QAA – DATA TRANSFORMATION

USING PYTHON

- Using python

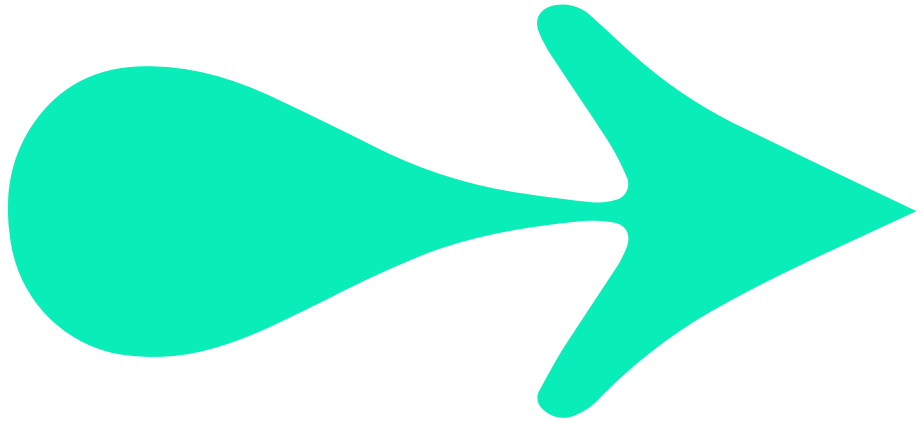
USING PYTHON

- Data Transformation

Importing packages

Import pandas and numpy

```
import pandas as pd
import numpy as np
```



Loading data

Hard-coded data entry

```
data = {
    'Name': ['Alice', 'Bob Gunner', '     Charlie', np.nan, '          Eve', 'Frank', 'Grace', 'Alice'],
    'Age': [25, np.nan, 35, 40, 50, 'Unknown', 30, 25],
    'Gender': ['Female', 'Male', np.nan, 'Female', 'Other', 'Male', 'Female', 'Female'],
    'Salary': [50000, 52000, 55000, np.nan, 48000, 'N/A', 51000, 50000],
    'Email': ['alice@email.com', 'bob@email', 'charlie--email.com', '', 'eve@email.com', '', 'grace@email.com', 'alice@email.
    'Retired':[False,True,False,False,False,False,False,False]
}
```

```
df = pd.DataFrame(data)
print(df)
```

```
           Name      Age  Gender Salary              Email  Retired
0         Alice       25  Female  50000    alice@email.com    False
1    Bob Gunner      NaN    Male  52000          bob@email     True
2       Charlie       35     NaN  55000  charlie--email.com    False
3           NaN       40  Female    NaN                       False
4           Eve       50   Other  48000      eve@email.com    False
5         Frank  Unknown    Male    N/A                       False
6         Grace       30  Female  51000    grace@email.com    False
7         Alice       25  Female  50000    alice@email.com    False
```

Loading data

CSV Loading

```
df_csv = pd.read_csv("Seattle_cycles_station.csv",sep=',')
```

```
df_csv.head(5)
```

]:

	station_id	name	lat	long	install_date	install_dockcount	modification_date	current_dockcount	decommission_date
0	BT-01	3rd Ave & Broad St	47.618418	-122.350964	10/13/2014	18	NaN	18	NaN
1	BT-03	2nd Ave & Vine St	47.615829	-122.348564	10/13/2014	16	NaN	16	NaN
2	BT-04	6th Ave & Blanchard St	47.616094	-122.341102	10/13/2014	16	NaN	16	NaN
3	BT-05	2nd Ave & Blanchard St	47.613110	-122.344208	10/13/2014	14	NaN	14	NaN
4	CBD-03	7th Ave & Union St	47.610731	-122.332447	10/13/2014	20	NaN	20	NaN

EXERCISE 1: LOADING THE DATAFRAME

- Install pandas into a notebook
- Load the data from a csv file called seattle_cycles_station.csv into a dataframe called df_cycle_station
- View the dataframe using the print function

Using tail(n)

```
df_csv.tail(8)
```

]:

	station_id	name	lat	long	install_date	install_dockcount	modification_date	current_dockcount	decommission_date
50	UW-07	UW Intramural Activities Building	47.653713	-122.302162	10/13/2014	20	2/20/2015	14	NaN
51	UW-10	UW Magnuson Health Sciences Center Rotunda / C...	47.650725	-122.311188	10/13/2014	16	NaN	16	NaN
52	WF-01	Pier 69 / Alaskan Way & Clay St	47.614315	-122.354093	10/13/2014	18	NaN	24	NaN
53	WF-04	Seattle Aquarium / Alaskan Way S & Elliott Bay...	47.607702	-122.341650	10/13/2014	18	NaN	18	NaN
54	CH-16	Broadway and E Denny Way	47.618640	-122.320777	3/18/2016	18	NaN	18	NaN
55	SLU-22	Thomas St & 5th Ave N	47.620879	-122.347377	7/3/2016	18	NaN	18	NaN
56	UW-11	NE Pacific St/UW Medical Center	47.649952	-122.306263	10/29/2015	16	NaN	16	NaN
57	WF-03	Pier 66 / Alaskan Way & Bell St	47.611370	-122.348702	8/9/2016	18	NaN	18	NaN

Using head(n)

```
df_csv.tail(8)
```

]:

	station_id	name	lat	long	install_date	install_dockcount	modification_date	current_dockcount	decommission_date
50	UW-07	UW Intramural Activities Building	47.653713	-122.302162	10/13/2014	20	2/20/2015	14	NaN
51	UW-10	UW Magnuson Health Sciences Center Rotunda / C...	47.650725	-122.311188	10/13/2014	16	NaN	16	NaN
52	WF-01	Pier 69 / Alaskan Way & Clay St	47.614315	-122.354093	10/13/2014	18	NaN	24	NaN
53	WF-04	Seattle Aquarium / Alaskan Way S & Elliott Bay...	47.607702	-122.341650	10/13/2014	18	NaN	18	NaN
54	CH-16	Broadway and E Denny Way	47.618640	-122.320777	3/18/2016	18	NaN	18	NaN
55	SLU-22	Thomas St & 5th Ave N	47.620879	-122.347377	7/3/2016	18	NaN	18	NaN
56	UW-11	NE Pacific St/UW Medical Center	47.649952	-122.306263	10/29/2015	16	NaN	16	NaN
57	WF-03	Pier 66 / Alaskan Way & Bell St	47.611370	-122.348702	8/9/2016	18	NaN	18	NaN

EXERCISE 2: REVIEWING THE FIRST AND LAST ROWS

- Showing the LAST and HEAD functions to review the content of the dataframe (5 rows only)

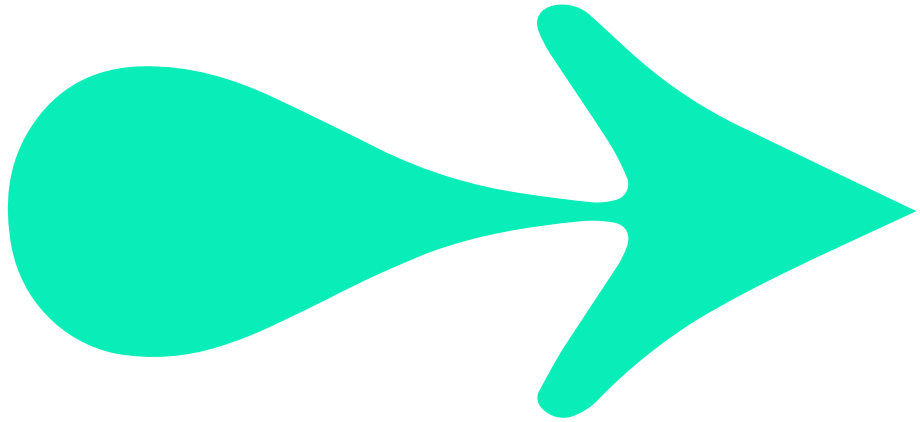
Showing the columns and data types

```
print(df_csv.columns)
```

```
Index(['station_id', 'name', 'lat', 'long', 'install_date',
       'install_dockcount', 'modification_date', 'current_dockcount',
       'decommission_date'],
      dtype='object')
```

```
print(df_csv.dtypes)
```

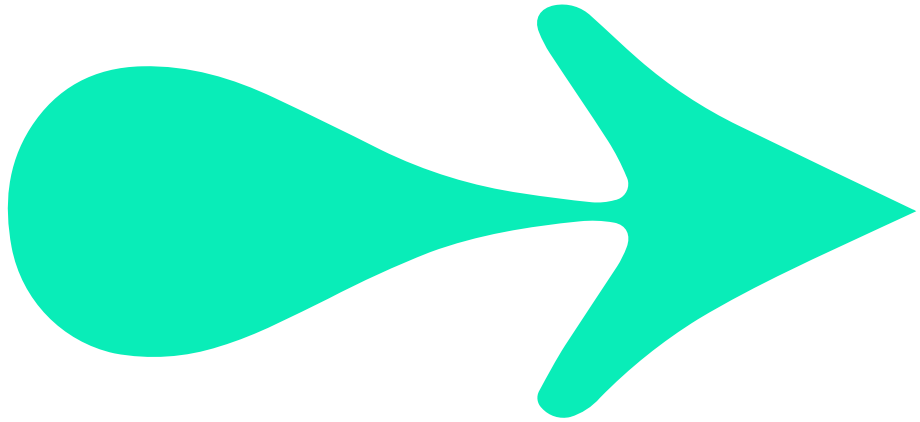
```
station_id            object
name                  object
lat                  float64
long                 float64
install_date          object
install_dockcount      int64
modification_date     object
current_dockcount      int64
decommission_date     object
dtype: object
```



EXERCISE 3: REVIEWING THE DATA TYPES

- Write the code to review the columns available in the df_cycle_station dataframe
- Write the code to review the data types of the dataframe

Showing number of cells, rows and columns



How many cells does the dataframe have?

```
print(df_cycle_station.size)
```

```
580
```

How the number of rows and coloumns?

```
df_cycle_station.shape[0]
```

```
58
```

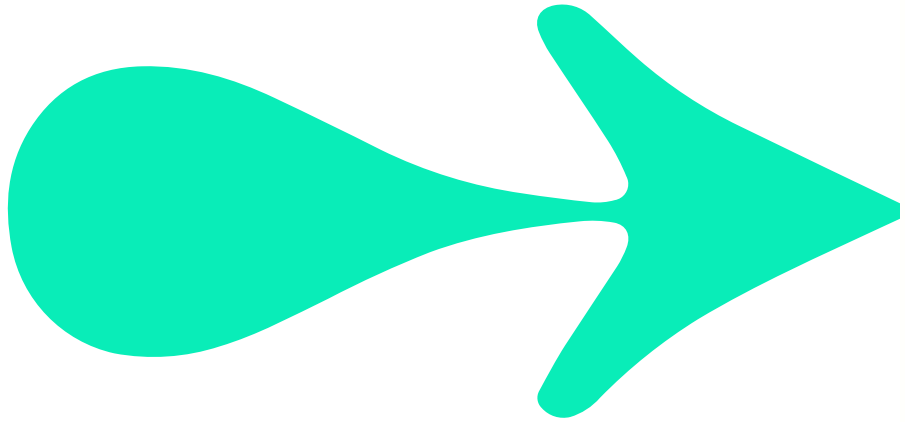
```
df_cycle_station.shape[1]
```

```
10
```

EXERCISE 4: REVIEWING THE DATAFRAME SIZE

- Write the code to review the columns available in the df_cycle_station dataframe
- Write the code to review the data types of the dataframe

Replacing certain values



Replacing the N/A values with a set value

```
df_filled = df.fillna({'Name': 'Unknown', 'Gender': 'Unknown'})
print(df_filled)
```

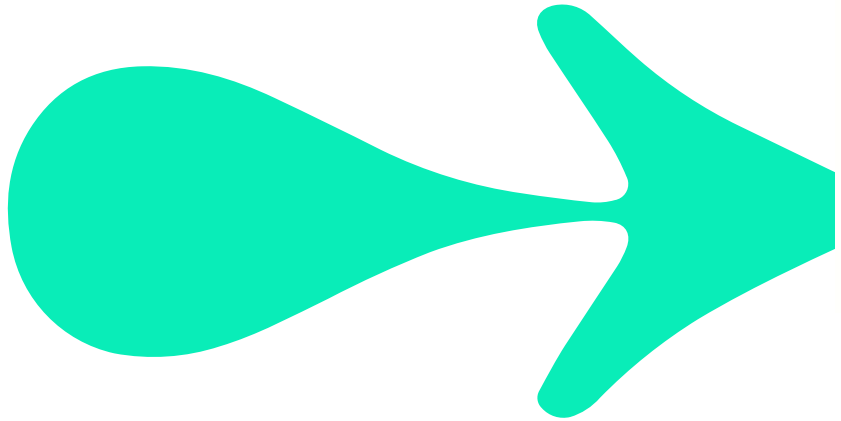
```
             Name      Age   Gender Salary              Email
0           Alice       25   Female  50000     alice@email.com
1     Bob Gunner      NaN     Male  52000           bob@email
2        Charlie       35  Unknown  55000  charlie--email.com
3        Unknown       40   Female    NaN
4            Eve       50    Other  48000       eve@email.com
5          Frank  Unknown     Male    N/A
6          Grace       30   Female  51000     grace@email.com
7          Alice       25   Female  50000     alice@email.com
```

replace NaN with set values

```
df.replace({'Age': 'Unknown', 'Salary': 'N/A'}, np.nan, inplace=True)
print(df)
```

```
             Name   Age  Gender   Salary              Email
0           Alice  25.0  Female  50000.0     alice@email.com
1     Bob Gunner   NaN    Male  52000.0           bob@email
2        Charlie  35.0     NaN  55000.0  charlie--email.com
3            NaN  40.0  Female      NaN
4            Eve  50.0   Other  48000.0       eve@email.com
5          Frank   NaN    Male      NaN
6          Grace  30.0  Female  51000.0     grace@email.com
7          Alice  25.0  Female  50000.0     alice@email.com
```

Calculate a replacement value



Replacing age and salary with a calculated replacement

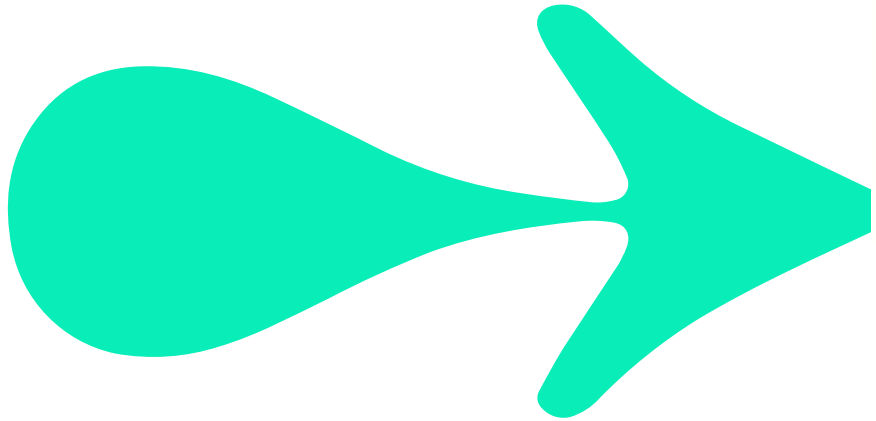
```
age_median = df['Age'].median(skipna=True)
salary_mean = df['Salary'].mean(skipna=True)
df['Age'].fillna(age_median, inplace=True)
df['Salary'].fillna(salary_mean, inplace=True)
print(df)
```

```
              Name   Age  Gender   Salary              Email
0            Alice  25.0  Female  50000.0    alice@email.com
1      Bob Gunner  32.5    Male  52000.0          bob@email
2          Charlie  35.0     NaN  55000.0  charlie--email.com
3              NaN  40.0  Female  51000.0
4              Eve  50.0   Other  48000.0      eve@email.com
5            Frank  32.5    Male  51000.0
6            Grace  30.0  Female  51000.0    grace@email.com
7            Alice  25.0  Female  50000.0    alice@email.com
```

EXERCISE 5: CLEANING DATA

- Write the code to review the decommission_date within the dataframe
- Write the code to replace any blank values with ‘12/31/2030’
- Review the column again to check the values have been changed
- Write the code to replace all decommission_date rows that have ‘12/31/2030’ with np.nan

Changing data types

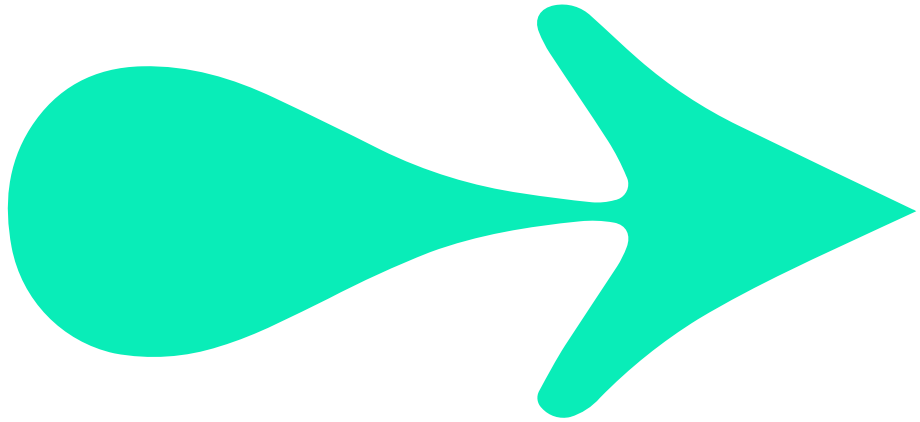


Change the data types of the numeric fields

```
df['Age'] = df['Age'].astype(int)
df['Salary'] = df['Salary'].astype(float)
print(df)
```

```
             Name  Age  Gender   Salary              Email
0           Alice   25  Female  50000.0    alice@email.com
1     Bob Gunner    32    Male  52000.0          bob@email
2         Charlie   35     NaN  55000.0  charlie--email.com
3             NaN   40  Female  51000.0
4             Eve   50   Other  48000.0      eve@email.com
5           Frank   32    Male  51000.0
6           Grace   30  Female  51000.0    grace@email.com
7           Alice   25  Female  50000.0    alice@email.com
```

Working with Dates



Working with dates

```
date_example_df = pd.DataFrame({
    'Name': ['Alice', 'Bob Newhart'],
    'DOB_String': ['2021-08-10 15:30:45','2021-08-10 15:30:45']
})
date_example_df['DOB_DateTime'] = pd.to_datetime(date_example_df['DOB_String'])

print(date_example_df)
```

```
          Name           DOB_String        DOB_DateTime
0        Alice  2021-08-10 15:30:45 2021-08-10 15:30:45
1  Bob Newhart  2021-08-10 15:30:45 2021-08-10 15:30:45
```

```
# Example: '2021-08-10 15:30:45'
date_example_df['Year'] = date_example_df['DOB_DateTime'].dt.year
date_example_df['Month'] = date_example_df['DOB_DateTime'].dt.month
date_example_df['day'] = date_example_df['DOB_DateTime'].dt.day
print(date_example_df)
```

```
          Name           DOB_String        DOB_DateTime  Year  Month  day
0        Alice  2021-08-10 15:30:45 2021-08-10 15:30:45  2021      8   10
1  Bob Newhart  2021-08-10 15:30:45 2021-08-10 15:30:45  2021      8   10
```

```
date_example_df['hour'] = date_example_df['DOB_DateTime'].dt.hour
date_example_df['minute'] = date_example_df['DOB_DateTime'].dt.minute
date_example_df['second'] = date_example_df['DOB_DateTime'].dt.second
print(date_example_df)
```

```
          Name           DOB_String        DOB_DateTime  Year  Month  day  \
0        Alice  2021-08-10 15:30:45 2021-08-10 15:30:45  2021      8   10
1  Bob Newhart  2021-08-10 15:30:45 2021-08-10 15:30:45  2021      8   10

   hour  minute  second
0    15      30      45
1    15      30      45
```

EXERCISE 6: CHANGING DATA TYPES

- Write the code to review the data types of all columns in the dataframe
- Change the data type of the following columns:
  - Station_id and name to string
  - Lat and long to float64
- Write the code to review the data types of the changed columns in the dataframe
- Use pd.to_datatime(df['column']) to change the install_date, modification_date and decomiision_date to a date column

Normalize data

Normalise the numeric columns

```
df['Salary_Normalized'] = (df['Salary'] - df['Salary'].min()) / (df['Salary'].max() - df['Salary'].min())
df['Age_Normalized'] = (df['Age'] - df['Age'].min()) / (df['Age'].max() - df['Age'].min())
print(df)
```

```
            Name  Age  Gender   Salary              Email  Salary_Normalized  \
0          Alice   25  Female  50000.0     alice@email.com           0.285714
1     Bob Gunner   32    Male  52000.0           bob@email           0.571429
2        Charlie   35     NaN  55000.0  charlie--email.com           1.000000
3            NaN   40  Female  51000.0                              0.428571
4            Eve   50   Other  48000.0       eve@email.com           0.000000
5          Frank   32    Male  51000.0                              0.428571
6          Grace   30  Female  51000.0     grace@email.com           0.428571
7          Alice   25  Female  50000.0     alice@email.com           0.285714

   Age_Normalized
0            0.00
1            0.28
2            0.40
3            0.60
4            1.00
5            0.28
6            0.20
7            0.00
```

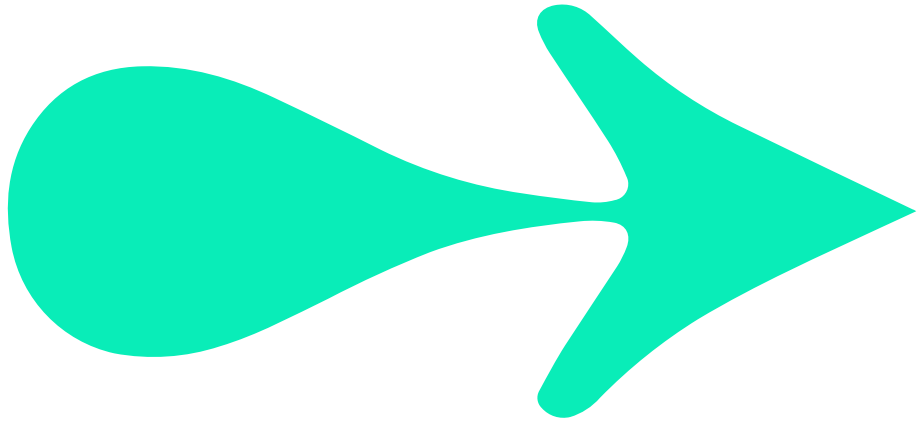
Placing in categories (Binning)

Binning of numeric values to group

```
bins = [0, 30, 40, 50, 60]
labels = ['20s', '30s', '40s', '50s']
df['Age Group'] = pd.cut(df['Age'], bins=bins, labels=labels, right=False)
print(df)
```

```
             Name  Age  Gender   Salary              Email  Salary_Normalized  \
0           Alice   25  Female  50000.0    alice@email.com           0.285714
1     Bob Gunner    32    Male  52000.0          bob@email           0.571429
2         Charlie   35     NaN  55000.0  charlie--email.com           1.000000
3             NaN   40  Female  51000.0                               0.428571
4             Eve   50   Other  48000.0      eve@email.com           0.000000
5           Frank   32    Male  51000.0                               0.428571
6           Grace   30  Female  51000.0    grace@email.com           0.428571
7           Alice   25  Female  50000.0    alice@email.com           0.285714

   Age_Normalized Age Group
0            0.00       20s
1            0.28       30s
2            0.40       30s
3            0.60       40s
4            1.00       50s
5            0.28       30s
6            0.20       30s
7            0.00       20s
```

Create category columns



Categorise the gender column (note the extra columns with boolean results)

```
gender_encoded = pd.get_dummies(df['Gender'], prefix='Gender')
df = pd.concat([df, gender_encoded], axis=1)
print(df)
```

```
             Name  Age  Gender   Salary              Email  Salary_Normalized  \
0           Alice   25  Female  50000.0    alice@email.com           0.285714
1     Bob Gunner    32    Male  52000.0          bob@email           0.571429
2         Charlie   35     NaN  55000.0  charlie--email.com           1.000000
3             NaN   40  Female  51000.0                              0.428571
4             Eve   50   Other  48000.0      eve@email.com           0.000000
5           Frank   32    Male  51000.0                              0.428571
6           Grace   30  Female  51000.0    grace@email.com           0.428571
7           Alice   25  Female  50000.0    alice@email.com           0.285714

   Age_Normalized Age Group  Gender_Female  Gender_Male  Gender_Other
0            0.00       20s              1            0             0
1            0.28       30s              0            1             0
2            0.40       30s              0            0             0
3            0.60       40s              1            0             0
4            1.00       50s              0            0             1
5            0.28       30s              0            1             0
6            0.20       30s              1            0             0
7            0.00       20s              1            0             0
```

EXERCISE 7: NORMALISING, BINNING AND CATEGORIES FROM VALUES

- Normalise the lat and long columns into the columns lat_normal and long_normal
- List the unique values from the install_dockcount amd current_dockcount columns
- Create bins for the dockcount as follows:
  - 0,1,10,15,24,30,100
  - Empty, Very-Small, Small, Medium, Large, Extra-Large
- Bin the install_dockcount column of the dataframe into the column install_group
- Bin the current_dockcount column of the dataframe into the column current_group
- Display the dataframe columns (install_dockcount, install_group, current_dockcount, current_group)

Function for checking email has @

create a function to validate email address (@) and replace if needed

```python
# extended reading on functions
def is_valid_email(email):
    if '@' in email:
        return email
    else:
        return np.nan

df['Email'] = df['Email'].apply(is_valid_email)
print(df['Email'])
```

```
0    alice@email.com
1          bob@email
2                NaN
3                NaN
4      eve@email.com
5                NaN
6    grace@email.com
7    alice@email.com
Name: Email, dtype: object
```

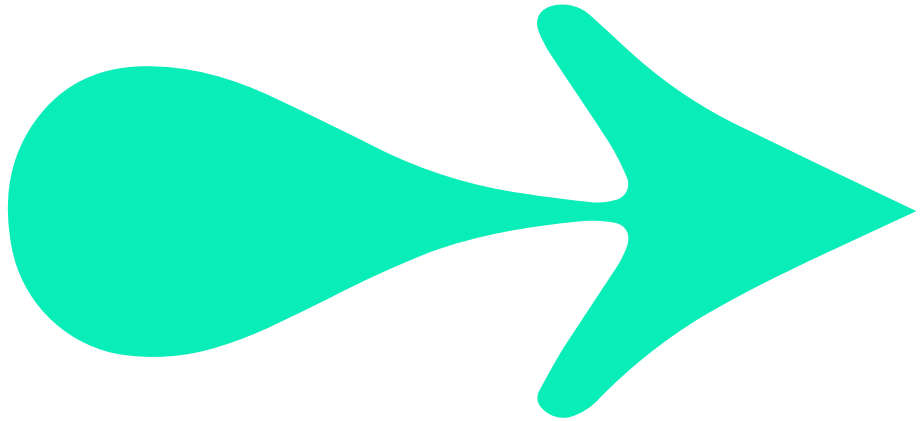
Working with strings

Splitting the address

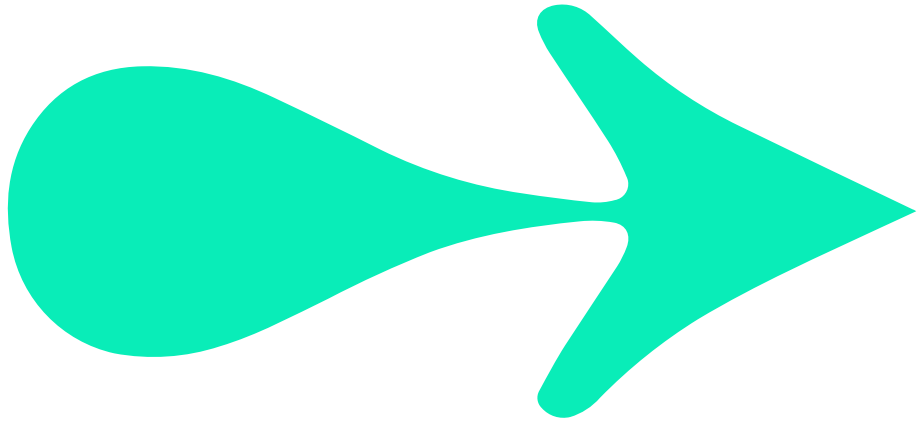
```
df_cycle_station['left'] = df_cycle_station['name'].str.split('&').str[0]
print(df_cycle_station)
```

Testing strings

```
df_cycle_station['Length_Name'] = df_cycle_station['name'].str.len()
df_cycle_station['name'] = df_cycle_station['name'].str.strip()
df_cycle_station['Length_NameAfterStrip'] = df_cycle_station['name'].str.len()
df_cycle_station
```



Checking lengths strings within columns



Show the lengths of two text fields (before and after strip)

```
df['Length_Email'] = df['Email'].str.len()
df['Length_Name'] = df['Name'].str.len()
print(df)
```

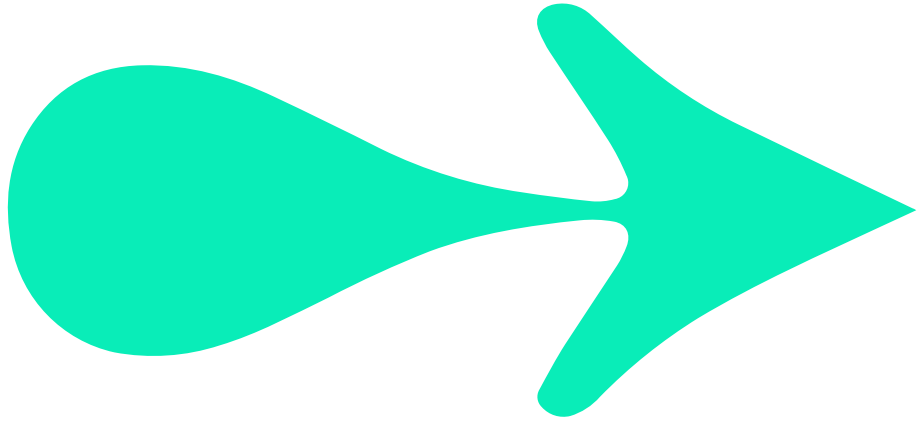
```
             Name  Age  Gender   Salary              Email  Salary_Normalized  \
0           Alice   25  Female  50000.0    alice@email.com           0.285714
1     Bob Gunner    32    Male  52000.0          bob@email           0.571429
2         Charlie   35     NaN  55000.0  charlie--email.com           1.000000
3             NaN   40  Female  51000.0                              0.428571
4             Eve   50   Other  48000.0      eve@email.com           0.000000
5           Frank   32    Male  51000.0                              0.428571
6           Grace   30  Female  51000.0    grace@email.com           0.428571
7           Alice   25  Female  50000.0    alice@email.com           0.285714

   Age_Normalized Age Group  Gender_Female  Gender_Male  Gender_Other  \
0            0.00       20s              1            0             0
1            0.28       30s              0            1             0
2            0.40       30s              0            0             0
3            0.60       40s              1            0             0
4            1.00       50s              0            0             1
5            0.28       30s              0            1             0
6            0.20       30s              1            0             0
7            0.00       20s              1            0             0

   Length_Email  Length_Name
0            15          5.0
1             9         10.0
2            18         12.0
3             0          NaN
4            13         13.0
5             0          5.0
6            15          5.0
7            15          5.0
```

Replace cleansed column

```
df['Name'] = df['Name'].str.strip()
df['Length_NameAfterStrip'] = df['Name'].str.len()
print(df)
```

```
          Name  Age  Gender   Salary              Email  Salary_Normalized  \
0        Alice   25  Female  50000.0    alice@email.com           0.285714
1  Bob Gunner   32    Male  52000.0          bob@email           0.571429
2      Charlie   35     NaN  55000.0  charlie--email.com           1.000000
3          NaN   40  Female  51000.0                              0.428571
4          Eve   50   Other  48000.0      eve@email.com           0.000000
5        Frank   32    Male  51000.0                              0.428571
6        Grace   30  Female  51000.0    grace@email.com           0.428571
7        Alice   25  Female  50000.0    alice@email.com           0.285714

   Age_Normalized Age Group  Gender_Female  Gender_Male  Gender_Other  \
0            0.00       20s              1            0             0
1            0.28       30s              0            1             0
2            0.40       30s              0            0             0
3            0.60       40s              1            0             0
4            1.00       50s              0            0             1
5            0.28       30s              0            1             0
6            0.20       30s              1            0             0
7            0.00       20s              1            0             0

   Length_Email  Length_Name  Length_NameAfterStrip
0            15          5.0                    5.0
1             9         10.0                   10.0
2            18         12.0                    7.0
3             0          NaN                    NaN
4            13         13.0                    3.0
5             0          5.0                    5.0
6            15          5.0                    5.0
7            15          5.0                    5.0
```

Standardising strings



Using the title, upper and lower function to format the strings

```
df['NameTitle'] = df['Name'].str.title()
print(df['NameTitle'])
```

```
0         Alice
1    Bob Gunner
2       Charlie
3           NaN
4           Eve
5         Frank
6         Grace
7         Alice
Name: NameTitle, dtype: object
```

```
df['NameUpper'] = df['Name'].str.upper()
print(df['NameUpper'])
```

```
0         ALICE
1    BOB GUNNER
2       CHARLIE
3           NaN
4           EVE
5         FRANK
6         GRACE
7         ALICE
Name: NameUpper, dtype: object
```

```
df['NameLower'] = df['Name'].str.lower()
print(df['NameLower'])
```

```
0         alice
1    bob gunner
2       charlie
3           NaN
4           eve
5         frank
6         grace
7         alice
Name: NameLower, dtype: object
```

EXERCISE 8: UPDATING STRINGS

- Split the name column at the character & placing the characters to the left into the column shorten_name
- Split the station_id column at the character - placing the characters to the left into the column code
- Review the two columns (shorten_name, code0
- Write code to:
  - Check the length of the name column (into length_name)
  - Clean the name column of any additional trailing spaces using strip (replacing the current name)
  - Check the length of the name column (into length_afterstrip)
  - Review the name, length_name, length_afterstrip

Dropping a column

```
df.drop(["NameUpper","NameTitle"], axis = 1,inplace = True)
df.reset_index(drop=True)
```

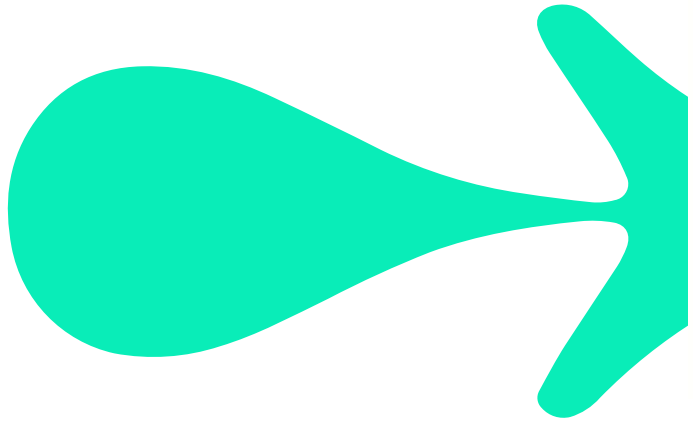
```
print(df)
```

```
          Name  Age  Gender   Salary            Email  Retired  \
0        Alice   25  Female  50000.0  alice@email.com    False
1  Bob Gunner   32    Male  52000.0        bob@email     True
2      Charlie   35     NaN  55000.0              NaN    False
3          NaN   40  Female  51000.0              NaN    False
4          Eve   50   Other  48000.0    eve@email.com    False
5        Frank   32    Male  51000.0              NaN    False
6        Grace   30  Female  51000.0  grace@email.com    False

   Salary_Normalized  Age_Normalized Age Group  Gender_Female  Gender_Male  \
0           0.285714            0.00       20s              1            0
1           0.571429            0.28       30s              0            1
2           1.000000            0.40       30s              0            0
3           0.428571            0.60       40s              1            0
4           0.000000            1.00       50s              0            0
5           0.428571            0.28       30s              0            1
6           0.428571            0.20       30s              1            0

   Gender_Other  Length_Email  Length_Name  Length_NameAfterStrip   NameLower
0             0            15          5.0                    5.0       alice
1             0             9         10.0                   10.0  bob gunner
2             0            18         12.0                    7.0     charlie
3             0             0          NaN                    NaN         NaN
4             1            13         13.0                    3.0         eve
5             0             0          5.0                    5.0       frank
6             0            15          5.0                    5.0       grace
```

EXERCISE 9: REMOVE COLUMNS NOT REQUIRED FROM DATAFRAME

- Review the shape of dimension 1 of the dataframe
- Use the drop function to remove the title, lower, upper columns individually
- Review the shape of dimension 1 of the dataframe (and see the difference)
- Use the drop function to remove the length_name and length_after strip columns (as a single line of code)

Summarising data



Summarisation

```
print(df.groupby('Gender').agg({'Age': ['mean', 'median','std','var','sum','first','last','min','max']}))
print(df.groupby('Gender').agg({'Age': ['count', 'mean', 'nunique','unique']}))
```

```
              Age
             mean median       std        var sum first last min max
Gender
Female  31.666667   30.0  7.637626  58.333333  95    25   30  25  40
Male    32.000000   32.0  0.000000   0.000000  64    32   32  32  32
Other   50.000000   50.0       NaN        NaN  50    50   50  50  50
         Age
       count       mean nunique        unique
Gender
Female     3  31.666667       3  [25, 40, 30]
Male       2  32.000000       1          [32]
Other      1  50.000000       1          [50]
```

Using describe

```
df.describe()
```

	Age	Salary	Salary_Normalized	Age_Normalized	Gender_Female	Gender_Male	Gender_Other	Length_Email	Length_Name	Length_Na
count	7.000000	7.000000	7.000000	7.000000	7.000000	7.000000	7.000000	7.000000	6.000000	
mean	34.857143	51142.857143	0.448980	0.394286	0.428571	0.285714	0.142857	10.000000	8.333333	
std	8.091736	2115.700942	0.302243	0.323669	0.534522	0.487950	0.377964	7.348469	3.777124	
min	25.000000	48000.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	5.000000	
25%	31.000000	50500.000000	0.357143	0.240000	0.000000	0.000000	0.000000	4.500000	5.000000	
50%	32.000000	51000.000000	0.428571	0.280000	0.000000	0.000000	0.000000	13.000000	7.500000	
75%	37.500000	51500.000000	0.500000	0.500000	1.000000	0.500000	0.000000	15.000000	11.500000	
max	50.000000	55000.000000	1.000000	1.000000	1.000000	1.000000	1.000000	18.000000	13.000000	

EXERCISE 10: SUMMARISE AND DESCRIBE

- Retrieve the following calculations
- (mean','median','std','var','sum','first','last','min','max','count', 'mean', 'nunique','unique
-)
- of the current_dockcount by code
- Use the describe function to review calculations for all numeric columns

Melting (unpivoting) Data

Moving columns to rows (unpivot in PBI)

```
df_melted = pd.melt(df, id_vars=['Name'], value_vars=['Age', 'Salary'])
print(df_melted)
```

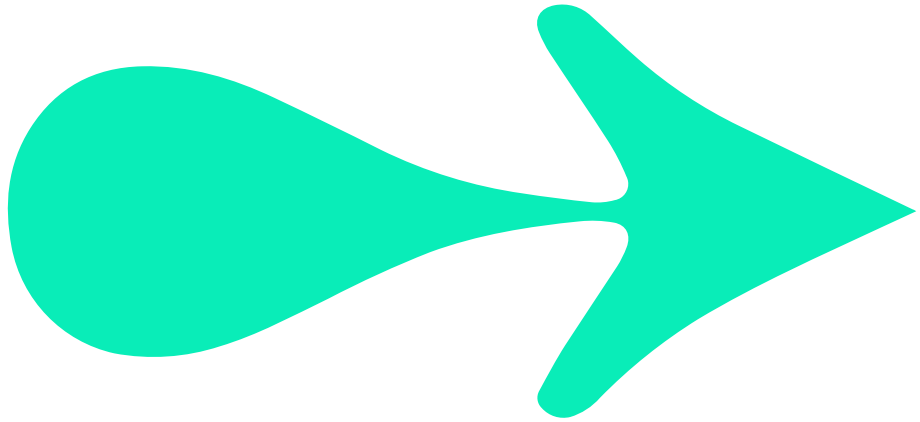
```
          Name variable    value
0        Alice      Age     25.0
1   Bob Gunner      Age     32.0
2      Charlie      Age     35.0
3          NaN      Age     40.0
4          Eve      Age     50.0
5        Frank      Age     32.0
6        Grace      Age     30.0
7        Alice   Salary  50000.0
8   Bob Gunner   Salary  52000.0
9      Charlie   Salary  55000.0
10         NaN   Salary  51000.0
11         Eve   Salary  48000.0
12       Frank   Salary  51000.0
13       Grace   Salary  51000.0
```

Pivoting data

Moving rows to columns (pivot in PBI)

```
df_pivoted = df.pivot(index='Name', columns='Gender', values='Salary')
print(df_pivoted)
```

```
Gender          NaN   Female     Male    Other
Name
NaN             NaN  51000.0      NaN      NaN
Alice           NaN  50000.0      NaN      NaN
Bob Gunner      NaN      NaN  52000.0      NaN
Charlie     55000.0      NaN      NaN      NaN
Eve             NaN      NaN      NaN  48000.0
Frank           NaN      NaN  51000.0      NaN
Grace           NaN  51000.0      NaN      NaN
```



Concat two data frames

	Name	Age	Gender	Salary	Email	DataGroup
0	Alice	25	Female	50000	alice@email.com	Company_A
1	Bob Newhart	NaN	Male	52000	bob@email	Company_A
2	Charlie	35	NaN	55000	charlie--email.com	Company_A
3	NaN	40	Female	NaN		Company_A
4	Eve	50	Other	48000	eve@email.com	Company_A
5	Frank	Unknown	Male	N/A		Company_A
6	Grace	30	Female	51000	grace@email.com	Company_A
7	Alice	25	Female	50000	alice@email.com	Company_A

	Name	Age	Gender	Salary	Email	DataGroup
0	Graham	85	Male	12000	g_able@email.com	Company_B
1	Simon	76	Male	52000	simon@here.com	Company_B
2	John	25	NaN	55000	jj@email.com	Company_B
3	Fred	43	Male	48000	fred_clampett@email.com	Company_B
4	Francis	62	Female	50000	frossi@quo.com	Company_B

```
df_concat = pd.concat([df1,df2], ignore_index = True, sort=True)
print(df_concat)
```

	Age	DataGroup	Email	Gender	Name	Salary
0	25	Company_A	alice@email.com	Female	Alice	50000
1	NaN	Company_A	bob@email	Male	Bob Newhart	52000
2	35	Company_A	charlie--email.com	NaN	Charlie	55000
3	40	Company_A		Female	NaN	NaN
4	50	Company_A	eve@email.com	Other	Eve	48000
5	Unknown	Company_A		Male	Frank	N/A
6	30	Company_A	grace@email.com	Female	Grace	51000
7	25	Company_A	alice@email.com	Female	Alice	50000
8	85	Company_B	g_able@email.com	Male	Graham	12000
9	76	Company_B	simon@here.com	Male	Simon	52000
10	25	Company_B	jj@email.com	NaN	John	55000
11	43	Company_B	fred_clampett@email.com	Male	Fred	48000
12	62	Company_B	frossi@quo.com	Female	Francis	50000

EXERCISE 11: USE THE CONCATENATE TO ADD A NEW ROW

- Add a newdata row using a list converted to a dataframe as:

```
newdata = {
    'station_id':['Downtown'],'name':['abc'],'lat':['47.7'],'long':['-122.3']}
df_cycle_stationNew = pd.DataFrame(newdata)
```

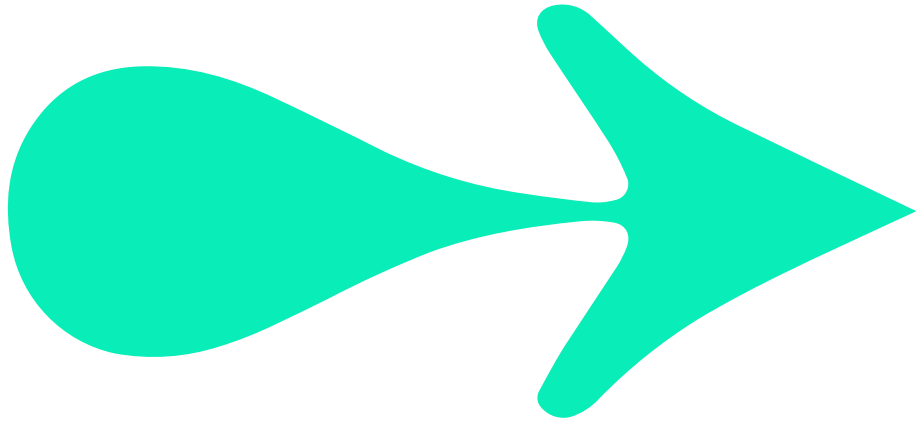
Use concat to add the newdata to the current dataframe

- Review the contents of the dataframe

Write to csv

write to csv file

```
df_stats.to_csv("stats.csv",sep=',')
```



EXERCISE 12: WRITE TO A CSV

Using the describe function from the last exercise output a new database (df_stats)

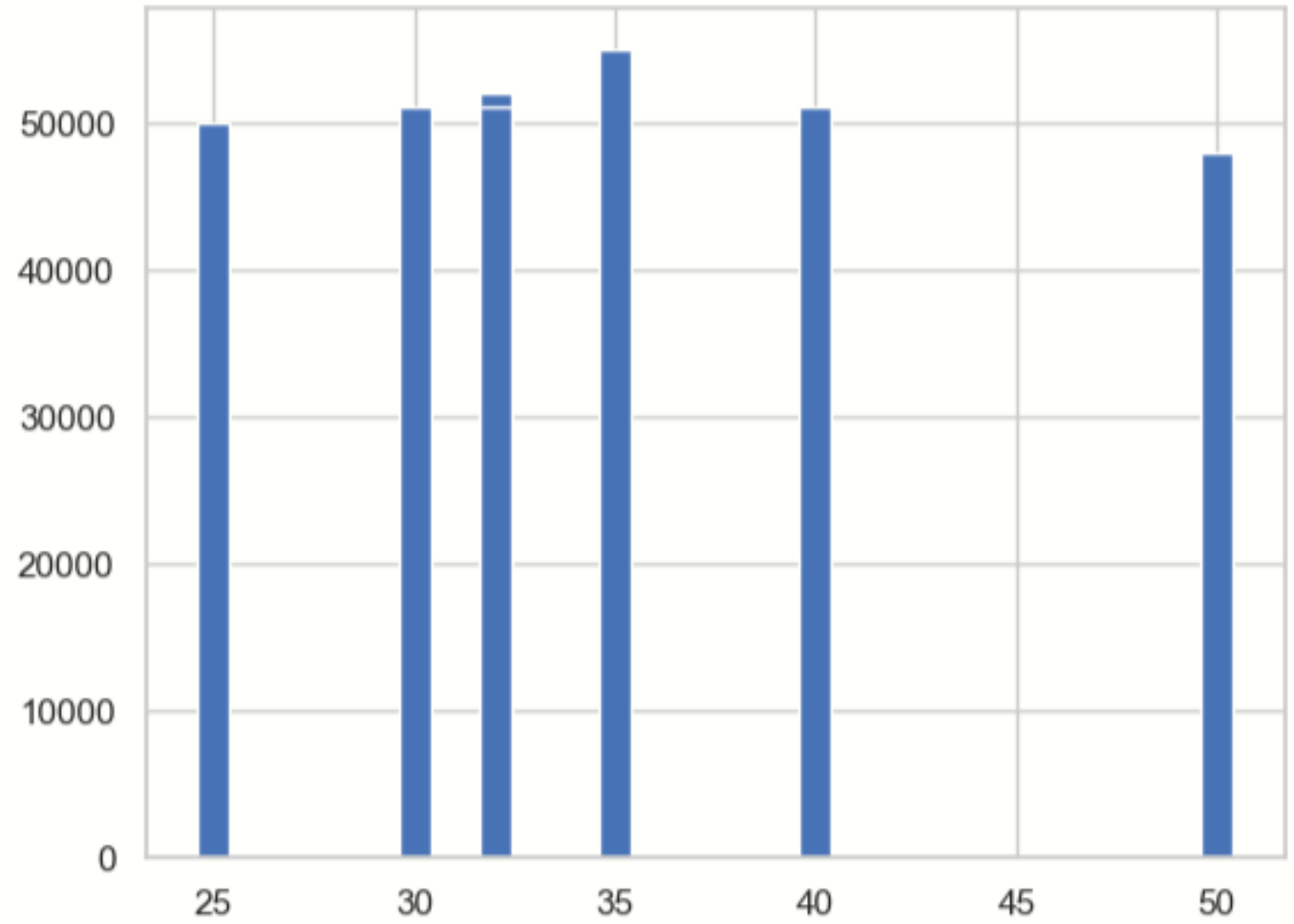
Write the dt_stats dataframe to a csv file

Matplotlib

Add a chart to the notebook using matlib

```
from matplotlib import pyplot as plt
plt.bar(x=df['Age'], height=df['Salary'])
```

```
<BarContainer object of 7 artists>
```

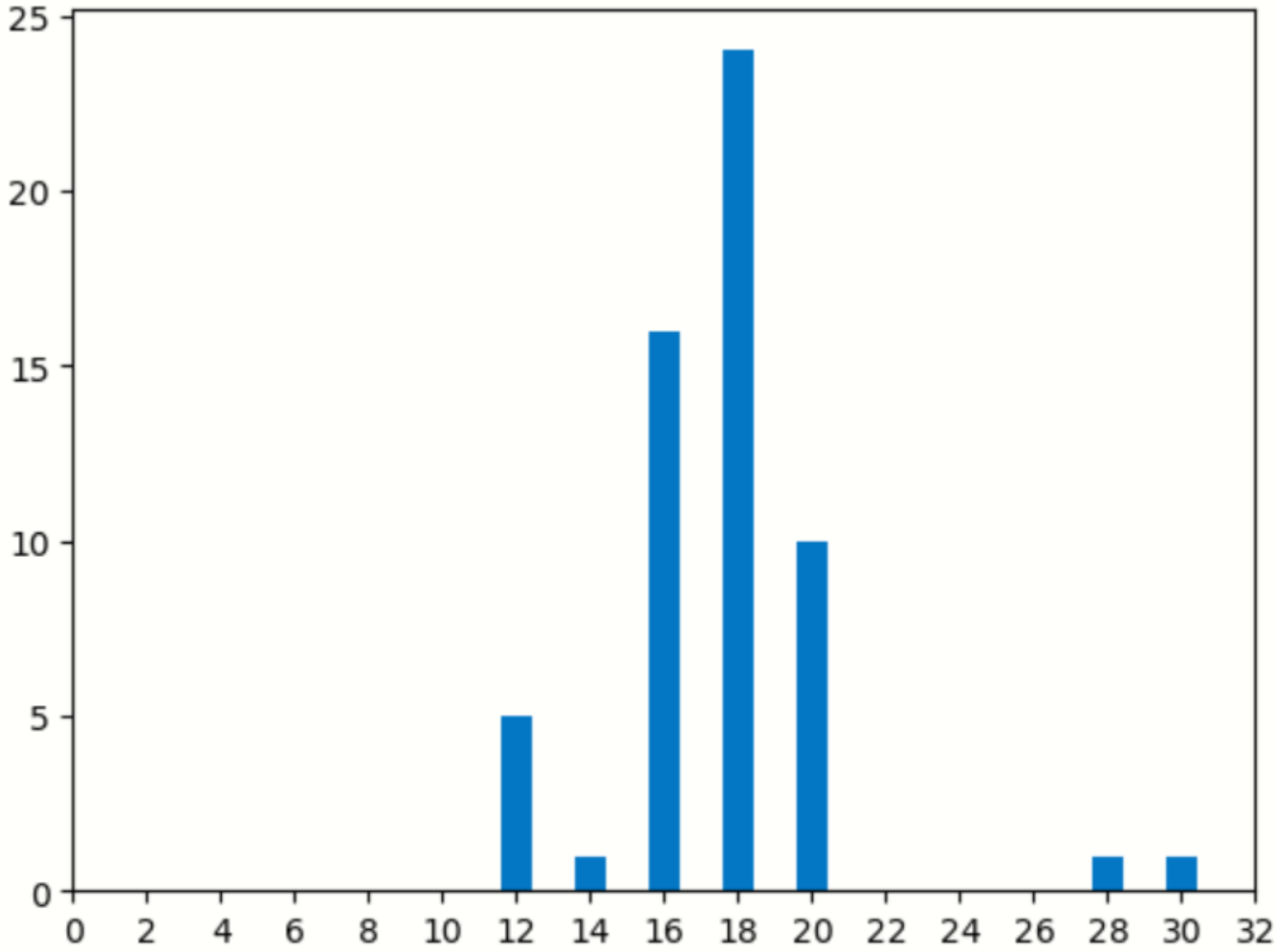


Updating styles

```python
# Clear the plot area
plt.clf()

# Create a bar plot of revenue by year
plt.bar(x=df['Age'], height=df['Salary'], color='orange')

# Customize the chart
plt.title('Salary by year')
plt.xlabel('Year')
plt.ylabel('Salary')
plt.grid(color='#95a5a6', linestyle='--', linewidth=2, axis='y', alpha=0.7)
plt.xticks(rotation=45)

# Show the figure
plt.show()
```

EXERCISE 13: CREATE A PLOT USING MATPLOTLIB

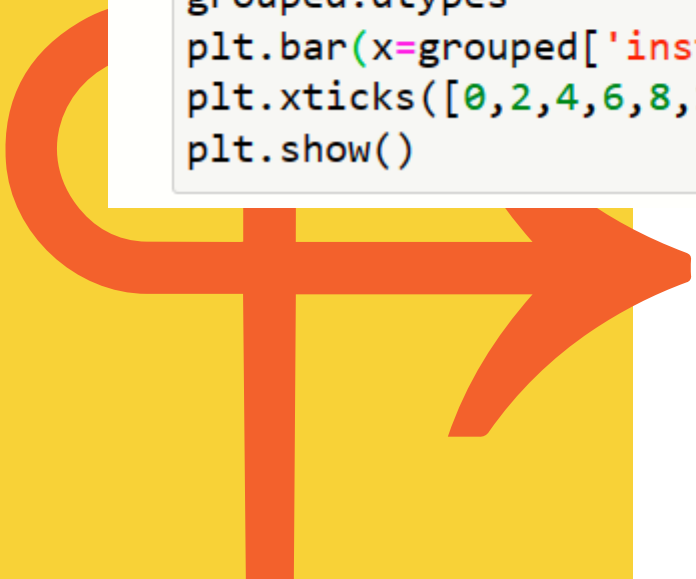
- Write the code to display a chart similar to the picture below
- Review the matplotlib help pages on the xticks to show 0 to 32 inclusive

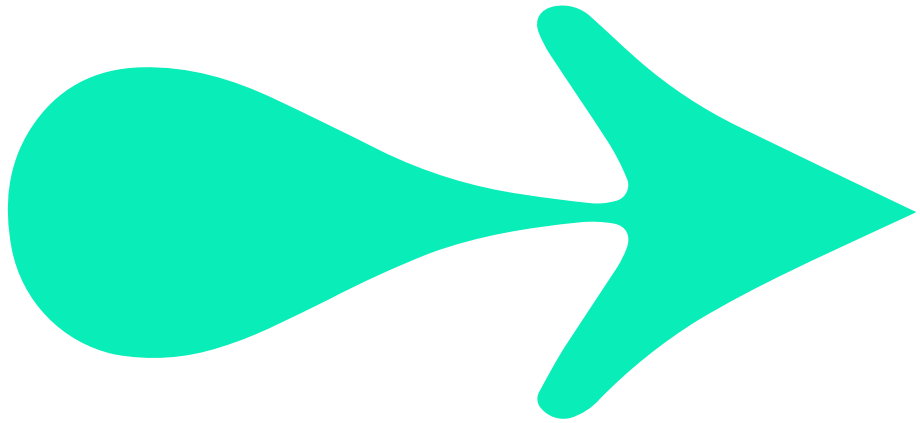


EXAMPLE CODE

```
import pandas as pd
import matplotlib.pyplot as plt
```

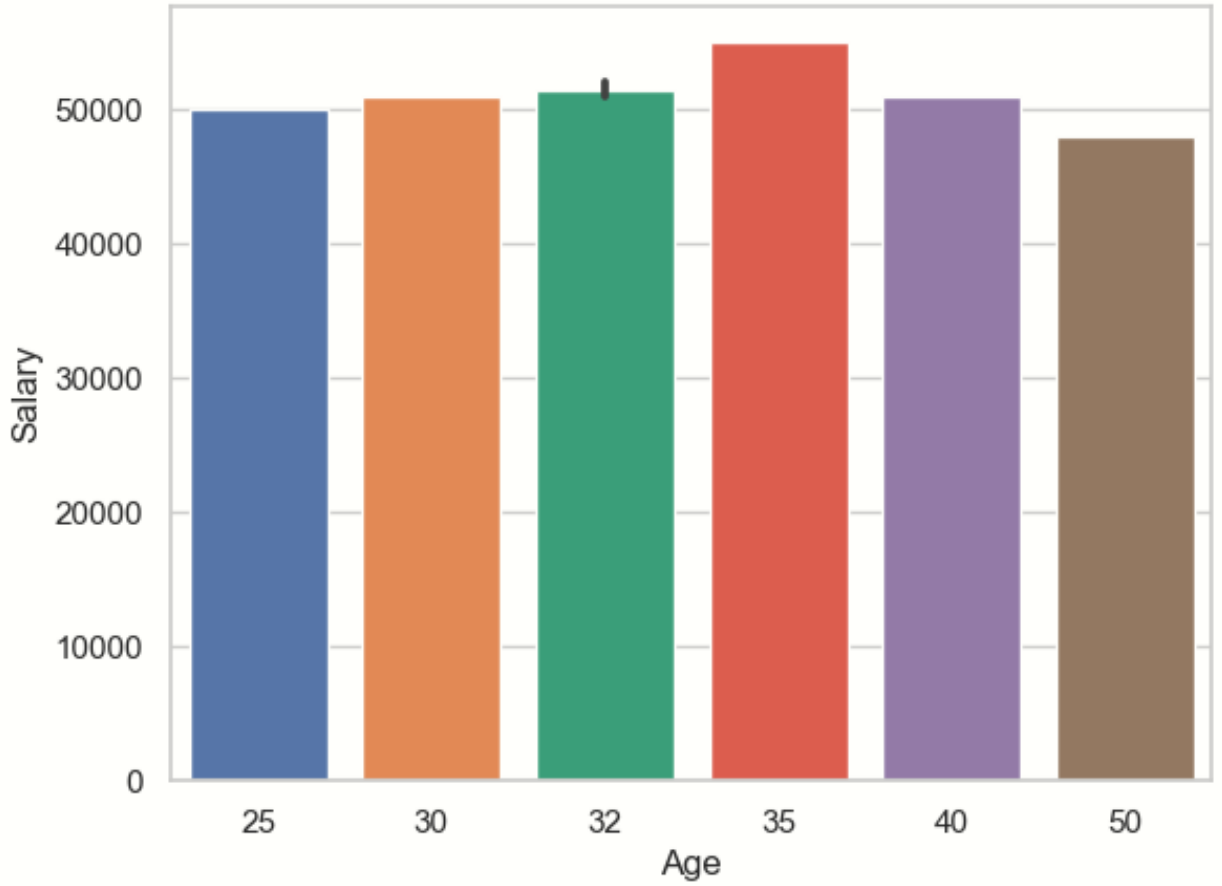
```
grouped = df_cycle_station.groupby(['install_dockcount']).size().reset_index(name='count')
grouped.dtypes
plt.bar(x=grouped['install_dockcount'],height=grouped['count'])
plt.xticks([0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32])
plt.show()
```





Using seaborn

```
import seaborn as sns

# Clear the plot area
plt.clf()
# Create a bar chart
ax = sns.barplot(x="Age", y="Salary", data=df)
plt.show()
```



EXERCISE 14:

USING SEABORN

- Clear the current plot
- Create a barplot within seaborn