Feature Specifications

**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 inserts a row into the customer table with the given ID, name, address, state, zip, email and credit (as zero). Please also print out the new customer ID.

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');

**Feature 2:** check customer based on email

Given a customer email, first check if there is a customer with that email. If not, print a message ‘no 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.

Input: email.

Output: If there isn't a client with the provided email, the screen will display "no such customer." In the absence of that, the page should output the customer's name, address, state, zip code, email, credit, and the total number of orders where status = 2.

A possible name for this feature is:

—— standard scenario, client in attendance with provided email address

exec show\_customer\_detail('Thor@gmail.com');

select \* from customer;

-- customer not existing

exec display\_customer\_detail(‘Thor@gmail.com');

**Feature 3:** Search restaurant by category.

Input is a category name or a part of the category name. If not found the system returns “category not found”. In case the category matches we print out name, average review score, average wait time, and zip code for restaurants that are open and match the input category name.

Input: category name

Output: Displays “category not found” if category is invalid. Otherwise displays a list of restaurants with name, average review score, average wait time, and zip code for restaurants that are open.

Example of calling this feature:

exec search\_restaurants\_by\_category(‘seafood’);

exec search\_restaurants\_by\_category(‘sea’);

**Feature 4:** Show dishes offered by a restaurant.

Input will be restaurant ID. We first need to check if it’s a valid restaurant ID. If not, print a message ‘no such restaurant’. Otherwise print out all dishes in this restaurant, along with dish name and price.

Input: Restaurant ID

Output: Displays “invalid restaurant ID” if invalid. Otherwise displays a list of dishes offered by that restaurant with name and price.

Example of calling this feature:

exec restaurant\_dishes (62);

**Feature 5:** Displaying all dishes in the shopping cart.

Show all dishes in a shopping cart. Input is a cart ID. First check 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.

Input: Cart ID

Output: Message on screen: "cart id is invalid" If the cart id is incorrect, the screen will display the dish name, price, and quantity.

**Feature 6:** Remove a dish from the 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'.

Input: Cart ID, dish ID.

Output: display of "invalid input" If the ID is invalid, update the row by reducing the quantity and print "quantity lowered" if the quantity is larger than 1. Otherwise, print "dish removed" if quantity is one and delete the row.

**Feature 7:** Update Status of an Order

Input: order\_ID, new\_status, Input time

Output:

1) From the given input it checks whether the Order\_ID exists in table Displays error message if the order\_ID is invalid.

2) Update the status of the Order\_ID with the given input (Example: 1 is in progress, 2 is delivered, 3 is cancelled)

3) Update the message table with input time and message body checking the condition status=’Delivered’ or ‘Cancelled’.

NOTE: Message body values are updated as below

'Your order X has been cancelled and refund issued!' where X is the order ID’

'Your order X has been delivered!' where X is the order ID.

Example of calling this feature:

exec update\_order\_status(70, ‘delivered’, timestamp '2022-10-11 21:30:00.00')

**Feature 8:** Enter a review.

Input includes a customer ID, a restaurant ID, a review date, a review score and review comment. First check if the customer ID is valid. If not prints a message saying “invalid customer ID” Then, checks if the restaurant ID is valid. If not, prints a message saying “invalid restaurant ID”. If both are valid, inserts a row into review table with the input customer id, restaurant ID, review date, score and comment. Finally updates the average review score of the restaurant to reflect the new review.

Input: customer ID, restaurant ID, review date, review score, review comment

Output: Displays “invalid input” if either customer ID or Restaurant ID is invalid. Otherwise displays “review successfully added”.

Example of calling this feature:

exec add\_review(6, 2, 40, date '2022-11-11', 4.2, 'Too good');

**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 “Invalid restaurant”. Else, print out all reviews of the restaurant, including review date, score, and comment.

Input: restaurant ID

Output: Displays “invalid input” if Restaurant ID is invalid. Otherwise displays all the reviews with date score and comments columns of that restaurant

Example of calling this feature:

exec view\_review(42);

**Feature 10:** Add a dish to shopping cart.

Input includes customer ID, restaurant ID, and a dish ID. First check whether the customer ID is valid. If not, print out a message “no such customer”. Then, check whether the restaurant ID is valid and the restaurant is open. If not print “invalid restaurant ID” or “the restaurant is closed”. Finally check the dish whether it belongs to the input restaurant. If it does not, print out message “invalid dish ID”. Otherwise where there is an existing shopping cart for the customer. If the cart does not exist, create a new cart for the customer and restaurant and print out the new cart ID. 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.

Input: customer ID, restaurant ID, dish ID

Output: Displays “invalid input” if Customer ID, Restaurant ID or Dish ID is invalid. Otherwise adds the input to the cart table.

Example of calling this feature:

exec add\_dish\_to\_cart(6, 42, 52);

**Feature 11:** Compute total amount for dishes in a cart.

Input includes 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 and display the final value on the screen

Input: cart ID, checkout time, delivery method

Output: Displays “invalid input” if Cart ID is invalid. Otherwise computes total amount for the cart and displays it along with delivery fee, tax, and amount for dishes.

Example of calling this feature:

exec cart\_total(62, timestamp '2022-01-01 09:00:00.00', 2);

**Feature 12:** Generate an order with dishes in a shopping cart.

The input is a cart ID, order time, delivery 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).

Input: cart ID, order time, delivery method, estimated time to deliver or pickup, tip, payment method

Output: Displays “invalid input” if Cart ID is invalid. Otherwise computes total amount for the cart updates payment details and delivery status and messages to create an order. Then displays “Order has been successfully placed and displays a message 'A new order X is placed at Restaurant Y with estimated time of Z and amount A”

Example of calling this feature:

exec order\_placed(62, timestamp '2022-01-01 09:00:00.00', 2, timestamp '2022-10-11 21:30:00.00', 2.00,3);

**Feature 13:** Advanced Search

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

The procedure first checks if the customer ID is valid. If not print a message “invalid Customer ID”, else it returns all the restaurants that satisfy all the given conditions.

Input: customer ID, list of category names, minimal review score, and wait time

Output: Displays “invalid input” if customer ID is invalid. Otherwise displays a list of restaurants that satisfy all conditions.

Example of calling this feature:

exec search\_restaurants(234, (‘seafood’,’mexican’), 4, 30);

\*passing varray containing list of categories to the procedure.

**Feature 14:** Restaurant Recommendation

The input is Customer ID. Display “invalid customer ” if the ID is invalid else display ID, names, address and reviews of the recommended restaurants.

Input: customer ID

Output: Displays “invalid input” if customer ID is invalid. Otherwise displays a list of restaurants with restaurant ID, name, address and.

Example of calling this feature:

exec Recommend\_rest(4);