IS 420: Database Application Development
Spring 2023

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 five individual features and five group features. Each member needs to implement one individual feature. So if your group has X members your group will implement any X individual features. Each group also needs to implement X (X=size of group) group features. For example, if you group has five members, your group will do all features. If you group has four members, your group will implement any four individual features plus any four group features.

Individual features will be graded individually (you group member's individual feature will have no impact on your grade), but group features will be graded group-wise. So each group should work together to make sure the group features are done correctly.

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.

Member 2:

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 3:

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.

Member 4:

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 5:

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.

Group features:

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'.

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.

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.

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.

Deliverables:

1. 10%. Due 2/21. Project Management Schedule.

    a. Include team members and a timeline showing each phase of your project with its activities and time duration, for the entire effort.

    b. It is expected that every member should participate in all phases of the project.

    c. Please specify which feature is assigned to which member (for group features it is still possible to assign a lead for each feature).

    d. Activities 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 such as 2-3 weeks are left for testing and integration).

2. 25%. Due 3/14. 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 create database tables and to insert some sample data (at least 3 rows per table). Please include drop table and drop sequence statements before create table, create sequence and insert.

   c. Specification for each required feature. The specification should include a description of input parameters and output (usually printing a message), and a few test cases (normally there should be one normal case and a few special cases). You don't need to implement any of these procedures at this point.

3. 35%. Due 5/10. Presentation of database design and demonstration of each feature. The project demo will be online (through Webex). You can sign up for the time of presentation/demo through a google form. To demo each feature, you need to prepare a couple of test cases, usually one normal case and the rest as special cases. For each test case you need to be able to explain why your answer is correct (this can be typically done by showing some tables or screen output). The instructor may ask you to only demo some of the features but you need to have all features ready.

4. 30%. Due 5/23. Please upload final code through blackboard. The code should include:

   a. Drop table and sequence statements to drop tables if they exist (remember to use cascade constraints).

   b. Create table statements and create sequence statements

   c. Insert statements

   d. Create procedure statements (with code for the procedures). Each feature can be implemented as one 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. Test script to show that all your features work correctly. The script shall include some 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).