--IS 620\_Group Project

--GROUP 14

--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).