IS 620 Advanced Database Projects
Fall 2022
Group Project
Online Food Ordering System

Overview

You will be assigned into groups of four to five people in this project. Please read the whole document carefully before starting your project.

Your assignment is to design a online food ordering system. You will design the database, insert some sample data, and implement a set of required features. Each feature will be implemented as one or more Oracle PL/SQL procedures/functions. You do **NOT** need to write a graphic user interface.

Assumptions:

You can make the following assumptions in this project.

1. The system will store information about customer, including customer ID, name, address, zip code, state, email, credit (the company may give a customer credit for canceled orders).

2. The system store information about types of discounts offered to customers, including discount ID, discount description, discount type (1 means free delivery, 2 means a fixed percent off the total charge, 3 means a fixed amount off the total charge (say $20)), and discount amount. E.g., if discount type is 2, and amount is 0.1, means 10% off each order. If discount type is 3 and amount is 10, means $10 off each order.

3. The system stores sales tax rate for each state.

4. The system stores each customer's available discounts, including customer ID, discount ID, discount start and end dates. The discount will apply between the start and end dates.

5. The system will store information about categories, each with a category ID, category name. E.g., Sample categories could be Fast food, Burgers, Pizza, Seafood, Asian, Mexican, Italian.

6. The system stores a list of restaurants, each restaurant has a restaurant ID, restaurant name, address, phone number, current status (open or closed), zip code, state, average wait time, and average review score.

7. The system stores the categories for each restaurant. The same restaurant may fall into multiple categories, e.g., both Italian and Pizza.

8. Each restaurant has a number of dishes. Each dish has a dish ID, restaurant ID, dish name, price.

9. Each customer can leave reviews for a restaurant, including review ID, customer ID, restaurant ID, review date, review score, comments.

10. A customer can select a restaurant and add one or more dishes a shopping cart. The cart table has cart id, customer id, restaurant id.

11. A separate table stores information about dishes put in a cart. These dishes must come from the same restaurant. The same dish can also appear multiple times in the same cart so you can use a quantity column to store the quantity of each dish.

12. A customer can place an order for all items in the shopping cart. The order has order ID, customer ID, restaurant ID, order time (time order placed), delivery time (null if not delivered yet), estimated time it is ready, status (in progress, delivered or canceled), payment status (paid or not paid), total cost. The order also contains a flag indicating delivery method (1 is delivery, 2 as pickup). If the order will be delivered, total cost includes prices for all ordered dishes, a delivery fee, tip, and sales tax (based on the state). If the order will be picked up, the total cost just includes prices of dishes and sales tax. The total prices should consider discount received by the customer (e.g., if it is 10% off, it is 10% off each dish's price).

13. The system needs to store dishes in an order, including order id and dish id.

14. The order also contains payment record for a customer, including payment ID, customer ID, payment time, order ID, payment amount, payment method (only contains whether it is credit/debit card, apple pay, or paypal).

15. The systems stores a message table which contains message ID, customer ID, message time, and message body.

Features: There are nine individual features (features 1 to 9) and five group features (features 10 to 14). Individual features will be assigned to individual group members and will be graded individually. Individual features are divided into five feature sets, one for each member. If your group only has four members then your group only need to implement any four feature sets. Group features will be assigned to the whole group and graded group-wise. **If your group has five members, your group need to implement all group features. If your group has four members, your group needs to implement group feature 10, 11, 12 plus one more group feature (feature 13 or 14).** The first two deliverables are also graded group-wise.

Member 1:

Feature 1: create a new customer. Input includes customer name, address, state, zip, email. This feature checks whether any customer with the same email exists. If so, it prints a message 'the client already exists' and updates address, state, and zip. Otherwise, it generates a new customer ID (using sequence) and insert a row into customer table with given ID, name, address, state, zip, email and credit (as zero). Please also print out the new customer ID.

Feature 2: Given a customer email, first check if there is a customer with that email. If not print a message now such customer. Otherwise 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.

Member 2:

Feature 3: Search restaurant by category. Input is a part of category name (e.g., for fast food the input could be just 'fast'). Please print out name, average review score, average wait time, and zip code for restaurants that are open and matches the input category name.

Feature 4: Show dishes offered by a restaurant. Input is a restaurant ID. The procedure first checks whether this is a valid restaurant ID. If not, please print a message 'no such restaurant'. Otherwise print out all dishes in this restaurant, along with dish name and price.

Member 3:

Feature 5: Show all dishes in a shopping cart. Input is a cart ID. First checks whether that cart ID is valid. If not print a message invalid cart ID. If the ID is valid, print out every dish in the shopping cart, including dish name, price, quantity.

Feature 6: Remove a dish from shopping cart. Input includes dish ID and cart ID.
First check whether the cart with the given ID has that dish. If not print a message 'Invalid input'. If the input ID is valid, check the quantity of that dish. If it is more than one, then reduce the quantity of that dish from the cart and print a message saying 'quantity reduced'. If the quantity is one, delete that row from the cart and print out 'dish removed'.

Member 4:

Feature 7: Update status of an order. Input is order ID, new status (1 is in progress, 2 is delivered, 3 is canceled), and input time. The procedure does the following:
1) First checks whether the order ID is valid. If not print a message saying invalid order id.
2) Update the status of the order to the input status. In case new status is in progress, no additional action is needed.
3) In case new status is 'delivered', insert a message into message table for the corresponding customer, with message time as input time, and message body saying 'Your order X has been delivered!' where X is the order ID.
4) In case new status is 'canceled', update the status to canceled, insert a message into message table for the corresponding customer, with message time as input time, and message body saying 'Your order X has been canceled and refund issued!' where X is the order ID. Please also insert 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 in the order, and payment method the same as the original payment record.

Member 5:

Feature 8: Enter a review. Input includes a customer ID, a restaurant ID, a review date, a review score and review comment. This procedure does the following:
1) first checks if the customer ID is valid. If not print a message saying invalid customer ID.
2) Check if the restaurant ID is valid. If not print a message saying invalid restaurant ID.
3) if both are valid, insert a row into review table with the input customer id, restaurant ID, review date, score and comment.
4) update the average review score of the restaurant to reflect the new review.

Feature 9: Display all reviews of a restaurant. Input is restaurant ID. First checks whether the restaurant ID is valid. If not print a message. Then print out all reviews of the restaurant, including review date, score, and comment.

Group features: Group features will be graded group-wise. Features 10-12 must be implemented by all groups. Group of five members also need to implement features 13 and 14. Group of four members can choose one out of feature 13 and 14.

Feature 10: Add a dish to shopping cart. Input includes customer ID, restaurant ID, and a dish ID.

1) First check whether the customer ID is valid. If not print out a message no such customer.
2) Then check whether the restaurant ID is valid and the restaurant is open. If not print out invalid restaurant ID or the restaurant is closed.
3) Finally check the dish whether it belongs to the input restaurant. If it does not print out message invalid dish ID.
4) Otherwise where there is an existing shopping cart for the customer. If the cart does not exit, create a new cart for the customer and restaurant and print out the new cart ID.
5) Now you can check whether the dish is already in the cart. If so just increase the quantity by one. Otherwise insert a new row to the table keeps dishes in a cart.

Feature 11: Compute total amount for dishes in a cart. Input: cart ID and a checkout time, delivery method (1 deliver, 2 pickup). The procedure first checks whether the cart ID is valid. If not it prints a message invalid cart ID. It then computes the total as follows (all steps MUST be done in just ONE procedure):

1) sum up price * quantity of each dish in the cart

2) discount is then applied for the customer if the discount is still valid at checkout time (check the discount start and end time associated with the customer).

For fixed percentage discount, the sum in step 1) is multiplied by 1-discount rate. E.g., 10% discount means the price should be multiplied by 0.9. For fixed amount discount, the sum is deducted by that amount. Free delivery discount will be handled in step 3).

3) delivery fee is added to the result in step 2 if the delivery method is not pickup. If the restaurant is at the same zip code as the customer's home address, the delivery fee is $2.00.

If the zip code is different, the delivery fee is $5.00. The delivery fee is zero if a free delivery discount is valid for the customer at time of order.

4) sales tax is added to the total computed in step 2) based on the state of the restaurant.

This is the total to be returned. Please also use output parameter to return delivery fee, tax, and amount for dishes (output of step 2).

Feature 12: Generate an order with dishes in a shopping cart. The input is a cart ID, order time, deliver method (1 deliver, 2 pickup), an estimated time to deliver or pickup, tip, and a payment method (1 credit/debit, 2 apple pay, 3 paypal). You can assume that the payment process is handled by a third party so you don't need to worry about it.

The steps are:

1) check if the cart ID is valid, if not print a message invalid cart ID and stop.

2) call feature 11 to compute total amount due.

3) insert a row into orders table with a newly generated order ID, customer ID as the cart's customer ID, restaurant ID as the cart's restaurant ID, order time as the input order time, delivery time is null, estimated time as the input estimated time, status is in progress, delivery method is the input delivery method, delivery fee, tax, and total are returned in feature 11, and tip is the input tip.

4) insert dishes in the shopping cart into the table that stores dishes in the order. Delete the shopping cart row and dishes in the cart.

5) insert into message table a message with customer ID as the customer ID associated with the cart, message time as the order time, and body as

'A new order X is placed at Restaurant Y with estimated time of Z and amount A', where X is the order ID, Y is the name of the restaurant, Z as input estimated time (in minutes), and A is total amount.

6) insert a payment record to payment table with order ID as the order ID in step 3),

payment time as order time, payment amount as total computed in step 2, and payment method as input payment method.

Feature 13: Advanced search.

The input is a customer ID, a list of category names, a minimal review score, and a wait time.

1) The procedure first checks whether the customer ID is valid. If not print a message invalid customer.

2) The procedure returns all restaurants that satisfy ALL of the following conditions:

a) under one of the input categories;

b) with an average review score greater or equal to the minimal score,

c) a wait time less or equal to the input wait time.

d) having a zip code either the same as the customer's zip code or differ only by the last digit (e.g., if the home zip code is 21042 then only restaurants in zip code 2104X will be returned where X is 0 to 9)

3) Please print out name of restaurant, address, status, average review score, zip code, and average wait time.

Hint: create a varray type to store list of input categories.

To get first 4 digits of zipcode, use function substr(string, first position, length).

Feature 14: Restaurant recommendation.

The input is a customer ID. This procedure does the following.

1) check whether the customer ID is valid. If not print an error message and stop.

2) Find restaurants that customer has placed an order. Please exclude the same customer in the input. Please print out these restaurants' IDs.

3) Find customers who have placed orders in any restaurant in step 2). Please print out these customers IDs.

4) Find other restaurants these customers (in step 3) go to. Please exclude those restaurants in step 2) (i.e., the customer already visited).

5) Print out id and names of these restaurants, their addresses and average reviews.

Here is an example that helps you understand feature 14 requirement, but your code needs to work for all possible cases. Customer 1 has ordered from restaurant 1 and 3, and customer 2 has ordered from restaurant 1,2. Input is customer 2. Step 2) prints restaurant 1 and 2 (these are ordered by customer 2), step 3 prints out customer 1 (who has also ordered from restaurant 1), and step 4 recommends restaurant 3 (which customer 1 has placed order but not by customer 2).

Deliverables:

There will be 4 deliverables. D1, D2, D4 will be due midnight of the due date. D3 is a demo. Delayed submission will result in a penalty of 30% of your score (e.g., if your score for part 2 is 20 but you are late, your score will be 14). The final presentation (D3) is due at class time and no delay is allowed.

1. 10%. Due 9/21. Project Management Schedule.
    a. Include team members and a timeline showing each phase of your project with its tasks and time duration, for the entire effort.
    b. It is expected that every member should participate in all phases of the project. For example, every member should be involved in writing the code.
    c. indicate which group member will do which set of individual features and how your group plan to work on group features.
    d. Tasks should include system design, populating tables, writing code, testing code, running example queries, writing documents, preparing for presentation, etc. Smaller milestones shall be set for deliverable 3 and 4.
    e. This deliverable will be graded based on whether items a) to d) are included and whether the schedule is reasonable (e.g., enough time is left for testing and integration).

2. 25%. Due 10/12. Design Document which includes the following:
    a. ER diagram of the database. You don't have to follow exact notations of ER diagram, but need to show tables, columns, primary keys, and foreign key links.
    b. SQL statements to drop and then create database tables and to insert some sample data (at least 5 rows per table).
    c. Specification for each required feature. The specification should include a description of input parameters and output (usually screen outputs), and an example of how a user can call this feature (e.g., exec XXX(…) where XXX is the procedure name). You don't need to implement any of these features at this point.

    Here is an example for feature 1: create a new customer
    Input: customer name, address, state, zip, email.
    Output: screen output of 'the client already exists' if a client with the same email exists and update of address, zip code and state. Otherwise a new row inserted client table and a screen output of the newly assigned client ID.

Example of calling this feature:

```
---- regular case
exec add_customer('Milly', '13459 Centennial Lane', 'MD','21042','milly@gmail.com');

select * from customer;
-- existing case (adding the same client again)
exec add_customer('Milly', '13458 Centennial Lane', 'MD','21042','milly@gmail.com');
```

3. 30%. Due 12/19. Demonstration. Your work will be demonstrated to the class in real time, where you will present the design of your system and you will run a demo. You don't need to submit anything.

4. 35% Due 12/19. The code should include:
    a. Drop table statements to drop tables if they exist (remember the order of drop should be inverse of create). Drop sequence statements.
    b. Create table statements, create sequence statements.
    c. Insert statements
    d. Create procedure statements (with code for the procedures). Each feature can be implemented as one or multiple PL/SQL procedure (in the procedure you may call other procedures or functions). Please include some comments in your code explaining the major steps. You should use create or replace to avoid procedure name conflict.

e. Demo script to show that all your features work correctly. The script shall include enough examples to test different cases. E.g., for feature 1, one example for new user (email is not in database) and one example for existing user (using existing email). Please include:
    i. PL/SQL script to call the appropriate PL/SQL procedure for this feature. E.g., exec procedure-name(parameter values)
    ii. Explanation of what should be the correct output. The output could be updated tables (you can have some select statement to show the updated tables), some print out, etc.
    iii. Make sure you have tested your examples from beginning to end. Remember that database tables may have been changed in the process. So you may need to start with a clean database (i.e., right after you execute all the drop table, create table, and insert statements).

Grading Guidelines

What I look for while grading software code (deliverable 4):

1. Existence of code and whether all code can be compiled without any error.
2. Comments: Both descriptive and inline for every procedure/function
3. Software quality
    a. Whether it is correct (giving correct results).
    b. Whether it is complete and clear.
    c. Efficiency of code. You shall not use too many SQL statements, and you shall put as much work as possible in SQL. For example, if you can do a join, do not use two select statements and then do a join in your program.
    d. Whether it has considered all special cases such as whether a user has already registered in Feature 1.

Regarding the presentation of your project: Each student must participate in the project demonstration by presenting to the entire class some slides. You will be graded on:

1. Timeliness of presentation
2. Presentation Style
3. Demo (running the code)

For the demo, you will be graded on the following items:

1. Existence of tables and data. You need to have at least 3 rows in each table.
2. The correctness of features. This can be shown by checking whether the screen output is correct and the database has been updated correctly.

Each member of the team shall contribute more or less equally. It is unfair for a few members to do most of the work while others do less. You will be asked to evaluate your teammate's effort at the end of the project. The instructor will adjust the grade based on the evaluation. Normally if most of your teammates agree that you do not contribute at all or contribute too little (e.g., your group has 4 members and you contribute only 5%), you may lose up to 80% of your project grade. If your teammates agree that you contribute much more than anyone else (e.g., your group has 4 members and you contribute 40%), you may gain up to 20% of your project grade (but not exceeding 100% of project grade). A peer evaluation will be conducted at the end of the semester to determine the contribution of each team member.

Tips:

1. Work as a team. Each member can do individual features by yourself but should work on other parts of the project including group features as a team. This means do NOT miss group meeting, do not miss internal deadlines, and help each other for tasks that belong to the whole group. You should also divide up group tasks fairly and according to everyone's strength.
2. Start early. Do not wait until last month to start coding. Do not wait until one week before the demo to start putting things together. Past experiences show that more than 50% of time shall be devoted to testing and putting things together.
3. Learn how to debug SQL and PL/SQL code. Most of time the error is from the SQL part of your code. So you can test SQL part separately (e.g., by copy & paste the SQL statement in a cursor and replace PL/SQL variables/parameters with values). You can insert screen output statements to check intermediate results. Oracle also returns error messages and error code. You can google the error messages and error code to find possible causes. You may also use Oracle SQL Developer which allows you to insert break points during debugging.
4. It is highly recommended to use SQL Developer rather than the web interface for the project.
5. Use homework, in class exercises, and programs in slides as templates of your PL/SQL program. For example, if you need to write a cursor, find a cursor example and use it as a starting point.
6. Make sure special cases are handled.
7. At demo time, different data in the database may lead to different results. So usually you will start with a standard database (with a fixed set of tables and rows), and keep track of the sequence of the demo (e.g., a course can only be scheduled if it has been added first).