Alfonso Ayala Paloma

alfonsoayalapaloma@gmail.com

Descripción breve

Instrucciones para realizar consultas a dataframes de pandas

CONSULTANDO PANDAS DATAFRAMES

Actividad de Contextualización ETL

Contenido

[Mejores películas calificadas del dataset 2](#_Toc128421059)

[Películas con mayor revenue con release date mayor a 2010-01-01 2](#_Toc128421060)

[Películas más rentables del dataset 4](#_Toc128421061)

[Películas Colombianas más rentables del dataset 4](#_Toc128421062)

[Películas más rentables de la historia 6](#_Toc128421063)

[Contenidos compartidos 7](#_Toc128421064)

[Cuales son los contenidos de amazon o disney que no tiene netflix 8](#_Toc128421065)

[Cuales son los contenidos con menor Budget pero con el mayor vote\_average que debe adquirir netflix 9](#_Toc128421066)

[Crear el pipeline 10](#_Toc128421067)

## Mejores películas calificadas del dataset

**Como pre-requisito, se debe tener un dataframe con la base de TMDB y los tres archivos .csv con los títulos de Netflix, amazon y Disney.**

**En colab, ejecutar:**

def query01(tmdb):

#Mejores peliculas calificadas del dataset

#Se buscan aquellas con mas de 20000 votos,

Df1=tmdb[tmdb.vote\_count>20000]

#Se ordenan de mayor a menor vote\_average

Df2=Df1.sort\_values(by=[‘vote\_average’],ascending=False)

#Se toman las diez primeras

print(Df2[:10])

# En una sola linea:

return tmdb[tmdb.vote\_count>20000].sort\_values(by=['vote\_average'], ascending=False)[:10]

## Películas con mayor revenue con release date mayor a 2010-01-01

**Como pre-requisito, se debe tener un dataframe con la base de TMDB.**

**En colab, ejecutar**

def query02(tmdb):

#Se buscan aquellas con release\_date > '2010-01-01'

#se ordenan de mayor a menor revenue

#Se toman las 10 primeras

return tmdb[tmdb.release\_date>'2010-01-01'].sort\_values(by=['revenue'], ascending=False)[:10]

## Películas más rentables del dataset

En colab, ejecutar

def query03(tmdb):

#Crear la columna “rentabilidad”

tmdb['rentabilidad']=tmdb['revenue']/ tmdb['revenue']

#ordenar descendentemente por rentabilidad y tomar las 10 primeras filas.

return tmdb.sort\_values(by=['rentabilidad'], ascending=False)[:10]

## Películas Colombianas más rentables del dataset

En colab, ejecutar

def query04(tmdb):

#define la columna a trabajar

col="production\_countries"

#elimina los NaN de la columna col

dfco=tmdb.dropna(subset=[col])

#reindexa el dataframe para que los índices sean consecutivos.

dfco=dfco.reset\_index(drop=True)

#Se seleccionan 5 columnas

dfco=dfco[[col,'revenue','budget','vote\_average','title']]

#Se un dataset con tantas columnas como elementos tenga el arreglo de json de países de producción.

data=pd.json\_normalize(dfco[col])

#Se muestra cuantas columnas tiene el dataset data

createdcols=len(data.columns)

#Por cada columna del dataset data (json) cree dos columnas una que tenga el nombre iso\_XXX y la otra name\_XXX donde XXX es el numero de la columna del dataset data

dfprod=""

for i in data.columns:

prefix="p\_"+str(i)+"."

dfn=pd.json\_normalize(data[i])

dfn.columns = ['iso\_'+str(i), 'name\_'+str(i)]

if i==0:

dfprod=pd.concat([dfco, dfn], axis=1)

else:

dfprod=pd.concat([dfprod, dfn], axis=1)

#coloque en un arreglo las columnas creadas

cols = []

for i in range(0,createdcols):

cols.append( 'iso\_'+str(i) )

#Aplique una function a las columnas del arreglo “cols” que una todas esas columnas en un solo campo texto separado por comas.

dfprod['countries'] = dfprod[cols].apply(lambda row: ','.join(row.values.astype(str)), axis=1)

#Busque en ese campo texto aquellas que tengan la palabra ‘CO’

dfprodcol =dfprod[dfprod.countries.str.contains('CO')]

#Ordenar descendentemente por vote\_average

return dfprodcol.sort\_values(by='vote\_average',ascending=False)[:10]

## Películas más rentables de la historia

Correr en colab

#Funcion para calcular la rentabilidad

def findprofit(df):

#tome los budgets mayores a cero

df=df[df["budget"]>0 ]

#tome los revenue mayores a cero

df=df[df["revenue"]>0 ]

#Cree la nueva columna “rentabilidad”

df["rentabilidad"]=df["revenue"]/df["budget"]

#ordene por rentabilidad descendentemente

df=df.sort\_values(by=['rentabilidad'] , ascending=False)

#print(df.head())

#retorne el dataframe

return df

def query05(tmdb):

dfa=findprofit(tmdb)

return dfa.head()

## Contenidos compartidos

Correr en colab

# Funcion para cargar los archivos de títulos en dataframes

def load\_titles(filename):

df=pd.read\_csv(filename)

return df

def query06():

#Cargue los archivos y defina la columna “Streaming” con el nombre de cada compañía.

dfdis = load\_titles('disney\_plus\_titles.csv')

dfdis['streaming']='disney'

dfnet = load\_titles('netflix\_titles.csv')

dfnet['streaming']='netflix'

dfama = load\_titles('amazon\_prime\_titles.csv')

dfama['streaming']='amazon'

#concatene todos los df

frames = [dfdis, dfnet, dfama]

dftotal = pd.concat(frames)

#Cree una Serie agrupando por cantidad de titulos

scantidad = dftotal.groupby(['title'])['title'].count()

#Cree una serie que tenga aquellas filas cuya cuenta sea 3.

compartidos =scantidad[scantidad==3]

return compartidos

## Cuales son los contenidos de amazon o disney que no tiene netflix

Correr en colab

def query07(dfdis, dfama, dfnet):

# Cree un df otros que contenga los titulos de Disney y amazon.

otros=[dfdis, dfama]

dfotros=pd.concat(otros)

#Cree un df únicamente con la columna título de disney y amazon.

dfotrost=dfotros['title']

#Cree un df con la columna titulo de netflix

dfnett=dfnet['title']

#obtenga los que están en otros y también en Netflix

Ambos=dfotrost.isin( dfnett )

#obtenga las filas cuyo indice no este en la serie Ambos

res = dfotrost[~ Ambos ]

return res

## Cuales son los contenidos con menor Budget pero con el mayor vote\_average que debe adquirir netflix

Prerequisito: la serie “res” que contiene los títulos que no tiene Netflix. Correr en colab

def query08(tmdb, res):

# Cree una lista de los faltantes

missinglist = res.tolist()

#obtenga del df de TMDB aquellos títulos que están en la lista de faltantes

dfmiss = tmdb[ tmdb.title.isin(missinglist) ]

# del df de faltantes, extraiga los que tienen presupuesto mayor a 0.

dfmiss=dfmiss[dfmiss.budget>0]

#ordene por budget ascendentemente y tome 100

dfmiss=dfmiss.sort\_values(by=['budget'])[:100]

#ordene por vote\_average descendentemente.

dfmiss=dfmiss.sort\_values(by=['vote\_average'], ascending=False )

#Presente los resultados

return dfmiss[['budget','vote\_average', 'title']]

## Crear el pipeline

El pipeline extrae de las dos tablas de postgres dos dataframes.

Luego extrae azure y de tmdb.

Crea un arreglo de dataframes y lo recorre mostrando .info() y .head()

# Pipeline

dfama = extract\_postgres("amazon\_movie")

dfdis = extract\_postgres("disneyplus\_movie")

dfnet = extract\_azure()

dftmdb = extract\_tmdb\_api()

dfarr=[ dfama, dfdis, dfnet, dftmdb]

for i in dfarr:

print("###############")

print(i.info() )

print(i.head() )

#Ejecuta consultas

#Insertar aqui el llamado a las funciones de consulta.

Y listo, ya está finalizada la extracción!

Felicitaciones!!!