Week 13 Quiz

Clarissa Tai - rt2822

Due Tues Dec 13th, 11:59pm ET

In this quiz we'll practice using SQL to extract and transform some US State population data.

We'll use pandasql to execute SQL on pandas dataframes.

If for some reason pandasql deosn't work, please just take a shot at what you think the SQL should be.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from pandasql import PandaSQL

%matplotlib inline
```

Set up pysqldf

In [2]:
```
# Setting up an instance of PandaSQL to pass SQL commands to
pysqldf = PandaSQL()
```

Load Data

In [3]:
```
# Load state population data
state_population = pd.read_csv('../data/state-population.csv')
state_population = state_population.rename({'state/region':'abbreviation'},
                                           axis=1)

# Load state area data
```

```
state_areas = pd.read_csv('../data/state-areas.csv')
state_areas = state_areas.rename({'area (sq. mi)':'area'},axis=1)

# Load state abbreviation data
state_abbreviations = pd.read_csv('../data/state-abbrevs.csv')
```

Practice SQL

In [4]:

```
# Write SQL to print out:
#    all columns from table state_areas limited to the first 3 rows
#    aka state_areas.head(3)
sql = """
SELECT *
FROM state_areas
LIMIT 3
"""
pysqldf(sql)
```

Out[4]:

	state	area
0	Alabama	52423
1	Alaska	656425
2	Arizona	114006

In [5]:

```
# Write SQL to print out the same dataframe as given by this pandas call:
#    state_population.loc[:,['abbreviation']].iloc[:3]
sql = """
SELECT S.abbreviation
FROM state_population as S
LIMIT 3
"""
pysqldf(sql)
```

Out[5]:

	abbreviation
0	AL
1	AL
2	AL

OPTIONAL

Everything below is optional if you'd like more practice with sql

In [ ]:
```
# Write SQL to print out:
#    columns state and area from table state_areas for rows with state starting with 'Mi'
sql = """

"""
pysqldf(sql)
```

In [ ]:
```
# Write SQL to print out:
#    columns state and area from table state_areas
#    for rows with state starting with 'Mi' and area greater than 80000
sql = """

"""
pysqldf(sql)
```

In [ ]:
```
# Write SQL to print out:
#    all columns from table state_areas
#    LEFT JOINed with state_abbreviations ON state
#    limited to the first 3 rows
sql = """

"""
pysqldf(sql)
```

In [ ]:
```
# Write SQL to print out:
#    all columns from table state_areas aliased as s_area
#    INNER JOINed with state_abbreviations aliased as s_abb ON state
#    INNER JOINed with state_population aliased as s_pop ON abbreviation
#    limited to the first 3 rows
sql = """

"""
pysqldf(sql)
```

In [ ]:

```
# Write SQL to print out:
#    s_area.state,
#    s_area.area,
#    s_pop.year,
#    s_pop.population
#    from table state_areas aliased as s_area
#    INNER JOINed with state_abbreviations aliased as s_abb ON state
#    INNER JOINed with state_population aliased as s_pop ON abbreviation
#    where s_pop.ages is 'total'
#    ordered by s_area.state, s_pop.year
sql = """

"""
pysqldf(sql)
# you should see 1224 rows and 4 columns
```

In [ ]:

```
# Feel free to experiment with additional SQL calls.
# For example, state_population contains more regions than there are states in state_areas
#     so different join types (left, right) will give different results

# As an additional challenge:
#  calculate the total average population per state in thousands over the years observed
sql = """

"""
pysqldf(sql)
```

In [ ]:

```

```