# Exercise: Add query functions

## **Scenario**

Little Lemon have started building a functional booking system from which they can manage their customer data. An integral part of a well-functioning system is the ability to query data. Little Lemon need you to help them add the search functionality required to query their data.

## **Prerequisites**

At this stage of the project, you should have set up the working environment, configured the connector/python and created a database with all the customer information. To implement a booking system, you will first have to review how to make queries on an existing database using Python syntax.

## **Task One**

In the previous exercise you created a Python environment. In the first task of this exercise, you are tasked with extending the environment to connect with your database and interact with the data it holds.

Your first step is to import the connector module, enter your user details and connect with the database (Hint: you can use an alias when importing the module).

This gives you access to all the functionality available from the connector API, which can be accessed through the variable named connector (or whichever alias you choose).

To connect with your database, you can call the connect method of the connector class and pass in your details using the following code:

connection = connector.connect(user = "your\_username", password = "your\_password", db = "database\_name")

This code should look very familiar to you from the previous course, apart from the parameter db. DB stands for database. When instantiating the connection, you can pass the database name here in place of calling the USE command later.

The final step is to instantiate an instance of cursor to pass queries and return results (Hint: the cursor is part of the connection class outlined above).

## **Task two**

In this second task, you now need to query the database to show all tables within the database.

Having established a connection in the first task, you need to execute a test query to ensure that there are no issues. You can do this by executing, or passing, a generic query that returns a snapshot of the database tables.

You need to execute the query on the cursor using the code that follows. The cursor, as you should recall, is the bridge through which you can pass queries and return results.

show\_tables\_query = "SHOW tables"

cursor.execute(show\_tables\_query)

As before, a variable is used to hold the query. To gain a general insight, the query asks to display all tables within the database.

The second line calls the cursor execute method. This method takes the Python string and ports it into a viable SQL statement. It then passes it to the database and returns the result.

To view the results of your query, you can create another variable called results (Hint: the cursor has a method that can return all results in one call).

To view the tables that are associated with a database, you can print out the results variable using the following code:

print(results)

## **Task 3**

Query with table JOIN

For the third and final task, Little Lemon need you to return specific details from your database. They require the full name and contact details for every customer that has placed an order greater than $60 for a promotional campaign.

You can use the following steps to implement this functionality in your database directory:

Step One: Identify which tables are required. To complete the query, you first need to identify which table has the required data.

The bill paid can be found in Orders as TotalCost and the customer contact information can be found in the Customers table.

When selecting attributes from a specific table, specify the table name, followed by a dot and the target attribute as below (Hint: select the column attributes that you will need).

Step Two: Next, specify a table (Hint: The FROM keyword allows you to identify a table.)

To join two tables, specify the type of JOIN and the attribute to join the table on. The tables must be joined on an attribute that is common to both tables (such as a common column).

Step Three: Finally, include a clause to filter the data on. (Hint: the WHERE clause can be used to add conditional parameters.)

When you have completed these steps, wrap this query as a string and pass it to the .execute() method of the cursor class. When executed, your SELECT query must extract the full name, contact details and bill amount for every customer who spent more than $60.

## **Conclusion**

In this exercise you reviewed the mechanisms for performing queries using Python.