Answer. 1 Python

from collections import Counter

data\_set = 'write write write all the number from from from 1 to 100'

split\_it = data\_set.split()

Counters\_found = Counter(split\_it)

most\_occur = Counters\_found.most\_common(1)

print(most\_occur)

len(most\_occur[0][0])

Explanation - From the given string we can note that the most frequent words are “write” and “from” and the maximum value of both the values is “write” and its corresponding length is 5

from collections import Counter

data\_set = 'Python is good Python dynamic'

split\_it = data\_set.split().

Counters\_found = Counter(split\_it)

most\_occur = Counters\_found.most\_common(1)

print(most\_occur)

len(most\_occur[0][0])

Explanation - From the given string we can note that the most frequent words is “Python” and

corresponding length is 6

Answer. 2 Python ->

def isValid(s):

d = Counter(s)

counts = Counter(d.values())

if len(counts) == 1:

return "YES"

elif len(counts) > 2:

return "NO"

else:

max\_v = max(counts.values())

k1, k2 = counts.keys()

if (max\_v == len(d) - 1):

if (abs(k1 - k2) == 1):

return "YES"

elif (min(k1, k2) == 1):

if counts[1] == 1:

return "YES"

else:

return "NO"

else:

return "NO"

else:

return "NO"

isValid('nnrrpp')

s = (“nnrrpp”). This is a valid string because frequencies are { “n”: 2, “r”: 2, “p”: 2 }-YES

s=('abcddc'). This string is not valid as we can remove only 1 occurrence of “c” & ‘d’. That leaves character frequencies of { “a”: 1, “b”: 1 , “c”: 2,’d’:2 }-NO

Answer. 3 Python ->

import wget

URL = "https://raw.githubusercontent.com/Biuni/PokemonGO-Pokedex/master/pokedex.json"

response = wget.download(URL,'data.json')

df = pd.read\_json("data.json")

df.to\_csv(‘data\_csv’,,index=False)

Answer. 4 Python ->

import wget

URL = "https://data.nasa.gov/resource/y77d-th95.json"

response = wget.download(URL,'data.json')

df = pd.read\_json("data.json")

df.to\_csv('data\_csv',index=False)

Answer. 5 Python ->

import requests

import json

response\_API = requests.get('http://api.tvmaze.com/singlesearch/shows?q=westworld&embed=episodes')

data = response\_API.text

parse\_json = json.loads(data)

Answer. 6 Python ->

df[df['spawn\_chance']<5]

df[df['weakness']<4]

df[df['multiplier']==NaN]

df[df['evolution']>2]

df[df['spawn time'].dt.second<300]

df[df['capabilities']>2]

Answer. 7 Python ->

df['year']=pd.to\_datetime(df.year,unit='ns',errors='coerce')

**Get all the Earth meteorites that fell before the year 2000**

**Plot:**

import seaborn as sns

fig = plt.figure(figsize =(50, 50))

sns.scatterplot(df,x=df['year'],y=df['name'])

**analysis**

df[df['year'].dt.year<2000]['name']

**● Get all the earth meteorites co-ordinates who fell before the year 1970**

**analysis**

new\_df=df[df['year'].dt.year<1970]['geolocation']

**Plot:**

import seaborn as sns

fig = plt.figure(figsize =(50, 50))

sns.scatterplot(new\_df,x=df['year'],y=df['name'])

● **Assuming that the mass of the earth meteorites was in kg, get all those whose mass was more than 10000kg**

**analysis**

df[df['mass']>10000]

**Plot:**

import seaborn as sns

fig = plt.figure(figsize =(50, 50))

sns.scatterplot(df,x=df['mass'],y=df['name'])

Answer. 8 Python ->

import pandas as pd

import requests

response = requests.get('http://api.tvmaze.com/singlesearch/shows?q=westworld&embed=episodes')

data = response.json()

df = pd.json\_normalize(data['\_embedded']['episodes'])

df=pd.DataFrame(df)

df.head()

**● Get all the overall ratings for each season and using plots compare the ratings for all the seasons, like season 1 ratings, season 2, and so on.**

**analysis**

df.groupby('season')['rating.average'].sum()

**Plot**

sns.barplot(data=df, x="season", y="rating.average")

**● Get all the episode names, whose average rating is more than 8 for every season**

**analysis**

df[df..groupby('season')& (df['rating.average']>8)]['name']

Or

df[(df['rating.average']>8)][['name','season']]

**analysis**

sns.barplot(data=df, x="name", y="rating.average")

● **Get all the episode names that aired before May 2019**

df['airdate']=pd.to\_datetime(df['airdate'])

df[(df['airdate'].dt.year)<2019]['name']

Answer. 9 Python ->

**● Get all the cars and their types that do not qualify for clean alternative fuel vehicle**

import pandas as pd

df=pd.read\_csv('C:\\Users\\LOKESHRAJ P\\Downloads\\Electric\_Vehicle\_Population\_Data.csv')

df[df['Clean Alternative Fuel Vehicle (CAFV) Eligibility']!='Clean Alternative Fuel Vehicle Eligible'][['Make','Model']]

**● Get all TESLA cars with the model year, and model type made in Bothell City.**

df[(df['Make']=='TESLA')&(df['City']=='Bothell')][['Model Year','Model']]

**● Get all the cars that have an electric range of more than 100, and were made after**

**2015**

df[(df['Electric Range']>100) & (df['Model Year']>2015)]

**● Draw plots to show the distribution between city and electric vehicle type**

import seaborn as sns

sns.set(rc={'figure.figsize':(50,50)})

sns.countplot(data=df, x="City", hue="Electric Vehicle Type")

sns.displot(data=df, x="City", hue="Electric Vehicle Type")

Answer. 10 Python ->

import nltk

from nltk.tokenize import word\_tokenize

from nltk.tag import pos\_tag

def count\_pos(phrase):

tokens = word\_tokenize(phrase)

tags = pos\_tag(tokens)

counts = {'verb': 0, 'noun': 0, 'pronoun': 0, 'adjective': 0}

for word, tag in tags:

if tag.startswith('VB'):

counts['verb'] += 1

elif tag.startswith('NN'):

counts['noun'] += 1

elif tag.startswith('PRP'):

counts['pronoun'] += 1

elif tag.startswith('JJ'):

counts['adjective'] += 1

return counts

Test case-1:

phrase = "The quick brown fox jumps over the lazy dog."

counts = count\_pos(phrase)

print('DIC=',counts)

DIC= {'verb': 1, 'noun': 3, 'pronoun': 0, 'adjective': 2}

Explanation: verb-jumps

Noun- fox,dog,brown

Pronoun- Not there

Adjective- quick,lazy

Test case-2:

phrase = "Python is easy for programming"

counts = count\_pos(phrase)

print('DIC=',counts)

DIC= {'verb': 2, 'noun': 1, 'pronoun': 0, 'adjective': 1}

Explanation: verb-is,programming

Noun- Pyhton

Pronoun- Not there

Adjective- easy