# 2. PandasIBasics HW

September 22, 2024

Name: Mrudu lahari Malayanur

## 0.1 Homework 2 - Pandas Basics

Let us review with a series of Pandas coding challenges!

Use comments and markdown cells to describe code.

Datasets we will use:

- https://raw.githubusercontent.com/mafudge/datasets/master/flights/sample-flights.csv
- https://raw.githubusercontent.com/mafudge/datasets/master/orders/sample-orders.csv

[3]:
```
import pandas as pd
import numpy as np
from IPython.display import display
from ipywidgets import widgets, interact_manual

pd.set_option('display.max_colwidth', None)
```

## 0.2 Reading a dataset into a dataframe.

The following code loads the airline flights dataset into the variable `flights`

[4]:
```
flights = pd.read_csv(" https://raw.githubusercontent.com/mafudge/datasets/
↪master/flights/sample-flights.csv")
flights.head()
```

[4]:

| | flight_number | departure_airport_code | arrival_airport_code | departure_date | arrival_date | departure_time | arrival_time | flight_duration | airline_name |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1350 | KJP | VOG | 2022-03-26 | 2022-03-07 | 5:04 | 23:25 | 10.96 | United |
| 1 | 5381 | FUN | POW | 2022-11-01 | 2022-07-05 | 19:32 | 13:09 | 10.29 | Southwest |
| 2 | 2892 | ROR | COO | 2022-11-09 | 2022-05-16 | 0:02 | 19:45 | 10.65 | Delta |
| 3 | 2406 | XGA | HCM | 2022-01-09 | 2022-02-13 | 19:32 | 11:45 | 12.20 | American |
| 4 | 1261 | TDK | LKU | 2022-02-07 | | | | | |

```
4   2022-01-26          4:25        16:50             7.05       United

  aircraft_type
0  Embraer E190
1  Embraer E190
2    Boeing 747
3    Boeing 737
4    Boeing 737
```

[6]:
```
# PROMPT 1 read orders data into variable called "orders" and display the first few rows
#using read_csv to read the data in the 2nd link. using head to print first 5 rows in dataset
orders=pd.read_csv("https://raw.githubusercontent.com/mafudge/datasets/master/orders/sample-orders.csv")
orders.head()
```

[6]:
```
   orderid   orderdate         custname                    custemail  \
0        2  2023-03-24   Frayda Pepperd  fpepperd0@sciencedaily.com
1        3  2020-02-23      Loy Siberry      lsiberry1@so-net.ne.jp
2        4  2022-04-28  Carree Henworth                         NaN
3        5  2019-11-22  Goldina Godsafe   ggodsafe3@dailymail.co.uk
4        6  2022-05-03   Marris Chatten     mchatten4@csmonitor.com

     custcountry orderstatus  ordertotal ordercreditcard ordershipvia  \
0         Canada   delivered      228.39        Discover          RPS
1         Canada   delivered       76.87        Discover         USPS
2         Canada     pending      152.30        Discover         USPS
3  United States     shipped      182.17            Amex          UPS
4         Mexico     pending      208.28        Discover          RPS

   shippingtotal
0          12.05
1           6.27
2          12.74
3           5.44
4           2.16
```

### 0.3 What does the data look like?

This code uses `info()` to get information about the columns and datatypes of the dataframe.

[7]:
```
flights.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
```

```
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   flight_number           1000 non-null   int64
 1   departure_airport_code  1000 non-null   object
 2   arrival_airport_code    1000 non-null   object
 3   departure_date          1000 non-null   object
 4   arrival_date            1000 non-null   object
 5   departure_time          1000 non-null   object
 6   arrival_time            1000 non-null   object
 7   flight_duration         1000 non-null   float64
 8   airline_name            993 non-null    object
 9   aircraft_type           1000 non-null   object
dtypes: float64(1), int64(1), object(8)
memory usage: 78.3+ KB
```

[8]:
```
# PROMPT 2 - get information for the "orders" dataframe
# does every order have an email?
orders.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   orderid          1000 non-null   int64
 1   orderdate        1000 non-null   object
 2   custname         1000 non-null   object
 3   custemail        946 non-null    object
 4   custcountry      1000 non-null   object
 5   orderstatus      1000 non-null   object
 6   ordertotal       1000 non-null   float64
 7   ordercreditcard  1000 non-null   object
 8   ordershipvia     1000 non-null   object
 9   shippingtotal    944 non-null    float64
dtypes: float64(2), int64(1), object(7)
memory usage: 78.3+ KB
```

No, there are 1000 orders, but only 946 customer mails. Hence not every order has a mail ID

### 0.4 What are the different aircraft names?

This code will use `value_counts()` to produce counts of the different aircraft names

[10]:
```
flights["aircraft_type"].value_counts()
```

[10]:
```
aircraft_type
Embraer E190    229
Boeing 777      217
Boeing 747      149
```

```
Airbus A350     141
Boeing 737      134
Airbus A320     130
Name: count, dtype: int64
```

[11]:
```
orders.columns
```

[11]:
```
Index(['orderid', 'orderdate', 'custname', 'custemail', 'custcountry',
       'orderstatus', 'ordertotal', 'ordercreditcard', 'ordershipvia',
       'shippingtotal'],
      dtype='object')
```

[13]:
```
# PROMPT 3  - get the value counts for order status
#using value counts to get unique values present in that column. here it is␣
 ↪delivered shipped and pending. Value count determines the number of such␣
 ↪values in each category
orders["orderstatus"].value_counts()
```

[13]:
```
orderstatus
delivered    584
shipped      210
pending      206
Name: count, dtype: int64
```

### 0.5 This prints a list of unique airline names

We use `unique()` on the series to get a unique list of value, and `dropna()` to get rid of the empty values.

[14]:
```
airlines = list(flights['airline_name'].dropna().unique())
airlines.sort()
print(airlines)
```

```
['American', 'Delta', 'Jetblue', 'Southwest', 'United']
```

[17]:
```
# PROMPT 4 - get a unique list of the customer country

unique_countries=list(orders['custcountry'].dropna().unique())
unique_countries.sort()
print(unique_countries)
```

```
['Canada', 'Mexico', 'United States']
```

[18]:
```
# PROMPT 4a : generalize this to a function:
def dedupe_series(series: pd.Series) -> list:
    # TODO
    return series.dropna().unique()
```

```
countries = dedupe_series(orders['custcountry'])
print(countries)
```

```
['Canada' 'United States' 'Mexico']
```

### 0.6 Create a drop-down list of airlines.

this creates a drop-down selection widget based on the airline values

[19]:
```
order_dropdown = widgets.Dropdown(options=orders, description="Airline")
display(airline_dropdown)
```

```
Dropdown(description='Airline', options=('American', 'Delta', 'Jetblue',␣
↪'Southwest', 'United'), value='Americ…
```

[20]:
```
# PROMPT 5 - create a dropdown of countries from orders
order_countries_dropdown = widgets.Dropdown(options=unique_countries,␣
↪description="Countries")
display(order_countries_dropdown)
```

```
Dropdown(description='Countries', options=('Canada', 'Mexico', 'United States'),␣
↪value='Canada')
```

### 0.7 Get stats on the numerical columns

The `describe()` method function will get statistics for the numerical values in the dataframe.

[21]:
```
flights.describe()
```

[21]:

| | flight_number | flight_duration |
|---|---|---|
| count | 1000.00000 | 1000.000000 |
| mean | 4990.50300 | 8.427190 |
| std | 2931.96522 | 4.312989 |
| min | 4.00000 | 1.020000 |
| 25% | 2345.75000 | 4.780000 |
| 50% | 5061.50000 | 8.455000 |
| 75% | 7488.25000 | 12.160000 |
| max | 9996.00000 | 15.990000 |

[27]:
```
# PROMPT 6 - that is the least expensive order? Most expensive shipping amount?
orders['ordertotal'].describe()
```

[27]:

| | |
|---|---|
| count | 1000.000000 |
| mean | 150.688370 |
| std | 56.781819 |
| min | 50.140000 |
| 25% | 103.535000 |
| 50% | 150.845000 |
| 75% | 198.552500 |

```
max       249.190000
Name: ordertotal, dtype: float64
```

From the above statistics, min of order total = 50.140000 which is the least expensive

[29]:
```
orders['shippingtotal'].describe()
```

[29]:
```
count    944.000000
mean      10.367352
std        5.259112
min        1.040000
25%        5.877500
50%       10.525000
75%       14.742500
max       20.000000
Name: shippingtotal, dtype: float64
```

From the above statistics, max of shipping total = 20.000000 which is the most expensive shipping amount

### 0.8 Storing Min and Max in variables

This example stores the shortest and longest flights in separate variables.

[31]:
```
shortest = flights['flight_duration'].min()
longest = flights['flight_duration'].max()
print(shortest, longest)
```

```
1.02 15.99
```

[34]:
```
# PROMPT 7 - store the largest and smallest orders order total in variables.
smallest = orders['ordertotal'].min()
largest = orders['ordertotal'].max()
print(smallest, largest)

#using min for smallest and max for largest order total values
```

```
50.14 249.19
```

### 0.9 Creating a Range Slider widget

This example creates a Range slider widget for flight duration, setting the upper and lower bounds to the min/max values.

[35]:
```
flight_duration_slider = widgets.FloatRangeSlider(
    min = shortest, max=longest, step=0.5, description="Duration")
display(flight_duration_slider)
```

```
FloatRangeSlider(value=(4.7625, 12.2475), description='Duration', max=15.99,␣
↪min=1.02, step=0.5)
```

[38]:
```
# PROMPT 8 - Create a range slider for orders using min/max approach

#using floatrange slider to get the slider with max and min values.
orders_total_slider = widgets.FloatRangeSlider(
    min = smallest, max=largest, step=0.5, description="Order Total")
display(orders_total_slider)
```

```
FloatRangeSlider(value=(99.9025, 199.42749999999998), description='Order Total',␣
↪max=249.19, min=50.14, step=0…
```

### 0.10 Let's engineer a column!

This example will create a YEAR column by slicing the first 4 characters from the date. Since the data type of the `departure_date` is Object we must use the `.str` property to get the string value.

[40]:
```
flights["departure_year"] = flights["departure_date"].str[:4]
flights.head()
```

[40]:

| | flight_number | departure_airport_code | arrival_airport_code | departure_date | arrival_date | departure_time | arrival_time | flight_duration | airline_name | aircraft_type | departure_year |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1350 | KJP | VOG | 2022-03-26 | 2022-03-07 | 5:04 | 23:25 | 10.96 | United | Embraer E190 | 2022 |
| 1 | 5381 | FUN | POW | 2022-11-01 | 2022-07-05 | 19:32 | 13:09 | 10.29 | Southwest | Embraer E190 | 2022 |
| 2 | 2892 | ROR | COO | 2022-11-09 | 2022-05-16 | 0:02 | 19:45 | 10.65 | Delta | Boeing 747 | 2022 |
| 3 | 2406 | XGA | HCM | 2022-01-09 | 2022-02-13 | 19:32 | 11:45 | 12.20 | American | Boeing 737 | 2022 |
| 4 | 1261 | TDK | LKU | 2022-02-07 | 2022-01-26 | 4:25 | 16:50 | 7.05 | United | Boeing 737 | 2022 |

[42]:
```
# PROMPT 9 - create an order year column!

#since the date of orders is the form of YYYY-MM-DD, using indexing to get the␣
↪first 4 charectes only using str[:4] function
orders["ordered_year"] = orders["orderdate"].str[:4]
orders.head()
```

[42]:
```
   orderid   orderdate         custname                   custemail  \
0        2  2023-03-24   Frayda Pepperd  fpepperd0@sciencedaily.com
1        3  2020-02-23      Loy Siberry      lsiberry1@so-net.ne.jp
2        4  2022-04-28  Carree Henworth                         NaN
3        5  2019-11-22  Goldina Godsafe   ggodsafe3@dailymail.co.uk
4        6  2022-05-03   Marris Chatten     mchatten4@csmonitor.com

     custcountry orderstatus  ordertotal ordercreditcard ordershipvia  \
0         Canada   delivered      228.39        Discover          RPS
1         Canada   delivered       76.87        Discover         USPS
2         Canada     pending      152.30        Discover         USPS
3  United States     shipped      182.17            Amex          UPS
4         Mexico     pending      208.28        Discover          RPS

   shippingtotal ordered_year
0          12.05         2023
1           6.27         2020
2          12.74         2022
3           5.44         2019
4           2.16         2022
```

[44]:
```
# prompt 10 - create an order month column!
#indexign from 5th to 6th charecter in the order date to get month values
orders["ordered_month"] = orders["orderdate"].str[5:7]
orders.head()
```

[44]:
```
   orderid   orderdate         custname                   custemail  \
0        2  2023-03-24   Frayda Pepperd  fpepperd0@sciencedaily.com
1        3  2020-02-23      Loy Siberry      lsiberry1@so-net.ne.jp
2        4  2022-04-28  Carree Henworth                         NaN
3        5  2019-11-22  Goldina Godsafe   ggodsafe3@dailymail.co.uk
4        6  2022-05-03   Marris Chatten     mchatten4@csmonitor.com

     custcountry orderstatus  ordertotal ordercreditcard ordershipvia  \
0         Canada   delivered      228.39        Discover          RPS
1         Canada   delivered       76.87        Discover         USPS
2         Canada     pending      152.30        Discover         USPS
3  United States     shipped      182.17            Amex          UPS
4         Mexico     pending      208.28        Discover          RPS

   shippingtotal ordered_year ordered_month
0          12.05         2023            03
1           6.27         2020            02
2          12.74         2022            04
3           5.44         2019            11
4           2.16         2022            05
```

### 0.11 United airlines flights

This example uses a boolean filter to create a smaller dataframe of just United airlines flights.

[45]:
```
ua_flights = flights[
    flights["airline_name"] == "United"
]
ua_flights
```

[45]:
```
     flight_number departure_airport_code arrival_airport_code departure_date  \
0             1350                   KJP                   VOG     2022-03-26
4             1261                   TDK                   LKU     2022-02-07
6             7066                   IXC                   YTJ     2022-03-23
7             5122                   YVC                   EGI     2022-03-19
12            2730                   LSZ                   DTD     2022-11-15
..             …                     …                     …            …
970           5726                   DBS                   TAC     2022-12-17
976           9830                   MQT                   BGA     2022-08-25
985           4517                   BMG                   MRF     2022-12-19
989           4029                   HEW                   NRB     2022-09-06
993           4898                   SNY                   YRM     2022-07-20

    arrival_date departure_time arrival_time  flight_duration airline_name  \
0     2022-03-07           5:04        23:25            10.96       United
4     2022-01-26           4:25        16:50             7.05       United
6     2022-12-19          10:51        21:11            13.01       United
7     2022-08-23          21:10         7:47            12.92       United
12    2022-07-02           9:21        11:19             6.95       United
..           …              …            …                …            …
970   2022-05-11           6:23         2:05            15.15       United
976   2022-06-23           8:14        14:59             3.05       United
985   2022-08-20           9:25        18:49             4.93       United
989   2022-04-30          15:19        19:07            14.78       United
993   2022-02-09          18:35        19:45            14.08       United

    aircraft_type departure_year
0    Embraer E190           2022
4      Boeing 737           2022
6      Boeing 747           2022
7    Embraer E190           2022
12     Boeing 777           2022
..            …              …
970   Airbus A350           2022
976    Boeing 747           2022
985  Embraer E190           2022
989    Boeing 777           2022
993   Airbus A350           2022
```

[256 rows x 11 columns]

[47]:
```
# Prompt 10 - display only orders that were delivered
#first applying boolean to get the orders for which order status = delivered. then applying the condiiton on array to get the list of item for which status is delivered
orders_delivered = orders[ orders["orderstatus"] == "delivered"]
orders_delivered
```

[47]:

| | orderid | orderdate | custname | custemail |
|---|---|---|---|---|
| 0 | 2 | 2023-03-24 | Frayda Pepperd | fpepperd0@sciencedaily.com |
| 1 | 3 | 2020-02-23 | Loy Siberry | lsiberry1@so-net.ne.jp |
| 5 | 7 | 2022-12-19 | Logan Jacobsson | ljacobsson5@wufoo.com |
| 7 | 9 | 2019-02-17 | Lowrance Sigsworth | lsigsworth7@youtube.com |
| 9 | 11 | 2020-01-20 | Renato Hue | rhue9@un.org |
| .. | … | … | … | … |
| 994 | 996 | 2019-04-03 | Barty Bell | bbellrm@eepurl.com |
| 995 | 997 | 2023-09-22 | Stewart Punter | spunterrn@yolasite.com |
| 996 | 998 | 2019-02-23 | Sherwynd Hardesty | shardestyro@deviantart.com |
| 997 | 999 | 2020-02-25 | Doll Kristiansen | dkristiansenrp@ca.gov |
| 999 | 1001 | 2023-08-20 | Ronna Crebott | rcrebottrr@cnet.com |

| | custcountry | orderstatus | ordertotal | ordercreditcard | ordershipvia |
|---|---|---|---|---|---|
| 0 | Canada | delivered | 228.39 | Discover | RPS |
| 1 | Canada | delivered | 76.87 | Discover | USPS |
| 5 | United States | delivered | 112.15 | Amex | USPS |
| 7 | United States | delivered | 141.94 | Discover | USPS |
| 9 | Canada | delivered | 120.52 | Visa | USPS |
| .. | … | … | … | … | … |
| 994 | United States | delivered | 114.47 | Discover | UPS |
| 995 | Canada | delivered | 155.50 | Visa | UPS |
| 996 | Mexico | delivered | 245.48 | Visa | UPS |
| 997 | Canada | delivered | 163.07 | Amex | FedEX |
| 999 | United States | delivered | 161.43 | Discover | USPS |

| | shippingtotal | ordered_year | ordered_month |
|---|---|---|---|
| 0 | 12.05 | 2023 | 03 |
| 1 | 6.27 | 2020 | 02 |
| 5 | 11.52 | 2022 | 12 |
| 7 | 7.31 | 2019 | 02 |
| 9 | 5.57 | 2020 | 01 |
| .. | … | … | … |
| 994 | 8.30 | 2019 | 04 |
| 995 | 17.27 | 2023 | 09 |
| 996 | 19.86 | 2019 | 02 |
| 997 | 6.09 | 2020 | 02 |
| 999 | 16.90 | 2023 | 08 |

[584 rows x 12 columns]

### 0.12 Dataframe Boolean Filters with logical And

Sometimes you want to filter a dataframe on two conditions for example:

- American Airlines AND
- Boeing 777 aircraft

To do this we must use the dataframe AND operator: `&`

Notice how we must include `()` around each boolean filter.

[48]:
```
special_flights = flights[
    (flights["airline_name"] == "American") &
    (flights["aircraft_type"] == "Boeing 777")
]
special_flights.head()
```

[48]:

| | flight_number | departure_airport_code | arrival_airport_code | departure_date | arrival_date | departure_time | arrival_time | flight_duration | airline_name | aircraft_type | departure_year |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 2329 | ELA | XPK | 2022-08-01 | 2022-03-02 | 22:09 | 7:53 | 7.79 | American | Boeing 777 | 2022 |
| 40 | 4116 | ZLX | ASY | 2022-03-01 | 2022-09-28 | 2:51 | 10:53 | 14.57 | American | Boeing 777 | 2022 |
| 52 | 2567 | DGF | OSP | 2022-03-06 | 2022-08-21 | 13:38 | 22:22 | 14.65 | American | Boeing 777 | 2022 |
| 78 | 9761 | SDY | BSF | 2022-05-02 | 2022-01-06 | 23:11 | 18:50 | 7.35 | American | Boeing 777 | 2022 |
| 114 | 6574 | GZP | FEB | 2022-04-30 | 2022-09-10 | 6:32 | 13:48 | 15.44 | American | Boeing 777 | 2022 |

[52]:
```
# PROMPT 11 - show "special orders": those orders delivered to the Canada in␣
 ↪year 2023

#applying & operator to get the orders that are "Delivered" to the country␣
 ↪"Canada" in the year "2023"
special_orders = orders[
    (orders["custcountry"] == "Canada") &
    (orders["ordered_year"] == "2023") &
```

```
    (orders["orderstatus"] == "delivered") ]
special_orders.head()
```

[52]:
```
     orderid   orderdate        custname                  custemail  \
0          2  2023-03-24  Frayda Pepperd  fpepperd0@sciencedaily.com
89        91  2023-10-02  Gardy Issacson   gissacson2h@123-reg.co.uk
248      250  2023-02-21    Jake Dibling   jdibling6w@cloudflare.com
266      268  2023-09-21   Donica Verden        dverden7e@github.com
306      308  2023-12-07  Felicdad Flegg          fflegg8i@apple.com

    custcountry orderstatus  ordertotal ordercreditcard ordershipvia  \
0        Canada   delivered      228.39        Discover          RPS
89       Canada   delivered       90.23        Discover          UPS
248      Canada   delivered       82.13        Discover         USPS
266      Canada   delivered      180.49      Mastercard         USPS
306      Canada   delivered      155.78      Mastercard          UPS

     shippingtotal ordered_year ordered_month
0            12.05         2023            03
89           13.56         2023            10
248          15.78         2023            02
266           8.87         2023            09
306           9.24         2023            12
```

### 0.13 Flight Tracker

Inputs:

- Range for the duration of the flight
- Airline

Outputs:

-DataFrame of flights matching that criteira

[53]:
```
# Get Data
airlines = sorted(list(flights['airline_name'].dropna().unique()))
shortest = flights['flight_duration'].min()
longest = flights['flight_duration'].max()

# Make widgets
airline_dropdown = widgets.Dropdown(options=airlines, description="Airline")
flight_duration_slider = widgets.FloatRangeSlider(
    min = shortest, max=longest, step=0.5, description="Duration")

@interact_manual(airline=airline_dropdown, duration=flight_duration_slider)
def on_click(airline, duration):
    filtered_flights = flights[
        (flights["airline_name"] == airline) &
```

```
        (flights["flight_duration"] >= duration[0]) &
        (flights["flight_duration"] <= duration[1])
    ]
    display(filtered_flights)
```

```
interactive(children=(Dropdown(description='Airline', options=('American',␣
 ↪'Delta', 'Jetblue', 'Southwest', 'U…
```

### 0.14 Order Report

Inputs:

```
- Range Slider for the order amount total
- Year of order, Order Status, Customer Country as drop downs
```

Outputs:

```
-DataFrame of orders matching the selected criteria
```

[ ]:
```
orders.columns # a refresher of the available columns
```

[ ]:
```
# PROMPT 12 - make the order report!

# Get Data for widgets

# Make widgets

# main interact
@interact_manual(country=countries, year=years, status=statuses)
def on_click(country, year, status):
    # TODO
```

### 0.15 Explain your report findings

[63]:
```
# Get Data

#Order year - creating dropdown for ordered year
order_year = sorted(list(orders['ordered_year'].dropna().unique()))
order_year_dropdown = widgets.Dropdown(options=order_year, description="Orderd␣
 ↪year")

#creating dropdown for Order Status
order_status = sorted(list(orders['orderstatus'].dropna().unique()))
order_status_dropdown = widgets.Dropdown(options=order_status,␣
 ↪description="Order Status")

# creating dropdown for Customer Country
custcountry = sorted(list(orders['custcountry'].dropna().unique()))
```

```
custcountry_dropdown = widgets.Dropdown(options=custcountry,␣
 ↪description="Customer Country")


#creating slider - Order Total. using min and max functions to give the range␣
 ↪of the slider
min_order_tot = orders['ordertotal'].min()
max_order_tot = orders['ordertotal'].max()

order_tot_slider = widgets.FloatRangeSlider(
    min=min_order_tot,max=max_order_tot, step=0.5, description="Order Total␣
 ↪Cost")


#creating the on click interact function with the input values - country, year,␣
 ↪status and total order cost
#assigning values to the parameters in the function to apply the filter␣
 ↪conditions
#using display function to display the filtered orders

@interact_manual(country=custcountry_dropdown, year=order_year_dropdown,␣
 ↪status=order_status_dropdown,amount=order_tot_slider)
def on_click(country, year, status, amount):
    # TODO
    filtered_orders = orders[
        (orders["orderstatus"] == status) &
        (orders["ordered_year"] == year) &
        (orders["custcountry"] == country) &
        (orders["ordertotal"] >= amount[0]) &
        (orders["ordertotal"] <= amount[1])
    ]

    display(filtered_orders)
```

```
interactive(children=(Dropdown(description='Customer Country',␣
 ↪options=('Canada', 'Mexico', 'United States'), …
```