assignment_3_solutions

January 11, 2023

1 Assignment 3: SQL and Pandas

1.1 Overview

Like Assignment 2, we will be using the Chinook database, this time focusing more on the sales side of the music store.

1.1.1 Question 0

Set up your connection and cursor for `sqlite3`.

[ ]:
```
import sqlite3
import pandas as pd
from pandasql import sqldf

connection = sqlite3.connect("../data/chinook.sqlite")
cursor = connection.cursor()
```

1.1.2 Question 1

Using `pandas`' built-in function `read_sql_query(query, connection)`, query the entire Customer and Invoice tables, merging by CustomerId.

[ ]:
```
customer_invoice_df = pd.read_sql_query('''
SELECT * FROM Customer
INNER JOIN Invoice
    ON Customer.CustomerId = Invoice.CustomerId
''', connection)

customer_invoice_df.head()
```

[ ]:
```
   CustomerId FirstName LastName Company                     Address       City  \
0           2    Leonie   Köhler    None  Theodor-Heuss-Straße 34  Stuttgart
1           4     Bjørn   Hansen    None         Ullevålsveien 14       Oslo
2           8      Daan  Peeters    None         Grétrystraat 63   Brussels
3          14      Mark  Philips   Telus          8210 111 ST NW   Edmonton
4          23      John   Gordon    None          69 Salem Street     Boston

  State   Country PostalCode              Phone  … SupportRepId InvoiceId  \
```

```
0  None  Germany      70174   +49 0711 2842222  …           5          1
1  None   Norway       0171    +47 22 44 22 22  …           4          2
2  None  Belgium       1000   +32 02 219 03 03  …           4          3
3    AB   Canada    T6G 2C7  +1 (780) 434-4554  …           5          4
4    MA      USA       2113  +1 (617) 522-1333  …           4          5

   CustomerId          InvoiceDate          BillingAddress BillingCity  \
0           2  2009-01-01 00:00:00  Theodor-Heuss-Straße 34   Stuttgart
1           4  2009-01-02 00:00:00         Ullevålsveien 14        Oslo
2           8  2009-01-03 00:00:00          Grétrystraat 63    Brussels
3          14  2009-01-06 00:00:00           8210 111 ST NW    Edmonton
4          23  2009-01-11 00:00:00          69 Salem Street      Boston

  BillingState BillingCountry BillingPostalCode  Total
0         None        Germany             70174   1.98
1         None         Norway              0171   3.96
2         None        Belgium              1000   5.94
3           AB         Canada           T6G 2C7   8.91
4           MA            USA              2113  13.86

[5 rows x 22 columns]
```

1.1.3 Question 2

Using sqldf, query a new DataFrame containing only the CustomerId, FirstName, LastName, InvoiceDate, and Total. **Drop any invoices before 2010**. Remember, when using an existing pandas DataFrame, you're query FROM the variable name, not the tables from previously.

[ ]:
```
table = sqldf('''
    SELECT CustomerId, FirstName, LastName, InvoiceDate, Total
    FROM customer_invoice_df
    WHERE InvoiceDate > 2010
''')

table
```

[ ]:
```
     CustomerId FirstName    LastName          InvoiceDate  Total
0            43  Isabelle     Mercier  2010-01-08 00:00:00   1.98
1            45  Ladislav      Kovács  2010-01-08 00:00:00   1.98
2            47     Lucas     Mancini  2010-01-09 00:00:00   3.96
3            51    Joakim   Johansson  2010-01-10 00:00:00   6.94
4            57      Luis       Rojas  2010-01-13 00:00:00  17.91
..          …         …          …                   …     …
324          25    Victor     Stevens  2013-12-05 00:00:00   3.96
325          29    Robert       Brown  2013-12-06 00:00:00   5.94
326          35  Madalena     Sampaio  2013-12-09 00:00:00   8.91
327          44     Terhi  Hämäläinen  2013-12-14 00:00:00  13.86
328          58     Manoj      Pareek  2013-12-22 00:00:00   1.99
```

```
[329 rows x 5 columns]
```

1.2 Question 3

We now have a simplified list of invoices containing only name, id, date, and total. Let's figure out how much every has spent since 2010! Recall that pandas has the `groupby()` function (documentation here). Using this, make a new DataFrame grouping by the CustomerId, giving the total/sum for each.

[ ]:
```
table.groupby(["CustomerId"]).sum()
```

[ ]:
```
            Total
CustomerId
1           39.62
2           12.87
3           39.62
4           28.73
5           38.64
6           40.71
7           40.64
8           30.69
9           31.68
10          28.71
11          21.78
12          36.63
13          25.74
14          28.71
15          22.78
16          36.63
17          27.74
18          37.62
19          13.87
20          39.62
21          27.72
22          39.62
23          14.85
24          43.62
25          35.69
26          45.64
27          28.71
28          27.78
29          36.63
30          31.68
31          28.71
32          21.78
33          36.63
```

```
34          27.74
35          37.62
36          21.78
37          42.63
38          25.74
39          38.62
40          13.87
41          37.62
42          28.73
43          40.62
44          32.71
45          45.62
46          38.69
47          35.64
48          31.71
49          21.78
50          36.63
51          32.68
52          28.71
53          21.78
54          36.63
55          25.74
56          37.62
57          30.78
58          38.62
59          26.74
```

1.2.1 Question 4

Turns out, the pandas `groupby()` statement has an equivalent in SQL! Putting

`GROUP BY <Table Name>.<Column Name>`

at the end of a query plus the keyword `SUM()` around the column you're grouping by creates a very similar table to the one we made in pandas!

Use what we've learned to query a table with FirstName, LastName, and SUM(Total). Do the results match what we did in pandas?

[ ]:
```
table = sqldf('''
    SELECT FirstName, LastName, SUM(Total)
    FROM customer_invoice_df
    WHERE InvoiceDate > 2010
    GROUP BY CustomerId
''')

table
```

[ ]:

	FirstName	LastName	SUM(Total)
0	Luís	Gonçalves	39.62
1	Leonie	Köhler	12.87
2	François	Tremblay	39.62
3	Bjørn	Hansen	28.73
4	František	Wichterlová	38.64
5	Helena	Holý	40.71
6	Astrid	Gruber	40.64
7	Daan	Peeters	30.69
8	Kara	Nielsen	31.68
9	Eduardo	Martins	28.71
10	Alexandre	Rocha	21.78
11	Roberto	Almeida	36.63
12	Fernanda	Ramos	25.74
13	Mark	Philips	28.71
14	Jennifer	Peterson	22.78
15	Frank	Harris	36.63
16	Jack	Smith	27.74
17	Michelle	Brooks	37.62
18	Tim	Goyer	13.87
19	Dan	Miller	39.62
20	Kathy	Chase	27.72
21	Heather	Leacock	39.62
22	John	Gordon	14.85
23	Frank	Ralston	43.62
24	Victor	Stevens	35.69
25	Richard	Cunningham	45.64
26	Patrick	Gray	28.71
27	Julia	Barnett	27.78
28	Robert	Brown	36.63
29	Edward	Francis	31.68
30	Martha	Silk	28.71
31	Aaron	Mitchell	21.78
32	Ellie	Sullivan	36.63
33	João	Fernandes	27.74
34	Madalena	Sampaio	37.62
35	Hannah	Schneider	21.78
36	Fynn	Zimmermann	42.63
37	Niklas	Schröder	25.74
38	Camille	Bernard	38.62
39	Dominique	Lefebvre	13.87
40	Marc	Dubois	37.62
41	Wyatt	Girard	28.73
42	Isabelle	Mercier	40.62
43	Terhi	Hämäläinen	32.71
44	Ladislav	Kovács	45.62
45	Hugh	O'Reilly	38.69

```
46      Lucas       Mancini       35.64
47   Johannes  Van der Berg       31.71
48  Stanisław        Wójcik       21.78
49    Enrique         Muñoz       36.63
50     Joakim     Johansson       32.68
51       Emma         Jones       28.71
52       Phil        Hughes       21.78
53      Steve        Murray       36.63
54       Mark        Taylor       25.74
55      Diego     Gutiérrez       37.62
56       Luis         Rojas       30.78
57      Manoj        Pareek       38.62
58       Puja   Srivastava       26.74
```

[ ]: