## Prerequisites

To complete this exercise, you must have access to the MySQL database. As an authorized user, you need to establish a connection between Python and MySQL via the connector API and create a “cursor” object using the following code:

!pip show mysql-connector-python

import mysql.connector as connector

# Establish connection between Python and MySQL database via connector API

connection=connector.connect(

user="root", # use your own

password="", # use your own

)

Once the connection is established and you have a cursor object, you can carry out the required tasks.

# Create a cursor object to communicate with entire MySQL database

cursor = connection.cursor()

Task 1 solution

Little Lemon needs to create a database that can hold their tables. Create a MySQL database for Little Lemon using Python. You can call the database little\_lemon.

Confirm that the database has been successfully executed using the SHOW DATABASE query.

**Solution**

# If exist, drop the database first

cursor.execute("drop database little\_lemon")

# Create database and checking all that we have!

cursor.execute("CREATE DATABASE little\_lemon")

cursor.execute("SHOW DATABASES")

for database in cursor:

print(database)

Task 2 solution:

Set the newly created database little\_lemon for use. Confirm that the database is available for use by using connection.database.

**Solution**

# Set little\_lemon database for use

cursor.execute("USE little\_lemon")

# Confirm the database in use

connection.database

cursor.execute("SHOW DATABASES")

for database in cursor:

print(database)

Task 3 solution

Create a MenuItems table using the following SQL query as a Python string:

create\_menuitem\_table = """CREATE TABLE MenuItems (

ItemID INT AUTO\_INCREMENT,

Name VARCHAR(200),

Type VARCHAR(100),

Price INT,

PRIMARY KEY (ItemID)

);"""

Task 4 solution

Create the Menus table to store data on menus using the following SQL query as a Python string.

create\_menu\_table = """CREATE TABLE Menus (

MenuID INT,

ItemID INT,

Cuisine VARCHAR(100),

PRIMARY KEY (MenuID,ItemID)

);"""

# The SQL query for Menu table is:

create\_menu\_table="""

CREATE TABLE Menus (

MenuID INT,

ItemID INT,

Cuisine VARCHAR(100),

PRIMARY KEY (MenuID,ItemID)

);"""

**Solution**

# Create Menu table

cursor.execute(create\_menu\_table)

# Confirm if the table is created

cursor.execute("SHOW TABLES")

for table in cursor:

print(table)

Task 5 solution

Create the Bookings table to store data on customer bookings using the following SQL query as a Python string:

Create\_booking\_table = """CREATE TABLE Bookings (

BookingID INT AUTO\_INCREMENT,

TableNo INT,

GuestFirstName VARCHAR(100) NOT NULL,

GuestLastName VARCHAR(100) NOT NULL,

BookingSlot TIME NOT NULL,

EmployeeID INT,

PRIMARY KEY (BookingID)

);"""

**Solution**

# The SQL query for Bookings table is:

create\_booking\_table="""

CREATE TABLE Bookings (

BookingID INT AUTO\_INCREMENT,

TableNo INT,

GuestFirstName VARCHAR(100) NOT NULL,

GuestLastName VARCHAR(100) NOT NULL,

BookingSlot TIME NOT NULL,

EmployeeID INT,

PRIMARY KEY (BookingID)

);"""

**Solution**

# Create Bookings table

cursor.execute(create\_booking\_table)

# Confirm if the table is created

cursor.execute("SHOW TABLES")

for table in cursor:

print(table)

Task 6 solution

Create an Orders table to store data on customer orders using the following SQL query as a Python string:

create\_orders\_table = """CREATE TABLE Orders (

OrderID INT,

TableNo INT,

MenuID INT,

BookingID INT,

BillAmount INT,

Quantity INT,

PRIMARY KEY (OrderID,TableNo)

);"""

**Solution**

# The SQL query for Bookings table is:

create\_orders\_table="""

CREATE TABLE Orders (

OrderID INT,

TableNo INT,

MenuID INT,

BookingID INT,

BillAmount INT,

Quantity INT,

PRIMARY KEY (OrderID,TableNo)

);"""

# Create Orders table

cursor.execute(create\_orders\_table)

# Confirm if the table is created

cursor.execute("SHOW TABLES")

for table in cursor:

print(table)

# Let's close the cursor and the connection

if connection.is\_connected():

cursor.close()

print("The cursor is closed.")

connection.close()

print("MySQL connection is closed.")

else:

print("Connection is already closed")