**Assignment 4 – MongoDB & PyMongo**

**# a)** Create collections “flights” inside database “airline\_delayDB”

from  pymongo import MongoClient

import pandas as pd

import json

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

def read\_flight(csv\_path):

    flight\_data=pd.read\_csv(csv\_path)

    fl\_data=json.loads(flight\_data.to\_json(orient='records'))

    collection.delete\_many({})

    collection.insert\_many(fl\_data)

if \_\_name\_\_=="\_\_main\_\_":

    print("welcome to pymongo")

    client=MongoClient("mongodb://localhost:27017")

    print(client)

    db=client['airline\_delayDB']

    collection=db['flights']

    read\_flight('D:\python trianings\Assignment 4 on Pymongo\_Flight\_Delay\Flights\_Delay.csv')

MongoClient(host=['localhost:27017'], document\_class=dict, tz\_aware=False, connect=True)

**# b)** Average arrival delay caused by airlines

average\_delay=collection.aggregate([{'$group':{'\_id':{},'avg\_Delay':

                            {'$avg':'$ARRIVAL\_DELAY'}}},{'$project':{'\_id':0}}])

for item in average\_delay:

    print(item)

{'avg\_Delay': 7.545457931394093}

**#c)** Days of months with respect to average of arrival delays.

    #   [Create a suitable plot using matplotlib/seaborn]

avg\_arvl\_day=collection.aggregate([{'$group': {'\_id':'$DAY','Avg\_day\_Delay':

                                    {'$avg':'$ARRIVAL\_DELAY'}}},{'$sort':{'Avg\_day\_Delay':-1}}])

avg\_arvl\_day\_df=pd.DataFrame(avg\_arvl\_day)

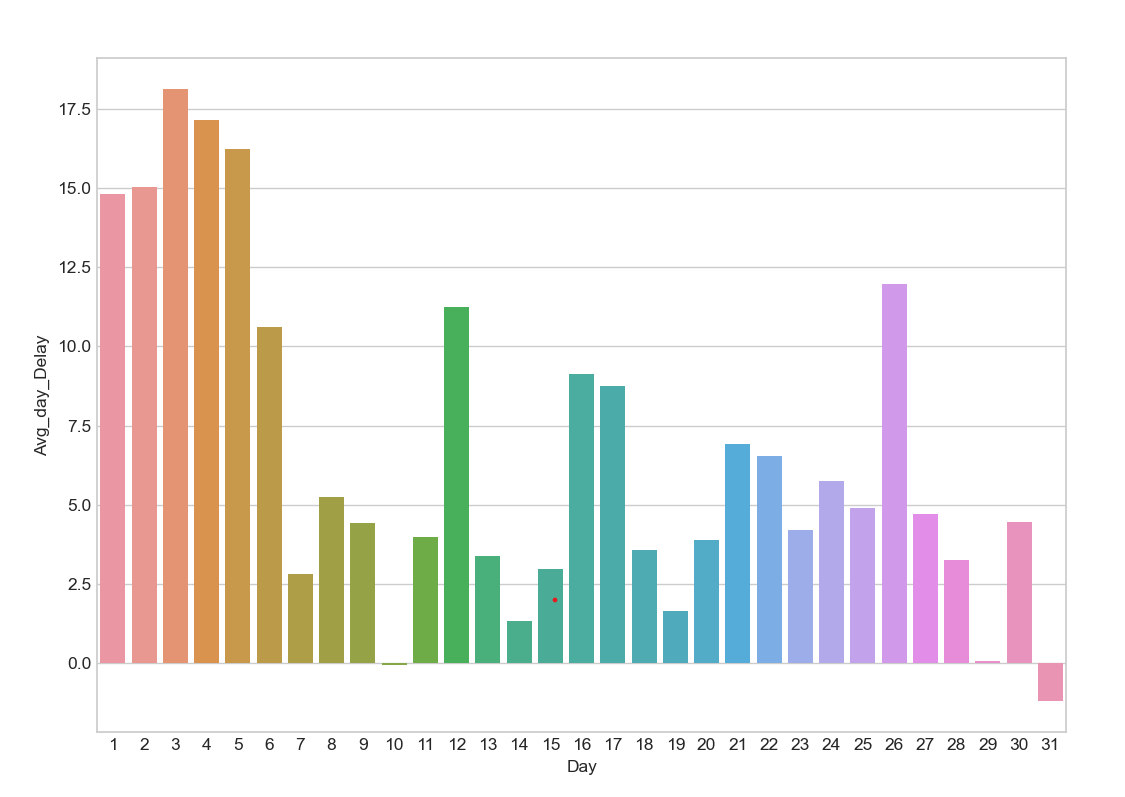
plt.style.use('seaborn-whitegrid')

plt.figure(figsize=(10,7))

sns.barplot(x=avg\_arvl\_day\_df['\_id'],y=avg\_arvl\_day\_df['Avg\_day\_Delay'])

plt.xlabel('Day')

plt.show()



**#d**  Arrange weekdays with respect to the average arrival days caused.

    #  [Create a suitable plot using matplotlib/seaborn]

avg\_arrival\_wkday=collection.aggregate([{'$group': {'\_id':'$DAY\_OF\_WEEK', 'Avg\_Arrival\_Delay':

                                          {'$avg': '$ARRIVAL\_DELAY'}}},{'$sort': {'Avg\_Arrival\_Delay': -1}}])

avg\_arrival\_wkday\_df=pd.DataFrame(avg\_arrival\_wkday)

plt.style.use('seaborn-whitegrid')

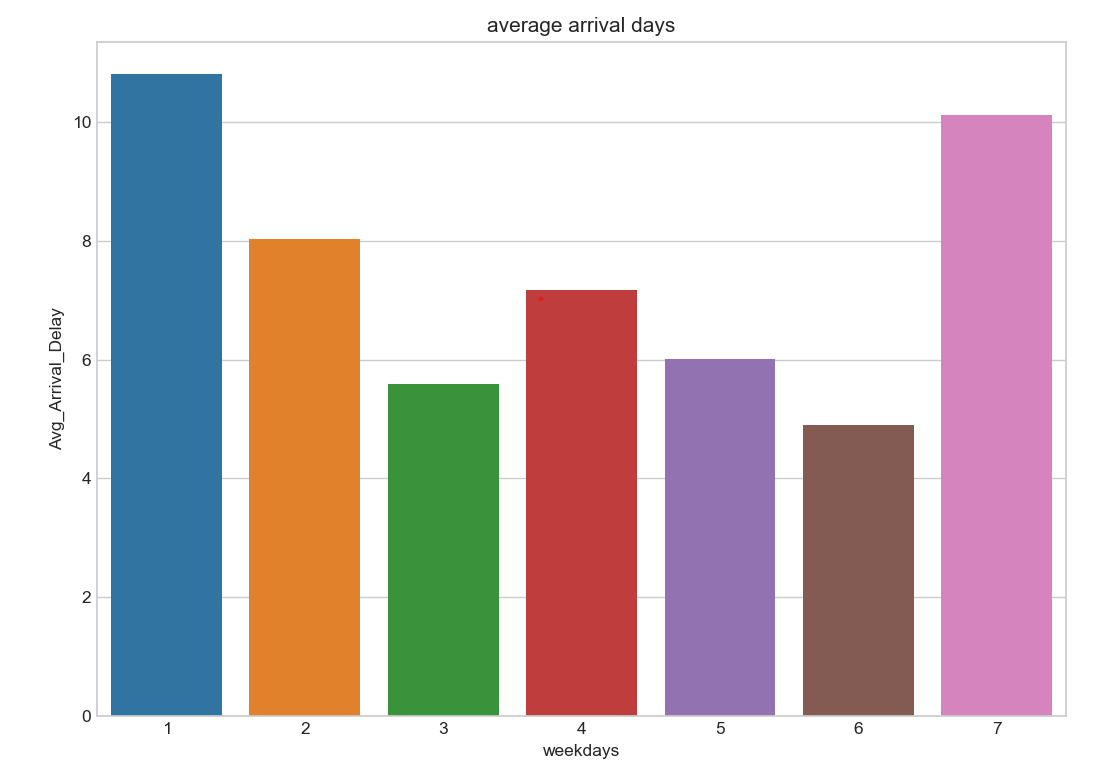
plt.figure(figsize = (10,7))

sns.barplot(x=avg\_arrival\_wkday\_df['\_id'],y=avg\_arrival\_wkday\_df['Avg\_Arrival\_Delay'])

plt.title('average arrival days')

plt.xlabel('weekdays')

plt.show()



**#e)** Arrange Days of month as per cancellations done in descending order.

    #   [Create a suitable plot using matplotlib/seaborn]

day\_per\_cancel = collection.aggregate([{'$match' : {'CANCELLED':1}},

                                    {'$group':{'\_id':'$DAY',

                                    'Cancellation\_Count':{'$count':{}}}},

                                     {'$sort':{'Cancellation\_Count': -1}}])

day\_per\_cancel\_df=pd.DataFrame(day\_per\_cancel)

plt.style.use('seaborn-whitegrid')

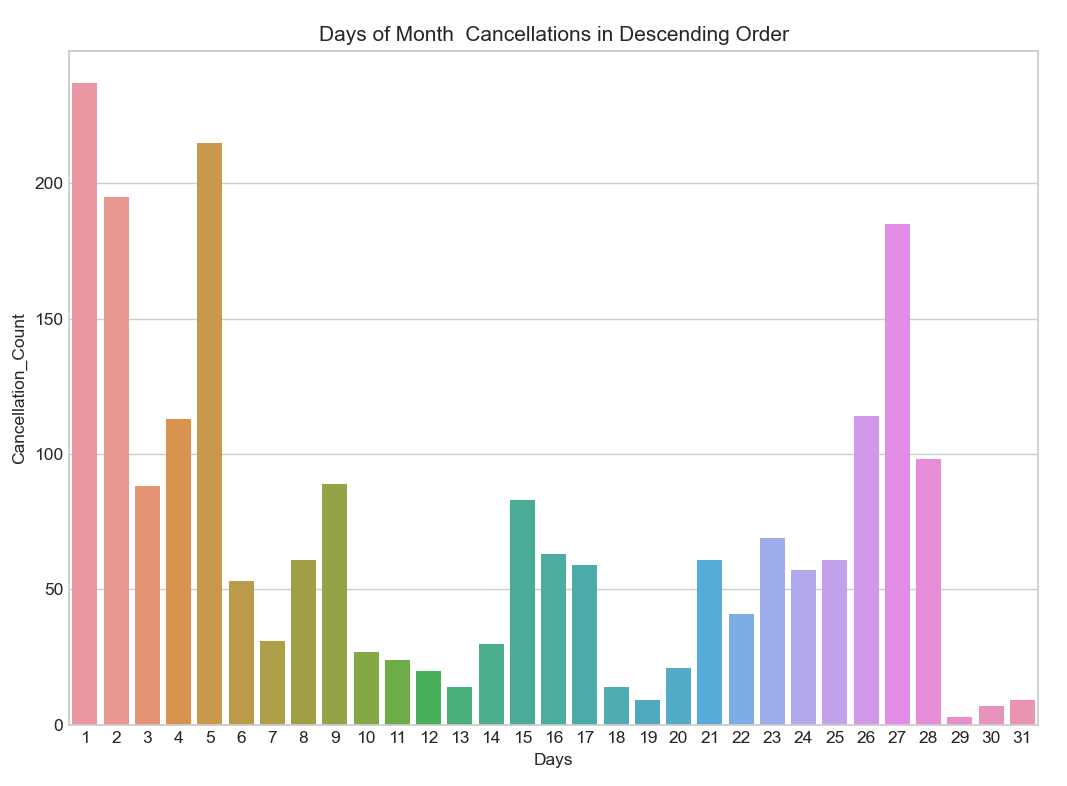
plt.figure(figsize = (10,7))

sns.barplot(x=day\_per\_cancel\_df['\_id'],y=day\_per\_cancel\_df['Cancellation\_Count'])

plt.title('Days of Month  Cancellations in Descending Order')

plt.xlabel('Days')

plt.show()



**#f)** Find the busiest airports with respect to day of week. Create a suitable plot using matplotlib/seaborn.

arrival\_airport\_cnt =collection.aggregate([{'$group':{'\_id':'$DAY\_OF\_WEEK'

                                              ,'count':{'$count':{}}}},

                                            {'$sort':{'DESTINATION\_AIRPORT': -1}}])

dep\_airport\_cnt = collection.aggregate([{'$group':{'\_id':'$DAY\_OF\_WEEK'

                                              ,'count':{'$count':{}}}},

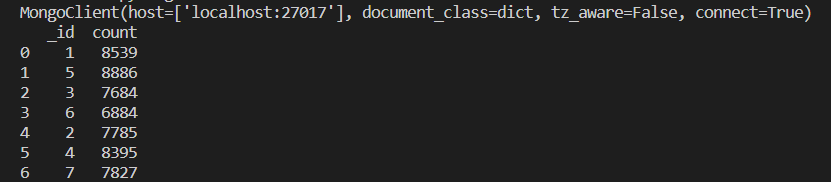
                                             {'$sort':{'ORIGIN\_AIRPORT': -1}}])

arrival\_airport\_cnt\_df=pd.DataFrame(arrival\_airport\_cnt)

dep\_airport\_cnt\_df =pd.DataFrame(dep\_airport\_cnt)

dict1= arrival\_airport\_cnt\_df.to\_dict()

print(arrival\_airport\_cnt\_df)



**#g** Find top 10 Airlines of US. Create a suitable plot using matplotlib/seaborn.

top\_US\_airlines=collection.aggregate([{'$group':{'\_id':'$AIRLINE'

                                          ,'Airline\_Count':{'$count':{}}}},

                            {'$sort':{'Airline\_Count': -1}}, {'$limit':10}])

top\_US\_airlines = collection.aggregate([{ '$match':{'AIRLINE':"US"}},

                                        {'$group' :{'\_id' : '$FLIGHT\_NUMBER', 'Airline\_Count':{'$count' : {}}}},

                                            {'$sort':{'Airline\_Count':-1}},

                                            {'$limit':10}])

airline\_df=pd.DataFrame(top\_US\_airlines)

plt.style.use('seaborn-whitegrid')

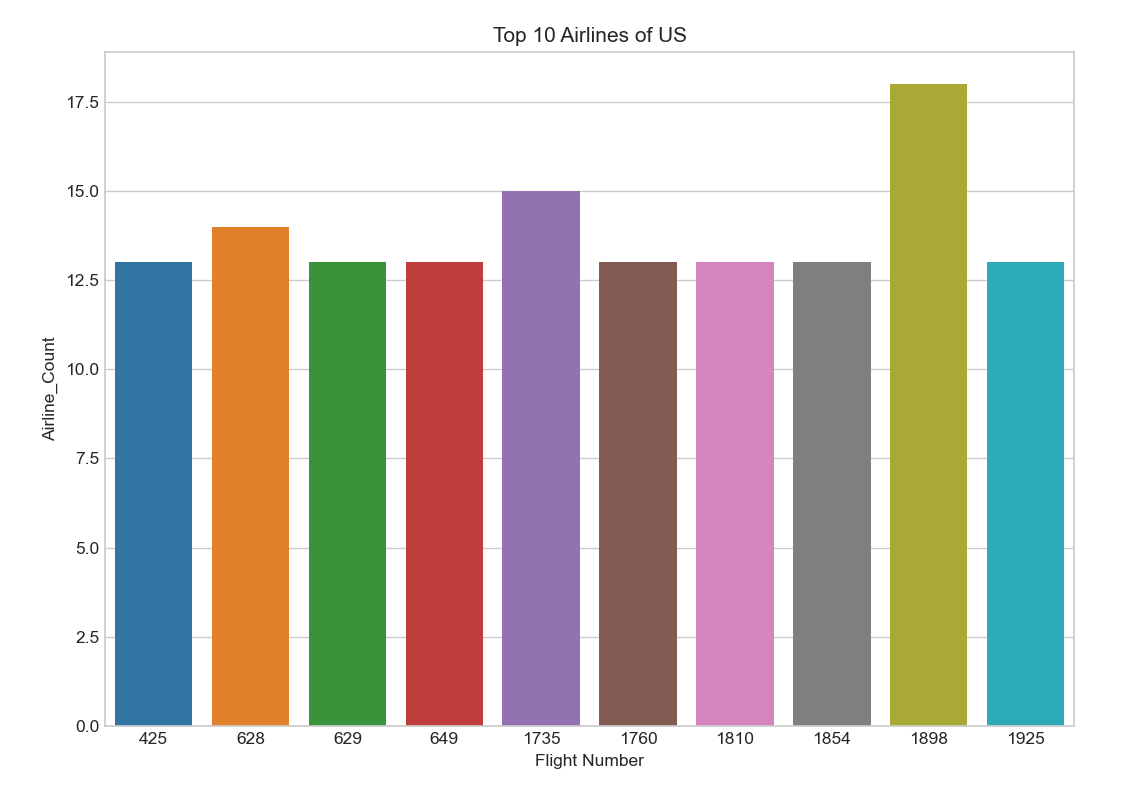
plt.figure(figsize = (10,7))

sns.barplot(x=airline\_df['\_id'],y=airline\_df['Airline\_Count'])

plt.title('Top 10 Airlines of US')

plt.xlabel('Flight Number')

plt.show()



**#h)** Finding airlines that make the maximum, minimum number of cancellations.

min\_cancellation = collection.aggregate([{'$match' : {'CANCELLED':1}},

                                   {'$group':{'\_id':'$AIRLINE',

                                    'min\_cancellation':{'$count':{}}}},

                                   {'$sort':{'min\_cancellation': 1}}, {'$limit':1}

                                         ])

for i in min\_cancellation:

        print(i)

max\_cancellation = collection.aggregate([{'$match' : {'CANCELLED':1}},

                                   {'$group':{'\_id':'$AIRLINE',

                                  'max\_cancellation':{'$count':{}}}},

                                  {'$sort':{'max\_cancellation': -1}}, {'$limit':1}

                                        ])

for i in max\_cancellation:

        print(i)

{'\_id': 'HA', 'min\_cancellation': 3} {'\_id': 'MQ', 'max\_cancellation': 414}

**# i)** Find and show airlines names in descending that make the most number of diversions made.

    #    [Create a suitable plot using matplotlib/seaborn]

airline\_diversion = collection.aggregate([{'$match' : {'DIVERTED':1}},

                                    {'$group':{'\_id':'$AIRLINE',

                                 'No\_of\_Diversions':{'$count':{}}}},

                                {'$sort':{'No\_of\_Diversions': -1}}])

airline\_diversion\_df=pd.DataFrame(airline\_diversion)

plt.style.use('seaborn-whitegrid')

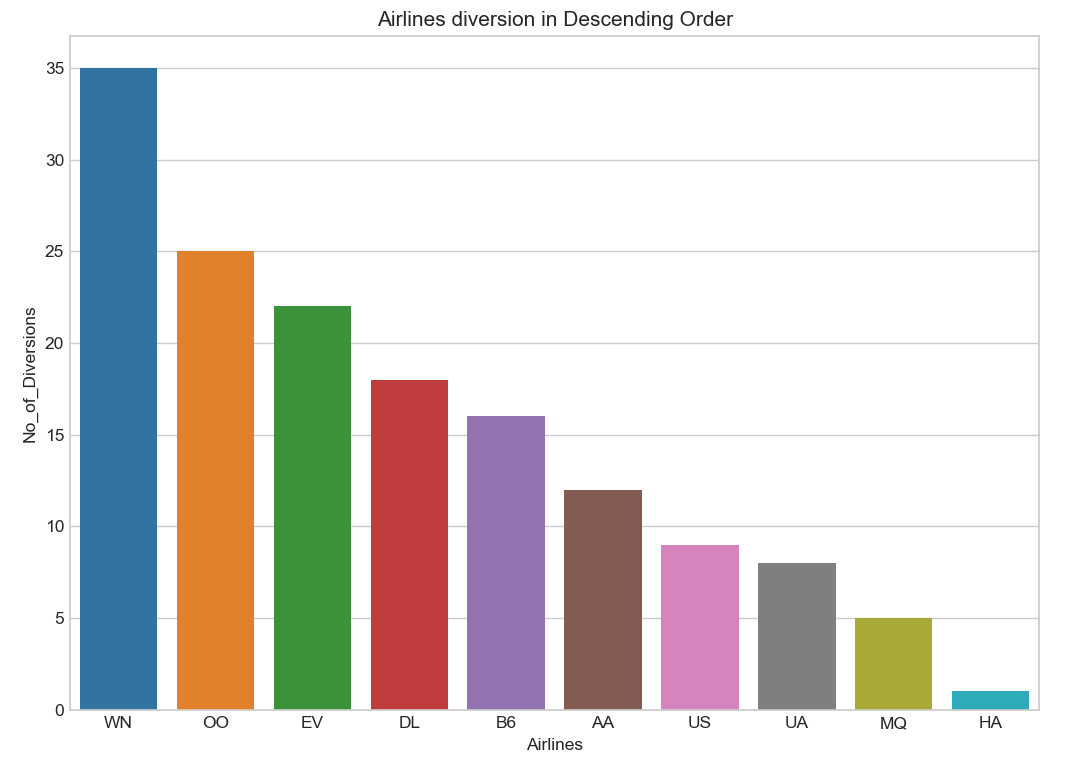
plt.figure(figsize = (10,7))

sns.barplot(x=airline\_diversion\_df['\_id'],y=airline\_diversion\_df['No\_of\_Diversions'])

plt.title('Airlines diversion in Descending Order')

plt.xlabel('Airlines')

plt.show()



**#j)** Finding days of month that see the most number of diversion

no\_of\_diversions = collection.aggregate([{'$match' : {'DIVERTED':1}},

                                     {'$group':{'\_id':'$DAY',

                                     'No\_of\_Diversions':{'$count':{}}}},

                                     {'$sort':{'No\_of\_Diversions': -1}},{'$limit':1}

                                           ])

for i in no\_of\_diversions:

         print(i)

{'\_id': 2, 'No\_of\_Diversions': 15}

**#k**) Calculating mean and standard deviation of departure delay for all flights in minutes

departure\_delay = collection.find({},{'DEPARTURE\_DELAY':1,'\_id':0})

departure\_delay\_df=pd.DataFrame(departure\_delay)

print("Mean:",departure\_delay\_df.mean())

print("Standard Deviation:",departure\_delay\_df.std())

Mean: DEPARTURE\_DELAY 11.329091 dtype: float64

Standard Deviation: DEPARTURE\_DELAY 39.620916 dtype: float64

**#l)** Calculating mean and standard deviation of arrival delay for all flights in minutes

arrival\_delay = collection.find({},{'ARRIVAL\_DELAY':1,'\_id':0})

arrival\_delay\_df=pd.DataFrame(arrival\_delay)

print("Mean:",arrival\_delay\_df.mean())

print("Standard Deviation:",arrival\_delay\_df.std())

Mean: ARRIVAL\_DELAY 7.545458 dtype: float64

Standard Deviation: ARRIVAL\_DELAY 42.378386 dtype: float64

**#m** Create a partitioning table “flights\_partition” using partitioned by schema “CANCELLED”

**#n)** Finding all diverted Route from a source to destination Airport & which route is the most diverted route.

**#o**) When is the best time of day/day of week/time of year to fly with minimum delay

total\_delay=collection.aggregate([{'$group':{'\_id':'$\_id','total':{'$sum':'$ARRIVAL\_DELAY'}}},{'$sort': {'total':1}},

                                  {'$limit':1}])

for i in total\_delay:

 print(i)

{'\_id': ObjectId('63649e59979ab9ec901003d4'), 'total': -76.0}