1. Definition of Class:

The '__init__' function of the RestaurantDatabase class is defined to initialize the class instance with the required parameters for creating a database connection.

The parameters for the "__init__" method are "host," "port," "database," "user," and "password." The database connection settings are configured using these options.

The supplied arguments are allocated to the appropriate instance attributes (self.host, self.port, self.database, self.user, and self.password) inside the __init__ method.

The initial value of the self.connection and self.cursor characteristics is None. Ultimately, the database connection is established by calling the _connect_() method.

2. Database Link:

Connecting to the MySQL database is the responsibility of the _connect_() method. It creates a connection object by giving the required connection parameters (host, port, database, user, and password) from the class instance to the mysql.connector.connect() function. The message "Successfully connected to the database" is printed if the connection is successful. The except block captures the Error exception and publishes an error message in the event that an error arises during the connection procedure.

3. Database Management:

The class provides a number of methods for working with the database tables in different ways.

a. _addReservation_(self, reservation_time, number_of_guests, special_requests, customer_id):-

• A new reservation is added using this approach to the reservations table.

• It initially uses self.connection.is_connected() to see if the database connection has been made.

• A cursor object is created if the connection is made (self.cursor = self.connection.cursor()).

• To enter the supplied values into the reservations table, a SQL enter query is created.

• The SQL query and the data are passed as parameters when the query is run using the self.cursor.execute() method.

• Self.connection.commit() is used to commit the modifications to the database.

• A printed success message appears.

b. self._getAllReservations_:-

• All reservations are retrieved using this method from the `reservations` table.

• It starts by determining whether the database connection has been made.

• A cursor object is created if the connection is made.

• All records in the `reservations` table are to be retrieved using the SQL `SELECT` query.

• `self.cursor.execute()` is used to run the query.

• Using `self.cursor.fetchall()}, the result records are retrieved and returned.

c. _addCustomer_(contact_info, customer_name, self):

• A new client is added using this approach to the customers table.

• It starts by determining whether the database connection has been made.

• A cursor object is created if the connection is made.

• For debugging purposes, the customer's contact information and name are printed.

• To enter the supplied values into the customers table, a SQL enter query is created.

• The SQL query and the values are passed as parameters when the query is run using self.cursor.execute().

• Self.connection.commit() is used to commit the modifications to the database.

• A printed success message appears.

d. Using self and customer_id, _getCustomerPreferences_:

• The eating preferences for a particular consumer are retrieved using this method from the diningPreferences table.

• It starts by determining whether the database connection has been made.

• A cursor object is created if the connection is made.

• To obtain entries from the diningPreferences table where the customerId matches the supplied customer_id, a SQL SELECT query is created.

• The SQL query and the customer_id are passed as parameters when the query is run using self.cursor.execute().

• Using self.cursor.fetchall(), the result records (preferences) are retrieved and returned.

Because the RestaurantDatabase class encapsulates the database connection and SQL operations within its methods, it offers a straightforward way to interact with the restaurant reservation database. This class can be created by developers, who can then utilize its methods to carry out a variety of operations pertaining to reservations, clients, and dining preferences.