Q1.

"""

- read_csv()
- head()
- describe()
- info()
- tail()

"""

In [1]:
```
import pandas as pd
```

In [3]:
```
df = pd.read_csv('services.csv')
```

In [4]:
```
df.head(2)
```

Out[4]:

	id	location_id	program_id	accepted_payments	alternate_name	application_process	audi
0	1	1	NaN	NaN	NaN	Walk in or apply by phone.	(adult 55 or e minori
1	2	2	NaN	NaN	NaN	Apply by phone for an appointment.	Resi o M Co age

2 rows × 22 columns

In [5]:
```
df.describe()
```

Out[5]:

	id	location_id	program_id
count	23.00000	23.000000	0.0
mean	12.00000	11.956522	NaN
std	6.78233	6.711444	NaN
min	1.00000	1.000000	NaN
25%	6.50000	6.500000	NaN
50%	12.00000	12.000000	NaN
75%	17.50000	17.500000	NaN
max	23.00000	22.000000	NaN

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23 entries, 0 to 22
Data columns (total 22 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       23 non-null     int64  
 1   location_id              23 non-null     int64  
 2   program_id               0 non-null      float64
 3   accepted_payments        1 non-null      object 
 4   alternate_name           1 non-null      object 
 5   application_process      23 non-null     object 
 6   audience                 14 non-null     object 
 7   description              23 non-null     object 
 8   eligibility              21 non-null     object 
 9   email                    1 non-null      object 
 10  fees                     21 non-null     object 
 11  funding_sources          21 non-null     object 
 12  interpretation_services  1 non-null      object 
 13  keywords                 21 non-null     object 
 14  languages                1 non-null      object 
 15  name                     23 non-null     object 
 16  required_documents       1 non-null      object 
 17  service_areas            21 non-null     object 
 18  status                   23 non-null     object 
 19  wait_time                19 non-null     object 
 20  website                  2 non-null      object 
 21  taxonomy_ids             1 non-null      object 
dtypes: float64(1), int64(2), object(19)
memory usage: 4.1+ KB
```

In [8]:
```
df.tail(2)
```

Out[8]:

	id	location_id	program_id	accepted_payments	alternate_name	application_process	auc
21	22	22	NaN	Cash, Check, Credit Card	Fotos para pasaportes	Walk in or apply by phone or mail	Pro nor busir the
22	23	22	NaN	NaN	NaN	Walk in or apply by phone or mail	S s nor busir

2 rows × 22 columns

Q2.

In [9]:
```
df.columns
```

Out[9]:
```
Index(['id', 'location_id', 'program_id', 'accepted_payments',
       'alternate_name', 'application_process', 'audience', 'description',
       'eligibility', 'email', 'fees', 'funding_sources',
       'interpretation_services', 'keywords', 'languages', 'name',
       'required_documents', 'service_areas', 'status', 'wait_time', 'website',
       'taxonomy_ids'],
      dtype='object')
```

In [9]:
```
df1 = df[['id', 'location_id', 'program_id']].head(3)
```

In [10]:
```
df1
```

Out[10]:

	id	location_id	program_id
0	1	1	NaN
1	2	2	NaN
2	3	3	NaN

In [11]:
```
def reindex_df(df1):
    df1_reindexed = df1.reset_index(drop=True)
    df1_reindexed.index = df1_reindexed.index * 2 + 1

    return df1_reindexed

df1_reindexed = reindex_df(df1)

print(df1_reindexed)
```

```
   id  location_id  program_id
1   1            1         NaN
3   2            2         NaN
5   3            3         NaN
```

Q3.

In [13]:
```
df1['Values'] = [10,20,30]
df1
```

Out[13]:

	id	location_id	program_id	Values
0	1	1	NaN	10
1	2	2	NaN	20
2	3	3	NaN	30

In [14]:
```
def sum_of_three(df):
    v_columns = df['Values']
    cal_sum = v_columns.head(3).sum()
    print(cal_sum)

sum_of_three(df1)
```

```
60
```

Q4.

In [18]:
```
df1['Text'] = ['Course name ' , 'Data science', 'pw_skills ']
```

In [19]:
```
df1['Word_Count'] = df1['Text'].str.split().apply(len)
df1
```

Out[19]:

	id	location_id	program_id	Values	Text	Word_Count
0	1	1	NaN	10	Course name	2
1	2	2	NaN	20	Data science	2
2	3	3	NaN	30	pw_skills	1

Q5.

""" DataFrame.size :

- returns the total number of elements in the DataFrame, which is equal to the product of the number of rows and columns.

DataFrame.shape :

- returns a tuple representing the dimensions of the DataFrame in the form number of rows, number of columns.

"""

Q6.

"""

- pd.read_excel('example.xlsx')

"""

In [20]:
```
df1['Email'] = ['abc@gmail.com','xyz@gmail.com','john@gmail.com']
df1
```

Out[20]:

	id	location_id	program_id	Values	Text	Word_Count	Email
0	1	1	NaN	10	Course name	2	abc@gmail.com
1	2	2	NaN	20	Data science	2	xyz@gmail.com
2	3	3	NaN	30	pw_skills	1	john@gmail.com

In [21]:
```
def extract_un(df):
    df['Username'] = df['Email'].str.split('@').str.get(0)
    print(df)

extract_un(df1)
```

```
   id  location_id  program_id  Values          Text  Word_Count  \
0   1            1         NaN      10  Course name            2
1   2            2         NaN      20  Data science           2
2   3            3         NaN      30    pw_skills            1

            Email Username
0   abc@gmail.com      abc
1   xyz@gmail.com      xyz
2  john@gmail.com     john
```

Q8.

In [22]:
```
df = pd.DataFrame({'A':[3,8,6,2,9] ,
                   'B' : [5,2,9,3,1],
                   'C' : [1,7,4,5,2]})
df
```

Out[22]:

	A	B	C
0	3	5	1
1	8	2	7
2	6	9	4
3	2	3	5
4	9	1	2

In [26]:
```
df[ (df['A'] > 5) & (df['B'] < 10) ]
```

Out[26]:

	A	B	C
1	8	2	7
2	6	9	4
4	9	1	2

Q9.

In [27]:
```
dff = pd.DataFrame({'Values': [10,20,30,40,50]})
dff
```

Out[27]:

	Values
0	10
1	20
2	30
3	40
4	50

In [28]:
```
dff.describe()
```

Out[28]:

	Values
count	5.000000
mean	30.000000
std	15.811388
min	10.000000
25%	20.000000
50%	30.000000
75%	40.000000
max	50.000000

In [29]:
```
dff.mean()
```

Out[29]:
```
Values    30.0
dtype: float64
```

In [30]:
```
dff.std()
```

Out[30]:
```
Values    15.811388
dtype: float64
```

In [31]:
```
dff.median()
```

Out[31]:
```
Values    30.0
dtype: float64
```

Q10.

In [32]:
```
df = pd.DataFrame({'Date': ['2023-01-01', '2023-01-02', '2023-01-03', '2023-01-0
                   'Sales': [10, 15, 20, 25, 30]})
df
```

Out[32]:

	Date	Sales
0	2023-01-01	10
1	2023-01-02	15
2	2023-01-03	20
3	2023-01-04	25
4	2023-01-05	30

In [33]:
```
df['MovingAverage'] = df['Sales'].rolling(window=7,min_periods=1).mean()
df
```

Out[33]:

	Date	Sales	MovingAverage
0	2023-01-01	10	10.0
1	2023-01-02	15	12.5
2	2023-01-03	20	15.0
3	2023-01-04	25	17.5
4	2023-01-05	30	20.0

Q11.

In [36]:
```
df['WeekDay'] = ['Sunday','Monday','Tuesday','Wednesday','Thursday']
df
```

Out[36]:

	Date	Sales	MovingAverage	WeekDay
0	2023-01-01	10	10.0	Sunday
1	2023-01-02	15	12.5	Monday
2	2023-01-03	20	15.0	Tuesday
3	2023-01-04	25	17.5	Wednesday
4	2023-01-05	30	20.0	Thursday

Q12.

In [37]:
```
df = pd.DataFrame({'Date': ['2023-01-05', '2023-01-15', '2023-02-05', '2023-02-1
                   'Value': [10, 20, 30, 40]})
df
```

Out[37]:

	Date	Value
0	2023-01-05	10
1	2023-01-15	20
2	2023-02-05	30
3	2023-02-15	40

In [38]:
```
df[(df['Date'] >= '2023-01-01') & (df['Date'] <= '2023-01-31')]
```

Out[38]:

	Date	Value
0	2023-01-05	10
1	2023-01-15	20

Q13.

In [39]:
```
import pandas as pd
```