processingF

April 10, 2020

Before you turn this problem in, make sure everything runs as expected. First, **restart the kernel** (in the menubar, select Kernel→Restart) and then **run all cells** (in the menubar, select Cell→Run All).

Make sure you fill in any place that says `YOUR CODE HERE`/`raise NotImplementedError` or "YOUR ANSWER HERE", as well as your name and collaborators below:

Jay Dickson and Jill Reiner

1 Processing and SQL for Relational Database Project

[1]:
```
import pandas as pd
import os
import os.path
import json
import sqlalchemy as sa
from IPython.display import Image

db_source = "sqlite"
datadir = "02_project/"
```

[2]:
```
def getsqlite_info(dirname=".",filename="creds.json"):
    """
    Purpose:open a credentials file and obtain the four parts needed for a␣
↪connection string to
        a remote provider using the "mysql" dictionary within
        an outer dictionary.
    Parameters:
        directory: directory name
        filename: filename to use
    Return
        scheme, server, user, and password
    """
    assert os.path.isfile(os.path.join(dirname, filename))
    with open(os.path.join(dirname, filename)) as f:
        D = json.load(f)
    sqlite = D["sqlite"]
    return sqlite["scheme"], sqlite["basepath"], sqlite["database"]
```

[3]:
```
def dataFrameToCSV(dataFrame, fileName):
    """
    Purpose: to write a dataframe to a CSV
    Parameters:
        dataFame: the dataframe to be written to a CSV
        fileName: the name of the file to be written given as a string
    Return:
        CSV: the CSV file in the local directory
    """
    CSV = dataFrame.to_csv(fileName + '.csv') #converts pandas dataframe to CSV␣
↪file
    return CSV
```

[4]:
```
def hr_so(dbcon, yearstart, yearend, ABthresh):
    """
    Purpose: create a dataframe to show the correlation between strikeouts and␣
↪home runs
    Parameters:
        dbcon: the database which will be connecting to
        yearstart: the start year to look at
        yearend: the end year to look at
        ABthresh: a cutoff threshold for at bats
    Return:
        df: a dataframe
    """
    query1 = """
    SELECT DISTINCT b.yearID AS Year, b.teamID, t.lgID, b.HR, b.SO AS K, --␣
↪selects year, team, league, home runs, strikeouts
        p.nameLast ||', '|| p.nameFirst AS name -- creates a new column of name␣
↪using the format lastname, firstname
    FROM batting AS b INNER JOIN teams as t -- join batting and teams
        ON t.teamID = b.teamId -- common field is teamID
    INNER JOIN people AS p -- join with people table
        USING(playerID) -- common field is playerID
    WHERE b.yearID BETWEEN '{}' AND '{}' -- user can input any year range
    AND b.AB > '{}' -- user can input any at bat threshold
    """

    query = query1.format(yearstart, yearend, ABthresh) #uses query and␣
↪parameters inputted
    df = pd.read_sql_query(query, con = dbcon) #converts query to dataframe
    return df
```

```
[5]: def RBI_byPOS(dbcon, yearstart, yearend, ABthresh, excludePOS):
         """
         Purpose: create a dataframe to show the correlation between RBIs and␣
      ↪Positions
         Parameters:
             dbcon: the database which will be connecting to
             yearstart: the start year to look at
             yearend: the end year to look at
             ABthresh: a cutoff threshold for at bats
             excludePOS: positions we do not want to include in the dataframe
         Return:
             df: a dataframe
         """
         query2 = """
         SELECT DISTINCT b.yearID, b.teamID, POS, RBI, -- select year, team,␣
      ↪position, RBI
                 nameLast ||', '|| nameFirst AS name -- creates new column with name,␣
      ↪same as last function
         FROM batting AS b LEFT JOIN people AS p -- join batting and people using␣
      ↪common field of playerID
             USING(playerID) INNER JOIN fielding AS f -- join with fielding
                 USING(playerID) -- using common field of playerID
         WHERE b.yearID >= :low AND b.yearID < :high AND b.AB > :thresh AND POS <> :
      ↪exclude -- bindparams operations, explained below
         ORDER BY RBI DESC
         LIMIT 250
         """

         prepare_stmt = sa.sql.text(query2) #prepare statement, combination of SQL␣
      ↪syntax and elements designating the places where the value of a variable␣
      ↪should be substituted
         bound_stmt = prepare_stmt.bindparams(low = yearstart, high = yearend,␣
      ↪thresh = ABthresh, exclude = excludePOS) #bindparams uses named parameters␣
      ↪as its arguments, where the named parameters are named similarly to the␣
      ↪required fields of our query
         df = pd.read_sql_query(bound_stmt, con = dbcon) #executes our bound_stmt,␣
      ↪same operation as fetchall() or execute(), turns into dataframe
         return df
```

```
[6]: def HR_HOF(dbcon, year=2019):
         """
         Purpose: to find the league leader in Home Runs for given years and␣
      ↪determine if they are in the Hall of Fame
         Parameters:
             dbcon: the database which will be connecting to
             year: the end year for filtering
```

```
    Return:
        df3: a dataframe which will be used to make into a CSV
    """
    query3 = """
    SELECT Name, Year, HomeRun, IFNULL(HOF,'N') AS HOF FROM
        (SELECT BT.yearID AS Year, pl.nameLast ||', '|| pl.nameFirst AS Name,␣
↪teamID AS Team, MAX(BT.HR) AS HomeRun, inducted AS HOF
        FROM batting AS BT LEFT JOIN halloffame
            USING(playerID)
        LEFT JOIN people AS pl
            USING(playerID)
        WHERE BT.yearID <= {}
        GROUP BY BT.yearID
        ORDER BY BT.yearID) AS BestHitters
    GROUP BY Name
    ORDER BY HomeRun DESC
    """

    query3 = query3.format(year) # change the year given in parameter
    df3 = pd.read_sql_query(query3, con = dbcon,index_col="Year") # a dataframe␣
↪from the SQL Query
    return df3
```

```
[7]: def team_avg(dbcon, BPthresh, yearstart, yearend):
    """
    Purpose: create a dataframe to show the correlation between strikeouts and␣
↪home runs
    Parameters:
        dbcon: the database which will be connecting to
        BPthresh: Batting average threshold, the minimum threshold we seek to␣
↪look at
        yearstart: the start year to look at
        yearend: the end year to look at
    Return:
        df: a dataframe
    """
    query4 = """
    SELECT yearID AS year, lgID AS league, teamID, avg(BP) AS avg_ba -- selects␣
↪year, league, team, and batting average from subquery
    FROM (SELECT b.teamID, b.yearID, l.lgID, (1.0 * b.H / b.AB) AS BP --␣
↪subquery computes batting percentage field
          FROM batting AS b LEFT JOIN leagues AS l -- join batting and leagues
             USING(lgID) -- using common field, lgID
          WHERE BP <> '{}' AND -- choose what you don't want batting percentage␣
↪to be equal to
            b.yearID > '{}' AND -- choose start year
            b.yearID < '{}') -- choose end year
    GROUP BY yearID, teamID -- group by year and team
    ORDER BY avg_ba DESC
    """

    query = query4.format(BPthresh, yearstart, yearend) #uses query and␣
↪parameters inputted
    df = pd.read_sql_query(query, con = dbcon) #converts to pandas dataframe
    return df
```

[8]:
```
def main():
    if db_source == "sqlite":
        scheme, basepath, db = getsqlite_info()
        template = '{}:///{}.db'
        cstring = template.format(scheme, os.path.join(basepath, db))
    elif db_source == "mysql":
        scheme, server, user, password, db = getmysql_creds()
        template = '{}://{}:{}@{}/{}'
        cstring = template.format(scheme, user, password, server,db)
    else:
        raise ValueEror

    engine=sa.create_engine(cstring)
#     connection = engine.connect()
    with engine.connect() as connection:
        %load_ext sql
        %sql $cstring

        yearstart = """2015"""
        yearend = """2020"""
        ABthresh = """500"""
        df1 = hr_so(connection, yearstart, yearend, ABthresh)
        dataFrameToCSV(df1,'hr_so')

        df2 = RBI_byPOS(connection, 2000, 2020, 0, 'P')
        dataFrameToCSV(df2, 'RBIbyPOS')

        df3 = HR_HOF(connection, 2005)
        dataFrameToCSV(df3, 'HR_HOF')

        BPthresh = """0"""
        yearstart = """1919"""
        yearend = """2020"""
        df4 = team_avg(connection, BPthresh, yearstart, yearend)
        dataFrameToCSV(df4, 'TeamAvg')
```

[9]:
```
main()
```