**Spark Dataframe Exercise**

1. The airlines data is has airline no, origin, destination, air time etc. Find out the following from given data

Exercise- from the flights dataset

* Select the airlines with their origin and destination

**df.select('airlines','origin','destination').show()**

* Select flights with origin 'IND'

**df.select('airlines','flight\_number','origin').filter(df.origin =='IND').show()**

**df.filter(df.origin =='IND').show()**

* Select flights with destination as 'DFW.

**df.filter(df.destination =='DFW').show()**

**df.select('airlines', 'destination').filter(df.destination =='IND').show()**

* Select flights with origin as 'SEA' and 'DFW'

**df.select('airlines','flight\_number','origin',).filter((df.origin =='SEA') | (df.origin == 'DFW')).show()**

* List the flights having air\_time more than 120 mins.

**df.filter(df.air\_time > 120).show()**

* Add a new columns to store the duration in hrs

**df.select('airlines','air\_time').withColumn('duration\_hrs',df.air\_time/60).show()**

* List flights covering distance more than 1500

**df.select('airlines','origin','destination','distance').filter(df.distance> 1500).show()**

* Find the flight with min airtime

**df.select('airlines','air\_time').agg({'air\_time':'min'}).show()**

**OR**

**df.groupBy('airlines').agg({'air\_time':'min'}).show()**

* Find the flight with max airtime

**df.groupBy('airlines').agg({'air\_time':'max'}).show()**

* Find the flight with min duration\_hrs

**df.select('airlines','air\_time').withColumn('duration\_hrs',df.air\_time/60).agg({'duration\_hrs':'min'}).show()**

* Find the flight with max duration\_hrs

**df.select('airlines','air\_time').withColumn('duration\_hrs',df.air\_time/60).agg({'duration\_hrs':'max'}).show()**

**OR**

**df.select('airlines','air\_time').withColumn('duration\_hrs',df.air\_time/60).groupby().max('duration\_hrs').show()**

* Find the flight with max distance covered

**df.agg({'distance':'max'}).show()**

**OR**

**df.groupby('flight\_number').agg({'distance':'max'}).show()**

* Find the flight with min distance covered

**df.agg({'distance':'min'}).show()**

* Find the average duration of the flights.

**df.groupby(['airlines']).agg({'air\_time':'avg'}).show()**

* Find the shortest flight from 'PDX' with respect to distance.

**df1=spark.read.csv('flights.csv', inferSchema=True,header=True)**

**df1.filter(df1.origin=='PDX').groupby('origin').min('distance').show()**

* Find the above for the flight 'OGG', 'BOS', 'JFK'

**df1.filter((df1.origin=='OGG')|(df1.origin=='BOS')|(df1.origin=='JFK')).groupby ('origin').min('distance').show()**

* Find the longest flight from 'SEA' with respect to duration

**df1.filter(df1.origin=='SEA').groupby('origin').max('air\_time').show()**

* find the above for the flight 'ORD','MIA', 'HNL'

**df1.filter((df1.origin=='ORD')|(df1.origin=='MIA')|(df1.origin=='HNL')).groupby ('origin').max('air\_time').show()**

* Find the average departure delay for the airline no 20366, 20398, 19805

**df1.filter((df1.airlines==20366)|(df1.airlines==20398)|(df1.airlines==19805)). groupby('airlines').avg('departure\_delay').show()**

* similarly, find the average arrival delay for all the airline nos.

**df1.groupby('airlines').avg('arrival\_delay').show()**

**Input File Name:** airlines.csv

1. Experimental sensor data used for binary classification (room occupancy) from Temperature, Humidity, Light and CO2, Ground-truth occupancy was obtained from time stamped pictures that were taken every minute. Find out the following from the given data.

* Find out the average humidity of the room.

**df.groupby('humidity').avg().show()**

* Filter all the timestamps where the temperature is greater than 20 degrees.

**df.filter(df.Humidity>20).show()**

* Find out the missing values of temperature, humidity, light and fill the values with mean of the corresponding column.
* What is the maximum and minimum temperature of the room with more than 3 occupants?

**df.filter(df.Occupants>3).agg({'Humidity':'max'}).show()**

**df.filter(df.Occupants>3).agg({'Humidity':'min'}).show()**

* Find out the number of instances where the room has more than 4 occupants.

**df.filter(df.Occupants >4).show()**

**Input File Name:** occupance\_sensor\_data

3. The data set of village amenities of Bangalore district is given. It has the data of 588 villages. Find out the following from the given data set.

* Find out the list of villages which are having a town within 10Km.

**df.select('Village Code','Village Name','Total Population of Village','Nearest Town Name', 'Nearest\_Town\_Distance\_Village').filter(df.Nearest\_Town\_Distance\_Village <= 10).show()**

* Find out the total number of villages which are having primary health centres (PHCs).

**df.filter(df.Primary\_Health\_Centre\_Numbers==1).count()**

* Find out the number of villages which cultivate paddy.

**df.filter(df.Agricultural\_Commodities\_First=='PADDY').count()**

**df.filter(df.Agricultural\_Commodities\_Second=='PADDY').count()**

**df.filter(df.Agricultural\_Commodities\_Third=='PADDY').count()**

**df.filter((df.Agricultural\_Commodities\_First=='PADDY')|(df.Agricultural\_Commodities\_Second=='PADDY') | (df.Agricultural\_Commodities\_Third=='PADDY')).count()**

**Input File Name:** DCHB\_Village\_Amenities\_Bangalore.csv