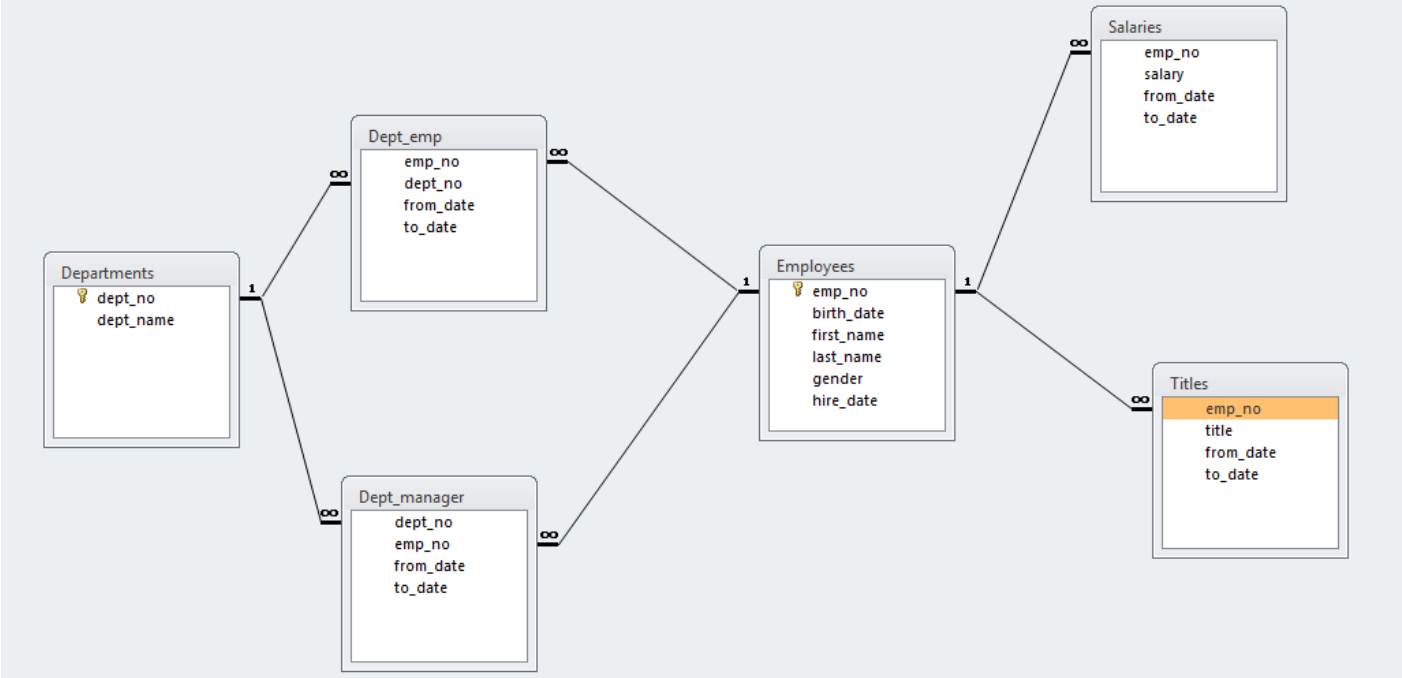
Consultas en Pandas de Python para datos en tablas relacionadas

Rafael Gamboa Hirales, Jean Paul Virueña

Abril de 2019

La evaluación consiste en llevar a cabo consultas a la base de datos que se le proporciona.
El esquema de entidad relación es el siguiente:



Lea previamente todas las preguntas y tome tiempo para estudiar los datos.
Los datos se proporcionan en archivos de texto de tipo ".txt"

Lectura de archivos

Después de haber importado las librerías que se van a usar (pandas, numpy y pickle), se deben leer los datos y si se desea trabajar en una misma computadora se recomienda fuertemente almacenar los datos de manera binaria en un archivo pickle o en un txt.

Para importar las librerías:

```
#Se importan las librerías pandas, numpy y pickle
import pandas as pd
import numpy as np
import pickle as pck
```

Para leer los archivos:

Para facilitar el trabajo se recomienda que una vez leídos los datos en el formato original (texto) se guarden en binario (en imagines compatible con pandas) y al leerlos de nuevo no tener que volver a aplicar la conversión de fechas.

```
# Se declara la trama con la que vamos a leer los archivos convirtiendo de texto a Datetime64 las
#fechas y convirtiendo aquellas con año igual a 9999 a un NaT (Not a Time)

tray="D:/jpvi/Employees/Python/EmployeesTxt/"
#Se leen los archivos con ayuda de pd.read_csv y se guardan en una variable

# tabla de empledos
emp=pd.read_csv(tray+"Employees.txt", parse_dates=['birth_date','hire_date'],
dayfirst=True,na_values=["1/1/9999 00:00:00"])

# tabla de departamentos
dept=pd.read_csv(tray+"Departments.txt")

# tabla de empleados por departamento
dept_emp=pd.read_csv(tray+"Dept_emp.txt", parse_dates=['from_date','to_date'],
dayfirst=True,na_values=["1/1/9999 00:00:00"])

# tabla  de gerentes por departamento
dept_manager=pd.read_csv(tray+"Dept_manager.txt", parse_dates=['from_date','to_date'],
dayfirst=True,na_values=["1/1/9999 00:00:00"])

# tabla de sueldos
salaries=pd.read_csv(tray+"Salaries.txt", parse_dates=['from_date','to_date'],
dayfirst=True,na_values=["1/1/9999 00:00:00"])

#tabla de títulos para los empleados
titles=pd.read_csv(tray+"Titles.txt", parse_dates=['from_date','to_date'],
dayfirst=True,na_values=["1/1/9999 00:00:00"])
```

Para guardar los datos en binario dentro de archivo:

```
#Se almacenan los datos en un archivo con ayuda de filehandler y pickle.dump()
# 'wb' significa write binary
filehandler = open("emp",'wb')
pck.dump(emp,filehandler)
filehandler.close()

filehandler = open("dept",'wb')
```

```
pck.dump(dept,filehandler)
filehandler.close()

filehandler = open("dept_emp",'wb')
pck.dump(dept_emp,filehandler)
filehandler.close()

filehandler = open("dept_manager",'wb')
pck.dump(dept_manager,filehandler)
filehandler.close()

filehandler = open("salaries",'wb')
pck.dump(salaries,filehandler)
filehandler.close()

filehandler = open("titles",'wb')
pck.dump(titles,filehandler)
filehandler.close()
```

Para leer los archivos en binario:

```
#Se cargan las variables con ayuda de open() y pickle.load()
# 'rb' significa read binary

        file = open("emp",'rb')
        emp = pck.load(file)
        file.close()

        file = open("dept",'rb')
        dept = pck.load(file)
        file.close()

        file = open("dept_emp",'rb')
        dept_emp = pck.load(file)
        file.close()

        file = open("dept_manager",'rb')
        dept_manager = pck.load(file)
        file.close()

        file = open("salaries",'rb')
        salaries = pck.load(file)
        file.close()
```

```
file = open("titles",'rb')
titles = pck.load(file)
file.close()
```

Preguntas

1. Considere la última fecha de contratación, salida, cambio de status. ¿Cuál es esa fecha?

Como se pide la última fecha de contratación, salida, cambio de status, … una estrategia es juntar todas las fechas de todas las tablas y de ahí tomar la fecha más reciente. Nótese que hay que omitir fechas con año 9999.

Para juntar todas las fechas de todas las tablas se usa el método .concat() de Pandas:

NOTA: las funciones de agregación que tienen que ver con fechas ignoran los valores NaT

```
#Se juntan las fechas con la ayuda de pandas.concat()
allDates=pd.concat([dept_emp.from_date, dept_emp.to_date, emp.hire_date,
salaries.from_date, salaries.to_date, dept_manager.from_date, dept_manager.to_date,
titles.to_date, titles.from_date])
```

Después, con .max() se consigue la mayor fecha

```
print('\n Pregunta 1:')
print("La última fecha de contratación, salida, cambio de status es:"+(str)(allDates.max()))
```

Así:

```
 Pregunta 1:
La última fecha de contratación, salida, cambio de status es: 2002-08-01 00:00:00
```

2. ¿A qué edad(es) contrata más personal la empresa?

Como se pide a qué edad(es) contrata más personal la empresa, se calculará la edad de los empleados cuando entraron y se contarán esas edades.

Para calcular esa edad, a la fecha de contratación del empleado se le resta la fecha de nacimiento de este:

```
#Se añade una serie (columna) al DataFrame emp donde se calcula la edad a la que el empleado fue contratado

#se divide entre un intervalo de tiempo (np.timedelta64) de un año para poder redondear la edad al año y se convierte a entero (np.int64) para que tenga el formato correcto

emp['hire_age']=((emp.hire_date - emp.birth_date)/np.timedelta64(1,'Y')).astype(np.int64)
```

Después se cuentan las edades y se ordenan de mayor a menor:

```
#Se cuentan las edades de los empleados y se ordenan a los empleados por edad de mayor a menor y

#se sacan los índices, después, se sacan el índice y los valores de la primera edad.

maxAgeIndex=emp.groupby(['hire_age']).hire_age.count().sort_values(ascending=False)[:1].index[0]

print('\n Pregunta 2:')

print("La edad a la que la empresa contrata más empleados es a los "+(str)(maxAgeIndex)+' años')
```

Así:

```
 Pregunta 2:
La edad a la que la empresa contrata más empleados es a los 32 años
```

3. ¿Cuál(es) es(son) la(s) edad(es) a la que se contratan más los hombres?

Como se piden la(s) edad(es) a la que se contratan más los hombres, se cuentan las edades de los empleados en el momento en el que fueron contratados y se ordenan de mayor a menor. Después, se filtran a los hombres.

Para esto se realiza un .groupby() en el que se cuenten y se ordenen el género y la edad de los empleados:

```
#Se cuentan las edades a las que fueron contratados los empleados y se ordenan de mayor a menor

x=emp.groupby(['gender','hire_age']).hire_age.count().sort_values(ascending=False)
```

Para filtrar a los hombres solo se pide que en la serie ‘gender’ contengan como valor ‘M’:

```
xm=x['M']
```

```
print('\n Pregunta 3:')

print('La empresa contrata más a los hombres de: '+str(xm.sort_values(ascending=False).index[0])+" años")
```

Así:

```
 Pregunta 3:
La empresa contrata más a los hombres de 32 años
```

4. ¿Cuál(es) es(son) la(s) edad(es) a la que se contratan más las mujeres?

Mismo procedimiento que en el inciso anterior solo que esta vez se filtraran a las mujeres.

Para esto como ya se tiene el .groupby() anterior solo será necesario filtrar a las mujeres, es decir pedirle que en la serie 'gender' contengan como valor 'F':

```
#Se filtran para solo tener a las mujeres

xf=x['F']

print('\n Pregunta 4:')

print('La empresa contrata más a las mujeres de: '+str(xf.sort_values(ascending=False).index[0])+" años")
```

Así:

```
 Pregunta 4:
La empresa contrata más a las mujeres de 32 años
```

5. ¿Cuántos empleados tiene actualmente la empresa?

Como se pide la cantidad de empleados que tiene la empresa en una fecha dada (la actual), se le pedirá al usuario que introduzca la fecha. Después, se filtran a los empleados que trabajan en la fecha dada.

Para pedirle al usuario la fecha se pide un input y se convierte a datetime:

```
#Se solicita como entreda la fecha actual

date=pd.to_datetime(input('\nDame la fecha de hoy (aaaa-mm-dd): \n'))
```

```
Dame la fecha de hoy (aaaa-mm-dd):
1999-01-01
```

Para filtrar a los empleados se buscan a aquellos que fueron contratados antes de la fecha ingresada y que dejaron de ser contratados o sigan trabajando a la fecha dada (que .to_date sea NaT):

```
#Mediante filtros se excluyen a los empleados que no están activos (que la fecha de
#cuando el empleado ingresó al departamento donde trabaja sea menor a la fecha actual
#o que la fecha de cuando se fue del departamento sea mayor a la fecha dada o que sea
#nula, ya que sigue estando en ese departamento)

f1 = dept_emp.from_date < date

f2_1 = dept_emp.to_date >= date

f2_2 = dept_emp.to_date.isnull()

f2 = f2_2 | f2_1

f3 = f2 == f1
```

Una vez aplicado este filtro, existen dos versiones para contestar a la pregunta.

Versión 1 (mediante índices):

Se le pide al DataFrame que devuelva solo los elementos que corresponden al filtro y se cuentan a los empleados:

```
#Se le pide al DataFrame los índices que corresponden al filtro y se cuentan los datos

p5v1 = dept_emp[f3].emp_no.count()

print('\n Pregunta 5 versión 1:')

print('Actualmente la empresa tiene '+str(p5v1)+' empleados')
```

Así:

```
 Pregunta 5 versión 1:
Actualmente la empresa tiene 244366 empleados
```

Versión 2 (se usa la función .where() de pandas):

Se aplica la la función .where() con el filtro y se cuentan a los empleados:

```
#Se le aplica la función .where() al DataFrame y se cuentan los datos

f5 = dept_emp.where(f3)

p5v2 = f5.emp_no.count()

print('\n Pregunta 5 versión 2:')

print('Actualmente la empresa tiene '+str(p5v2)+' empleados')
```

6. Obtenga la distribución actual de las edades.

Como se pide la distribución de las edades en una fecha dada, se calcularán las edades como en la pregunta 2 y se le aplicará el filtro de la pregunta 5.

Para esto, además de añadir la edad actual de los empleados al DataFrame, también se deberán juntar los datos de dept_emp, dept y emp con un .merge() y al final se filtrarán y se contarán las edades:

```
#Se le añade una serie (columna) a solo los datos filtrados del DataFrame de empleados.
#En esta serie se calcula la edad del empleado en la fecha dada

emp['actual_age']=((date - emp.birth_date)/np.timedelta64(1,'Y')).astype(np.int64)[f3]

#Se juntan los DataFrames dept_emp y dept con la ayuda de pd.merge() a partir de su clave única

m1 = pd.merge(dept_emp, dept, on='dept_no', how = 'left')

#Este DataFrame generado se junta con emp con solo los datos de los empleados actuales

f8 = pd.merge(m1, emp, on = 'emp_no')[f3]

#Se cuentan y agrupan las edades actuales

p6 = f8.groupby(['actual_age']).actual_age.count()

print('\n Pregunta 6:')

print('La distribución actual de las edades: \n'+str(p6))
```

Así:

```
 Pregunta 6:
La distribución actual de las edades:
actual_age
33.00     1173
34.00    13762
35.00    13806
36.00    13847
37.00    13877
38.00    13814
39.00    14088
40.00    13990
41.00    13711
42.00    13702
43.00    13847
44.00    14023
45.00    13712
46.00    12868
Name: actual_age, dtype: int64
```

7. ¿Cuál es el departamento con más empleados?

Como se pide el departamento con más empleados a una fecha dada, se usará la consulta de la pregunta 5 versión 2 o bien, se aplicará el filtro de la versión 1 a dept_emp y se contaran los empleados agrupándolos por número de departamento, ordenándolos de mayor a menor y extrayendo el primero. Después se solicitará el nombre de ese departamento:

```
#Se cuentan y ordenan de mayor a menor a los empleados y se ordenan por número de
#departamento pidiendo la información del primer dato para después ver en dept cómo se
#llama este departamento

f6 = f5.groupby(['dept_no']).emp_no.count().sort_values(ascending=False)[:1].index[0]

f7 = dept.where(dept.dept_no == f6)

p7 = f7.dept_name.sort_values(ascending=False)[:1].values[0]

print('\n Pregunta 7: ')

print(p7+' es el departamento con más empleados')
```

Así:

```
 Pregunta 7:
Development es el departamento con más empleados
```

8. Obtenga la distribución actual de hombres y mujeres por departamento.

Como se pide la distribución de hombres y mujeres por departamento en una fecha dada, se usará el DataFrame que une a dept_emp, dept y emp que se creó en la pregunta 6 y se le pedirá que cuente el género de las personas agrupando por nombre del departamento en el que trabaja el empleado y por género:

```
#Se cuenta el genero de los empleados que trabajan en ese momento y se agrupan por departamento

p8 = f8.groupby(['dept_name', 'gender']).gender.count()

print('\n Pregunta 8:')

print('La distribución actual de hombres y mujeres por departamento es: \n'+ str(p8))
```

Así:

```
 Pregunta 8:
La distribución actual de hombres y mujeres por departamento es:
dept_name           gender
Customer Service    F          6441
                    M          9748
Development         F         25727
                    M         38806
Finance             F          5259
                    M          7739
Human Resources     F          5337
                    M          8059
Marketing           F          5702
                    M          8730
Production          F         21787
                    M         32351
Quality Management  F          5791
                    M          8601
Research            F          5946
                    M          8937
Sales               F         15722
                    M         23683
Name: gender, dtype: int64
```

9. Obtenga el importe del total del sueldo anual (el que aparece en los datos es sueldo anual) y sueldo promedio por departamento clasificado por género.

Como se solicita el sueldo anual y sueldo promedio por departamento clasificado por género, se filtran los salarios de manera similar a como se hizo en la pregunta 5 y se le piden al DataFrame salaries solo los datos que cumplen con las condiciones del filtro:

```
#Se filtran los salarios actuales

f9_1 = salaries.from_date < date

f9_21 = salaries.to_date >= date

f9_22 = salaries.to_date.isnull()

f9_2 = f9_21 | f9_22

f9_3 = f9_2 == f9_1

f9 = salaries[f9_3]
```

Después, se juntas los DataFrames de dept_emp, dept, emp y salaries y se calculan la suma y el promedio de los salarios agrupando departamento y género:

```
#Se juntas los Dataframes de los salarios actuales con la información de los empleados y su
#departamento, en este nuevo DataFrame se calcula la suma de los salarios y su promedio
#y se concatenan los resultados

m3 = pd.merge(f9, f8, on = 'emp_no', suffixes = ('_dep', '_salaries'))

p9_1 = m3.groupby(['dept_name', 'gender']).salary.sum()

p9_2 = m3.groupby(['dept_name', 'gender']).salary.mean()

p9 = pd.concat([p9_1, p9_2], axis = 1)

print('\n Pregunta 9:')

print('El importe es: \n' + str(p9))
```

Así:

```
 Pregunta 9:
El importe es:
                                 salary    salary
dept_name          gender
Customer Service   F         386189195 59,957.96
                   M         580428358 59,543.33
Development        F        1572477078 61,121.67
                   M        2375627722 61,218.05
Finance            F         381439185 72,530.74
                   M         560037722 72,365.64
Human Resources    F         305090374 57,165.14
                   M         459933831 57,070.83
Marketing          F         417983676 73,304.75
                   M         644202950 73,791.86
Production         F        1335077566 61,278.63
                   M        1990599682 61,531.32
Quality Management F         342850017 59,203.94
                   M         505867362 58,814.95
Research           F         366992349 61,720.88
                   M         550923898 61,645.28
Sales              F        1297506898 82,528.11
                   M        1953878105 82,501.29
```

10. Obtenga el sueldo promedio por puesto (“title”).

Como se pide el sueldo promedio por puesto, se juntan los puestos con los salarios y se calcula el promedio de los salarios agrupados por puesto:

```
#Se juntan los títulos y los salarios por el número de empleado y se calcula el sueldo
#promedio por puesto

m4 = pd.merge(titles, salaries, on = 'emp_no', suffixes = ('_titles', '_salaries'))

p10 = m4.groupby(['title']).salary.mean()

print('\n Pregunta 10:')

print('El sueldo promedio por puesto es : \n'+str(p10))
```

Así:

```
 Pregunta 10:
El sueldo promedio por puesto es :
title
Assistant Engineer   59,304.99
Engineer             59,508.04
Manager              66,924.27
Senior Engineer      60,543.22
Senior Staff         70,470.84
Staff                69,309.10
Technique Leader     59,294.37
Name: salary, dtype: float64
```

11. Verifique si actualmente todo “manager” de un departamento es empleado de ese departamento.

Como se quiere ver que el manager de cada departamento es empleado de ese departamento, se filtran a los “managers” para tener solo a los que son “manager” a la fecha dada y se buscan los índices de esos “managers” en el DataFrame dept_emp. Por último, se valida que todos los “managers” actuales estén en ese filtro:

```
#Se filtran a los "managers" actuales, se buscan en la tabla dept_emp y se verifica que los
#"managers" actuales estén como empleados actuales

f11_11 = dept_manager.to_date.isnull()

f11_1 = dept_manager[f11_11]

m5 = f5[f5.emp_no.isin(f11_1.emp_no)]

f12 = m5.emp_no.values == f11_1.emp_no.values

f13 = m5.dept_no.values == f11_1.dept_no.values

p11 = (False in f12) and (False in f13)

print('\n Pregunta 11: ')

if(not p11):

   print('Actualmente todo “manager” de un departamento es empleado de ese
departamento')
```

else:

print('Actualmente no todo “manager” de un departamento es empleado de ese departamento')

Así:

```
 Pregunta 11:
Actualmente todo “manager” de un departamento es empleado de ese departamento
```