```sql
-- Drop Tables
--please drop all the tables before and then run the create table statements.
drop table customer cascade constraints;
drop table discount cascade constraints;
drop table tax cascade constraints;
drop table available_discount cascade constraints;
drop table categories cascade constraints;
drop table restaurants cascade constraints;
drop table mcategories cascade constraints;
drop table cart cascade constraints;
drop table dish cascade constraints;
drop table cart_dish cascade constraints;
drop table reviews cascade constraints;
drop table orders cascade constraints;
drop table order_dishes cascade constraints;
drop table payment_record cascade constraints;
drop table message cascade constraints;

drop sequence "CUSTID_SEQ";
CREATE SEQUENCE custid_seq START WITH 101 INCREMENT BY 1;

drop sequence "RID_SEQ";
CREATE SEQUENCE rid_seq START WITH 401 INCREMENT BY 1;

drop sequence "DID_SEQ";
CREATE SEQUENCE did_seq START WITH 501 INCREMENT BY 1;

drop sequence "REVID_SEQ";
CREATE SEQUENCE revid_seq START WITH 601 INCREMENT BY 1;

drop sequence "CID_SEQ";
CREATE SEQUENCE cid_seq START WITH 701 INCREMENT BY 1;

drop sequence "ORDID_SEQ";
CREATE SEQUENCE ordid_seq START WITH 801 INCREMENT BY 1;

drop sequence "PAYID_SEQ";
CREATE SEQUENCE payid_seq START WITH 901 INCREMENT BY 1;

drop sequence "MSGID_SEQ";
CREATE SEQUENCE msgid_seq START WITH 1001 INCREMENT BY 1;


create table customer(
customerID NUMBER DEFAULT custid_seq.nextval NOT NULL PRIMARY KEY,
names varchar(25),
address varchar(25),
zipcode number,
state varchar(20),
email varchar(50),
credit number);

insert into customer
values(custid_seq.nextval,'Stark','209 Maiden Choice Lane',21222,'Maryland','tonystark@gmail.com',28);
insert into customer
values (custid_seq.nextval, 'Rogers', '289 Mayson Choice Avenue', 21228, 'New York', 'steverogers78@gmail.com', 89);
insert into customer
values (custid_seq.nextval, 'Banner', '246 Symington Lane', 21237, 'Maryland', 'bannerbruce345@gmail.com', 9);
insert into customer
values (custid_seq.nextval, 'Natasha', '655 Fredrick Avenue', 21237, 'Chicago', 'natasharomanoff@gmail.com', 4);
insert into customer
values (custid_seq.nextval, 'Clint', '829 Taylor Lane', 21232, 'Texas', 'clintbarton89@gmail.com', 28);

create table discount(
discountID int,
discount_description varchar(100),
discount_type int,
discount_amount float,
primary key(discountID));


insert into discount
values (201, 'Coupon', 2, 0.05);
insert into discount
values (202, 'Coupon', 2, 0.1);
insert into discount
values (203, 'FreeDel', 1, 0);
insert into discount
values (204, 'FreeDel', 1, 0);
insert into discount
values (205, 'FixedAmt', 3, 5);

create table available_discount(
--availablediscID int,
customerID int,
discountID int,
start_date date,
end_date date,
primary key(discountID),
--primary key(customerID),
--primary key(availablediscID),
foreign key(customerID) references customer,
foreign key(discountID) references discount);

insert into available_discount
values(101,201,date'2019-12-1',date '2025-1-1');
insert into available_discount
values(102,202,date '2013-5-1',date '2013-6-3');
insert into available_discount
values(103,203,date '2015-1-8',date '2015-2-10');
insert into available_discount
values(104,204,date '2016-1-1',date '2016-1-31');
insert into available_discount
values(105,205,date '2018-7-7',date '2023-12-31');

create table tax(
state varchar(30),
taxrate number,
primary key(state));

insert into tax
values('MD', 2.67);
insert into tax
values ('BF', 2.5);
insert into tax
values('CD', 3.5);
insert into tax
values('TX', 4.5);
insert into tax
values('WA', 5);

create table restaurants(
restaurantID int,
restaurant_name varchar(50),
address varchar(200),
phone_number number,
current_status varchar(100),
zip_code number,
state varchar(30),
average_wait_time number,
average_review_score float,
primary key(restaurantID),
foreign key(state) references tax);

insert into restaurants
values (401,'Starbucks','1000 Hilltop Circle',4434357899,'open',21226,'MD',5,7);
insert into restaurants
values (402,'Paradise','605 Fredrick Road',6685547894,'closed',21223,'TX',10,8.1);
insert into restaurants
values (403,'Chick-fil-A','899 Arundel Avenue',6634512345,'open',21226,'MD',8,6.3);
insert into restaurants
values (404,'Subway','255 Maiden Choice Lane',4325678899,'closed',21233,'TX',15,4.8);
insert into restaurants
values (405,'Dunkins','219 Baltimore National Pike',4437896547,'open',21237,'WA',3,9.1);


create table categories(
categoryID int,
categoryname varchar(50),
restaurantID int,
primary key(categoryID),
foreign key(restaurantID) references restaurants);

insert into categories
values(301,'fast food',401);
insert into categories
values(302,'pizza',402);
insert into categories
values(303,'burger',403);
insert into categories
values(304,'asian',404);
insert into categories
values(305,'seafood',405);

create table mcategories (
mcategoryID int,
restaurantID int,
category varchar(300),
primary key(mcategoryID),
foreign key(restaurantID) references restaurants);

insert into mcategories
values(11,401,'both italian and pizza');
Insert into mcategories
```

```
values(22,402,'both burger and asian');
insert into mcategories
values(33,403,'both asian and seafood');
insert into mcategories
values (44,404,'both seafood and pizza');
insert into mcategories
values(55,405,'both mexican and fastfood');

create table cart(
cartID NUMBER DEFAULT cid_seq.nextval NOT NULL PRIMARY KEY,
customerID int,
restaurantID int,
foreign key (customerID) references customer,
foreign key (restaurantID) references restaurants);

insert into cart
values(cid_seq.nextval,101,401);
insert into cart
values(cid_seq.nextval,102,402);
insert into cart
values(cid_seq.nextval,103,403);
insert into cart
values(cid_seq.nextval,104,404);
insert into cart
values(cid_seq.nextval,105,405);

create table cart_dish(
cartID int,
dish_quantity int,
dishID int,
primary key(cartID, dishID),
foreign key(cartID) references cart);

insert into cart_dish
values(701, 21,501);
insert into cart_dish
values(702, 1, 502);
insert into cart_dish
values(703, 3,503);
insert into cart_dish
values(704, 4,504);
insert into cart_dish
values(705, 5,505);

create table dish(
dishID int,
restaurantID int,
dish_name varchar(50),
price float,
primary key(dishID),
foreign key(restaurantID) references restaurants);

insert into dish
values(501,401,'Chicken',100.1);
insert into dish
values(502,402,'Mutton',100.2);
insert into dish
values(503,403,'Biryani',100.3);
insert into dish
values(504,404,'Rice',100.4);
insert into dish
values(505,405,'Chapathi',100.5);

create table reviews (
reviewID INT DEFAULT revid_seq.nextval NOT NULL PRIMARY KEY,
customerID int,
restaurantID int,
review_date date,
review_score int,
comments varchar(300),
foreign key(customerID) references customer,
foreign key(restaurantID) references restaurants);

insert into reviews
values(revid_seq.nextval,101,401,date '2022-10-1',90,'Delicious and we probably gonna visit soon');
insert into reviews
values(revid_seq.nextval,102,402,date '2022-12-23',91,'Great food, great staff');
insert into reviews
values(revid_seq.nextval,103,402,date '2022-11-19',92,'Good Indian food');
insert into reviews
values(revid_seq.nextval,104,404,date '2022-9-11',93,'Biryani is the star of the Restaurant');
insert into reviews
values(revid_seq.nextval,105,405,date '2022-3-17',94,'Worth a try');

create table orders(
orderID INT DEFAULT ordid_seq.nextval NOT NULL PRIMARY KEY,
restaurantID int,
customerID int,
order_time timestamp,
delivery_time timestamp,
estimated_time timestamp,
status varchar2(50),
payment_status varchar2(50),
total_cost float,
prices float,
delivery_fee float,
tip float,
taxrate float,
delivery int,
--foreign key (price) reference dish,
--foreign key(taxrate) references tax,
foreign key(restaurantID) references restaurants,
foreign key(customerID) references customer);

insert into orders
values(ordid_seq.nextval,401,101,timestamp '2022-10-13 10:05:00.00',timestamp '2022-10-13 11:05:00.00',
        timestamp '2022-10-13 11:15:00.00','Delivered','Paid',20.1,100.1,37,2,1.5,1);
insert into orders
values(ordid_seq.nextval,402,102,timestamp '2022-11-14 11:23:00.00',null,
        timestamp '2022-11-14 12:56:00.00','In progress','Paid',23.2,100.2,34,8,2.5,2);
insert into orders
values(ordid_seq.nextval,403,103,timestamp '2022-9-21 08:35:50.40',null,
        timestamp '2022-9-21 09:18:37.00','In progress','Unpaid',65.3,100.3,31,2,3.5,2);
insert into orders
values(ordid_seq.nextval,404,104,timestamp '2022-4-11 1:05:57.10',timestamp '2022-4-11 2:47:58.00',
        timestamp '2022-4-11 02:59:00.10','Delivered','Paid',78.14,100.4,6.77,32,4.5,2);
insert into orders
values(ordid_seq.nextval,405,105,timestamp '2022-3-7 09:25:56.00',timestamp '2022-3-7 09:43:47.50',
        timestamp '2022-3-7 09:55:49.00','Delivered','Unpaid',27.15,100.5,6.87,13.4,5.5,1);
insert into orders
values(ordid_seq.nextval,402,105,timestamp '2022-3-7 09:25:56.00',timestamp '2022-3-7 09:43:47.50',
        timestamp '2022-3-7 09:55:49.00','Delivered','Unpaid',27.15,100.5,6.87,13.4,5.5,1);
insert into orders
values(ordid_seq.nextval,403,101,timestamp '2022-3-7 09:25:56.00',timestamp '2022-3-7 09:43:47.50',
        timestamp '2022-3-7 09:55:49.00','Delivered','Unpaid',27.15,100.5,6.87,13.4,5.5,1);


create table order_dishes(
orderID int,
dishID int,
dish_quantity int,
primary key(orderID),
foreign key(orderID) references orders,
foreign key(dishID) references dish);

insert into
order_dishes values(801,501, 1);
insert into
order_dishes values(802,502, 2);
insert into
order_dishes values(803,503, 3);
insert into
order_dishes values(804,504, 4);
insert into
order_dishes values(805,505, 5);

create table payment_record (
paymentID INT DEFAULT payid_seq.nextval NOT NULL PRIMARY KEY,
customerID int,
payment_time timestamp,
orderID int,
payment_amount float,
payment_method varchar(100),
foreign key(customerID) references customer,
foreign key(orderID) references orders);

insert into payment_record
values(payid_seq.nextval,101,timestamp '2022-6-19 12:05:05.00',801,20.1,'Debit card');
insert into payment_record
values(payid_seq.nextval,102,timestamp '2022-7-21 10:45:00.00',802,23.2,'Credit card');
insert into payment_record
values(payid_seq.nextval,103,timestamp '2022-8-13 09:07:00.00',803,65.3,'Zelle');
insert into payment_record
values(payid_seq.nextval,104,timestamp '2022-9-23 07:06:00.00',804,78.14,'Credit card');
insert into payment_record
values(payid_seq.nextval,105,timestamp '2022-2-28 01:43:00.00',805, 27.15,'Paypal');

create table message(
messageID INT DEFAULT msgid_seq.nextval NOT NULL PRIMARY KEY,
customerID int,
message_time timestamp,
message_body varchar(500),
foreign key(customerID) references customer);
```

```
insert into message
values(msgid_seq.nextval,101,timestamp '2022-10-14 10:05:00.00','Paneer Butter Masala is exceptional');
insert into message
values(msgid_seq.nextval,102,timestamp '2022-8-18 12:55:08.50','Rotis could have been better');
insert into message
values(msgid_seq.nextval,103,timestamp '2022-9-21 08:47:19.13','I have had the best food in the recent times at your restaurant');
insert into message
values(msgid_seq.nextval,104,timestamp '2022-12-31 06:57:40.17','Keep up the good work!');
insert into message
values(msgid_seq.nextval,105,timestamp '2022-2-19 11:56:50.34','Would love to visit again');

select * from customer;
select * from discount;
select * from available_discount;
select * from categories;
select * from restaurants;
select * from mcategories;
select * from cart;
select * from dish;
select * from cart_dish;
select * from reviews;
select * from orders;
select * from order_dishes;
select * from payment_record;
select * from message;
/
--INDIVIDUAL FEATURES
SET SERVEROUTPUT ON;
set autoprint on;
--- needed to print ouputs

-- feature 1 (KHYATI BIRUDUGANTI)
CREATE OR REPLACE PROCEDURE add_customer(ns varchar, addr varchar, zip number,
st varchar, em varchar)
AS

r_count NUMBER;
cust_id NUMBER;

BEGIN
--checks whether any customer with the same email exists
SELECT count(*) INTO r_count
FROM customer
WHERE email = em;

IF r_count = 0 THEN

cust_id := custid_seq.nextval;
--a row into customer table with given ID, name, address, state, zip, email,  and credit (as zero)
insert into customer values(cust_id, ns, addr,
zip, st, em, 0);

COMMIT;

dbms_output.put_line('New CustomerID : '|| cust_id);

ELSE
-- it prints a message 'the client already exists' and updates address, state, and zip
dbms_output.put_line('the client already exists');
UPDATE customer
SET address = addr,
zipcode = zip,
state = st
WHERE email = em;

END IF;

END;
/



--- feature 2 ( KHYATI BIRUDUGANTI)

CREATE OR REPLACE PROCEDURE check_email(em varchar)
AS
r_count NUMBER;
p_cid NUMBER;
p_n VARCHAR(256);
p_add VARCHAR(256);
p_zip NUMBER;
p_st VARCHAR(256);
p_em VARCHAR(256);
p_cr NUMBER;
p_del NUMBER;
p_cost NUMBER;
BEGIN
```

```
--checks if there is a customer with that email
SELECT count(*) INTO r_count
FROM customer
WHERE email = em;

IF r_count = 0 THEN
dbms_output.put_line('No Such Customer');


ELSE
-- print out the profile of the customer, including name, address, state, zip code, email, credit, total number of orders with status 2
(delivered) in the last six months and total amount spent (sum of total cost for orders with status 2) in the last six months.
select cu.customerid, names, address, zipcode, state, email, credit,
        count(ord.orderid), sum(total_cost)
        INTO p_cid, p_n, p_add, p_zip, p_st, p_em, p_cr, p_del, p_cost
        from customer cu
        left join orders ord
        on cu.customerid = ord.customerid
        where email=em and TRUNC(order_time) >= sysdate - 180 and ord.status = 'Delivered'
        GROUP BY cu.customerid, names, address, zipcode, state, email, credit;

        dbms_output.put_line(p_cid || ' | ' || p_n || ' | ' || p_add || ' | '
        || p_zip || ' | ' || p_st || ' | '
        || p_em || ' | ' || p_cr || ' | '|| p_del || ' | '|| p_cost);
END IF;

END;
/


--- feature 3 (JYOTHI CHINTKULA)

CREATE OR REPLACE PROCEDURE check_restaurant_by_category(cat varchar)
AS
p_w NUMBER;
p_n VARCHAR(256);
p_add NUMBER;
p_zip NUMBER;
BEGIN
--prints out name, average review score, average wait time, and zip code for restaurants that are open
select restaurant_name, average_review_score, average_wait_time, zip_code
INTO p_n, p_add, p_w, p_zip
from restaurants rs
left join categories ca
on rs.restaurantid = ca.restaurantid
where ca.categoryname LIKE '%'||cat||'%' and rs.current_status = 'open';

dbms_output.put_line(p_n || ' | ' || p_add || ' | ' || p_w || ' | ' || p_zip);

END;
/



--- feature 4 (JYOTHI CHINTAKULA)

CREATE OR REPLACE PROCEDURE check_id(rid number)
AS
r_count NUMBER;
p_n VARCHAR(256);
p_cr NUMBER;
BEGIN
--checks whether this is a valid restaurant ID
SELECT count(*) INTO r_count
FROM dish
WHERE restaurantid = rid;

IF r_count = 0 THEN
dbms_output.put_line('No Such Restaurant');


ELSE
-- prints out all dishes in this restaurant, along with dish name and price.
select dish_name, price
INTO p_n, p_cr
from dish ds
where ds.restaurantid = rid;
dbms_output.put_line(p_n || ' | ' || p_cr);
END IF;

END;
/



--- feature 5( ARJUN VEDULA)
```

```
CREATE OR REPLACE PROCEDURE check_cartid(rid number)
AS
r_count NUMBER;
p_n VARCHAR(256);
p_cr NUMBER;
p_dq NUMBER;

BEGIN
--checks whether that cart ID is valid
SELECT count(*) INTO r_count
FROM cart
WHERE cartid = rid;

IF r_count = 0 THEN
dbms_output.put_line('Invalid Cart Id');


ELSE
-- print dishname, price,dish quantity
select dish_name, price, dish_quantity
INTO p_n, p_cr, p_dq
from cart cs
left join cart_dish ds
on cs.cartid = ds.cartid
right join dish dd
on dd.dishid = ds.dishid
where cs.cartid = rid;

dbms_output.put_line(p_n || ' | ' || p_cr || ' | ' || p_dq);

END IF;

END;
/

--- feature 6 ( ARJUN VEDULA)

CREATE OR REPLACE PROCEDURE check_quantity(cid number , did number)
AS
r_count NUMBER;
d_count NUMBER;

BEGIN
--checks whether the cart with the given ID has that dish
SELECT count(*) INTO r_count
FROM cart_dish
WHERE cartid = cid and dishid = did;

-- dbms_output.put_line('r count: ' || r_count);

IF r_count = 0 THEN
dbms_output.put_line('Invalid input');


ELSE
--check the quantity of that dish
    select dish_quantity into d_count
    from cart_dish ds
    where ds.cartid = cid and ds.dishid = did;

    IF d_count = 1 THEN
--delete that row from the cart and print out 'dish removed'
    DELETE FROM cart_dish
    WHERE cartid = cid and dishid = did;
    dbms_output.put_line('dish removed');

    ELSE
--reduce the quantity of that dish from the cart and print a message saying ‘quantity reduced’.
    UPDATE cart_dish
    SET dish_quantity = d_count - 1
    where cartid = cid and dishid = did;
    dbms_output.put_line('dish quantity reduced');

    END IF;

END IF;


END;
/



--- feature 7 ( SUJIT KANDALA)


CREATE OR REPLACE PROCEDURE update_status(ordid number , ost number, tst timestamp)
```

```
AS
r_count NUMBER;
cid NUMBER;
orc NUMBER;
ostatus varchar(256);
pmet varchar(256);


BEGIN
--checks whether the order ID is valid
SELECT count(*) INTO r_count
FROM orders
WHERE orderid = ordid;

IF r_count = 0 THEN
dbms_output.put_line('invalid order id');


ELSE
SELECT customerid, status, total_cost INTO cid, ostatus, orc
FROM orders WHERE orderid = ordid;

IF ost = 2 THEN
--insert a message into message table for the corresponding customer, with message time as input time, and message body.
UPDATE orders
SET status = 'Delivered'
WHERE orderid = ordid;
insert into message values(msgid_seq.nextval, cid, tst,
'your order ' ||ordid||' has been delivered');


ELSIF ost = 3 then
--updating the status to canceled.
UPDATE orders
SET status = 'Cancelled'
WHERE orderid = ordid;
--inserting into payment table a refund record with a new payment ID, the corresponding customer id and order id, time as input time, and
amount as the negative of the total amount, and payment method the same as the original payment record.
insert into payment_record values(payid_seq.nextval, cid, tst,
ordid, -orc, 'Credit Card');

--inserting a message into message table for the corresponding customer, with message time as input time, and message body saying.
insert into message values(msgid_seq.nextval, cid, tst,
'your order '||ordid||' has been canceled and refund issued!');

ELSE
--if status is 1, that is in progress.
dbms_output.put_line('No additional changes as the status is in progess');


END IF;

END IF;

END;
/


-- feature 8 ( SANJAY SANGARAJU)

CREATE OR REPLACE PROCEDURE write_comment(cid number , rid number , rdt date, rsc number, rvc varchar)
AS
c_count NUMBER;
r_count NUMBER;
avg_rs NUMBER;

BEGIN
--checks if the customer ID is valid
SELECT count(*) INTO c_count
FROM customer
WHERE customerid = cid;
--Checks if the restaurant ID is valid
SELECT count(*) INTO r_count
FROM restaurants
WHERE restaurantid = rid;

IF c_count = 0 THEN
    dbms_output.put_line('invalid customer id');

ELSIF r_count = 0 then
    dbms_output.put_line('invalid restaurant id');

ELSE
--insert a row into review table with the input customer id, restaurant ID, review date, score and comment
    insert into reviews values(revid_seq.nextval, cid, rid, rdt, rsc, rvc);
--update the average review score of the restaurant
    select avg(review_score) into avg_rs
    from reviews
    where restaurantid = rid
    group by restaurantid;

    UPDATE restaurants
    SET average_review_score = avg_rs
    WHERE restaurantid = rid;

    END IF;

END;
/


--- feature 9 ( SANJAY SANGARAJU)
CREATE OR REPLACE PROCEDURE print_all_reviews(rid number)
AS
rc sys_refcursor;
c_count NUMBER;
r_count NUMBER;
avg_rs NUMBER;

BEGIN
--checks whether the restaurant ID is valid
SELECT count(*) INTO r_count
FROM restaurants
WHERE restaurantid = rid;

IF r_count = 0 THEN
    dbms_output.put_line('invalid restaurant id');

ELSE
--prints out all reviews of the restaurant, including review date, score, and comment.
    open rc for select * from reviews where restaurantid = rid;
    dbms_sql.return_result(rc);

    END IF;

END;
/

------------------ START OF GROUP FEATURES ------------------

--- feature 10

--SET SERVEROUTPUT ON;
--set autoprint on;

CREATE OR REPLACE PROCEDURE feature10(cid number , rid number , did number)
AS
c_count NUMBER;
r_count NUMBER;
rd_count NUMBER;
cr_count NUMBER;
ccr_count NUMBER;
crd_count NUMBER;
cust_cartid NUMBER;
dsq NUMBER;
res_stat varchar(64);

BEGIN
--checks whether the customer ID is valid
SELECT count(*) INTO c_count
FROM customer
WHERE customerid = cid;
--checks whether the restaurant ID is valid
SELECT count(*) INTO r_count
FROM restaurants r
WHERE r.restaurantid = rid;
--check the dish whether it belongs to the input restaurant
SELECT count(*) INTO rd_count
FROM restaurants r
inner join dish d
on r.restaurantid = d.restaurantid
WHERE r.restaurantid = rid and d.dishid=did;

SELECT count(*) INTO cr_count
FROM cart
WHERE customerid = cid and restaurantid = rid;


IF c_count = 0 THEN
    dbms_output.put_line('no such customer');

ELSIF r_count = 0 then
    dbms_output.put_line('invalid restaurant id');
```

```
ELSE
--check whether the restaurant is open
    select current_status into res_stat
    from restaurants
    where restaurantid = rid;

    select count(*) into ccr_count
    from cart
    where customerid=cid and restaurantid=rid;

    select count(*) into crd_count
    from cart c
    inner join cart_dish d
    on c.cartid = d.cartid
    where customerid=cid and restaurantid=rid;

    dbms_output.put_line(crd_count);

    IF res_stat = 'closed' then
        dbms_output.put_line('Restaurant Closed');
    ELSIF rd_count = 0 then
        dbms_output.put_line('invalid dish id');
    ELSIF crd_count > 0 then

        select cartid into cust_cartid
        from cart
        where customerid=cid and restaurantid=rid;
        dbms_output.put_line(cust_cartid);

        select dish_quantity into dsq
        from cart c
        left join cart_dish cd
        on c.cartid = cd.cartid
        where customerid=cid and restaurantid=rid and dishid = did
        and c.cartid = cust_cartid;
-- increase the quantity by one
        UPDATE cart_dish
        SET dish_quantity = dsq + 1
        where cartid = cust_cartid and dishid = did;
        dbms_output.put_line('Increase dish quantity');

    ELSIF ccr_count > 0 then
        dbms_output.put_line('Add dish to cart_dish');
        select cartid into cust_cartid
        from cart where customerid=cid and restaurantid=rid;
        insert into cart_dish values(cust_cartid, 1, did);

    ELSE
        cust_cartid := cid_seq.nextval;
        dbms_output.put_line('Added dish to New Cart created with Id: '|| cust_cartid);
        insert into cart values(cust_cartid, cid, rid);
        insert into cart_dish values(cust_cartid, 1, did);

    END IF;

END IF;

END;
/



--- feature 11
CREATE OR REPLACE PROCEDURE feature11(carid number , ddm number , cct timestamp,
                        del_fee OUT NUMBER, ord_amt OUT NUMBER, tax_amt OUT NUMBER, tot_amt OUT NUMBER)
AS
c_count NUMBER;
r_count NUMBER;
cart_price NUMBER;
czip NUMBER;
rzip NUMBER;
rtax NUMBER;

distyp varchar(64);
disper NUMBER;

dis_amt NUMBER;

BEGIN
--checks whether the cart ID is valid
SELECT count(*) INTO c_count
FROM cart
WHERE cartid = carid;

IF c_count = 0 THEN
    dbms_output.put_line('invalid cart id');
```

```
ELSE
--sum up price * quantity of each dish in the cart
        select SUM(cd.dish_quantity * d.price) into cart_price
        from cart c
        inner join cart_dish cd
        on c.cartid = cd.cartid
        inner join dish d
        on c.restaurantid = d.restaurantid
        and cd.dishid = d.dishid
        where c.cartid = carid
        group by c.cartid;

    dbms_output.put_line('sum up price * quantity of each dish in the cart is '||cart_price);

--check the discount start and end time associated with the customer,if the discount is still valid at checkout time
        select cd.zipcode, r.zip_code, d.discount_type, d.discount_amount, t.taxrate
        into czip, rzip, distyp, disper, rtax
        from cart c
        inner join customer cd
        on c.customerid = cd.customerid
        inner join restaurants r
        on c.restaurantid = r.restaurantid
        left join available_discount ad
        on ad.customerid = c.customerid
        inner join discount d
        on d.discountid = ad.discountid
        inner join tax t
        on t.state = r.state
        where c.cartid = carid
        and ad.start_date <= trunc(cct) and ad.end_date >= trunc(cct);
--If the zip code is different, the delivery fee is $5.00
        IF czip <> rzip
            then del_fee := 5;
        ELSE
--If the restaurant is at the same zip code as the customer's home address, the delivery fee is $2.00.
            del_fee := 2;
        END IF;
--Delivery method is pickup
        IF ddm = 2
            then del_fee := 0;
        END IF;
-- discount type is 1 so free delivery
        IF distyp = 1
            then del_fee := 0;
            dis_amt := 0;
--discoun tye is 2 means fixed percent off the total charge
        ELSIF distyp = 2 then
            dis_amt := cart_price * disper;
-- fixed amount off the total charge
        ELSE
            dis_amt := disper;
        END IF;

    ord_amt := cart_price - dis_amt;
    tax_amt := ord_amt * rtax * 0.01;
    tot_amt :=  ord_amt + del_fee + tax_amt;
--delivery fee,tax,amount for dishes are returned
    dbms_output.put_line('the total amount is '||tot_amt);
    dbms_output.put_line('the order amount is  '|| ord_amt || ' the tax amount is  ' || tax_amt || ' the delivery fee is  ' || del_fee);

    END IF;

END;
/

--- feature 12

CREATE OR REPLACE PROCEDURE generating_order(carid number , ordtime timestamp,
                                    ddm number, estime timestamp, tip number, paym varchar)
AS
c_count NUMBER;
del_fee NUMBER;
ord_amt NUMBER;
tax_amt NUMBER;
tot_amt NUMBER;
new_orid NUMBER;
cusid NUMBER;
resid NUMBER;
msg_str varchar(512);

BEGIN
--checks if the cart ID is valid
SELECT count(*) INTO c_count
FROM cart
WHERE cartid = carid;
```

```
IF c_count = 0 THEN
    dbms_output.put_line('invalid cart id');

ELSE

    select customerid, restaurantid
    into cusid, resid
    from cart
    where cartid = carid;
--calling feature 11
    feature11(carid, ddm, ordtime, del_fee, ord_amt, tax_amt, tot_amt);

    new_orid := ordid_seq.nextval;
--inserting a row into orders table
    insert into orders
    values(new_orid, resid, cusid, ordtime, null, estime, 'In progress', 'Paid',
            tot_amt, ord_amt, del_fee, tip, tax_amt, ddm);

--inserting a row into order_dishes
    INSERT INTO order_dishes
    select new_orid, dishid, dish_quantity
    from cart_dish
    where cartid=carid;
--deleting a row from cart_dish and cart
    delete from cart_dish
    where cartid = carid;

    delete from cart
    where cartid = carid;
--defining the message body
    msg_str := 'A new order ' ||  new_orid || ' is placed at Restaurant ' ||  resid || ' with estimated time of ' || to_char(estime-ordtime,
'MI') || ' and amount '|| tot_amt;
--inserting a row into message table
    insert into message values(msgid_seq.nextval, cusid, ordtime, msg_str);
--inserting a row into payments_record
    insert into payment_record values(payid_seq.nextval, cusid, ordtime,
                        new_orid, tot_amt, paym);

    dbms_output.put_line('complete. Details inserted into Orders, Message and Payment Tables.');

    END IF;

END;
/

--- feature 13

CREATE OR REPLACE TYPE cat_array AS TABLE OF VARCHAR2(20);
/
CREATE OR REPLACE PROCEDURE feature13(cid number , mrs number ,  wt number, cats cat_array)
AS
rc sys_refcursor;
c_count NUMBER;
czc NUMBER;
d_count NUMBER;
rcs varchar(256);

BEGIN
--checks whether the customer ID is valid
SELECT count(*) INTO c_count
FROM customer
WHERE customerid = cid;

IF c_count = 0 THEN
    dbms_output.put_line('invalid customer id');

ELSE
--checking the zipcode for the given customer id
    SELECT zipcode INTO czc
    FROM customer
    WHERE customerid = cid;
    -- dbms_output.put_line(czc);

    open rc for select restaurant_name, categoryname, current_status,
    average_wait_time, average_review_score,
    zip_code, address
    FROM restaurants r left join categories c on r.restaurantid = c.restaurantid
    WHERE average_review_score >= mrs --checks whether the average review score greater or equal to the minimal score
    and categoryname in (select * from table( cats )) --under one of the input categories
    and average_wait_time <= wt -- checks whether wait time less or equal to the input wait time
    and substr(r.zip_code,1,4) = substr(czc,1,4); --comparing the customer zip code with restaurant zip code
--print out name of restaurant, address, status, average review score, zip code, and average wait time.
    dbms_sql.return_result(rc);
    END IF;
```

```
END;
/



-- feature 14

CREATE OR REPLACE PROCEDURE feature14(cid number)
AS
rc sys_refcursor;
c_count NUMBER;
cust_rest NUMBER;
custs NUMBER;
rec_rest NUMBER;

BEGIN
--checks whether the customer ID is valid
SELECT count(*) INTO c_count
FROM customer
WHERE customerid = cid;

IF c_count = 0 THEN
    dbms_output.put_line('invalid customer id');

ELSE
--Finds restaurants that customer has placed an order
    SELECT restaurantid into cust_rest
    FROM orders
    WHERE customerid = cid;
--Finds other customers who have placed orders in any restaurant excluding the input customer
    SELECT distinct customerid into custs
    FROM orders
    WHERE restaurantid in cust_rest and customerid != cid ;
    dbms_output.put_line('other Customer Ids: '||custs);
--finds other restaurants these customers go to excluding the restaurants in the previous step
    SELECT distinct restaurantid into rec_rest
    FROM orders
    WHERE customerid in custs and restaurantid not in cust_rest;
    dbms_output.put_line('restaurant Ids: '||rec_rest);
--Prints out id and names of these restaurants, their addresses and average reviews.
open rc for select restaurantID, restaurant_name, address, average_review_score
from restaurants
where restaurantid in rec_rest;
dbms_sql.return_result(rc);

    END IF;

END;
/
SET SERVEROUTPUT ON;
set autoprint on;
--- needed to print ouputs


-- feature 1 ( KHYATI BIRUDUGANTI)
select * from customer;
--this email already exists so it prints 'the customer already exists'
EXECUTE add_customer('Clinton', '829 Taylor Lane', 22256, 'Texas', 'tonystark@gmail.com');

--This is an new email so this customer is getting inserted into the customer table with this mail id
EXECUTE add_customer('harsha', '888 mount ridge', 22234, 'swizerland', 'harsha@gmail.com');
select * from customer;
/

-- feature 2( KHYATI BIRUDUGANTI)
select * from customer;
EXECUTE check_email('rohanp@gmail.com');
--this prints no such customer as there are no customers with 'rohanp@gmail.com'


EXECUTE check_email('tonystark@gmail.com');
--This prints the details of customer in last six months and status is delivered
select * from customer,orders where customer.customerID=orders.customerID;

/

--- feature 3 (JYOTHI CHINTAKULA)
select * from  restaurants, categories where restaurants.restaurantid=categories.restaurantid;
EXECUTE check_restaurant_by_category('sea');
--here the input(sea) is a part of seafood so this prints restaurant name, average review score, average wait time, zip code where restaurants
that are open

/

--- feature 4(JYOTHI CHINTAKULA)
EXECUTE check_id('409');
--This prints out ' no such restuarant'
```

```
EXECUTE check_id('403');
--This  print out all dishes in this restaurant, along with dish name and price.
--There is only one dish name 'biryani'  with id =403 so it prints that.
select * from dish;
/

--- feature 5( ARJUN VEDULA)
-- can conflict with later features (CreateTables.sql rerun required).
EXECUTE check_cartid('708');
--This is an invalid cart id

EXECUTE check_cartid('701');
--This prints every dish in the cart with the id 701 including dish name, price, quantity .

select * from cart, cart_dish where cart.cartid=cart_dish.cartid;
--There is only one dish with id 501 in the cart with 701
select  * from dish;



/

--- feature 6(ARJUN VEDULA)
EXECUTE check_quantity('701', '502');
--There is no dish with id = 502 in the cart which has id = 701, hence it prints "invalid input"
select * from cart_dish;
EXECUTE check_quantity('701', '501');
-- dish quantity for 701/501 is 21, it should be now reduced to 20, and prints "quantity reduced"
EXECUTE check_quantity('702', '502');
-- dish quantity for 702/502 is 1, will be removed now
select * from cart_dish;
/

--- feature 7(SUJIT KANDALA)
select * from orders;
EXECUTE update_status(808,1,timestamp '2022-11-13 10:09:00.00');
--This is an invalid case
EXECUTE update_status(806,1,timestamp '2022-11-13 10:09:00.00');
--Here the status is 1 therefore no additional changes are done.
select * from orders;
EXECUTE update_status(801,2,timestamp '2022-11-13 10:09:00.00');
--Here the status is 2 so with id 801 a new row is inserted into message table
select * from message;
EXECUTE update_status(802,3,timestamp '2022-11-14 10:11:00.00');
--here the status is 3 so with id 802 a new row is inserted into message and a new row is inserted into payment_record.
select * from message;
select * from payment_record;

/
--- feature 8(SANJAY SANGARAJU)

EXECUTE write_comment(109,402,date '2022-11-23',91,'very nice and delicious');
select * from customer;
--This is an invalid case becuase there is no customer with customer id 109
EXECUTE write_comment(102,411,date '2022-11-23',91,'very nice and delicious');
select * from restaurants;
-- This is also an invalid case becuase there is no restaurant with restuarant id 411
EXECUTE write_comment(102,402,date '2022-11-23',91,'very nice and delicious');
select * from reviews;
select * from restaurants;
--The review score for restaurant with id 402 is 91,92,91 therefore avg review score is 91.3333
/

--- feature 9(SANJAY SANGARAJU)
EXECUTE print_all_reviews(411);
--There is no restaurant with id = 411. Hence, it prints invalid restaurant id
EXECUTE print_all_reviews(402);
--print out all reviews of the restaurant, including review date, score, and comment.
/

--GROUP FEATURES---
--- feature 10
EXECUTE feature10(106,403,501);
--no such customer
EXECUTE feature10(102,406,501);
--invalid restaurant id
SELECT * FROM restaurants;


EXECUTE feature10(102,402,501);
-- restaurant closed

EXECUTE feature10(102,405,501);
--invalid dish id as the dish does not belong to the input restaurant
select * from dish;

select * from cart_dish;
```

```
EXECUTE feature10(101,401,501);
-- increases dish quantity for existing cart
select * from cart_dish;



EXECUTE feature10(102,401,501);
-- creates new cart and add dish
select * from cart_dish;
/

--- feature 11
--invalid cart id
DECLARE
del_fee NUMBER;
ord_amt NUMBER;
tax_amt NUMBER;
tot_amt NUMBER;
BEGIN
feature11(700, 1, timestamp'2022-10-13 11:05:00.00', del_fee, ord_amt, tax_amt, tot_amt);
END;
--return delivery fee,order amount, tax, and amount for dishes
DECLARE
del_fee NUMBER;
ord_amt NUMBER;
tax_amt NUMBER;
tot_amt NUMBER;
BEGIN
feature11(701, 1, timestamp'2022-10-13 11:05:00.00', del_fee, ord_amt, tax_amt, tot_amt);
END;

/
--- feature 12
--invalid cart id
EXECUTE generating_order(710,timestamp '2022-10-13 10:05:00.00', 1, timestamp '2022-10-13 11:15:00.00', 2, 'debit card');
--valid cart id
EXECUTE generating_order(701,timestamp '2022-10-13 10:05:00.00', 1, timestamp '2022-10-13 11:15:00.00', 2, 'debit card');
select * from orders;
select * from order_dishes;
select * from message;
select * from payment_record;
/

--- feature 13
--invalid customer id
EXECUTE feature13(106, 1, 5, cat_array('seafood'));
--valid customer id
EXECUTE feature13(105, 3, 15, cat_array('seafood','fast food'));
/

--- feature 14
--- invalid customer
select * from customer;
EXECUTE feature14(116);
--valid customer and prints out id and names of these restaurants, their addresses and average reviews.

select * from orders;
select * from restaurants;
EXECUTE feature14(103);
/
```