UMBC

IS620 Advanced Database Projects

Fall 2023

Group Project

An Online Shopping DB System

Assume that your team has been in contract with the headquarters of a company that wants to create a new online marketplace, and compete against the known behemoths, like Amazon, Walmart, etc. Your team is to provide software that enables this new online platform based on an Oracle database, and performing several operations in the software. These operations actually manipulate various components of a database. Example operations are to add new products to sell, store customer information, enable ordering of products, issue invoices, generate recommendations to customers for new products, etc. In addition, there are reports to be run for the managers of the enterprise, to observe how the business is doing overall, or during a specific month or quarter of the year, which products sell, which do not, etc.

Your team will implement the above by designing a database, creating the appropriate tables, and then writing, testing and deploying PL/SQL stored procedures that implement the operations and the reports.

This project is a group project. The work is to be divided into tasks which are to be distributed among the group members, each one will work individually on their own tasks and also consolidate the individual tasks with the tasks of the other group members into an integrated project. The entire project is to be completed incrementally based on deliverables. Some deliverables are group deliverables meaning that the entire group must work together to accomplish them, while other deliverables are individual where each group member is to work on their own and upload their completed deliverable separately.

TABLES

Customers: Contains customer ID, name, email, city, state, zip.

Product Categories: Contains information about the various categories of products for sale. Attributes are: Category ID, category name, description.

Products: Product ID, product name, available quantity in store, unit price, category ID.

Orders: These are the contents of the shopping carts where customers add products for purchase. Attributes: Order ID, customer ID, product ID, quantity, order date. A customer may have multiple orders, and in each order there can be multiple items (products).

Credit Cards: Customer ID, Credit Card #, Credit Card Type (VISA, MC, Discover, AMEX), expiration year, expiration month. A customer may have multiple credit cards.

Invoices: Invoice ID, Order ID, Customer ID, Credit Card #, Amount

Reviews: Can be supplied by anyone with an email (not just customers). Attributes: Review ID, Product ID, Reviewer email, Stars Given, Review text.

Recommendations: Information about possible recommendations to be suggested to a customer. This is going to be calculated based on other products in the same category as those that a customer has bought and also according to the reviews of the products. Attributes: Recommendation ID, Customer ID, Recommended Product ID, Recommendation Date. There can be many recommendations per customer based on products they ordered in the past.

OPERATIONS

These are the operations that need to be implemented by each member and they correspond to the functional operations within our restaurant ecosystem. Their purpose is to implement each numbered operation listed below as a PL/SQL **stored procedure** or **function**. For example, Add Product Category / Product / Customer / Order / Invoice / Review / Recommendation, etc. Create reports that show how financial information regarding orders, inventory, income, etc. A detailed set of operations to be performed is shown below.

NOTE: All IDs must be automatically created using sequences.

- **Member 1**: Responsible for the Customers and CreditCards tables. Must create these tables.

    1. Add_Customer: This procedure takes as input all necessary information to insert a new customer to the Customers table.

    2. Show_all_customers_in_state: Input parameter is a state. Print a list of all customers in that state including customer name, email, address, credit card numbers, and type.

    3. Add_CreditCard: This procedure takes as input the CustomerID and credit card information and inserts a new credit card to the CreditCards table. Use a helper function to find the customer ID that is needed in adding a credit card.

    4. Report_Cards_Expire: Given as input a specific date, the online store needs to notify customers about their pending credit card expiration. Provide a report with a list of every single customer whose credit card will expire in the time window of two months before, until the given date. Show the customer last name, first name, credit card #, credit card type, expiration year, month. Sort the output by last name, then first name.

- **Member 2**: Responsible for Product Categories and Products. Must create these tables.

    5. Add_Category: This procedure takes as input all necessary information to insert a new category to the Category table.

    6. Add_Product: This procedure takes as input all necessary information to insert a new product to the Products table. Use a helper function to find the category ID that is needed in adding a product.

    7. Update_Inventory: When an order is placed, the Products table must be updated with the new remaining quantity. Create a procedure that will take as input the productID and the units (quantity) of a product in an order, and will update the inventory in the Products table to reflect the new quantity after the order.

8. Report_Inventory: This procedure will display a report of available inventory. For each product category print a list with the name of the category and the quantity of all available products in that category.

- **Member 3**: Responsible for Orders. Must create this table.

9. Place_Order: A customer orders a product at a given quantity, in a given date. Input parameters are the email of the customer, the product name, the quantity, and the credit card number to charge. User helper functions to find the customer ID, product ID. Every time the Place_Order procedure is called, the Update_Inventory procedure (created by Member 2) must be called from within the Place_Order procedure. In addition the Invoice_Customer procedure (created by Member 4) must be called.

10. Show_Orders: Show all orders in the online market place: Customer name, product, quantity ordered, amount charged. At the end add an additional line showing the grand total of the number of all orders placed in the system.

11. Report_Orders_by_State: Create a procedure that takes a state as input. Report all orders placed in the online market from that state by customer and print for each customer: the name, email, total number of orders placed and total amount of dollars spent. At the end, print the grand total amount of dollars spent.

12. Report_Low_Inventory: Create a trigger which automatically shows on the screen, all products that are below 50 units in stock. Show Product ID, product name, available quantity. In a separate call (outside of the trigger) invoke Update_Inventory procedure (created by Member 2) separately to restock a specific product. Inputs: Product name, units (quantity) to be added to existing quantity.

- **Member 4**: Responsible for Invoices. Must create this table.

13. Invoice_Customer: This procedure enters a record in the Invoice table. Inputs: Order ID, Customer ID, Credit Card #, Amount. You may need to call helper functions to find the required IDs.
14. Report_Best_Customers: This procedure takes as input a dollar amount and will produce a report of all customers (names, and total amount spent) who have spent more than the input amount in the online store.
15. Payments_to_CC: This procedure calculates the fee that the online store must pay to credit card companies for the orders that were placed on it. The fees per purchase are: (VISA: 3%, MC: 3%, AMEX:5%, Discover: 2%). Print a list with five lines each showing the total fee to be paid to each credit card type (VISA, MC, AMEX, Discover)
16. Thrifty_Customer: The store needs to identify all the stingy customer in order to give them an incentive to make purchases. Create a procedure that takes as input a number X, and print the X most stingy customers based on dollar amount of purchases sorted by amount spent in descending order (e.g. if X=5 print the 5 customers who spent the least)

- **Member 5**: Responsible for the Reviews and Recommendations. Must create these tables.

17. Add_Review: This procedure receives a review from a person (anyone with an email) for a particular product and adds the review in the Reviews table. Input to this

procedure is the reviewer email, stars given (a number from 1-5), product ID, and review text. You will need to call a helper function to find the product ID.

18. Buy_Or_Beware: This procedure takes as input a number X and prints the best X products and the worst X products based on the average star ratings of customers who wrote a review for each product. It should first print: 'Top rated products' then for each line show the average number of stars (in descending order), product ID, product name, and the standard deviation. Then print 'Buyer Beware: Stay Away from…" then each consecutive line shows the average number of stars (in ascending order), product ID, product name, and the standard deviation.
19. Recommend_To_Customer: This procedure calculates a recommendation for a customer. The input is the customer ID. This procedure will insert a new recommendation in the Recommendations table for a specific input customer as follows: it will identify the best rated product (based on reviews) in the same category that the customer has bought a product before, but has not bought that particular recommended product yet.
20. List_Recommendations: Provide a report that lists all customer recommendations. This report prints the name of the customer, the recommended product, and its average stars (one line per recommended product).

- **Member 6** (only for teams that have 6 members. If your team does not have six members you do not have to implement these operations). Responsible for manager reports.
    21. Income_By_state: This procedure prints all income to the online store by state. Each line lists the state and then calculates and prints the total amount of purchases (in dollars) by customers in that state.
    22. Best_Selling_Products: This report lists the top X (where X is an input parameter) best selling products based on total number of units sold. Each line lists the product name, category, total number of units sold, and total amount collected (in dollars) in descending number or units sold.
    23. Recommendations_Follow_Up: The store managers want to find out if the recommendations actually work. They want to identify the customers who actually bought the personally recommended products after they were notified of the recommendation. Create a report with customer names, recommended product name, and if they bought it after it was recommended print 'recommendation followed'; if not print 'recommendation not followed yet'. One product per line.
    24. Products_Ordered_By_Time_Interval: This procedure takes as input a start and an end date. Within this time interval identify all products that were ordered. For each product print its name, the total number of units that were ordered, and the total number of customers who ordered it.

Helper functions/procedures: You will need some helper functions to make implementation easier and more structured. Feel free to add more procedures and/or functions if needed. You will definitely need to create a FIND_table_name_ID function that returns the ID of a row in that table. For example, several operations require customer ID. How can you find the customer ID? You create a function FIND_CUSTOMERS_ID (email). Given an email of a customer it will return the customer ID. FIND ID functions must exist as follows:

- **FIND_CUSTOMER_ID (email):** Assume input parameter email is unique. This function returns the customer ID.
- **FIND_PRODUCT_CATEGORY_ID (category name):** Assume category name is unique (e.g. Furniture). This function returns the category ID.
- **FIND_PRODUCT_ID (product name):** Assume product name is unique (e.g. 40inchTV). This function returns the product ID.
- **FIND_CREDIT_CARD_ID (credit card #):** Given the credit card number as input, this function returns the credit card ID.
- You may create additional functions for other IDs if needed.

Who creates these functions? Members who are responsible for the corresponding tables need to create the FIND_x_ID functions, where x is the table name they are responsible for. These functions are going to be used by all members of the team, and it is very likely that a FIND_x_ID created by one member will be called by several other members.

IMPORTANT NOTES

- GUI: There is no Graphical User Interface (GUI) for this project. You need to create PL/SQL procedures and functions that carry out the tasks identified above. Each task will be a separate PL/SQL stored procedure or function.
- Input/output: For tasks that require input parameters, you need to call the corresponding PL/SQL procedure or function and pass to it all necessary input parameters. This means that you need to create another script that contains calls to your procedures and functions.
- **Do not hard-code ID values for primary keys**. Use sequences instead. Primary keys should be automatically generated based on sequences, otherwise the penalty is 10% off of the grade.
- It is fine (actually it is a necessity) to call procedures created by other members of your team to get information that you need for your own procedure. For example, if you need a customer's ID, you can call the Find a Customer function which returns the customer ID, even if this is a procedure that another team member is responsible for, and then you can use that ID in your procedure/function
- How to speed up your work: First start by writing and completing the simple tasks. Make sure that you are DEBUGGING your code:
    1. First thing to do in each procedure/function is to print out the values of the input parameters, to make sure they are passed correctly, before you start working on the main part of the procedure/function.
    2. Occasionally within the procedure/function print out the values of variables, just to make sure your procedure is progressing correctly.
    3. Always use EXCEPTIONs to explain what went wrong. This will definitely speed up the implementation time and you can figure out errors much faster. In addition,

EXCEPTIONS ARE REQUIRED for every single procedure and function. There will be points taken off for missing exceptions.

- Experiment in your own Oracle account. Make sure your code works well and then transfer it to the Team Oracle account for integration with the other team members' code

A complete scenario will be given later in the semester to the entire class. Each team will take the scenario and write a driver program by calling the appropriate procedures that carry out the items mentioned in the scenario. The driver program is a script of PL/SQL anonymous code that implements a given scenario. A test scenario will be given earlier, and a final scenario will be given about a week before the final demonstrations.

Project Grading: The project is divided into several deliverables that are part of the project and they are due throughout the semester. Most deliverables are individual work, whereas a few are group deliverables. In essence, each student needs to take ownership of specific components, work individually on them, and then work with all team members to integrate his/her components together into a single working project. So there are operations that are stand-alone, and there are other operations that rely on data generated/updated/maintained by other members. This is the way that the software industry works: Each developer works separately on their own tasks, then they all get together and consolidate their work, test it out, fix any integration bugs, then then deploy the final project to the customer (the latter corresponds to running a demo at the end of the semester). Regarding the demo:

- Towards the end of the semester there will be a demo of the entire project. All team members are required to participate and demo their own components. Each student will present his or her individual procedures in front of the Professor.

- The project must be run using the group Oracle account on a consolidated schema. Running the project on individual accounts reveals inconsistencies among group members and will incur significant penalty. In other words, questions like "May I run my part on my account?" is not acceptable.

- The demo is the final deliverable, and it represents the highest weight of the project. In that deliverable each member must demonstrate that their procedures (stand-alone and consolidated) run correctly.

Submission/Upload instructions

Each deliverable must be uploaded on blackboard. Whenever source code is requested, you must create a text file containing the SQL and PL/SQL commands. The text file must have the following:

- Team name, member names, Which member operations you are implementing (for example Member 1, Member 2, … etc.)
- Must start with drop table commands
- Must have Comments
    - Inline comments (description of each parameter)
    - Descriptive comments (multi-line for each procedure/function explaining how it works)
- Must run from beginning to end without errors and must comply with all instructions for full points.

The project deliverables are:

- **Deliverable D1:** Database design – November 14, 2023
- **Deliverable D2:** Initial PL/SQL operations – November 21, 2023
- **Deliverable D3:** Complete PL/SQL operations – December 12, 2023
- **Deliverable D4:** Demo – December 19, 2023

Helpful suggestions for the project:

- You may use your Oracle team account in conjunction to your personal Oracle account for the project. Test your code in your personal account and when it is free of bugs move the code to the team account.
- Identify a process to follow before every deliverable. Examples:
    - All code must be ready x days before the deliverable for others to see.
    - No member changes the schema without the okay from all team members
- Make sure you have agreed on the 'Kick out policy' (part of Deliverable D1). Team members who are 'kicked-out' of the team have to do the entire project themselves.
    - Every team member must respond to communication requests from other team members. Identify after how many non-responses from a member the team must notify the instructor by email
    - How will the team enforce the 'kick out policy'?
- Scenarios: As we progress through the deliverables, there will be scenarios provided to the class to test the stored procedures with example data so that all teams will be using the same set of data (except from the deliverable D1 where each team is free to INSERT any data)
- Make sure to use the 'Ask the Professor' forum to post questions and get answers about the project.