**UMBC**

**IS620 Advanced Database Projects**

Spring 2022

Group Project

**A Food Industry DB System**

Assume that your team has been in contract with the headquarters of a company that owns several restaurants in different states in the US. Your team is to provide software that manages these restaurants, by performing several operations in the software. These operations actually manipulate various components of a database. Example operations are to establish a new restaurant, update a menu, hire restaurant employees, place an order by a customer, provide financial statements, etc. In addition, there are reports to be run for the managers of the restaurants, to observe how the business is doing in a specific state, or during a specific month or quarter of the year, 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**

**Cuisine Types**: Contains a cuisine type ID and the cuisine type names: American, Indian, Italian, BBQ, Ethiopian.

**Restaurants**: Each restaurant has an ID, name, street address, city, state, zip, and the cuisine type it specializes in.

**Waiters**: Each waiter has an ID, name, ID of restaurant they work at.

**Menu Items**: This is the food that is served in each restaurant and shows up as items in the menu. Contains cuisine type ID, menu item ID, name of item for each cuisine type, and price.

* American cuisine menu items: burger, fries, pasta, salad, salmon
* BBQ cuisine menu items: steak, burger, pork loin, fillet mignon
* Indian cuisine menu items: dal soup, rice, tandoori chicken, samosa
* Italian cuisine menu items: lasagna, meatballs, spaghetti, pizza
* Ethiopian cuisine menu items: meat chunks, legume stew, flatbread

Since there is one large company that owns all these restaurants, there is a single consolidated menu. So for simplicity let us assume that:

* All restaurants of the same type (e.g. BBQ) have the same plates (steak, burger, etc.)
* The same plate has the same price across all restaurants that offer it.

**Restaurant Inventory**: Restaurants need to stock food items (menu items) to serve to customers. This table has food inventory for each restaurant. Include menu item ID, menu item name, restaurant ID, and quantity for the menu item. This table is populated with information from the Menu Items table. Every time food is ordered from the menu, the appropriate quantity must be reduced accordingly.

**Customers**: Contains customer ID, name, email, street address, city, state, zip, credit card number.

**Orders**: This table contains information about an order of a customer at a restaurant. Include order ID, restaurant ID, customer ID, order date, menu item ID, waiter ID, amount paid (order amount without tip), tip (calculated as 20% of the order amount).

Assumptions: Each order contains a single menu item.

**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 New Cuisine Type / Restaurant / Customer / Waiter / Menu / Inventory / Order etc. Create reports that show how each restaurant is doing regarding orders, food inventory, waiters’ tips, 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 Cuisine Types and Restaurants tables. Must create these tables.
  1. Add cuisine type: Given the name of a cuisine type, add it to the table. The cuisine type is the input parameter to the PL/SQL procedure.
  2. Add restaurant: Add a new restaurant in the table with all pertinent input information.
  3. Display restaurant by cuisine: Given a cuisine type, show name and address about all restaurants that offer that cuisine.
  4. Report Income by state. Generate a report that lists the income of restaurants per cuisine type and per state.
* **Member 2**: Responsible for the Waiters table. Must create the table.

1. Hire waiter: Given all pertinent information as parameters, hire a waiter at a restaurant. The waiter name and restaurant ID must be input parameters. Use the function FIND\_RESTAURANT\_ID (see helper functions below) first to get the rID.
2. Show list of waiters: Given a restaurant ID (you need to call the appropriate helper function), show all info about each employee
3. Report tips: Show total tips by each waiter.
4. Report tips by state: Show total tips earned by waiters per state.

* **Member 3**: Responsible for the Menu Items and Restaurant Inventory tables. Must create these tables.

1. Create menu item: Given a cuisine type id, create a menu item (name and price) for that cuisine type. Use the function FIND\_CUISINE\_TYPE\_ID (see helper functions below) first to get the ID.
2. Add menu item to Restaurant Inventory: Given all pertinent information, add a menu item with a given quantity to a given restaurant in the Restaurant Inventory table. You will need to call helper functions to find IDs (see helper functions below).
3. Update menu item inventory: Given a restaurant id, a menu item id, along with a given quantity, reduce the inventory of that menu item by the amount specified by the quantity. This is to keep the inventory updated every time there is an order of an item.
4. Report Menu items: Generate a report to show totals of each menu item by type of cuisine.

* **Member 4**: Responsible for the Orders table. Must create this table.

1. Place an order: Given all required information, add an order in the Orders table. Use the FIND\_x\_ID helper functions (where x is the name of a table - see helper functions below) first to retrieve the IDs that are needed.
2. List all orders at a given restaurant on a given date.
3. List the most popular menu item ordered for each cuisine type
4. Report: Generate a report showing the top 3 restaurants of each state. The ranking is based on the total of ‘amount paid’ per restaurant per state.

* **Member 5**: Responsible for the Customers table. Must create this table.

1. Add a customer: Given all necessary information add a customer to the DB
2. List names of all customers who live in a given zip code
3. Report: Generate a report with the names of customers who spent the most money (top 3) so we can send them discount coupons, and also the names of the most frugal customers (bottom 3).
4. Report: States of generous customers. Generate a report that lists the states based on customers who tip generously. Show the total amount of tips by state in descending order of tip amount.

**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 restaurant ID. How can you find the restaurant ID? You create a function FIND\_RESTAURANT\_ID (restaurant name). Given a name of a restaurant it will return its rID. FIND ID functions must exist as follows:

* **FIND\_CUISINE\_TYPE\_ID (cuisine name):** Assume cuisine name is unique (e.g. Italian). This function returns the cuisine ID.
* **FIND\_RESTAURANT\_ID (restaurant name):** Assume restaurant name is unique (e.g. HotDawg). This function returns the restaurant ID.
* **FIND\_MENU\_ITEM\_ID (item name):** Assume item name is unique (e.g. spaghetti). This function returns the menu item ID.
* **FIND\_CUSTOMER\_ID (customer name):** Assume customer name is unique (e.g. Pat). This function returns the customer ID.
* **FIND\_WAITER\_ID (waiter name):** Assume waiter name is unique (e.g. Joe). This function returns the waiter ID
* **FIND\_ORDER\_ID (…):** You may or may not need this function. If you do, you need to call other FIND\_x\_ID functions first, and then use them as parameters to FIND\_ORDER\_ID. This function returns the order ID

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)

* 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
* 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.