A Food Industry DB System

I have been contracted by the headquarters of a company that owns several restaurants in different states in the US. I am 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.

I 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.

The project will be completed incrementally based on deliverables.

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 the restaurant they work at.

Menu Items: This is the food that is served in each restaurant and shows up as items on 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, 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.

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

Recommendations: Information about possible recommendations to be suggested to a customer. This is calculated based on stars given to restaurants with the same cuisine type (e.g. BBQ) that a customer has not visited yet. Attributes: Recommendation ID, Customer ID, Recommended restaurant ID, Recommendation Date. There can be many recommendations per customer.

OPERATIONS

These are the operations that need to be implemented, corresponding to the functional operations within the 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 / Add Restaurant / Add Customer / Add Waiter / Add 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.

Month 1

Week 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.

Add restaurant: Add a new restaurant in the table with all pertinent input information.

Week 2

Display restaurant by cuisine: Given a cuisine type, show the name and address of all restaurants that offer that cuisine.

Report Income by state: Generate a report that lists the income of restaurants per cuisine type and per state.

Week 3

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 first to get the ID.

Show list of waiters: Given a restaurant ID (you need to call the appropriate helper function), show all info about each employee.

Month 2

Week 4

Report tips: Show total tips by each waiter.

Report tips by state: Show total tips earned by waiters per state.

Week 5

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 first to get the ID.

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.

Week 6

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.

Report Menu items: Generate a report to show totals of each menu item by type of cuisine.

Month 3

Week 7

Add a customer: Given all necessary information add a customer to the DB.

Place an order: Given all required information, add an order in the Orders table. Use the FIND\_x\_ID helper functions first to retrieve the IDs that are needed.

Week 8

List all orders at a given restaurant on a given date.

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.

Week 9

Add\_Review: This procedure receives a review from a person (anyone with an email) for a particular restaurant and adds the review in the Reviews table. Input to this procedure is the reviewer email, stars given (a number from 1-5), and review text. You will need to call a helper function to find IDs.

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 restaurant. It should first print: ‘Top rated restaurants’ then for each line show the average number of stars (in descending order), restaurant ID, cuisine type, and the standard deviation. Then print ‘Buyer Beware: Stay Away from…’ then each consecutive line shows the average number of stars (in ascending order), restaurant ID, restaurant name, cuisine type, and the standard deviation.

Week 10

Recommend\_To\_Customer: This procedure calculates a recommendation for a customer. The input is the customer ID and a cuisine type. This procedure will insert a new recommendation in the Recommendations table for a specific customer as follows: it will identify the best-rated restaurant (based on reviews) for the input cuisine type. Must check if a customer has had an order for that particular cuisine type in the past. Recommendations must be made for restaurants that the customer has not visited yet.

List\_Recommendations: Provide a report that lists all customer recommendations. This report prints the name of the customer, the recommended restaurant, cuisine type, and its average stars (one line per recommended product).

Helper Functions

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.