**Q.1 By Using Food Delivery App orders & order\_items table answer Below Question.**

Write sql query to get the date when highest order placed on app.

SELECT TOP 1 concat(CONCAT(YEAR(ordered\_at),'-',month(ordered\_at)),'-',day(ordered\_at))

as year\_month\_day,

count(\*) as count

FROM orders

GROUP BY concat(CONCAT(YEAR(ordered\_at),'-',month(ordered\_at)),'-',day(ordered\_at))

ORDER BY count DESC

**Q.2 By Using Food Delivery App orders & order\_items table answer Below Question.**

Write sql query to get the highest ordered food\_item placed on app.

SELECT TOP 1 count(name) number\_order, name FROM order\_items

GROUP BY name

ORDER BY number\_order desc

**Q.3 By Using Game Database App gamplays & purchases table answer Below Question.**

Write sql query to get the most used platform for gaming.

SELECT top 1 count(\*) no\_of\_plays , platform FROM gameplays

GROUP BY platform

ORDER BY no\_of\_plays desc

**Pandas and Numpy Questions**

**Q.4 By Using Mental Health Data write a program to answer below questions.**

How does the frequency of mental health illness vary by geographic location?

#top country with mental illness

df[(df['mental\_health\_consequence']!='No')].groupby(['country']).count()[['count']].sort\_values('count', ascending=False).head(1)

#top state with mental illness

df[df['mental\_health\_consequence']!='No'].dropna(subset=['state'],axis=0).groupby(['state']).count()[['count']].sort\_values('count', ascending=False).head(1)

#top 3 countries with mental illness df[(df['mental\_health\_consequence']!='No')].groupby(['country']).count()[['count']].sort\_values('count', ascending=False).head(3

# frequency distribution and percentage of mental illiness in top 3 contry

frq=df[(df['country'].isin(['United States','United Kingdom','Canada']))].groupby(['country','work\_interfere'])[['count']].count()

frq['Percentage']=frq.groupby(level=0).apply(lambda x:100 \* x / float(x.sum()))['count']

print(frq)

#Gender clean function for next question

male\_list=['M', 'Mail', 'Make', 'Mal', 'Male', 'Male ', 'Malr', 'Man', 'm', 'maile', 'male', 'male leaning androgynous', 'msle', 'cis male', 'Male (CIS)','Cis Male', 'Cis Man']

female\_list=['f', 'femail', 'female', 'F', 'Femake', 'Female', 'Female ', 'Female (cis)','cis-female/femme','Cis Female' 'woman','Woman']

def gender\_create(gender):

if gender in male\_list:

return 'male'

elif gender in female\_list:

return 'female'

else:

return 'trans'

df['gender\_clean']=df['gender'].apply(gender\_create)

**Q.5 By Using Mental Health Data write a program to answer below questions.**

How many people did go for treatment on the basis of gender for the top 3 countries?

#getting the top 3 coutries with illness

df[(df['mental\_health\_consequence']!='No')].groupby(['country']).count()[['count']].sort\_values('count', ascending=False).head(3)

#we got top 3 countries from above, we have passed those countries in below code to filter the data

#gender clean function ,mentioned above

df[(df['country'].isin(['United States','United Kingdom','Canada'])) & (df['treatment']=='Yes')].groupby(['country','gender\_clean'])[['count']].count()

**Q.6 By Using Mental Health Data write a program to answer below questions.**

Draw Seaborn swarm plot between (work\_interface & age\_clean)

#deriving age\_clean

def clean\_age(age):

if age>=0 and age <=100:

return age

else:

return np.nan

#calling function of column

df['age\_clean']=df['age'].apply(clean\_age)

Fig,ax =plt.subplots(figsize=(15,9)) #setting the size of figure

sns.set\_theme(style="whitegrid")

sns.swarmplot(data=df,y='age\_clean',x='work\_interfere')

**Pyspark Solution**

**Q.7 By Using airlines, flights and airports data file answer below Question.**

Parse the rows in the csv files.

# Importing libraries:

import pyspark

from pyspark import SparkContext

sc = SparkContext()

# Loading the data into RDD:

airlinesPath = "airlines.csv"

airlines = sc.textFile(airlinesPath)

# Split rows into list:

import csv

from io import StringIO

def split(line):

reader = csv.reader(StringIO(line))

return next(reader)

AIRLINE\_Parsed = airlines.filter(lambda x:'Code' not in x).map(split)

# Parsing rows in the csv:

from collections import namedtuple

fields = ('code', 'description')

Airline = namedtuple('Airline', fields)

def parse(row):

return Airline(\*row[:2])

AIRLINE\_FINAL= AIRLINE\_Parsed.map(parse)

AIRLINE\_FINAL.take(10)

**for flight data: data loading can be done as done as above**

# Removing NULL values:

Blanks = flight.map(lambda x: ','.join(x or '00.00' for x in x.split(',')))

flight\_clean = blanks.map(lambda x:x.replace(',""',',"0000"'))

# Split rows into list:

import csv

from io import StringIO

def split(line):

reader = csv.reader(StringIO(line))

return next(reader)

flightParsed = flight\_clean.filter(lambda x:’FL\_DATE' not in x).map(split)

# Parsing rows in the csv:

from datetime import datetime

from collections import namedtuple

fields = ('date','airline','flightnum','origin','dest','dep',

'dep\_delay','arv','arv\_delay','airtime','distance')

Flight = namedtuple('Flight', fields, verbose=False)

DATE\_FMT = '%Y-%m-%d'

TIME\_FMT = '%H%M%S'

def parse(row):

row[0] = datetime.strptime(row[0],DATE\_FMT).date()

row[5] = datetime.strptime(row[5],TIME\_FMT).time()

row[6] = float(row[6])

row[7] = datetime.strptime(row[7],TIME\_FMT).time()

row[8] = float(row[8])

row[9] = float(row[9])

row[10] = float(row[10])

return Flight(\*row[:11])

flight\_final= flightParsed.map(parse)

flight\_final.take(10)

**Q.8 By Using airlines, flights and airports data file answer below Question.**

Compute the % of flights which had delays

FINAL\_FLIGHT.filter(lambda x:x.dep\_delay>0).count()/float(flight\_final.count())

Or (as it is not clear arrival delay or dept delay is to be computed)

flight\_final.filter(lambda x:x[8] < 0).count()/float(flight\_final.count())

**Q.9 By Using airlines, flights and airports data file answer below Question.**

Compute the average delay

sumCount = flight\_final.map(lambda x:x.dep\_delay).aggregate((0,0), (lambda acc,value: (acc[0]+value,acc[1]+1)), (lambda acc1,acc2: (acc1[0]+acc2[0],acc1[1]+acc2[1])))

**Q.10 By Using airlines, flights and airports data file answer below Question.**

Compute a frequency distribution of delays

#using map function we have mapped the delay values and using count by value to frequency distribution

flight\_final.map(lambda x:int(x.dep\_delay/60)).countByValue()