Table of Contents

What You Need

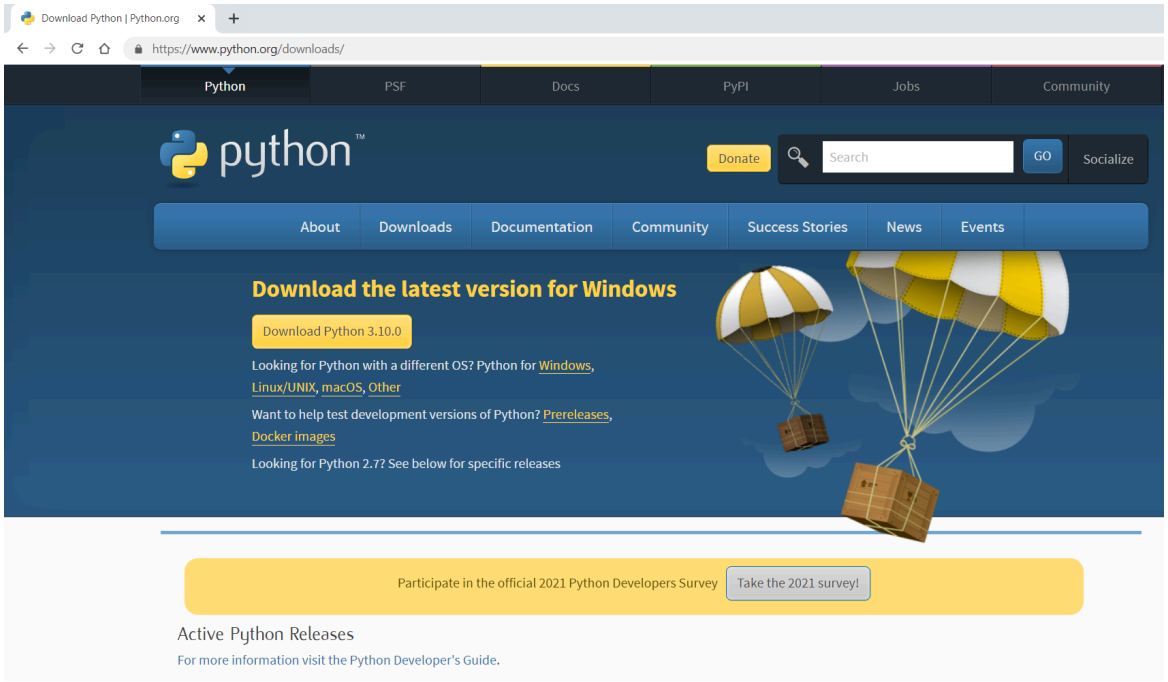
- **Python**: Head over to https://www.python.org/ and navigate to specific Downloads section for instructions based on your operating system. Download the latest stable release of Python for your computer and operating system. Verify the successful installation by entering 'python' in the command window terminal - a response from python interpreter should be received including its version number.
- **Jupyter Notebook**: The Jupyter notebook environment helps in analyzing the SAU data present in the S3 public bucket of SAU. Navigate to getting started with the classic Jupyter Notebook section in the official Jupyter website https://jupyter.org/install and install Jupyter Notebook as per the given instructions. If you are not already familiar with usage of Jupyter, refer to https://jupyter.readthedocs.io/en/latest/running.html#running for further details on how to run Jupyter Notebooks.
- **Anaconda:** Instead of installing the above tools individually, a popular alternative is to use open source Anaconda Individual Edition distribution that includes Python, R, necessary tools like Jupyter Notebooks, and commonly used packages such as Pandas and NumPy by default as part of its installation. Refer to https://www.anaconda.com/products/individual for further details.
- **Additional Packages**: Additional packages necessary for data analysis and visualizations can be installed by referring to the install instructions from official

sources. For example, pandas library for python can be installed using either conda install pandas or pip install pandas command as mentioned in https://pypi.org/project/pandas/. These packages can also be installed from within the notebooks as highlighted on top in the example notebooks. List of packages that need to be installed:pandas, numpy, ipywidgets and altair.

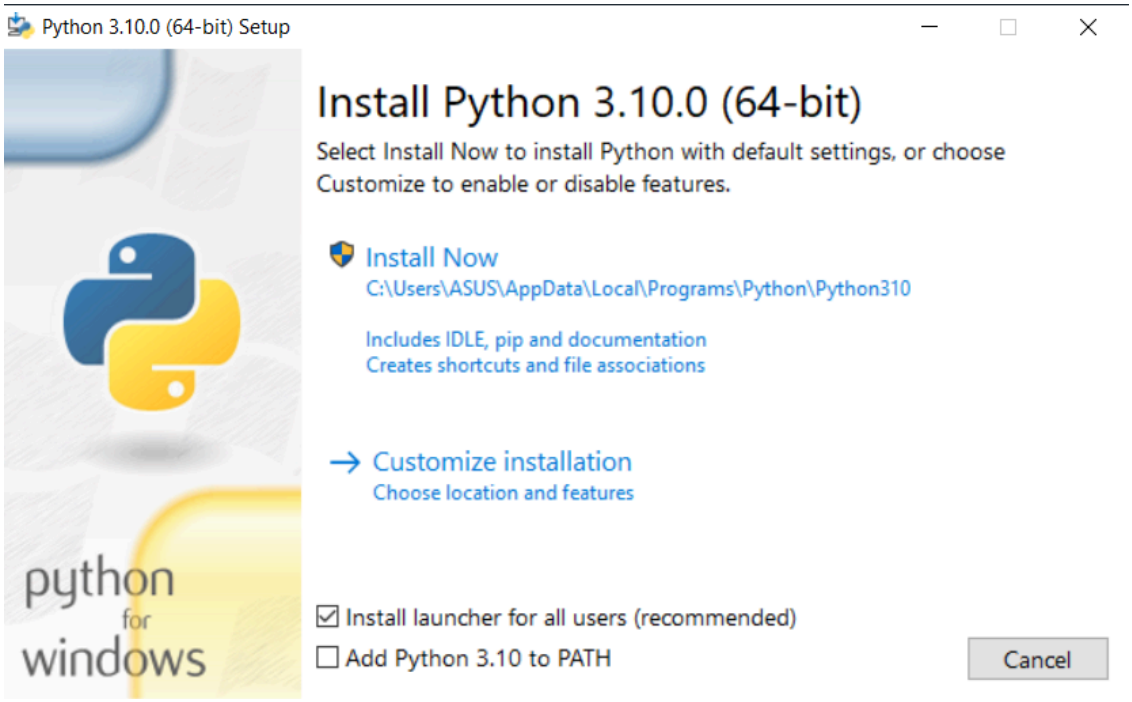
How to Install the Necessary Applications

Python

1. Go to https://www.python.org/ and head over to Downloads. Example here is for Windows operating system.

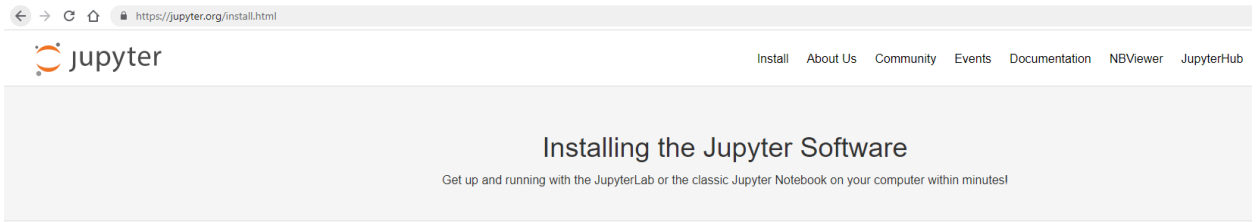


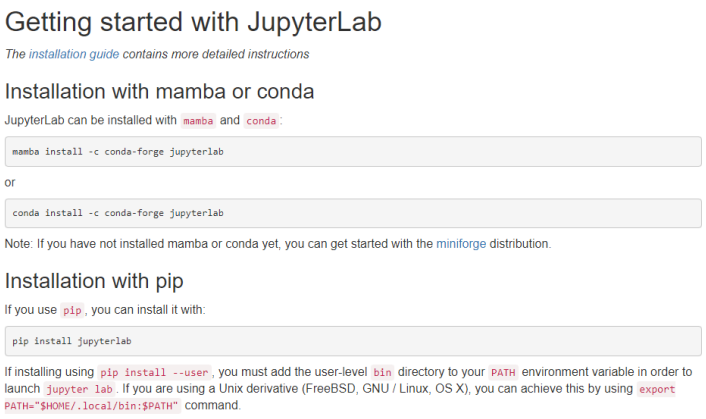
2. Download the latest Python by double clicking the downloaded installer and clicking install now.



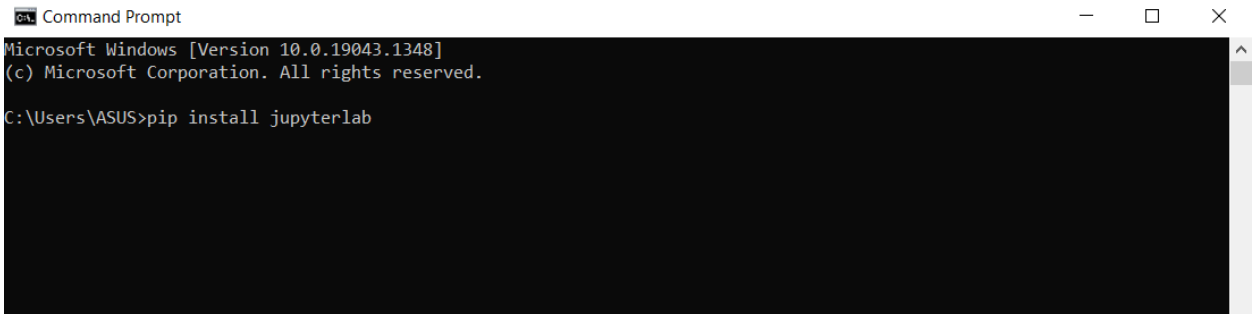
Jupyter Notebook

1. Go to https://jupyter.org/install



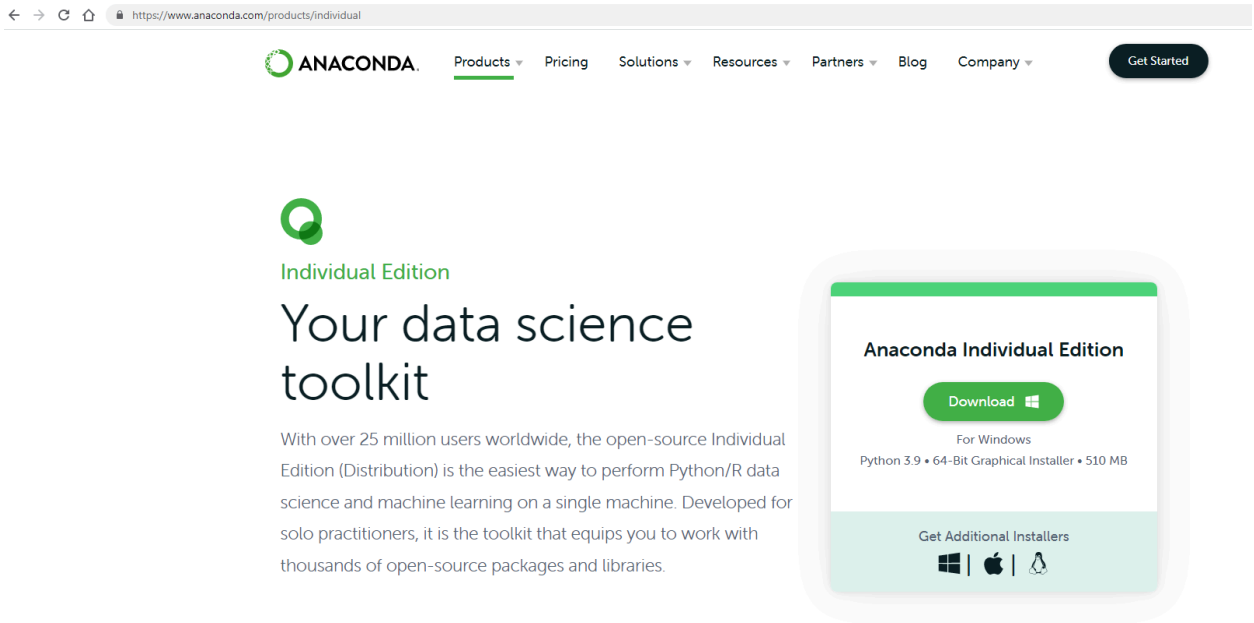


2. Install jupyterlab via mamba, conda or pip. Shown example was using pip.

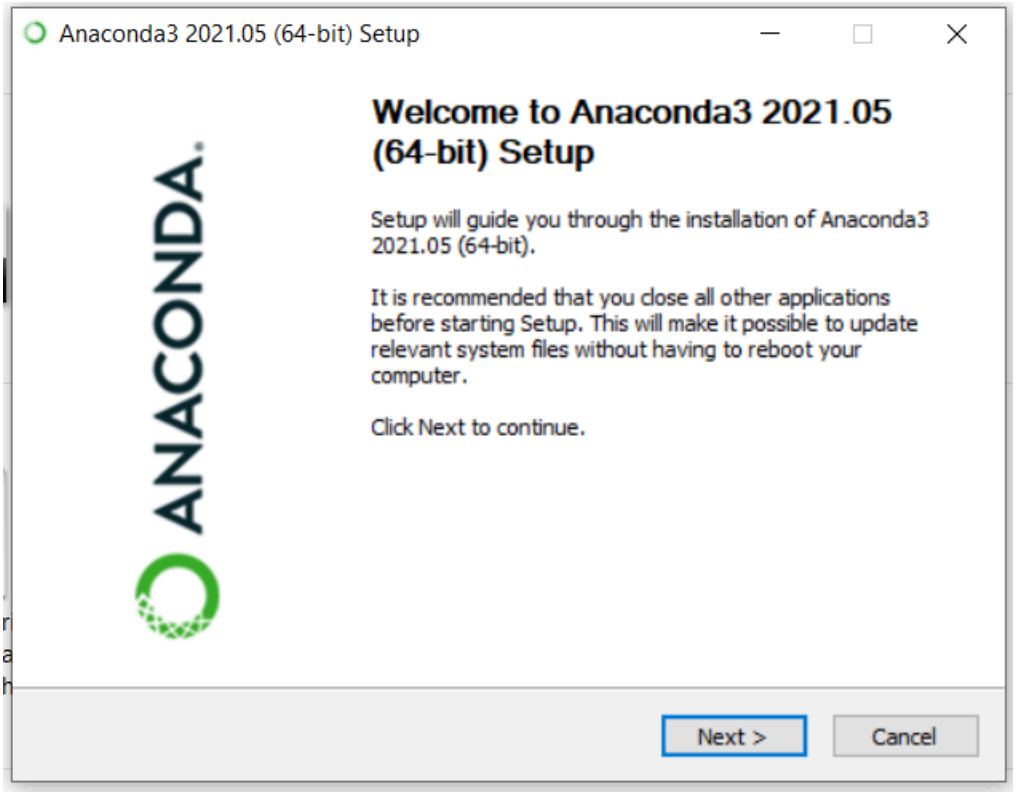


Anaconda

1. Go to https://www.anaconda.com/products/individual then download the installer for your operating system.



2. Double click the installer and follow the setup guide.



Additional Packages

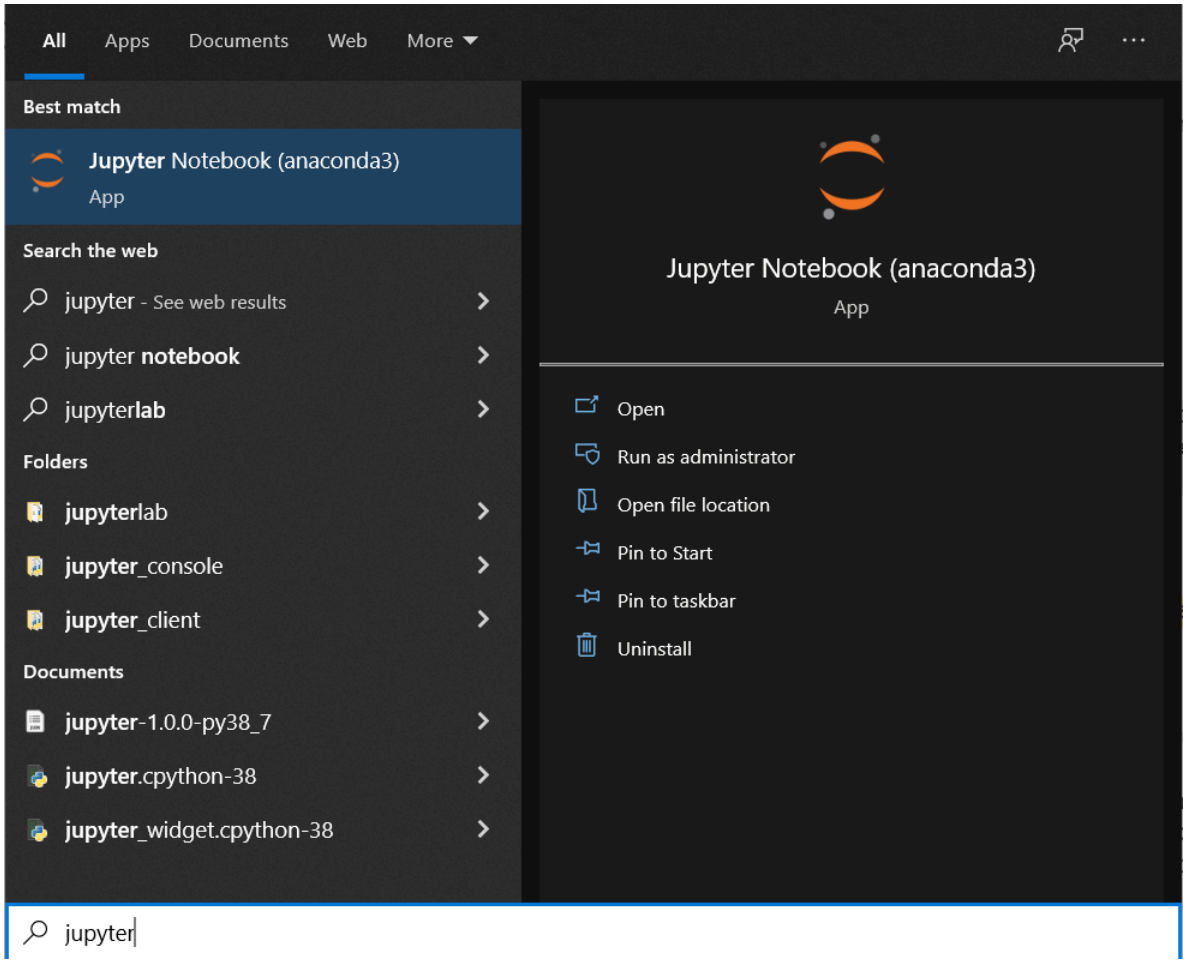
1. If you used Anaconda then you don't need to install pandas, numpy and ipywidgets separately but if you use other methods, you need to install them individually. Example shown here is using pip install. You can just change the name of the package you want to install.

Download the Jupyter Notebook

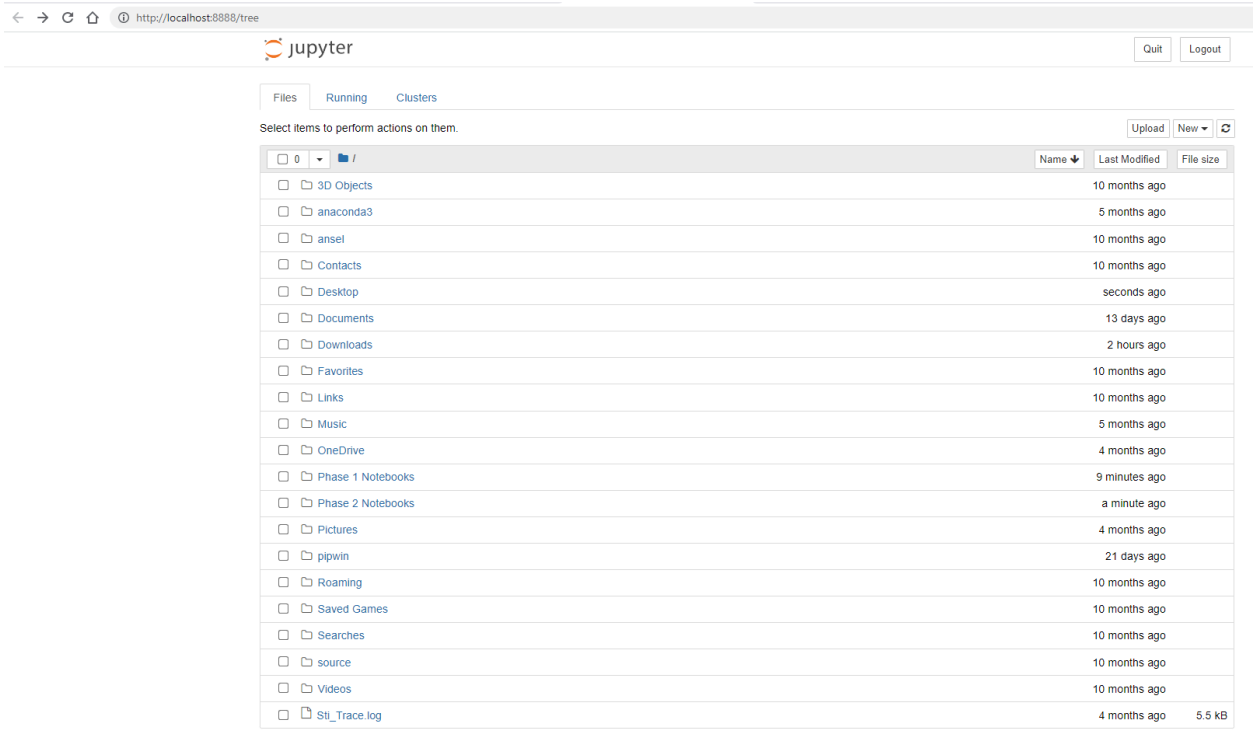
From Github?

How to Start Testing

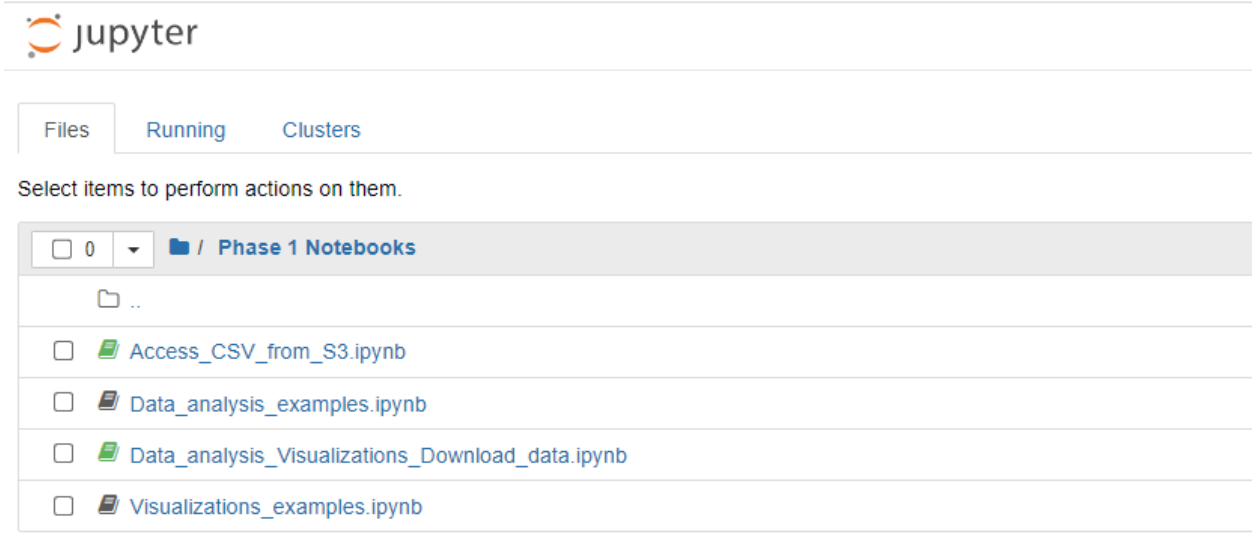
1. Open the Jupyter Notebook application on your computer.



2. It will open the Jupyter notebook in your browser. Identify the root directory of Jupyter, e.g. Drive C user folder.



3. Copy and paste the .ipynb files to your desired target directory. Example here was located inside Phase 1 Notebooks folder.



4. Open the notebook by double clicking the .ipynb file. It will open the file in a new tab.

5. Run the cells in order of appearance in the notebook. Refer to https://jupyter.readthedocs.io/en/latest/running.html#running for further details.
6. Refer to examples in the document reference to be guided in testing the notebooks: https://github.com/SeaAroundUs/sau-ubc-cic-catchdata#user-interface.

Data explanations

Concepts, definitions and methods: https://www.seaaroundus.org/tools-guide/
GitHub – Repository: https://github.com/SeaAroundUs/sau-ubc-cic-catchdata
See Data-Dictionnary folder.

Jupyter Notebook Files to Test

Access_CSV_from_S3

- Install and import pandas. If installed already, no need to install again.

```
In [ ]: # Uncomment and run this cell if Pandas library is not already installed

        # ! pip install --upgrade pip

        # ! pip install pandas
```

```
In [1]: # import library

        import pandas as pd
```

- Define CloudFront domain name (to access S3) and data

```
In [2]: # define CloudFront domain name (to access S3) and data

        cloudfront = 'https://d3cu2src083cxg.cloudfront.net'
        data_key = 'rfmo_query_result.csv'

        data_location = '{}/{}'.format(cloudfront, data_key)
```

- Loading the data and outputs row and column count

In [3]:
```
# df_rfmo = pd.read_csv(data_location) # if all the columns need to be loaded

# load select columns
df_rfmo = pd.read_csv(data_location,
                      usecols = ['year', 'name_rfmo', 'name_comm_group', 'name_fishing_entity', 'name_sector_type', 'catch_sum',

# rename columns as needed
df_rfmo.rename(columns = {"name_rfmo": "rfmo", "name_comm_group": "commercial_group", "name_fishing_entity": "fishing_entity", "
               inplace = True)

# print shape (rows, columns) of dataframe
df_rfmo.shape
```

Out[3]: (1027995, 7)

- Displaying sample data

In [4]:
```
# sample data

df_rfmo.head(3)
```

Out[4]:

	rfmo	year	commercial_group	fishing_entity	sector_type	catch_sum	real_value
0	NEAFC	1950	Other fishes & inverts	Belgium	Industrial	1082.915358	1.587554e+06
1	NEAFC	1950	Other fishes & inverts	Belgium	Industrial	112.044683	1.642575e+05
2	NEAFC	1950	Other fishes & inverts	Belgium	Industrial	1937.004283	2.839648e+06

- Displaying data information

In [5]:
```
# dataframe information

df_rfmo.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1027995 entries, 0 to 1027994
Data columns (total 7 columns):
 #   Column            Non-Null Count    Dtype
---  ------            --------------    -----
 0   rfmo              1027995 non-null  object
 1   year              1027995 non-null  int64
 2   commercial_group  1027995 non-null  object
 3   fishing_entity    1027995 non-null  object
 4   sector_type       1027995 non-null  object
 5   catch_sum         1027995 non-null  float64
 6   real_value        1027995 non-null  float64
dtypes: float64(2), int64(1), object(4)
memory usage: 54.9+ MB
```

- Data analysis example here is mean catch sum and real value in Iceland by year (after 2010). You can choose a different year and fishing entity.

Data Analysis example

Mean catch sum and real value in Iceland by year (after 2010)

In [6]:
```
df_rfmo[(df_rfmo['year'] > 2010) & (df_rfmo['fishing_entity'] == 'Iceland')].groupby('year', as_index = False).agg({'catch_sum':
```

Out[6]:

	year	catch_sum	real_value
0	2011	1138.40	1.172804e+08
1	2012	1357.82	2.524238e+08
2	2013	1297.89	2.108246e+08
3	2014	1033.93	1.146456e+08
4	2015	1579.63	2.522778e+08
5	2016	1232.25	1.652062e+08

Data_analysis_examples

- First five cells are the same with example number by importing pandas, loading data and displaying sample data and its information.

In [ ]:
```
# Uncomment and run this cell if Pandas library is not already installed

# ! pip install --upgrade pip

# ! pip install pandas
```

Import Libraries

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

Load data from S3

In [2]:
```
# define CloudFront domain name (to access S3) and data

cloudfront = 'https://d3cu2src083cxg.cloudfront.net'
data_key = 'rfmo_query_result.csv'

data_location = '{}/{}'.format(cloudfront, data_key)
```

In [3]:
```
# load select columns
df_rfmo = pd.read_csv(data_location, usecols = ['year', 'name_rfmo', 'name_comm_group', 'name_fishing_entity', 'name_sector_type

# rename columns as needed
df_rfmo.rename(columns = {"name_rfmo": "rfmo", "name_comm_group": "commercial_group", "name_fishing_entity": "fishing_entity", "
               inplace = True)

# print shape (rows, columns) of dataframe
df_rfmo.shape
```

Out[3]: (1027995, 7)

In [4]:
```
# sample data

df_rfmo.head(3)
```

Out[4]:

	rfmo	year	commercial_group	fishing_entity	sector_type	catch_sum	real_value
0	NEAFC	1950	Other fishes & inverts	Belgium	Industrial	1082.915358	1.587554e+06
1	NEAFC	1950	Other fishes & inverts	Belgium	Industrial	112.044683	1.642575e+05
2	NEAFC	1950	Other fishes & inverts	Belgium	Industrial	1937.004283	2.839648e+06

In [5]:
```
# dataframe information

df_rfmo.info()
```

- Function definition for data analysis scenarios

Data analysis

In [6]:
```
# Function definitions for simple data analysis scenarios

def average_catch_and_value_by_year(start_year = 2010, end_year = 2015):
    return df_rfmo[(df_rfmo['year'] >= start_year) & (df_rfmo['year'] <= end_year)].groupby('year', as_index = False).agg({'catch

def average_catch_and_value_by_region():
    return df_rfmo.groupby('fishing_entity', as_index = False)[['catch_sum', 'real_value']].mean().copy()

def average_catch_and_value_by_year_and_region(start_year = 2010, end_year = 2015, fishing_entity = 'Canada'):
    return df_rfmo[(df_rfmo['year'] >= start_year) & (df_rfmo['year'] <= end_year) & (df_rfmo['fishing_entity'] == fishing_entity

def catch_and_value_by_commercial_groups():
    return df_rfmo.groupby('commercial_group', as_index = False)[['catch_sum', 'real_value']].sum().copy()

def catch_and_value_by_commercial_groups_and_year(start_year = 2010, end_year = 2015):
    return df_rfmo[(df_rfmo['year'] >= start_year) & (df_rfmo['year'] <= end_year)].groupby(['year', 'commercial_group'], as_inde
```

- Displaying average catch sum and real value by year with start and end year parameters

Average catch sum & real value by year (parameters: start year, end year)

In [7]:
```
# Aggregate the data by Year, and display the result dataframe

df_table1 = average_catch_and_value_by_year(2005, 2010)
df_table1
```

Out[7]:

	year	catch_sum	real_value
0	2005	595.99	1.463770e+08
1	2006	550.30	1.081786e+08
2	2007	523.11	1.035868e+08
3	2008	488.62	1.027873e+08
4	2009	496.13	1.055332e+08
5	2010	471.13	8.674963e+07

- Displaying first 10 records of catch sum and real value by fisihing entity or region

Average catch sum & real value by fishing entity or region

In [8]:
```
# Aggregate the data by Fishing entity, and display the result dataframe

df_table2 = average_catch_and_value_by_region()
df_table2.head(10) # display only first 10 records
```

Out[8]:

	fishing_entity	catch_sum	real_value
0	Algeria	5.422545	7.949450e+03
1	Angola	905.784217	2.310087e+06
2	Argentina	0.044136	6.244362e+01
3	Azores Isl. (Portugal)	43.395882	5.108110e+04
4	Barbados	0.024249	1.358519e+01
5	Belgium	185.975452	9.072482e+05
6	Belize	25.080581	5.104515e+04
7	Brazil	169.085920	7.730256e+04
8	Bulgaria	84.628133	4.645546e+04
9	Canada	2.973976	1.354338e+03

- Displaying average catch sum and real value by year and fishing entity with start year, end year and region parameters

Average catch sum & real value by year and fishing entity (parameters: start year, end year, region)

In [9]:
```
# Aggregate the data by Year, filter the data to a Region, and display the result dataframe

df_table3 = average_catch_and_value_by_year_and_region(2010, 2015, 'Iceland')
df_table3
```

Out[9]:

	year	catch_sum	real_value
0	2010	1214.09	7.148157e+07
1	2011	1138.40	1.172804e+08
2	2012	1357.82	2.524238e+08
3	2013	1297.89	2.108246e+08
4	2014	1033.93	1.146456e+08
5	2015	1579.63	2.522778e+08

- Displaying total catch sum and real value of all commercial groups

Total catch sum & real value of all commercial groups

In [10]:
```
# Aggregate the data by Commercial group, and display the result dataframe

df_table4 = catch_and_value_by_commercial_groups()
df_table4
```

Out[10]:

	commercial_group	catch_sum	real_value
0	Anchovies	3.147476e+06	1.650981e+11
1	Cod-likes	2.645346e+08	5.743172e+13
2	Crustaceans	1.491285e+07	2.433204e+11
3	Flatfishes	3.200471e+07	5.282933e+11
4	Herring-likes	1.247234e+08	3.990954e+13
5	Molluscs	1.425190e+07	4.582541e+11
6	Other fishes & inverts	9.073546e+07	3.704571e+12
7	Perch-likes	7.088165e+07	9.046523e+12
8	Salmon, smelts, etc	7.130895e+07	5.658565e+13
9	Scorpionfishes	3.391901e+07	2.181282e+12
10	Sharks & rays	7.341301e+06	6.326200e+10
11	Tuna & billfishes	3.059225e+06	4.710837e+10

- Displaying catch sum and real value of all commercial groups by year with start and end year parameter

Total catch sum & real value of all commercial groups by year (parameters: start year, end year)

In [11]:
```
# Aggregate the data by Year and Commercial group, and display the result dataframe

df_table5 = catch_and_value_by_commercial_groups_and_year(2010, 2011)
df_table5
```

Out[11]:

	year	commercial_group	catch_sum	real_value
0	2010	Anchovies	1.933953e+04	4.019488e+08
1	2010	Cod-likes	3.081363e+06	5.602136e+11
2	2010	Crustaceans	2.390992e+05	3.289283e+09
3	2010	Flatfishes	3.416338e+05	3.186101e+09
4	2010	Herring-likes	1.969670e+06	9.452535e+11
5	2010	Molluscs	1.932697e+05	3.390055e+09
6	2010	Other fishes & inverts	1.537563e+06	3.746845e+10
7	2010	Perch-likes	1.277467e+06	1.281852e+11
8	2010	Salmon, smelts, etc	4.623862e+05	4.825351e+10
9	2010	Scorpionfishes	2.181623e+05	6.276247e+09
10	2010	Sharks & rays	5.463626e+04	7.432728e+08
11	2010	Tuna & billfishes	3.928175e+04	4.133071e+08
12	2011	Anchovies	3.056227e+04	9.004567e+08
13	2011	Cod-likes	2.131002e+06	3.792220e+11
14	2011	Crustaceans	2.248422e+05	2.928293e+09
15	2011	Flatfishes	3.181560e+05	2.788364e+09
16	2011	Herring-likes	1.327073e+06	4.288981e+11
17	2011	Molluscs	2.059345e+05	3.967590e+09
18	2011	Other fishes & inverts	1.564210e+06	5.187609e+10
19	2011	Perch-likes	1.190501e+06	1.198960e+11
20	2011	Salmon, smelts, etc	7.640422e+05	1.432698e+11
21	2011	Scorpionfishes	1.489860e+05	3.033474e+09
22	2011	Sharks & rays	6.456752e+04	1.059135e+09
23	2011	Tuna & billfishes	3.153618e+04	2.067479e+08

Visualizations_examples

- The first cell Installs and imports packages suchs as altair, ipywidgets, pandas and numpy.

Setup

In [1]:
```
# Uncomment this cell if necessary libraries are not already installed

# ! pip install --upgrade pip

# ! pip install altair
# ! pip install ipywidgets
# ! pip install pandas
# ! pip install numpy
```

Import Libraries

In [1]:
```
from IPython.display import display
from IPython.display import Javascript

import ipywidgets as widgets
import numpy as np
import pandas as pd
import altair as alt
```

In [2]:
```
# enable extension for widgets
! jupyter nbextension enable --py widgetsnbextension

# disable the default 5000 limit on number of rows
alt.data_transformers.disable_max_rows()

# enable renderer
alt.renderers.enable('default')
```

```
Enabling notebook extension jupyter-js-widgets/extension...
      - Validating: ok
```

Out[2]:
```
RendererRegistry.enable('default')
```

- Loading data and displaying sample data and information

Load data from S3

In [3]:
```
# define CloudFront domain name (to access S3) and data

cloudfront = 'https://d3cu2src083cxg.cloudfront.net'
data_key = 'rfmo_query_result.csv'

data_location = '{}/{}'.format(cloudfront, data_key)
```

In [5]:
```
# load select columns
df_rfmo = pd.read_csv(data_location, usecols = ['year', 'name_rfmo', 'name_comm_group', 'name_fishing_entity', 'name_sector_type

# rename columns as needed
df_rfmo.rename(columns = {"name_rfmo": "rfmo", "name_comm_group": "commercial_group", "name_fishing_entity": "fishing_entity", "
               inplace = True)

# print shape (rows & columns) of dataframe
df_rfmo.shape
```

Out[5]:
```
(1027995, 7)
```

In [6]:
```
# sample data

df_rfmo.head(3)
```

Out[6]:

	rfmo	year	commercial_group	fishing_entity	sector_type	catch_sum	real_value
0	NEAFC	1950	Other fishes & inverts	Belgium	Industrial	1082.915358	1.587554e+06
1	NEAFC	1950	Other fishes & inverts	Belgium	Industrial	112.044683	1.642575e+05
2	NEAFC	1950	Other fishes & inverts	Belgium	Industrial	1937.004283	2.839648e+06

In [7]:
```
# dataframe information

df_rfmo.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1027995 entries, 0 to 1027994
Data columns (total 7 columns):
 #   Column            Non-Null Count    Dtype
---  ------            --------------    -----
 0   rfmo              1027995 non-null  object
 1   year              1027995 non-null  int64
 2   commercial_group  1027995 non-null  object
 3   fishing_entity    1027995 non-null  object
 4   sector_type       1027995 non-null  object
 5   catch_sum         1027995 non-null  float64
 6   real_value        1027995 non-null  float64
dtypes: float64(2), int64(1), object(4)
memory usage: 54.9+ MB
```

- Defining variables and functions for chart 1

Visualizations ¶

Scripts for widgets, variables, and charts

In [4]:
```
# Chart 1

# get unique list of years from the dataframe
years = list(df_rfmo['year'].unique())

# get unique list of regions from the dataframe
regions = list(df_rfmo['fishing_entity'].unique())

# define start year dropdown for chart 1
start_year_dropdown1 = widgets.Dropdown(
    options = years,
    value = 2010,
    description = 'Start Year :',
    disabled = False,
)

# define end year dropdown for chart 1
end_year_dropdown1 = widgets.Dropdown(
    options = years,
    value = 2015,
    description = 'End Year :',
    disabled = False,
)

# define region dropdown for chart 1
region_dropdown1 = widgets.Dropdown(
    options = regions,
    value = 'Iceland',
    description = 'Region',
    disabled = False,
)
```

- Defining variables and functions for chart 2

In [9]:
```
# Chart 2

# get unique list of years from the dataframe
years = list(df_rfmo['year'].unique())

# get unique list of regions from the dataframe
regions = list(df_rfmo['fishing_entity'].unique())

# define start year dropdown for chart 2
start_year_dropdown2 = widgets.Dropdown(
    options = years,
    value = 2010,
    description = 'Start Year :',
    disabled = False,
)

# define end year dropdown for chart 2
end_year_dropdown2 = widgets.Dropdown(
    options = years,
    value = 2015,
    description = 'End Year :',
    disabled = False,
)

# define region dropdown for chart 2
region_dropdown2 = widgets.Dropdown(
    options = regions,
    value = 'Iceland',
    description = 'Region',
    disabled = False,
)

def display_chart_2_filters():

    # display both start year and end year dropdowns, and also region
    display(start_year_dropdown2)
    display(end_year_dropdown2)
    display(region_dropdown2)

def display_chart_2():
```

- Defining variables and functions for chart 3

In [10]:

```
# Chart 3

# get unique list of years from the dataframe
years = list(df_rfmo['year'].unique())

# get unique list of regions from the dataframe
regions = list(df_rfmo['fishing_entity'].unique())

# define start year dropdown for chart 3
start_year_dropdown3 = widgets.Dropdown(
    options = years,
    value = 2010,
    description = 'Start Year :',
    disabled = False,
)

# define end year dropdown for chart 3
end_year_dropdown3 = widgets.Dropdown(
    options = years,
    value = 2015,
    description = 'End Year :',
    disabled = False,
)

# define region dropdown for chart 3
region_dropdown3 = widgets.Dropdown(
    options = regions,
    value = 'Iceland',
    description = 'Region',
    disabled = False,
)

def display_chart_3_filters():

    # display both start year and end year dropdowns, and also region
    display(start_year_dropdown3)
    display(end_year_dropdown3)
    display(region_dropdown3)
```

- Defining variables and functions for chart 4

In [11]:

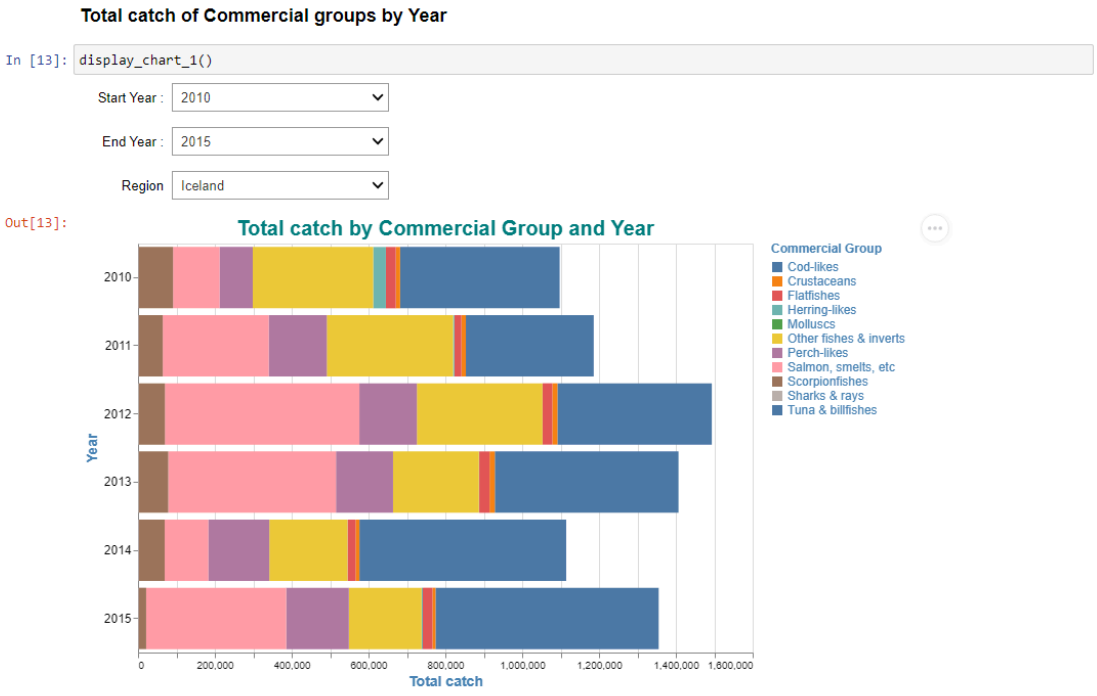
```
# Chart 4

# get unique list of years from the dataframe
years = list(df_rfmo['year'].unique())

# get unique list of regions from the dataframe
regions = list(df_rfmo['fishing_entity'].unique())

# define start year dropdown for chart 4
start_year_dropdown4 = widgets.Dropdown(
    options = years,
    value = 2000,
    description = 'Start Year :',
    disabled = False,
)

# define end year dropdown for chart 4
end_year_dropdown4 = widgets.Dropdown(
    options = years,
    value = 2015,
    description = 'End Year :',
    disabled = False,
)

# define region dropdown for chart 4
region_dropdown4 = widgets.Dropdown(
    options = regions,
    value = 'Iceland',
    description = 'Region',
    disabled = False,
)

# define radio button to select the measure for chart 4
measure_chart_4 = widgets.RadioButtons(
    options = [('Total Catch', 'catch_sum'), ('Real Value', 'real_value')],
    value = 'catch_sum',
    description = 'Measure :',
    disabled = False
)

def display_chart_4_filters():

    # display both start year and end year dropdowns, region and measure selection
    display(start_year_dropdown4)
    display(end_year_dropdown4)
    display(region_dropdown4)
```

- Defining variables and functions for chart 5

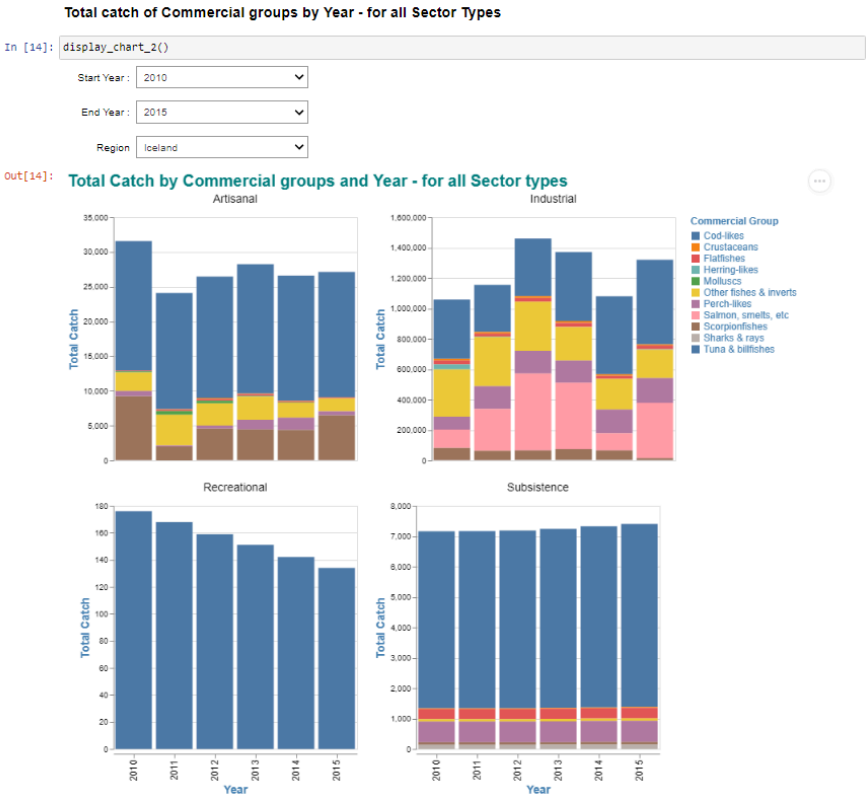
In [12]:

```
# Chart 5

# get unique list of years from the dataframe
years = list(df_rfmo['year'].unique())

# get unique list of regions from the dataframe
regions = list(df_rfmo['fishing_entity'].unique())

# define start year dropdown for chart 5
start_year_dropdown5 = widgets.Dropdown(
    options = years,
    value = 1950,
    description = 'Start Year :',
    disabled = False,
)

# define end year dropdown for chart 5
end_year_dropdown5 = widgets.Dropdown(
    options = years,
    value = 2015,
    description = 'End Year :',
    disabled = False,
)

# define region dropdown for chart 5
region_dropdown5 = widgets.Dropdown(
    options = regions,
    value = 'Iceland',
    description = 'Region',
    disabled = False,
)

# define radio button to select the measure for chart 5
measure_chart_5 = widgets.RadioButtons(
    options = [('Total Catch', 'catch_sum'), ('Real Value', 'real_value')],
    value = 'catch_sum',
    description = 'Measure :',
    disabled = False
)

def display_chart_5_filters():

    # display both start year and end year dropdowns, region and measure selection
    display(start_year_dropdown5)
    display(end_year_dropdown5)
    display(region_dropdown5)
```

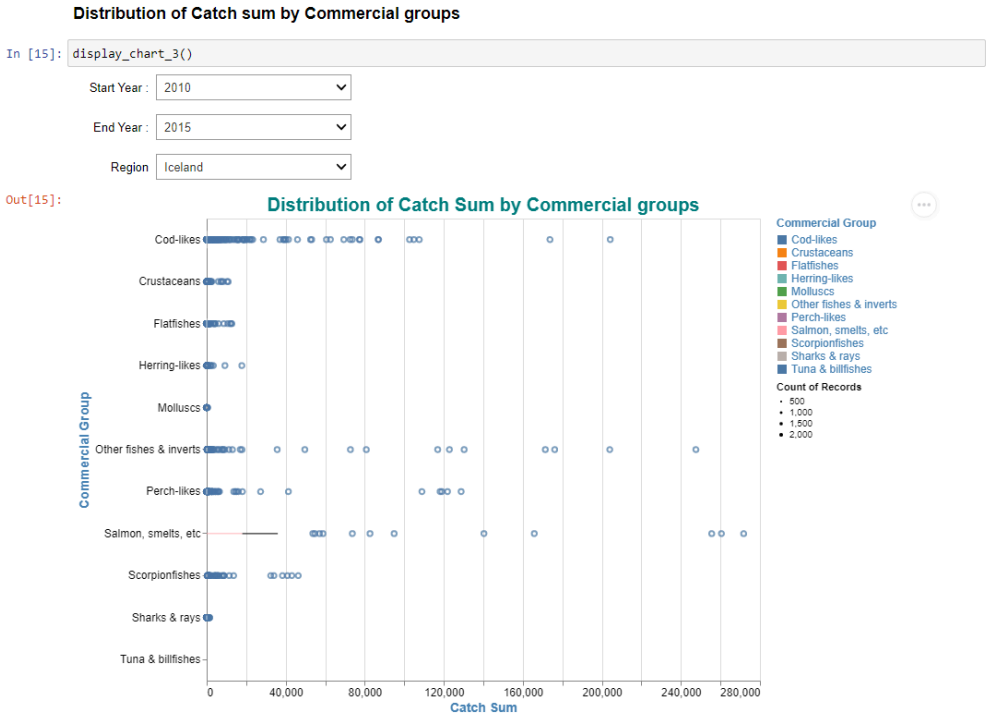
- Displaying total catch of commercial groups by year with 2010 as start year, 2015 as end year and Iceland as region (chart 1)



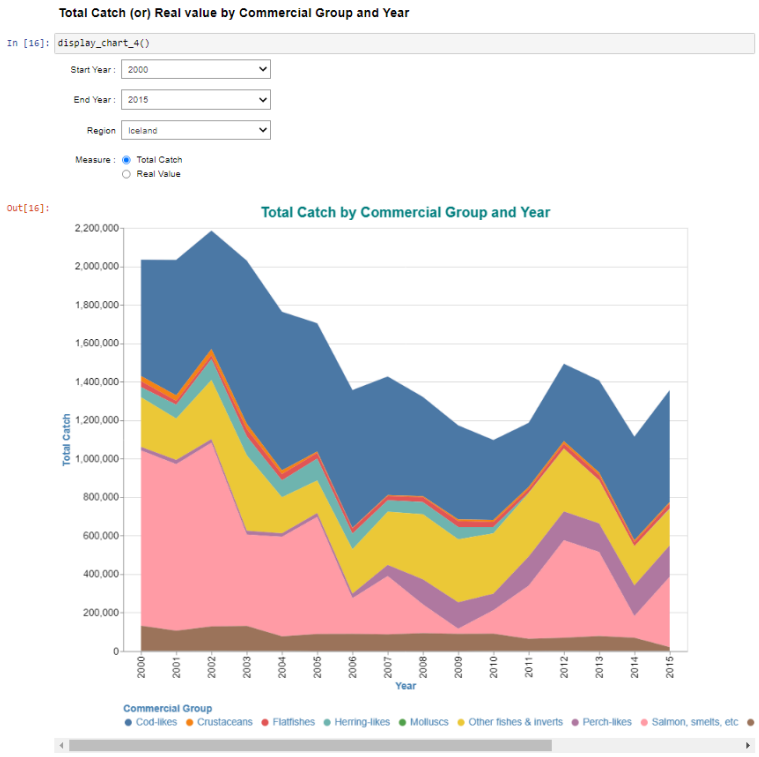
- Displaying total catch of commercial groups by year for all sectors with 2010 as start year, 2015 as end year and Iceland as region



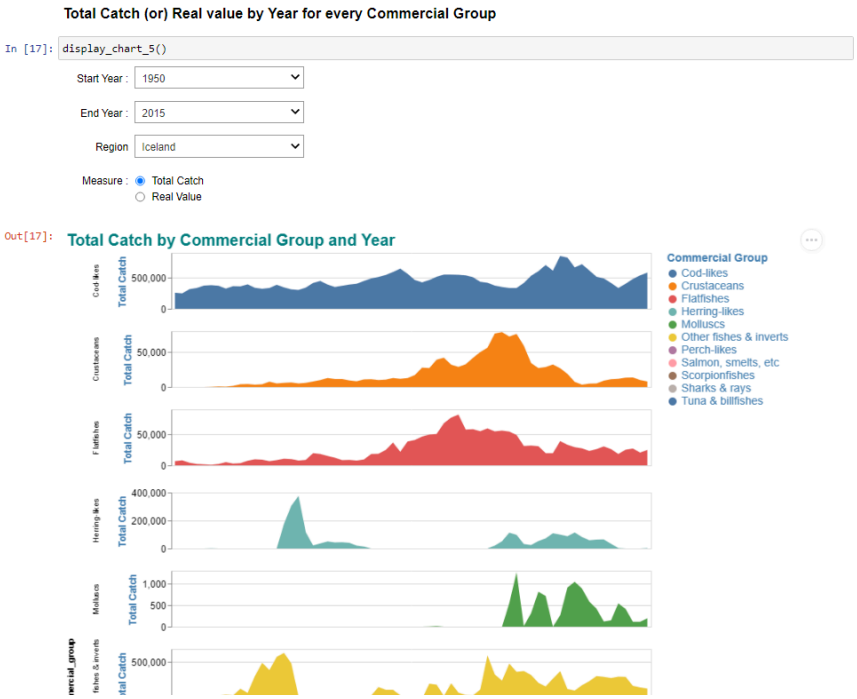
- Displaying distribution of catch sum by commercial groups with 2010 as start year, 2015 as end year and Iceland as region



- Displaying total catch or real value by commercial group and year with 2000 as start year, 2015 as end year, Iceland as region and total catch as measure



- Displaying total catch or real value by year for every commercial group and year with 2000 as start year, 2015 as end year, Iceland as region and total catch as measure



[Data_analysis_visualizations_download_data]

- Installs and imports packages suchs as altair, ipywidgets, pandas and numpy.

Setup

In [1]:
```
# Uncomment this cell if necessary libraries are not already installed

# ! pip install --upgrade pip

# ! pip install altair
# ! pip install ipywidgets
# ! pip install pandas
# ! pip install numpy
```

Import Libraries

In [1]:
```
from IPython.display import display
from IPython.display import Javascript

import ipywidgets as widgets
import numpy as np
import pandas as pd
import altair as alt
```

In [2]:
```
# enable extension for widgets
! jupyter nbextension enable --py widgetsnbextension

# disable the default 5000 limit on number of rows
alt.data_transformers.disable_max_rows()

# enable renderer
alt.renderers.enable('default')
```

```
Enabling notebook extension jupyter-js-widgets/extension...
      - Validating: ok
```

Out[2]:
```
RendererRegistry.enable('default')
```

- Loading data and displaying sample data and information

Load data from S3

In [3]:
```
# define CloudFront domain name (to access S3) and data

cloudfront = 'https://d3cu2src083cxg.cloudfront.net'
data_key = 'rfmo_query_result.csv'

data_location = '{}/{}'.format(cloudfront, data_key)
```

In [5]:
```
# load select columns
df_rfmo = pd.read_csv(data_location, usecols = ['year', 'name_rfmo', 'name_comm_group', 'name_fishing_entity', 'name_sector_type

# rename columns as needed
df_rfmo.rename(columns = {"name_rfmo": "rfmo", "name_comm_group": "commercial_group", "name_fishing_entity": "fishing_entity", "
               inplace = True)

# print shape (rows & columns) of dataframe
df_rfmo.shape
```

Out[5]: (1027995, 7)

In [6]:
```
# sample data

df_rfmo.head(3)
```

Out[6]:

	rfmo	year	commercial_group	fishing_entity	sector_type	catch_sum	real_value
0	NEAFC	1950	Other fishes & inverts	Belgium	Industrial	1082.915358	1.587554e+06
1	NEAFC	1950	Other fishes & inverts	Belgium	Industrial	112.044683	1.642575e+05
2	NEAFC	1950	Other fishes & inverts	Belgium	Industrial	1937.004283	2.839648e+06

In [7]:
```
# dataframe information

df_rfmo.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1027995 entries, 0 to 1027994
Data columns (total 7 columns):
 #   Column            Non-Null Count    Dtype
---  ------            --------------    -----
 0   rfmo              1027995 non-null  object
 1   year              1027995 non-null  int64
 2   commercial_group  1027995 non-null  object
 3   fishing_entity    1027995 non-null  object
 4   sector_type       1027995 non-null  object
 5   catch_sum         1027995 non-null  float64
 6   real_value        1027995 non-null  float64
dtypes: float64(2), int64(1), object(4)
memory usage: 54.9+ MB
```

- Defines javascript function for downloading data as csv file

Java script for downloading data

In [8]:
```
# java script function to download a dataframe into csv

def download_csv(dataframe_to_download, output_filename = 'results.csv'):

    js_download = """
    var csv = '%s';
    var filename = '%s';

    var blob = new Blob([csv], { type: 'text/csv;charset=utf-8;' });
    if (navigator.msSaveBlob) { // IE 10+
        navigator.msSaveBlob(blob, filename);
    } else {
        var link = document.createElement("a");
        if (link.download !== undefined) { // feature detection
            // Browsers that support HTML5 download attribute
            var url = URL.createObjectURL(blob);
            link.setAttribute("href", url);
            link.setAttribute("download", filename);
            link.style.visibility = 'hidden';
            document.body.appendChild(link);
            link.click();
            document.body.removeChild(link);
        }
    }
    """ % (dataframe_to_download.to_csv(index = False).replace('\n','\\n').replace("'","\'").replace('\r',''), output_filename)

    return Javascript(js_download)
```

- Test downloading data

In [9]:
```
# example for using java script to download dataframe

df_test = df_rfmo[(df_rfmo['year'] >= 2010) & (df_rfmo['year'] <= 2015)].groupby('year', as_index = False).agg({'catch_sum':'mean

download_csv(df_test)
```

Out[9]:

- Downloading data with new_file_name as file name

In [10]:
```
# output file name can be defined while calling this function as shown here

download_csv(df_test, 'new_file_name.csv')
```

Out[10]:

- Function definition for data analysis scenarios

Data analysis

In [11]:
```
# Function definitions for simple data analysis scenarios

def average_catch_and_value_by_year(start_year = 2010, end_year = 2015):
    return df_rfmo[(df_rfmo['year'] >= start_year) & (df_rfmo['year'] <= end_year)].groupby('year', as_index = False).agg({'catch

def average_catch_and_value_by_region():
    return df_rfmo.groupby('fishing_entity', as_index = False)[['catch_sum', 'real_value']].mean().copy()

def average_catch_and_value_by_year_and_region(start_year = 2010, end_year = 2015, fishing_entity = 'Canada'):
    return df_rfmo[(df_rfmo['year'] >= start_year) & (df_rfmo['year'] <= end_year) & (df_rfmo['fishing_entity'] == fishing_entity

def catch_and_value_by_commercial_groups():
    return df_rfmo.groupby('commercial_group', as_index = False)[['catch_sum', 'real_value']].sum().copy()

def catch_and_value_by_commercial_groups_and_year(start_year = 2010, end_year = 2015):
    return df_rfmo[(df_rfmo['year'] >= start_year) & (df_rfmo['year'] <= end_year)].groupby(['year', 'commercial_group'], as_inde
```

- Displaying average catch sum and real value by year with start and end year parameters and downloading data as csv

Average catch sum & real value by year (parameters: start year, end year)

In [12]:
```
# Aggregate the data by Year, and display the result dataframe

df_table1 = average_catch_and_value_by_year(2005, 2010)
df_table1
```

Out[12]:

	year	catch_sum	real_value
0	2005	595.99	1.463770e+08
1	2006	550.30	1.081786e+08
2	2007	523.11	1.035868e+08
3	2008	488.62	1.027873e+08
4	2009	496.13	1.055332e+08
5	2010	471.13	8.674963e+07

In [13]:
```
# download the dataframe into a csv

download_csv(df_table1, 'table1.csv')
```

Out[13]:

- Displaying first 10 records of catch sum and real value by fishing entity or region and downloading data as csv

Average catch sum & real value by fishing entity or region

In [14]:
```
# Aggregate the data by Fishing entity, and display the result dataframe

df_table2 = average_catch_and_value_by_region()
df_table2.head(10)
```

Out[14]:

	fishing_entity	catch_sum	real_value
0	Algeria	5.422545	7.949450e+03
1	Angola	905.784217	2.310087e+06
2	Argentina	0.044136	6.244362e+01
3	Azores Isl. (Portugal)	43.395882	5.108110e+04
4	Barbados	0.024249	1.358519e+01
5	Belgium	185.975452	9.072482e+05
6	Belize	25.080581	5.104515e+04
7	Brazil	169.085920	7.730256e+04
8	Bulgaria	84.628133	4.645546e+04
9	Canada	2.973976	1.354338e+03

In [15]:
```
# download the dataframe into a csv

download_csv(df_table2, 'table2.csv')
```

Out[15]:

- Displaying average catch sum and real value by year and fishing entity with start year, end year and region parameters and downloading data as csv

Average catch sum & real value by year and fishing entity (parameters: start year, end year, region)

In [16]:
```
# Aggregate the data by Year, filter the data to a Region, and display the result dataframe

df_table3 = average_catch_and_value_by_year_and_region(2010, 2015, 'Iceland')
df_table3
```

Out[16]:

	year	catch_sum	real_value
0	2010	1214.09	7.148157e+07
1	2011	1138.40	1.172804e+08
2	2012	1357.82	2.524238e+08
3	2013	1297.89	2.108246e+08
4	2014	1033.93	1.146456e+08
5	2015	1579.63	2.522778e+08

In [17]:
```
# download the dataframe into a csv

download_csv(df_table3, 'table3.csv')
```

Out[17]:

- Displaying total catch sum and real value of all commercial groups and downloading data as csv

Total catch sum & real value of all commercial groups

In [18]:
```
# Aggregate the data by Commercial group, and display the result dataframe

df_table4 = catch_and_value_by_commercial_groups()
df_table4
```

Out[18]:

	commercial_group	catch_sum	real_value
0	Anchovies	3.147476e+06	1.650981e+11
1	Cod-likes	2.645346e+08	5.743172e+13
2	Crustaceans	1.491285e+07	2.433204e+11
3	Flatfishes	3.200471e+07	5.282933e+11
4	Herring-likes	1.247234e+08	3.990954e+13
5	Molluscs	1.425190e+07	4.582541e+11
6	Other fishes & inverts	9.073546e+07	3.704571e+12
7	Perch-likes	7.088165e+07	9.046523e+12
8	Salmon, smelts, etc	7.130895e+07	5.658565e+13
9	Scorpionfishes	3.391901e+07	2.181282e+12
10	Sharks & rays	7.341301e+06	6.326200e+10
11	Tuna & billfishes	3.059225e+06	4.710837e+10

In [19]:
```
# download the dataframe into a csv

download_csv(df_table4, 'table4.csv')
```

Out[19]:

- Displaying catch sum and real value of all commercial groups by year with start and end year parameter and downloading data as csv

Total catch sum & real value of all commercial groups by year (parameters: start year, end year)

In [20]:
```
# Aggregate the data by Year and Commercial group, and display the result dataframe

df_table5 = catch_and_value_by_commercial_groups_and_year(2010, 2011)
df_table5
```

Out[20]:

	year	commercial_group	catch_sum	real_value
0	2010	Anchovies	1.933953e+04	4.019488e+08
1	2010	Cod-likes	3.081363e+06	5.602136e+11
2	2010	Crustaceans	2.390992e+05	3.289283e+09
3	2010	Flatfishes	3.416338e+05	3.186101e+09
4	2010	Herring-likes	1.969670e+06	9.452535e+11
5	2010	Molluscs	1.932697e+05	3.390055e+09
6	2010	Other fishes & inverts	1.537563e+06	3.746845e+10
7	2010	Perch-likes	1.277467e+06	1.281852e+11
8	2010	Salmon, smelts, etc	4.623862e+05	4.825351e+10
9	2010	Scorpionfishes	2.181623e+05	6.276247e+09
10	2010	Sharks & rays	5.463626e+04	7.432728e+08
11	2010	Tuna & billfishes	3.928175e+04	4.133071e+08
12	2011	Anchovies	3.056227e+04	9.004567e+08
13	2011	Cod-likes	2.131002e+06	3.792220e+11
14	2011	Crustaceans	2.248422e+05	2.928293e+09
15	2011	Flatfishes	3.181560e+05	2.788364e+09
16	2011	Herring-likes	1.327073e+06	4.288981e+11
17	2011	Molluscs	2.059345e+05	3.967590e+09
18	2011	Other fishes & inverts	1.564210e+06	5.187609e+10
19	2011	Perch-likes	1.190501e+06	1.198960e+11
20	2011	Salmon, smelts, etc	7.640422e+05	1.432698e+11
21	2011	Scorpionfishes	1.489860e+05	3.033474e+09
22	2011	Sharks & rays	6.456752e+04	1.059135e+09
23	2011	Tuna & billfishes	3.153618e+04	2.067479e+08

In [21]:
```
# download the dataframe into a csv

download_csv(df_table5, 'table5.csv')
```

Out[21]:

- Defining variables and functions for chart 1 and download function

Visualizations

Scripts for widgets, variables, and charts

In [22]:
```
# Chart 1

# get unique list of years from the dataframe
years = list(df_rfmo['year'].unique())

# get unique list of regions from the dataframe
regions = list(df_rfmo['fishing_entity'].unique())

# define start year dropdown for chart 1
start_year_dropdown1 = widgets.Dropdown(
    options = years,
    value = 2010,
    description = 'Start Year :',
    disabled = False,
)

# define end year dropdown for chart 1
end_year_dropdown1 = widgets.Dropdown(
    options = years,
    value = 2015,
    description = 'End Year :',
    disabled = False,
)

# define region dropdown for chart 1
region_dropdown1 = widgets.Dropdown(
    options = regions,
    value = 'Iceland',
    description = 'Region',
    disabled = False,
)
```

- Defining variables and functions for chart 2 and download function

In [23]:

```
# Chart 2

# get unique list of years from the dataframe
years = list(df_rfmo['year'].unique())

# get unique list of regions from the dataframe
regions = list(df_rfmo['fishing_entity'].unique())

# define start year dropdown for chart 2
start_year_dropdown2 = widgets.Dropdown(
    options = years,
    value = 2010,
    description = 'Start Year :',
    disabled = False,
)

# define end year dropdown for chart 2
end_year_dropdown2 = widgets.Dropdown(
    options = years,
    value = 2015,
    description = 'End Year :',
    disabled = False,
)

# define region dropdown for chart 2
region_dropdown2 = widgets.Dropdown(
    options = regions,
    value = 'Iceland',
    description = 'Region',
    disabled = False,
)

def display_chart_2_filters():

    # display both start year and end year dropdowns, and also region
    display(start_year_dropdown2)
    display(end_year_dropdown2)
    display(region_dropdown2)

def display_chart_2():
```

- Defining variables and functions for chart 3 and download function

In [24]:

```
# Chart 3

# get unique list of years from the dataframe
years = list(df_rfmo['year'].unique())

# get unique list of regions from the dataframe
regions = list(df_rfmo['fishing_entity'].unique())

# define start year dropdown for chart 3
start_year_dropdown3 = widgets.Dropdown(
    options = years,
    value = 2010,
    description = 'Start Year :',
    disabled = False,
)

# define end year dropdown for chart 3
end_year_dropdown3 = widgets.Dropdown(
    options = years,
    value = 2015,
    description = 'End Year :',
    disabled = False,
)

# define region dropdown for chart 3
region_dropdown3 = widgets.Dropdown(
    options = regions,
    value = 'Iceland',
    description = 'Region',
    disabled = False,
)

def display_chart_3_filters():

    # display both start year and end year dropdowns, and also region
    display(start_year_dropdown3)
    display(end_year_dropdown3)
    display(region_dropdown3)

def display_chart_3():

    # call the function to display both the start year and end year dropdowns
    display_chart_3_filters()

    # get values selected in the start year and end year dropdowns, and also region
    start_year = start_year_dropdown3.value
```

- Defining variables and functions for chart 4 and download function

In [25]:

```
# Chart 4

# get unique list of years from the dataframe
years = list(df_rfmo['year'].unique())

# get unique list of regions from the dataframe
regions = list(df_rfmo['fishing_entity'].unique())

# define start year dropdown for chart 4
start_year_dropdown4 = widgets.Dropdown(
    options = years,
    value = 2000,
    description = 'Start Year :',
    disabled = False,
)

# define end year dropdown for chart 4
end_year_dropdown4 = widgets.Dropdown(
    options = years,
    value = 2015,
    description = 'End Year :',
    disabled = False,
)

# define region dropdown for chart 4
region_dropdown4 = widgets.Dropdown(
    options = regions,
    value = 'Iceland',
    description = 'Region',
    disabled = False,
)

# define radio button to select the measure for chart 4
measure_chart_4 = widgets.RadioButtons(
    options = [('Total Catch', 'catch_sum'), ('Real Value', 'real_value')],
    value = 'catch_sum',
    description = 'Measure :',
    disabled = False
)

def display_chart_4_filters():
```

- Defining variables and functions for chart 5 and download function

In [26]:

```
# Chart 5

# get unique list of years from the dataframe
years = list(df_rfmo['year'].unique())

# get unique list of regions from the dataframe
regions = list(df_rfmo['fishing_entity'].unique())

# define start year dropdown for chart 5
start_year_dropdown5 = widgets.Dropdown(
    options = years,
    value = 1950,
    description = 'Start Year :',
    disabled = False,
)

# define end year dropdown for chart 5
end_year_dropdown5 = widgets.Dropdown(
    options = years,
    value = 2015,
    description = 'End Year :',
    disabled = False,
)

# define region dropdown for chart 5
region_dropdown5 = widgets.Dropdown(
    options = regions,
    value = 'Iceland',
    description = 'Region',
    disabled = False,
)

# define radio button to select the measure for chart 5
measure_chart_5 = widgets.RadioButtons(
    options = [('Total Catch', 'catch_sum'), ('Real Value', 'real_value')],
    value = 'catch_sum',
    description = 'Measure :',
    disabled = False
)
```

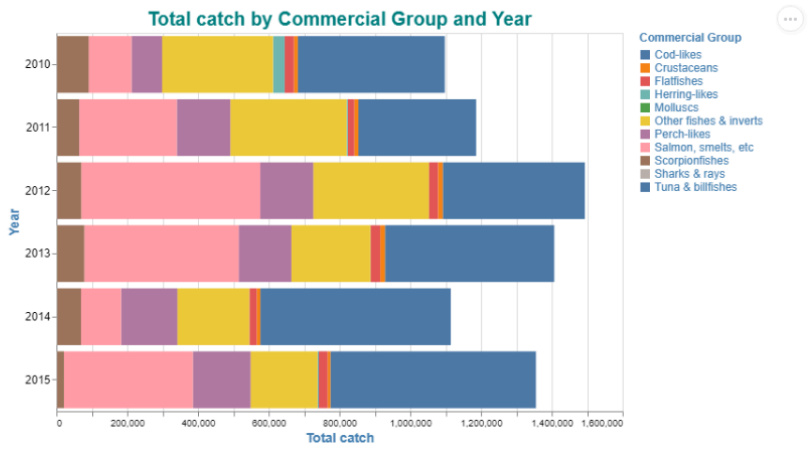
- Displaying total catch of commercial groups by year with 2010 as start year, 2015 as end year and Iceland as region and downloading graph data as csv

Total catch of Commercial groups by Year

```
In [27]: display_chart_1()
```

Start Year : 2010
End Year : 2015
Region Iceland

Out[27]:




```
In [28]: # download data behind the chart

chart1_data = download_chart_1()
download_csv(chart1_data, 'chart1_data.csv')
```

- Displaying total catch of commercial groups by year for all sectors with 2010 as start year, 2015 as end year and Iceland as region

Total catch of Commercial groups by Year - for all Sector Types

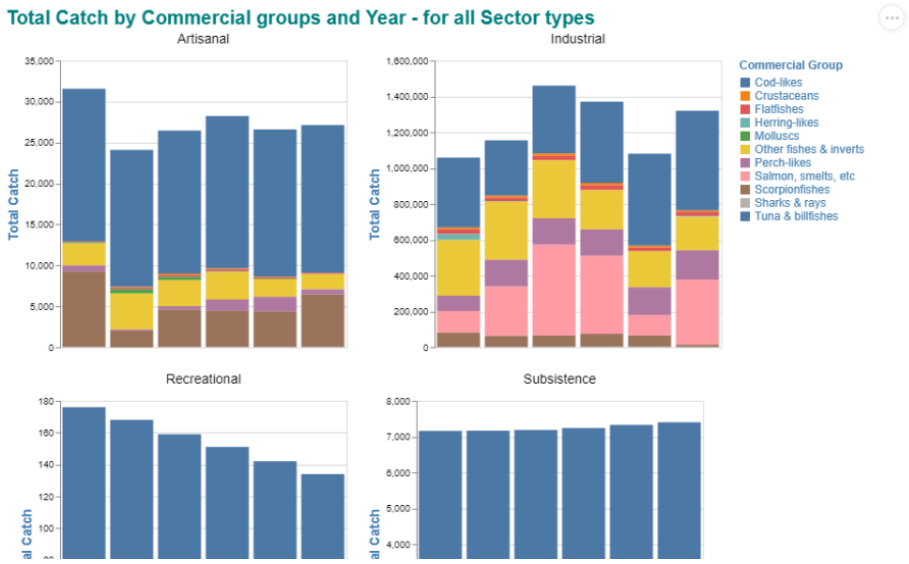
```
In [29]: display_chart_2()
```

Start Year : 2010

End Year : 2015

Region Iceland

Out[29]:



- Downloading graph data as csv

```
In [30]: # download data behind the chart

chart2_data = download_chart_2()
download_csv(chart2_data, 'chart2_data.csv')
```

Start Year : 2010

End Year : 2015

Region Iceland

Out[30]:

- Displaying distribution of catch sum by commercial groups with 2010 as start year, 2015 as end year and Iceland as region

Distribution of Catch sum by Commercial groups

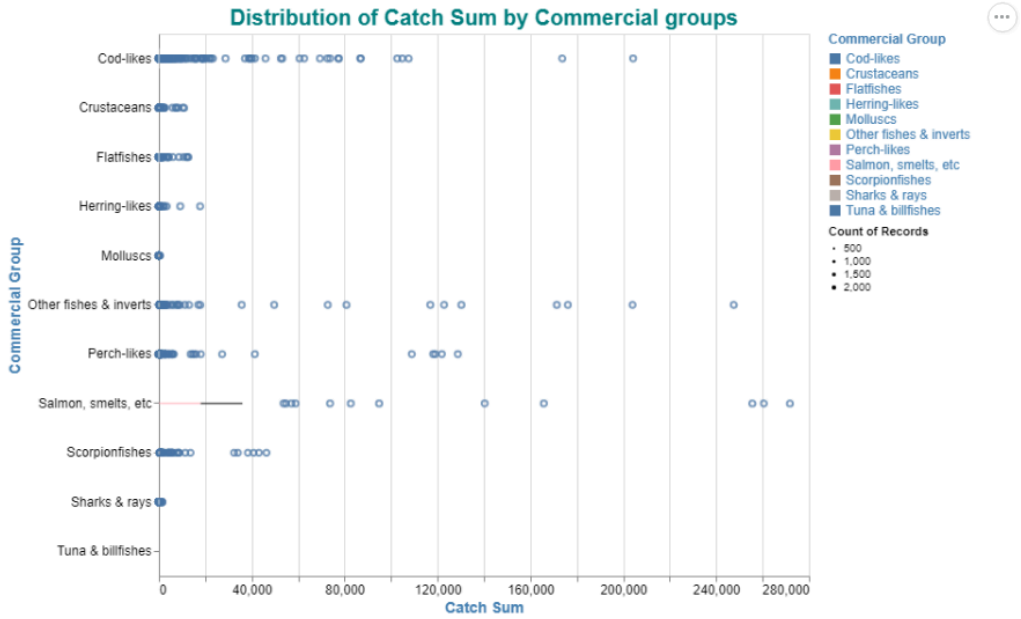
```
In [31]: display_chart_3()
```

Start Year : 2010

End Year : 2015

Region Iceland

Out[31]:



- Downloading graph data as csv

```
In [32]: # download data behind the chart

chart3_data = download_chart_3()
download_csv(chart3_data, 'chart3_data.csv')
```

Start Year : 2010

End Year : 2015

Region Iceland

Out[32]:

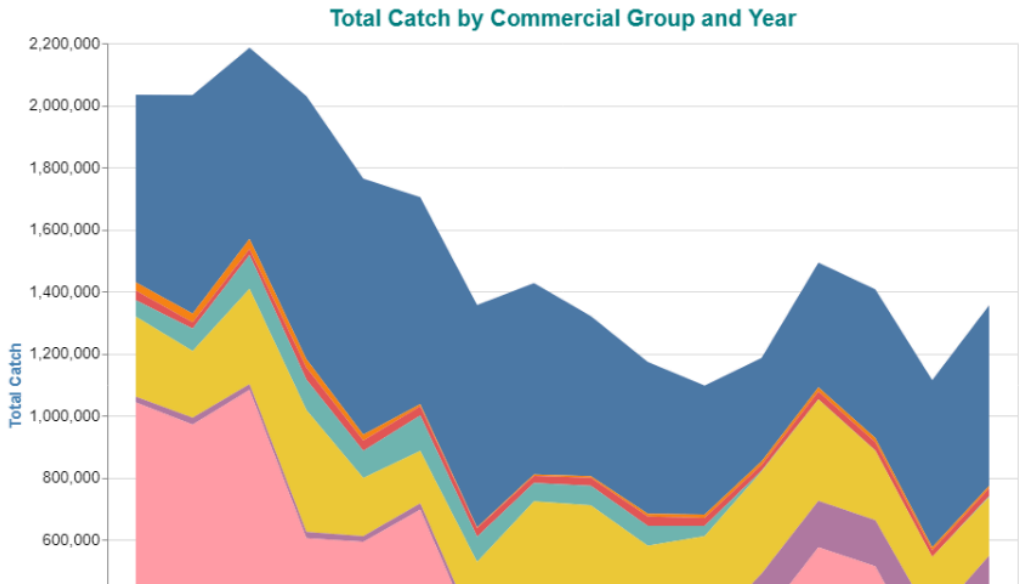
- Displaying total catch or real value by commercial group and year with 2000 as start year, 2015 as end year, Iceland as region and total catch as measure

Total Catch (or) Real value by Commercial Group and Year

```
In [33]: display_chart_4()
```

Start Year : 2000
End Year : 2015
Region : Iceland
Measure : Total Catch / Real Value

Out[33]:



- Downloading graph data as csv

```
In [34]: # download data behind the chart

chart4_data = download_chart_4()
download_csv(chart4_data, 'chart4_data.csv')
```

Start Year : 2000
End Year : 2015
Region : Iceland
Measure : Total Catch / Real Value

Out[34]:

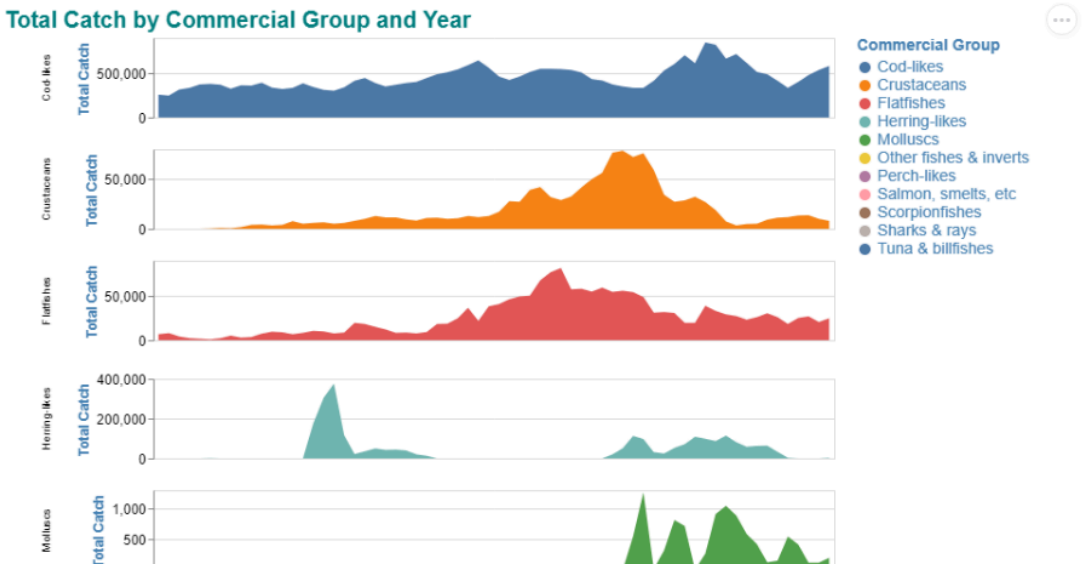
- Displaying total catch or real value by year for every commercial group and year with 2000 as start year, 2015 as end year, Iceland as region and total catch as measure

Total Catch (or) Real value by Year for every Commercial Group

```
In [35]: display_chart_5()
```

Start Year : 1950
End Year : 2015
Region : Iceland
Measure : (•) Total Catch ( ) Real Value

Out[35]:



- Downloading graph data as csv

```
In [36]: # download data behind the chart

chart5_data = download_chart_5()
download_csv(chart5_data, 'chart5_data.csv')
```

Start Year : 1950
End Year : 2015
Region : Iceland
Measure : (•) Total Catch ( ) Real Value

Out[36]: