A picture containing logo

Description automatically generated

Rusty DeGarmo, Mitchell Kwon, Aurora Lopez, Lucas Ramsey and Parrish Ward

Bellevue University

CSD 310: Database Development and Use

Project Milestone: Outland Adventures

Milestone 3

July 15, 2021

# Equipment Sales Report

This report is attempting to help Outland Adventures decide whether it is worth it for the company to continue selling equipment instead of just renting. “Do enough customers buy equipment to keep equipment sales?” We queried the customer table to count how many customers rented their equipment and how many bought it.

A problem I ran across that I thought was interesting was that returning {info} instead of {info[0][0]} returned something like [(7,)]. I was stumped for a little bit but then I realized that this looked like a tuple inside an array object. I decided to try to get an index and voila, it works. Any insights as to why cursor.fetchall() returned this object structure or how I could have gotten the desired result without indexing?

Python Script:

import mysql.connector

#connect to the database

db = mysql.connector.connect(

    host="localhost",

    user="root",

    password= "password",

    database= "outlandadventures"

)

#create a cursor

cursor = db.cursor()

#Count how many customers rented equipment

cursor.execute('SELECT COUNT(supplies) FROM customer WHERE supplies = "rented";')

info = cursor.fetchall()

print()

print("-- DISPLAYING CUSTOMER RENTAL RECORDS --")

print(f"Number of customers who rented equipment: {info[0][0]}")

print()

#Count how many customers purchased equipment

cursor.execute('SELECT COUNT(supplies) FROM customer WHERE supplies = "bought";')

info = cursor.fetchall()

print("-- DISPLAYING CUSTOMER PURCHASE RECORDS --")

print(f"Number of customers who purchased equipment: {info[0][0]}")

print()

#close the database

db.close()

Results:

Text

Description automatically generated

# Number of Treks per Location Report

This report is attempting to help them answer their question. “So far, they have conducted treks in Africa, Asia, and Southern Europe. Is there any one of those locations that has a downward trend in bookings?” We queried the trip table and counted how many trips were for each destination.

Python Script:

import mysql.connector

#connect to the database

db = mysql.connector.connect(

    host="localhost",

    user="root",

    password= "rBillie-0928t",

    database= "outlandadventures"

)

#create a cursor

cursor = db.cursor()

#Count how many treks were to Africa

cursor.execute('SELECT COUNT(destination\_name) FROM trip WHERE destination\_name = "Africa";')

info = cursor.fetchall()

print()

print("-- DISPLAYING AFRICA TREK RECORDS --")

print(f"Number of treks to Africa: {info[0][0]}")

print()

#Count how many treks were to Asia

cursor.execute('SELECT COUNT(destination\_name) FROM trip WHERE destination\_name = "Asia";')

info = cursor.fetchall()

print("-- DISPLAYING ASIA TREK RECORDS --")

print(f"Number of treks to Asia: {info[0][0]}")

print()

#Count how many treks were to Southern Europe

cursor.execute('SELECT COUNT(destination\_name) FROM trip WHERE destination\_name = "Southern Europe";')

info = cursor.fetchall()

print("-- DISPLAYING SOUTHERN EUROPE TREK RECORDS --")

print(f"Number of treks to Southern Europe: {info[0][0]}")

print()

#close the database

db.close()

Results:

Text

Description automatically generated

# Aged Equipment Report

We wanted to help Outland Adventures answer their question. “They are a little concerned about the age of some of the inventory. Are there inventory items that are over five years old?” We needed to query the ‘supplies’ table for the date the equipment was added to inventory. Then we needed to check if that date was more than five (5) years old.

I had to reformat the date\_added column in our supplies table to get this to work. When converting our date\_added column from a string to a date I was getting “2003-05-21” instead of the anticipated “03-05-2021”. This is because of the way SQL formats dates yyyy-mm-dd. The result was that I was returning that all of our equipment was older than 5 years old. The other, probably better-in-the-long-run, option would have been to update all of the dates in all of our tables to a date data type. I went forward assuming that there is a good reason that the database is structured the way it is. I had to run this from the MySQL command line:

UPDATE supplies SET date\_added = "2021/03/05";

I tried to run this command from a Python script but for some reason, it wouldn’t work. I didn’t get any errors but it didn’t actually update the table either. I don’t know what was wrong. This one command worked for our data, though, because all of the equipment in stock at the time of the report was added on that same date.

Python Script:

import mysql.connector

#connect to the database

db = mysql.connector.connect(

    host="localhost",

    user="root",

    password= "rBillie-0928t",

    database= "outlandadventures"

)

#create a cursor

cursor = db.cursor()

#Count how much equipment older than five years old

cursor.execute('SELECT COUNT(\*) FROM supplies WHERE CAST(date\_added AS DATE) < NOW() - interval 5 year;')

info = cursor.fetchall()

print()

print("-- DISPLAYING SUPPLIES RECORDS --")

print(f"Equipment older than 5 years: {info[0][0]}")

print()

#close the database

db.close()

Output:

Text

Description automatically generated