exec showDishes(1142);

--Valid Cart
set serveroutput on;
exec showDishes(114);

--select * from dish_cart;
--select * from cart;
--select * from dish;



--Member 3: Savita Ningappa Navalgi
--Feature 6:

create or replace procedure removeDish
    (dishId in dish.dish_id%type, cartId in cart.cart_id%type)
    IS
    dish_qty_cart number;
    count_rows int;
BEGIN
    count_rows:=0;
    --to check if dish id and cart id are valid
```

```
   select count(*)into count_rows from dish_cart where dish_cart.cart_id = cartId and
dish_cart.dish_id=dishId;
   if(count_rows=0) then
      dbms_output.put_line('Invalid Input');
   else
   --fetch the dish quantity
      select DISH_QUANTITY into dish_qty_cart from dish_cart where dish_id=dishId and
cart_id=cartId;
      --if dish quantity is greater than 1 then reduce quantity by 1
      if dish_qty_cart>1 then
         update dish_cart set dish_cart.dish_quantity=dish_cart.dish_quantity-1 where
dish_id=dishId and cart_id=cartId;
         dbms_output.put_line('Dish Quantity Reduced !');
      else
      --if dish quantity is not greater than 1 then delete the dish from cart
         delete from dish_cart where dish_id=dishId and cart_id=cartId;
         dbms_output.put_line('Dish Removed from Cart !');
      end if;
   end if;
Exception
   when no_data_found then
   dbms_output.put_line('Invalid Cart ID');
END;


select * from dish_cart;

--1. Invalid Input
set serveroutput on;
exec removeDish(3112,114);

--2. dish quantity reduced by 1
set serveroutput on;
exec removeDish(312,114);
select * from dish_cart;

--to insert a dish into cart
--insert into dish_cart values(114,312,4);
--to update the quantity of the dish
--update dish_cart set dish_cart.dish_quantity=10 where dish_id=312 and cart_id=114;
```

```
--3.Remove dish from cart (dish quantity 1);
--First Insert a dish with single quantity ;

--select * from dish_cart;
insert into dish_cart values(114,311,1);
select * from dish_cart;

set serveroutput on;
exec removeDish(311,114);

select * from dish_cart;


--Member 4: Vrushali Vishal Patil
--Feature 7:

--procedure to update the status of an order
create or replace procedure update_status(orderId in int,
orderStatus number, inputTime timestamp) is
orderCount int := 0;
final_message varchar(200);
begin
select count(*) into orderCount from orders where order_id = orderId;
--check if order is valid
if orderCount = 0 then
dbms_output.put_line('Invalid Order ID');
else
--update the status
update orders set status = orderStatus where order_id = orderId;
if orderStatus = 2 then
--if delivered add message
final_message := 'Your order ' || orderId || ' has been delivered!';
add_message(orderId, inputTime, final_message);
elsif orderStatus = 3 then
--if canceled add message & record payment
final_message := 'Your order ' || orderId ||
' has been canceled and refund issued!';
add_message(orderId, inputTime, final_message);
record_payment(orderId, inputTime);
```

```
end if;
end if;
end;

--procedure to add message in message table
create or replace procedure add_message(orderId in int,
inputTime in timestamp, final_message in varchar) is
customerId int;
begin
--Fetch customer id from orders and insert a message
select customer_id into customerId from orders where order_id = orderId;
insert into message values (messageid_seq.nextval,
customerId, inputTime, final_message);
end;

--procedure to record payment in payment table
create or replace procedure record_payment(orderId in int,
inputTime in timestamp) is
customerId int;
orderAmount number;
paymentMethod number;
begin
--Fetch customer id and total amount from orders table.
select customer_id, total into customerId, orderAmount
from orders where order_id = orderId;
--Fetch original payment method.
select payment_method into paymentMethod
from payment where order_id = orderId;
--negative amount as order is canceled
orderAmount := -orderAmount;
insert into payment values (paymentid_seq.nextval,customerId, orderId,
inputTime, orderAmount, paymentMethod);
end;


SET SERVEROUTPUT ON;
--Invalid Order ID
exec update_status(1000,1, timestamp '2022-11-30 11:07:09.00');
--This will print message as ‘Invalid Order ID’
```

```
--Valid Order ID with order status as 1 i.e ‘in progress’
exec update_status(1,1, timestamp '2022-11-30 11:07:09.00');
--This will update the status of the order  1 as 1.
--To check run below sql statement.
select * from orders where order_id = 1;

--Valid order id with order status as 2 i.e ‘delivered’
exec update_status(2,2, timestamp '2022-11-30 11:07:09.00');
--This will update the status of the order 2 as 2.
--Also it will insert a entry in payment table with orderid 2 & in message table.
--To check run below sql statements.
select * from orders where order_id = 2;
select * from payment where order_id = 2;
select * from message;

--Valid order id with order status as 3 i.e ‘canceled’
exec update_status(3,3, timestamp '2022-11-30 11:07:09.00');
--This will udate the status of the order 3 as 3.
--Also will insert a entry in payment table with orderid 2 & in message table.
--To check run below sql statements.
select * from orders where order_id = 3;
select * from payment where order_id = 3;
select * from message;


--Member 5: Parthiv Gandhi
--Feature 8:

– Creating a trigger that will be fired on every insert query on review
CREATE OR REPLACE TRIGGER update_average_score
AFTER INSERT ON review
FOR EACH ROW
BEGIN
– on every insert clause, we update the avg_review_score that will take into account the new added review_score
   UPDATE restaurant
   set average_review_score=(select avg(review_score) from review where restaurant_id = :new.restaurant_id)
   where restaurant_id = :new.restaurant_id;
END;
```

```
/

DROP TRIGGER update_average_score;

Create or replace procedure addReview(cust_id in number,res_id in number, rev_date in
timestamp, rev_score in number, rev_comment in varchar)
IS
   cust_id_exists pls_integer;
   res_id_exists pls_integer;
BEGIN
--check for count of a particular customer id, print ‘invalid customer id’ if no ids found
   SELECT COUNT(*) into cust_id_exists from customer where customer_id = cust_id;
   if cust_id_exists = 0 THEN
   dbms_output.put_line('INVALID CUSTOMER ID');
   end if;
– same case but for restaurant id
   SELECT COUNT(*) into res_id_exists from restaurant where restaurant_id = res_id;
   if res_id_exists = 0 THEN
   dbms_output.put_line('INVALID RESTAURANT ID');
   end if;
  --if both res id and cust id exists, then we insert the record in the review table
   if cust_id_exists= 1 AND res_id_exists = 1 THEN
      INSERT INTO
review(customer_ID,restaurant_id,review_date,review_score,review_comment)
               VALUES(cust_id,res_id, rev_date, rev_score, rev_comment);
    – after insertion, we update the average_review_score that will take into account the new
added review

      UPDATE restaurant
      set average_review_score=(select avg(review_score) from review where restaurant_id =
res_id)
      where restaurant_id = res_id;

   end if;

END;
/

--successful insertion of a record
```

```
exec addReview(2,5,timestamp '2022-10-13 15:45:00.00', 8, 'Nice and a health alternative to other food chains');

--insert attempt with invalid customer id, will print “invalid customer id”
exec addReview(356,5,timestamp '2022-10-13 15:45:00.00', 8, 'Nice and a health alternative to other food chains');

--insert attempt with invalid restaurant id, will print “invalid restaurant id”
exec addReview(2,534,timestamp '2022-10-13 15:45:00.00', 8, 'Nice and a health alternative to other food chains');


set SERVEROUTPUT on;

select * from review;
select * from restaurant;


--Member 5: Parthiv Gandhi
--Feature 9:



--procedure that will display all reviews of a particular restaurant
Create or replace procedure displayReviews(res_id in number)
IS
   res_id_exists pls_integer;

BEGIN
  --condition to check if the input restaurant id exists or not, if not prints an error message
   SELECT COUNT(*) into res_id_exists from restaurant where restaurant_id = res_id;
   if res_id_exists = 0 THEN
   dbms_output.put_line('INVALID RESTAURANT ID');
   end if;


   if res_id_exists = 1 THEN
         --creating an implicit cursor that loops on the select statement, useful in display the reviews of a restaurant
      for revs in ( select review_date, review_score,review_comment from review
```

```
                where restaurant_id =  res_id)
    loop
     dbms_output.put_line('Review Date: '|| revs.review_date ||
                  ' | Comment: ' || revs.review_comment ||
                  ' | Score: '|| revs.review_score);

    end loop;
  end if;
END;
/

--successful output
exec displayReviews(13);

--invalid case of an input restaurant id
exec displayReviews(133);
select count(*) from review where restaurant_id = 1;



 --GROUP FEATURES

--Feature 10:


--procedure creation
create or replace procedure feature10(v_custID customer.customer_id%type, v_restID
restaurant.restaurant_id%type, v_dish dish.dish_id%type)

IS
v_count number;
cart_seq_id cart.cart_id%type;
Begin
--check valid customer id
select count(*) into v_count from customer where customer_id=v_custID;

if(v_count=0) then --custid not valid
dbms_output.put_line('no such customer');
else --custid if valid
--check whether the restaurant ID is valid
```

```
select count(*) into v_count from restaurant where restaurant_id=v_restID and status=1;

if v_count=0 then --if restid invalid
dbms_output.put_line('invalid restaurant ID');

else --if restid valid
--check whether the restaurant is open
select count(*) into v_count from restaurant where restaurant_id=v_restID and status=1;

if v_count=0 then -- if  restaurant closed
dbms_output.put_line('restaurant is closed');
else --if  restaurant open
--check whether input dishid belongs to input restaurantid

select count(*) into v_count from dish where restaurant_id = v_restID and dish_id=v_dish;

if v_count=0 then --if invalid dishid
dbms_output.put_line('Invalid dish Id');
else --if valid dishid
--check for existing shopping cart
select count(*) into v_count from cart where restaurant_id = v_restID and
customer_id=v_custID;
if v_count=0 then -- if not exist, create new and print cart id

cart_seq_id := 7;
insert into cart values(cart_seq_id,v_custID,v_restID);

dbms_output.put_line('New cart id : '||cart_seq_id);

else
select cart_id into cart_seq_id from cart where restaurant_id = v_restID and
customer_id=v_custID;
end if;
--check whether the dish is already in cart
select count(*) into v_count from dish_cart where dish_cart.dish_id = v_dish and
dish_cart.cart_id=cart_seq_id;
if v_count=0 then -- if not in cart, insert new row

insert into dish_cart values(cart_seq_id,v_dish,1);
```

```
else -- if in cart, increase quantity
update dish_cart set dish_quantity=dish_quantity+1 where dish_id=v_dish and
cart_id=cart_seq_id;
end if;
end if;
end if;
end if;
end if;
End;

--Existing cart
set serveroutput on;
exec feature10(2,1,213);
select * from cart;
select * from dish_cart;
select * from dish;
--correct output
exec feature10(2,1,213);
--if customer id is invalid
set serveroutput on;
exec feature10(10, 1, 213);
--if restaurant id is invalid
set serveroutput on;
exec feature10(2, 122, 213);
--if dish belongs to same restaurant
set serveroutput on;
exec feature10(2, 1, 313);



--Feature 11:

--Procedure to calculate total price of the dishes in the cart at checkout time
set serveroutput on;
Create or replace function totalPrice(cartId in int,flag in int, checkout_time in timestamp,
delivery_fee out int, v_sales_tax out float, dishprice out float)
return float IS
totalPrice float;
--cursor to go through the list of dishes and its quantity
Cursor c1 is select dish_id, dish_quantity from dish_cart where cart_id = cartId;
```

```
dishId int;
dish_qty int;
sd date;
ed date;
v_discount_type int;
v_discount_id int;
v_count int;
v_discount_amount int;
v_customer_id int;
v_restaurant_id int;
v_customer_zip_code int;
v_zipcode number;
v_dish_price number;
v_state_restaurant varchar(50);
BEGIN
delivery_fee:= 0;
v_sales_tax:=0;
dishprice:=0;

--to check if cart id is valid
select count(*) into v_count from cart where cart_id=cartId;
if(v_count=0) then
dbms_output.put_line('Invalid Cart Id');
return -1;
else
-- fetch the discount start and end date to check if its valid on the checkout time
select Discount_start_date, Discount_end_date, discount_id, cart.customer_id into
sd,ed,v_discount_id,v_customer_id from Customer_Discount,Cart where
Customer_Discount.Customer_ID=cart.Customer_ID
and cart.cart_id=cartId and rownum=1;
totalPrice:= 0;

--calculate total dish price
Open c1;
Loop
fetch c1 into dishId, dish_qty;
exit when c1%notfound;
select dish_price into v_dish_price from dish where dish_id = dishId;
dbms_output.put_line('dish id : '||dishId||' dish Qty: '||dish_qty||' dish price: '||v_dish_price);
totalPrice:= totalPrice+(v_dish_price*dish_qty);
```

```
end loop;
dishprice := totalPrice;  --to return the dish price
dbms_output.put_line('total Price before discount : '||totalPrice);
close c1;
-- to check if discount is valid for the customer
if checkout_time<ed and checkout_time>sd then
select discount_type,discount_amount into v_discount_type, v_discount_amount from discount
where discount_id = v_discount_id;

-- apply type of discounts
if v_discount_type = 1 then
delivery_fee:=0;
dbms_output.put_line('total Price after discount : '||totalPrice);
elsif v_discount_type = 2 then
totalPrice:= (1- v_discount_amount) * totalPrice;
dbms_output.put_line('total Price after discount : '||totalPrice);
elsif v_discount_type = 3 then
totalPrice:= totalPrice - v_discount_amount;
dbms_output.put_line('total Price after discount : '||totalPrice);
else
dbms_output.put_line('Invalid Discount/Discount is expired');
end if;
end if;

--fetch zip code for the customer as well as restaurant too calculate tax based on locations
select restaurant_id into v_restaurant_id from cart where cart_id = cartId;
select state into v_state_restaurant from restaurant where restaurant_id = v_restaurant_id;
select tax_rate into v_sales_tax from tax_rate where state=v_state_restaurant;
v_sales_tax:=totalPrice*v_sales_tax;

--calculate total price after adding sales tax
totalPrice := totalPrice+v_sales_tax;
dbms_output.put_line('Sales Tax on dish price: '||v_sales_tax);
dbms_output.put_line('total Price after tax : '||totalPrice);

-- check if its a pickup order or delivery
if flag = 1 then
dbms_output.put_line('Delivery Method: Deliver');
select customer_zip_code into v_customer_zip_code from customer where customer_id =
v_customer_id;
```

```
select zipcode into v_zipcode from restaurant where restaurant_id = v_restaurant_id;
if v_customer_zip_code = v_zipcode then
delivery_fee:= 2;
else
delivery_fee:= 5;
end if;
elsif flag=2 then
delivery_fee:=0;
dbms_output.put_line('Delivery Method: Pickup');
end if;

--calculate total price after adding delivery fee
if v_discount_type = 1 then
delivery_fee:=0;
end if;
totalPrice := totalPrice+delivery_fee;
return totalPrice;
end if;
return -1;
END;


--1. Invalid Cart ID
set serveroutput on;
declare
delivery_fee_v int;
sales_tax_v  float;
dishprice_v float;
totalPrice_v float;
begin
totalPrice_v:= totalPrice(113,1,timestamp '2022-12-19 2:00:00', delivery_fee_v, sales_tax_v,
dishprice_v);
if totalPrice_v>0 then
dbms_output.put_line('Delivery Fee: '||delivery_fee_v);
dbms_output.put_line('Sales Tax: '||sales_tax_v);
dbms_output.put_line('Total Dish Amount: '||dishprice_v);
dbms_output.put_line('Total Price:'||totalPrice_v);
end if;
end;
```

```
--2. Valid Cart, Delivery Method - 1(deliver),
--valid discount with discount type - fixed amount ($30)
set serveroutput on;
declare
delivery_fee_v int;
sales_tax_v  float;
dishprice_v float;
totalPrice_v float;
begin
totalPrice_v:= totalPrice(114,1,timestamp '2022-12-19 2:00:00', delivery_fee_v, sales_tax_v,
dishprice_v);
if totalPrice_v>0 then
dbms_output.put_line('Delivery Fee: '||delivery_fee_v);
dbms_output.put_line('Sales Tax: '||sales_tax_v);
dbms_output.put_line('Total Dish Amount: '||dishprice_v);
dbms_output.put_line('Total Price:'||totalPrice_v);
end if;
end;


--3. Valid Cart, Delivery Method - 2(pickup),
--valid discount with discount type - fixed amount ($30)
set serveroutput on;
declare
delivery_fee_v int;
sales_tax_v  float;
dishprice_v float;
totalPrice_v float;
begin
totalPrice_v:= totalPrice(114,2,timestamp '2022-12-19 2:00:00', delivery_fee_v, sales_tax_v,
dishprice_v);
if totalPrice_v>0 then
dbms_output.put_line('Delivery Fee: '||delivery_fee_v);
dbms_output.put_line('Sales Tax: '||sales_tax_v);
dbms_output.put_line('Total Dish Amount: '||dishprice_v);
dbms_output.put_line('Total Price:'||totalPrice_v);
end if;
end;

--4. Valid Cart, Delivery Method - 1(deliver),
```

```
--valid discount with discount type - free delivery
set serveroutput on;
declare
delivery_fee_v int;
sales_tax_v  float;
dishprice_v float;
totalPrice_v float;
begin
totalPrice_v:= totalPrice(111,1,timestamp '2022-12-19 2:00:00', delivery_fee_v, sales_tax_v,
dishprice_v);
if totalPrice_v>0 then
dbms_output.put_line('Delivery Fee: '||delivery_fee_v);
dbms_output.put_line('Sales Tax: '||sales_tax_v);
dbms_output.put_line('Total Dish Amount: '||dishprice_v);
dbms_output.put_line('Total Price:'||totalPrice_v);
end if;
end;



--Feature 12:

--create procedure
create or replace procedure feature12(cartid IN int, ordertime IN timestamp,

delmethod in int, esttime in timestamp, tippay in number, paymethod in int) IS

counter number;
tamount number;
cid int;
rid int;
delfee number;
taxno number;
dishtotal number;
rname varchar (50);
msg varchar (1000);
timeInMinutes number;
--cursor declaration
cursor c1 is select dish_id from dish_cart where cart_id=cartid;
oid int;
```

```
BEGIN
--check valid cart_id
select count(1) into counter from cart where cart_id=cartid;
if counter>0 then
--calculate total price using feature 11
tamount:=totalPrice(cartid, delmethod, ordertime, delfee, taxno, dishtotal);

select customer_id into cid from cart where cart_id=cartid;
select restaurant_id into rid from cart where cart_id=cartid;
oid:=oid_seq.nextval;

--insert new order based on input cart is and new order sequence
insert into orders values (oid, ordertime, NULL, esttime, 'Y',
1,delmethod, tamount, dishtotal,delfee,taxno,tippay,cid,rid);

--insert in dish_order table for input cart id's, dish_id
for i in c1 loop
insert into dish_order values(oid, i.dish_id);
end loop;

--delete cart details with input cart id
delete from dish_cart where cart_id = cartid;
delete from cart where cart_id=cartid;

select restaurant_name into rname from restaurant
where restaurant_id=rid;


timeInMinutes:= ((extract(day from esttime))*60*24)+
((extract(hour from esttime))*60)+(extract(minute from esttime)) +
((extract(second from esttime)/60));

msg:='A new order '||oid||' is placed at restaurant '|| rname ||
with estimated time of ' || timeInMinutes || ' minutes and amount '||
tamount;

--insert order place message in message table
insert into message values (messageid_seq.nextval, cid,ordertime,msg);

--insert payment details in payment table
```

```
insert into payment values (paymentid_seq.nextval,cid,
oid,ordertime,tamount, paymethod);
else
dbms_output.put_line('Invalid Cart Id');
end if;
end;

SET SERVEROUTPUT ON;
--Invalid Order ID
exec feature12(11038,timestamp '2022-12-11 16:05:00.00',1,timestamp '2022-12-21
16:00:00.00',1.25,3);
--This print message Invalid Order ID.

--Valid Order ID
exec feature12(114,timestamp '2022-12-11 16:05:00.00',1,timestamp '2022-12-21
16:00:00.00',1.25,3);
--This will insert a row in an orders table with a newly generated order id.
--Also, it will delete the respective cart.
--Additionally, it will add rows in payment and massage tables.
select * from orders;
select * from message;
select * from payment;



--Feature 13:
 -- advanced search feature
set SERVEROUTPUT on;

-- creating a varray of varchar, which will be used to store list of input categories
CREATE OR REPLACE TYPE cat_arr_type AS VARRAY(20) OF VARCHAR2(50);
/

--this procedure returns all restaurant that are a) under of on the input categories b) --
avg_review_score >= given minimum score, c) wait time <= input wait time and d) having --
restaurant zip code same as that of the customer's zip code or only differ by the last digit

Create or replace procedure search_restaurant(cust_id in number, categories cat_arr_type,
min_rev_score in number, max_wait_time in interval day to second)
AS
   cust_id_exists pls_integer;
```

```
   cust_zip_code pls_integer;

BEGIN
   -- checks for a valid customer id, prints invalid message if not
   SELECT COUNT(*) into cust_id_exists from customer where customer_id = cust_id;
   if cust_id_exists = 0 THEN
   dbms_output.put_line('INVALID CUSTOMER ID');
   end if;

    --if it exists, filter according to a),b), c) & d)
   if cust_id_exists= 1 THEN
      --store customer's zip code into a variable for future use
       SELECT Customer_Zip_code into cust_zip_code from customer where customer_id =
cust_id;

      -- implicit for loop that performs join on restaurant and restaurant category, helping us to
retrieve information about the restaurant based on the input ids
       for res in (
       SELECT restaurant_name, address, status, zipcode, average_wait_time,
average_review_score, restaurant.restaurant_id, category_id
       FROM restaurant INNER JOIN restaurant_category ON restaurant_category.restaurant_id =
restaurant.restaurant_id
       where average_review_score >= min_rev_score and average_wait_time <= max_wait_time
AND SUBSTR(cust_zip_code,1,4) = SUBSTR(zipcode,1,4)
       AND
       CATEGORY_id IN (SELECT category_id FROM category where category_name in (select
column_value from TABLE(categories)))
       )
       loop
        dbms_output.put_line('Restaurant : '|| res.restaurant_name ||
                        ' | Address: ' || res.address ||
                        ' | Status: ' || res.status ||
                        ' | Average review score: ' ||  res.average_review_score ||
                        ' | Zip code: ' || res.zipcode ||
                        ' | Average wait time: ' || res.average_wait_time);

       end loop;
   end if;

END;
```

```
/


-- successful run case, displays restaurant name, status, score, zip code and wait time
exec search_restaurant(4,cat_arr_type('fried chicken','cold beverage'),4.2, interval '47' MINUTE
);

-- Invalid case, outputs invalid customer id
exec search_restaurant(434,cat_arr_type('fried chicken','cold beverage'),4.2, interval '47'
MINUTE );

--Feature 14:

--Procedure to first find restaurant visited by given customers and then find
--customers who visited the same restaurants.
create or replace procedure feature_14(customerId in number) is

customerCount int;
--cursor to get restaurants where input customer has put order
cursor c1 is select restaurant_id from orders where customer_id = customerId;

--cursor to get other customers who have put orders in same restaurant
cursor c2 is select o2.customer_id from orders o1, orders o2
where o2.customer_id != customerId and o1.customer_id = customerId
and o1.restaurant_id = o2.restaurant_id
group by o2.customer_id;

begin

--To check if customer_id is valid or not
select count(*) into customerCount from customer where customer_id = customerId;


--Valid Customer
if customerCount > 0 then
dbms_output.put_line('Customer '||customerId||' visited following restaurants');

--Print Restaurants
for i in c1 loop
dbms_output.put_line('Restaurant ' || i.restaurant_id);
```

```
end loop;

dbms_output.new_line;
dbms_output.put_line('Following customers visited above restaurants');

--Print customers who visited same restaurant
for i in c2 loop
dbms_output.put_line('Customer ' || i.customer_id);
end loop;

dbms_output.new_line;
dbms_output.put_line('Restaurant recommendations listed below.');

--calling another procedure to get restaurants for each customer
for i in c2 loop
restaurant_recommendation(i.customer_id, customerId);
end loop;

--Invalid Customer
else
dbms_output.put_line('Customer Not Found!');
end if;
end;

--procedure to get restaurants visited by individual customers.
create or replace procedure restaurant_recommendation(newCustomerId in number,

inputCustomerId in number) is
--cursor to get restaurant name, address, review of the restaurant
cursor c1 is select o.restaurant_id, r.address, r.restaurant_name,
r.average_review_score
from orders o, restaurant r
where o.restaurant_id not in ( select restaurant_id
from orders where customer_id = inputCustomerId)
and o.customer_id = newCustomerId and o.restaurant_id = r.restaurant_id;

begin
--print restaurant information
for i in c1 loop
dbms_output.put_line ('ID: ' || i.restaurant_id);
```

```
dbms_output.put_line ('Name: ' || i.restaurant_name);
dbms_output.put_line ('Address: ' || i.address);
dbms_output.put_line ('Average Review Score: ' || i.average_review_score);
dbms_output.new_line;
end loop;
end;

SET SERVEROUTPUT ON:
--Calling procedure with invalid customer id.
exec feature_14(1000);
--This will prinnt message as customer not found.

--Calling procedure with valid customer id
exec feature_14(2);
--This will print restaurant visited by customer 2.
--Also, it will print other customer who visited same restaurant.
--Finally, it will print restaurant visited by those customers(recommended restaurants).
```