# Untitled

September 20, 2024

[17]:
```
# Q1
import pandas as pd
url ="https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/
 ↪2020/2020-05-05/villagers.csv"
df = pd.read_csv(url)
df.isna().sum()
```

[17]:
```
row_n           0
id              1
name            0
gender          0
species         0
birthday        0
personality     0
song           11
phrase          0
full_id         0
url             0
dtype: int64
```

[5]:
```
# Q2
import pandas as pd

# Load the dataset
url = "https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/
 ↪data/2020/2020-05-05/villagers.csv"
villagers_df = pd.read_csv(url)

# Check the columns and the number of rows
columns = villagers_df.columns
row_count = villagers_df.shape[0]

columns, row_count
```

[5]:
```
(Index(['row_n', 'id', 'name', 'gender', 'species', 'birthday', 'personality',
        'song', 'phrase', 'full_id', 'url'],
       dtype='object'),
```

```
391)
```

[16]:
```
# Observations refer to individual records or data points within a dataset.
# Each observation represents a single unit of analysis and typically␣
 ↪corresponds to a single row in a dataset.
# In this dataset means a single Animal Crossing character, like "Bulbasaur".
```

[ ]:
```
# Variables are the different types of information or attributes that are␣
 ↪recorded for each observation.
# Each variable corresponds to a column in a dataset.
# In this dataset like "name", "species" etc.
```

[11]:
```
# Q3
df.describe()
```

[11]:
```
            row_n
count  391.000000
mean   239.902813
std    140.702672
min      2.000000
25%    117.500000
50%    240.000000
75%    363.500000
max    483.000000
```

[18]:
```
df['species'].value_counts()
```

[18]:
```
species
cat          23
rabbit       20
frog         18
squirrel     18
duck         17
dog          16
cub          16
pig          15
bear         15
mouse        15
horse        15
bird         13
penguin      13
sheep        13
elephant     11
wolf         11
ostrich      10
deer         10
eagle         9
```

```
gorilla       9
chicken       9
koala         9
goat          8
hamster       8
kangaroo      8
monkey        8
anteater      7
hippo         7
tiger         7
alligator     7
lion          7
bull          6
rhino         6
cow           4
octopus       3
Name: count, dtype: int64
```

[20]:
```
# Q4
# df.shape returns the total number of rows and columns in the dataset(does not␣
 ↪report information about missing values).
# df.describe() only analyzes numerical columns. It does not include␣
 ↪non-numeric columns in its summary.
# The "count" column shows the number of non-null values, which might be less␣
 ↪than the total number of rows if there are missing values.
# If the dataset contains both numeric and non-numeric columns,
# the number of columns analyzed by df.describe() will generally be fewer than␣
 ↪the total number of columns in the dataset as given by df.shape.
url = "https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.
 ↪csv"
df = pd.read_csv(url)
print(df.shape[0], df.shape[1])
df.describe()
```

```
891 15
```

[20]:

| | survived | pclass | age | sibsp | parch | fare |
|---|---|---|---|---|---|---|
| count | 891.000000 | 891.000000 | 714.000000 | 891.000000 | 891.000000 | 891.000000 |
| mean | 0.383838 | 2.308642 | 29.699118 | 0.523008 | 0.381594 | 32.204208 |
| std | 0.486592 | 0.836071 | 14.526497 | 1.102743 | 0.806057 | 49.693429 |
| min | 0.000000 | 1.000000 | 0.420000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 0.000000 | 2.000000 | 20.125000 | 0.000000 | 0.000000 | 7.910400 |
| 50% | 0.000000 | 3.000000 | 28.000000 | 0.000000 | 0.000000 | 14.454200 |
| 75% | 1.000000 | 3.000000 | 38.000000 | 1.000000 | 0.000000 | 31.000000 |
| max | 1.000000 | 3.000000 | 80.000000 | 8.000000 | 6.000000 | 512.329200 |

[ ]:
```
# Q5
# The shape attribute of a pandas DataFrame provides the dimensions of the␣
 ↪DataFrame-specifically, the number of rows and columns.
# It accesses attributes directly without parentheses.
# Methods require parentheses, even if no arguments are passed, to be executed.
# The describe() method generates descriptive statistics of the numerical␣
 ↪columns in the DataFrame, such as count, mean, standard deviation, min, and␣
 ↪max.
# Accessed with parentheses & Perform actions or computations and can return␣
 ↪results
# "Shape" just shows these data, "describe" needs to be calculated.
```

[ ]:
```
# Q6 https://chatgpt.com/share/66e38233-6c30-800b-a5d9-8e778ca5d6c2
# I can't put the picture of the formulas in here, so I put the link about␣
 ↪their formulas.
# count: Simply the total number of non-missing values.
# mean: The average value of the entries in the column, is calculated by␣
 ↪summing all the values and dividing by the count of entries.
# Standard Deviation (std): A measure of the amount of variation or dispersion␣
 ↪of the values in the column. It indicates how much the values deviate from␣
 ↪the mean.
# Minimum (min): The smallest value in the column.
# 25th Percentile (25%): The value below which 25% of the entries in the column␣
 ↪fall. Also known as the first quartile (Q1).
# Median (50%): The middle value of the column when the entries are sorted.␣
 ↪Half of the values are below this point and half are above it.
# 75th Percentile (75%): The value below which 75% of the entries in the column␣
 ↪fall. Also known as the third quartile (Q3).
# Maximum (max): The largest value in the column.
```

[1]:
```
# Q7
# 1 When 97% of data  in a set are valid, 3% of data are invalid.
# 1 In this situation, most of the data is available, "df.dropna()" is better.
# 2 When the vast majority of a set of data is invalid,"del df['col']" might be␣
 ↪preferred over using "df.dropna()".
# 3 Removing irrelevant columns beforehand reduces the complexity of the␣
 ↪dataset.
# 3 By removing unnecessary columns first, "df.dropna()" can operate more␣
 ↪efficiently, focusing on a smaller, more relevant set of data.
# 4 justification:
# 4 Before: The dataset contains missing values in various columns, making it␣
 ↪less reliable for analysis.
# 4 After: By removing the irrelevant Name column and then dropping rows with␣
 ↪missing values, we are left with a cleaner dataset containing only complete␣
 ↪and relevant data.
```

```
# 4 This approach ensures that the remaining data is both complete and␣
 ↪pertinent to the analysis.
# 4
import pandas as pd

# Sample DataFrame with missing values
data = {
    'EmployeeID': [1, 2, 3, 4, 5],
    'Name': ['Alice', 'Bob', None, 'David', 'Ella'],
    'Age': [25, None, 30, 45, None],
    'Department': ['HR', 'IT', 'Finance', None, 'IT'],
    'Salary': [50000, 55000, None, 60000, 62000]
}

df = pd.DataFrame(data)

print("Before:")
print(df)
```

```
Before:
   EmployeeID   Name   Age Department   Salary
0           1  Alice  25.0         HR  50000.0
1           2    Bob   NaN         IT  55000.0
2           3   None  30.0    Finance      NaN
3           4  David  45.0       None  60000.0
4           5   Ella   NaN         IT  62000.0
```

[2]:
```
# Remove the 'Name' column
del df['Name']
```

[3]:
```
# Drop rows with any missing values
df_cleaned = df.dropna()
print("After:")
print(df_cleaned)
```

```
After:
   EmployeeID   Age Department   Salary
0           1  25.0         HR  50000.0
```

[12]:
```
# Q8 1 https://chatgpt.com/share/66e385ec-a62c-800b-a3f9-77592945e7b8
df.shape
df.columns
```

[12]:
```
Index(['EmployeeID', 'Age', 'Department', 'Salary'], dtype='object')
```

[17]:
```
df. groupby ("Age")["Salary"].describe()
```

[17]:

| Age | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| 25.0 | 1.0 | 50000.0 | NaN | 50000.0 | 50000.0 | 50000.0 | 50000.0 | 50000.0 |
| 30.0 | 0.0 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 45.0 | 1.0 | 60000.0 | NaN | 60000.0 | 60000.0 | 60000.0 | 60000.0 | 60000.0 |

[21]:
```
# Q8 2 In df.describe(), missing values are simply omitted from the count, which gives you a sense of overall completeness.
# Q8 2 In df.groupby("col1")["col2"].describe(), missing values are considered within each group. If some groups have more missing values than others, this will show up as varying counts across groups.
```

[23]:
```
# Q8 3 https://jupyter.utoronto.ca/user/nicolez.huang@mail.utoronto.ca/files/Untitled1.ipynb?_xsrf=2%7C80c9eef3%7C7b7c3f249d39640a25227183828dfd72%7C1726154395
```

[ ]:
```
# yes, I review the wiki-textbook.
# https://chatgpt.com/share/66e37ecf-448c-800b-9a74-2a3d89d61eeb (Q1-Q3)
# https://chatgpt.com/share/66e38233-6c30-800b-a5d9-8e778ca5d6c2 (Q6)
# https://chatgpt.com/share/66e385ec-a62c-800b-a3f9-77592945e7b8 (Q8)
```