**P2 – Retail Sales – Pandas and Postgres Instructions**

**Overview:**

**This is a group project**. Assume you are given a sample of sale data from an online retailer in the form of an excel file. The retailer wants your team to test how they could transfer their data into a postgres database and read data programmatically back from the database. You’ll use Pandas to read the data in, to transfer it to postgres, and to read it back from postgres to display a summary of different product categories.

**Libraries Required:**

* import sqlalchemy
  + note: for sqlalchemy to work properly with a postgres database, you should also have installed psycopg2, even though you don’t need to list it as an import in your python file
* import pandas as pd
* import matplotlib.pyplot as plot
* optional, but you could use them:
  + import openpyxl
  + from sqlalchemy.sql import text
  + import psycopg2

**External Files Required:**

* see the Learning Suite project description for the Excel file you’ll need to start from downloads. The excel file is mostly in good shape, except the Name column needs to be separated into first name and last name, and the Category column has the incorrect categories listed. You’ll need to fix the categories based on the product names.

**Logical Flow:**

Your program will ask the user:

* *“If you want to import data, enter 1. If you want to see summaries of stored data, enter 2. Enter any other value to exit the program: “*

**PART 1:** If they enter 1, do the following:

1. Read the Retail\_Sales\_Data.xlsx into python. I recommend using pandas for this, though you could use openpyxl.
2. Separate the "name” column into a “first\_name” and “last\_name” column and delete (or overwrite) the original “name” column
3. Fix the “category” column so that the categories actually match the product that was sold (see the hints section for specific details the new category values)
4. Save the results as a table called ‘sale’ in your is303 postgres database.
5. Print out the message: *“You've imported the excel file into your postgres database.”*

**PART 2:** If they enter 2, do the following:

1. Print out: *“The following are all the categories that have been sold:”*
2. Print out each of the categories stored in your database from the ‘sale’ table with a number preceding it. You can’t just hardcode the categories in, your program must read them from the database. It should look like this:

*1: Technology*

*2: Apparel*

*3: Accessories*

*4: Household Items*

*5: Stationery*

1. Print out: *“Please enter the number of the category you want to see summarized: “*
2. Then, for the entered category, calculate and display the sum of total sales, the average sale amount, and the total units sold.
3. Then, display a bar chart with the x axis as the products in that category and the y axis as the sum of the total sales of that product.
   1. The title of the chart should be “Total Sales by Product in Category (but put the actual category name)
   2. The x label should be “Product”, the y label should be “Total Sales”

If they enter anything other than 1 or 2:

1. Print: "Closing the program."

Similar to your other projects, there are multiple ways to do this, and all that matters is that you fulfill the requirements. It doesn’t matter exactly how you implement it. For example, in Part 1, you could use openpyxl to do the splitting and fixing the category names. However, I recommend just using pandas. Below, I’ll give some recommendations and hints for completing the project:

**Hints/Suggestions (BUT NOT REQUIRED):**

**PART 1:** If they enter 1, do the following:

1. Read the Retail\_Sales\_Data.xlsx into python.
   1. pd.read\_excel() makes this very easy.
2. Separate the Name column into a First\_Name and Last\_Name column.
   1. In your last project, you used the .split() function to do something similar. Pandas actually has their own version of split, and it can split a column of a dataframe into multiple columns
      1. dfSeparatedNames = dfImportedFile["name"].str.split("\_", expand = True)
   2. To insert new columns into a dataframe, you can use the .insert() function.
   3. To delete the “name” column, the easiest way is to use “del” but you can also use the .drop function.
3. Fix the “category” column so that the categories match the product that was sold.
   1. Here is a dictionary that has the keys as the product names and the new category as the value:

productCategoriesDict = {

'Camera': 'Technology',

'Laptop': 'Technology',

'Gloves': 'Apparel',

'Smartphone': 'Technology',

'Watch': 'Accessories',

'Backpack': 'Accessories',

'Water Bottle': 'Household Items',

'T-shirt': 'Apparel',

'Notebook': 'Stationery',

'Sneakers': 'Apparel',

'Dress': 'Apparel',

'Scarf': 'Apparel',

'Pen': 'Stationery',

'Jeans': 'Apparel',

'Desk Lamp': 'Household Items',

'Umbrella': 'Accessories',

'Sunglasses': 'Accessories',

'Hat': 'Apparel',

'Headphones': 'Technology',

'Charger': 'Technology'}

* 1. You can change the values of the “category” column with just one line of code using the .map function and the productCategoriesDict above. I recommend looking that up, but if you have trouble getting that to work, you could also loop through the dataframe and update the values in “category” one by one.

1. Save the results as a table called ‘sale’ in your is303 postgres database.
   1. Use the sqlalchemy.create\_engine() function to create a connection to your postgres database
   2. Then use your dataframe and the .to\_sql() function to write it to sql.
   3. I recommend these settings:
      1. dfImportedFile.to\_sql("sale", engine, if\_exists = 'replace', index = False)
2. Print out the message: *“You've imported the excel file into your postgres database.”*

**PART 2:** If they enter 2, do the following:

1. Print out: *“The following are all the categories that have been sold:”*
2. Print out each of the categories stored in your database from the ‘sale’ table with a number preceding it.
   1. To do this, you need to read the category names from the “sale” table that you created in Part 1. That means using the sqlalchemy.create\_engine() function and the pandas pd.read\_sql\_query() function.
   2. There are many ways to get the category names, so do what seems natural to you. I personally would just grab everything from the “sale” table, put it in a dataframe (let’s call it dfImported) and then you can see unique values by doing dfImported[“column name”].unique(). You could loop through that.
   3. Be sure to include numbers like 1, 2 before displaying the category name. If you display the names in a loop, you can create your own number variable that increases each time. Or you can look up using enumerate() in loops, which basically does that for you.
3. Print out: *“Please enter the number of the category you want to see summarized: “*
   1. You’ll need a way to translate a number (e.g. 1) into a category name (e.g. “Technology). A dictionary is often a good way to do this.
4. Then, for the entered category, calculate and display the sum of total sales, the average sale amount, and the total units sold.
   1. Because you only want to display summary information for the selected category, you should filter your dataframe so you are only showing data for the selected category. dfExample.query() is a good way to do this.
   2. Pandas includes ways to do sums and averages on columns easily. For example, you could store the sum of a column like this:
      1. iColumnSum = dfExample[“column\_name”].sum()
      2. for average, you can use .mean() instead of .sum()
   3. You want to display the sum of total\_price, the average of total\_price, and the sum of quantity\_sold
5. Then, display a bar chart with the x axis as the products in that category and the y axis as the sum of the total sales of that product.
   1. There’s so much that you can do with pandas and charts, but because we don’t cover it in detail in the class, I’ll just give you example code here. You’ll obviously need to change it depending on what you called your variables:
   2. A screen shot of a computer code

      Description automatically generated

Upload just the python file to Learning Suite. Only one person per group needs to upload.

**Example Output:**

*Part 1:*

If you want to import data, enter 1. If you want to see summaries of stored data, enter 2. Enter any other value to exit the program: 1

You've imported the excel file into your postgres database.

Note, you don’t need to upload anything from postgres/pgadmin4, but if you did part 1 correctly, if you do select \* from sale; in pgAdmin4 it should look like this:

A screenshot of a computer

Description automatically generated

*Part 2:*

If you want to import data, enter 1. If you want to see summaries of stored data, enter 2. Enter any other value to exit the program: 2

The following are all the categories that have been sold:

1: Technology

2: Apparel

3: Accessories

4: Household Items

5: Stationery

Please enter the number of the category you want to see summarized: 3

Total sales for Accessories: 175,150.00

Average sale amount for Accessories: 1,684.13

Total units sold for Accessories: 328

Then this should pop up:

A screenshot of a computer

Description automatically generated

Don’t worry if the sizing of the chart cuts off some of the text.

**Requirements:**

|  |  |
| --- | --- |
| **Requirement** | **Sub Requirements** |
| asks the user to enter 1 or 2 |  |
| reads the retail excel file into python |  |
| separates the name column into a first\_name and last\_name column and deletes the name column |  |
| fixes the category column | * + - The new category values should be those given in the example dictionary |
| creates table called sale in postgres with the fixed data and prints message after doing so. |  |
| Lists all the categories and asks for user to select category by number | * The list of categories can’t be hardcoded if you want full credit. It should pull the category names from postgres. |
| Prints out the sum of total\_price, the average total\_price, and the sum of quantity\_sold for the selected category |  |
| Displays a chart showing each product in the selected category and the sum of that product's total\_price |  |
| includes useful comments | * Should include a name and description at the top * Should also include comments throughout |