Homework

DATA QUALITY

Data Management & Ethics

By Arthy UTHAYARAJAH, Chelsie LOUIS, Nourhene LIHEOUEL, and Salma TARIFA

18/01/2025

Define Data Quality Rules

1. **Set up the collaboration environment you will do!**

Data Quality starts with a good environment to make the collaboration process effective. It ensures we all have access to the same data at the same time. (Availability/Accessibility/Coherence/Durability using GitHub and workflows to automatically secure the incoming changes).

2. **Careful data extraction you will perform!**

Then, it's important to perform the proper extraction of the datasets - here by setting up the right delimiters.

3. **Data Analysis Tool you will use!**

Most of the time, we perform Data Analysis step right in the Jupyter Notebook. But now, we will perform differently and use a Data Analysis Tool called *Dataiku* to help us in the process, advise us, and reinforce our observations/statements. We can easily miss out important elements.

4. **Into the data cleaning you will deep dive!**

The Data Analysis Tool was just help. Now, you must deepen the observation by continuing the data analysis part and by including some relevant metrics on your own to watch for incoherences and things to change (data cleaning).

5. **The Different Levels of Confidentiality for encryption you must know!**

Encryption is something very important in case of a data leak. Here our encryption file will not be publicly available to not be able to decrypt the data.

Public: Information that can be shared openly with anyone, like a company's website content.

Internal: Information meant for use within the organization, like internal memos or non-sensitive employee communications. (We don't encrypt but in some business policies it might be required to store the data)

Confidential: Sensitive information that requires protection, such as business strategies, customer data, or financial records. (To encrypt)

Restricted: Highly sensitive information with strict access controls, like trade secrets or personal health information. (To encrypt)

6. Of highly help can be the visualizations

Can be a good way to visualize the different trends of the data, especially when there are a lot of them. We will use Plotly as it is a library that provide interactive visualizations.

7. Code reviewing you must!

It's a good practice to make several people read your code and use a tool to detect potential breach in your code (e.g. visible passwords). It's not only necessary at the end but during all the project time. For example, on GitHub you can set up rules to pushing to the main branch only if a specific person validates your request (accept your pull request after reviewing your code), or if the code has no merge conflicts, or make the main branch "read-only" (disabling the pushes etc.).

Data Extraction

```
import pandas as pd
import numpy as np
import plotly.express as px

from encryption import encrypt_column, decrypt_column
```

```
customers_with_errors = pd.read_csv("datasets/customers_with_errors.csv", delimiter=';')
customers_with_errors.head()
```

	Customer ID	Title	First Name	Middle Name	Last Name	Email	Phone Number	Street	City	Postal Code	Country	Birthday	Age	Subscription Date	Update Date
0	CUST001	Mr.	Robert	Anna	Cain	robert.cain@outlook.com	14165551234	75944 John Forges	NaN	NaN	United States	25/10/1961	63.0	17/07/2021	01/06/2022
1	CUST002	Ms.	Lisa	Brian	Williams	lisa.williams@example.fr	19055556789	34633 Sosa Fork	NaN	NaN	United States	11/06/1969	55.0	09/01/2023	22/11/2024
2	CUST003	NaN	Richard	Cameron	Beard	richard.beard@example.fr	16475552345	8474 Crystal Unions Suite 449	NaN	NaN	United States	29/09/1988	36.0	18/04/2021	05/01/2023
3	CUST004	Mr.	Nicole	Gina	Obrien	nicole.obrien@outlook.com	16045553456	8603 Scott Turnpike Suite 266	South Madisonside	25568.0	United States	30/07/1997	27.0	27/02/2020	13/03/2021

```
deco_sales = pd.read_csv("datasets/deco_sales")
deco_sales.head()
```

	Product ID	Customer ID	Transaction Date	Currency	Amount	Quantity	Update Date
0	1	CUST001	2024-11-05	USD	250.0	10	2024-11-06
1	2	CUST002	2024-11-07	USD	220.0	4	2024-11-08
2	3	CUST003	2024-11-10	EUR	72.0	4	2024-11-12
3	4	CUST004	2024-11-09	EUR	60.0	4	2024-11-10
4	5	CUST005	2024-11-11	USD	150.0	5	2024-11-12

```
deco_product = pd.read_csv("datasets/deco_product", delimiter='₩t')
deco_product.head()
```

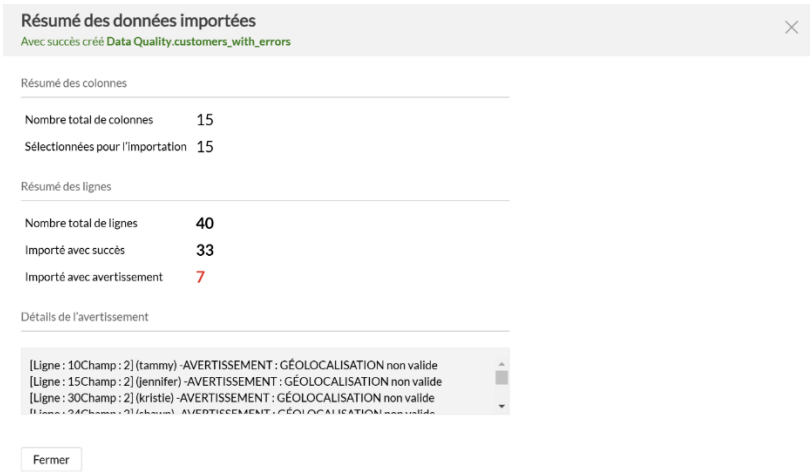
	Product ID	Product Name	Category	Starting Price	Selling Price	Currency	Current Stock	Supplier	Update Date	Availability
0	1	Decorative Vase	Vases	15.0	25.0	USD	100	HomeDecor Inc.	01/11/2024	Available
1	2	Elegant Wall Mirror	Mirrors	50.0	55.0	USD	50	Luxury Living	01/11/2024	Available
2	3	Wooden Frame	Frames	12.0	18.0	EUR	200	Rustic Charm	15/10/2024	Available
3	4	Candle Holder	Candles	10.0	15.0	EURO	150	Cozy Creations	20/10/2024	Available
4	5	Throw Pillow	Pillows	25.0	30.0	USD	100	Plush Home	10/11/2024	Out of Stock

Data Analysis with *Zoho Analytics*

N.B. The free version of Zoho Analytics isn't relevant and doesn't provide detailed or advanced metrics. That's why in the next session we will use Dataiku because it gives us access to more functionnalities.

Anyway, we will try to understand the given insights.

Zoho Analytics Insights for *customers_with_errors:*



The detailed advertisements **(7 in red)** appear to occur because the localization associated with some lines can't be found geographically surely due to missing information or misspelling error(s).

The solution can be to search the correct address and risk to put a wrong one, to delete it, or not to change it.

According to this use case, we will consider that the localization is a valuable insight and choose not to change it.

```
customers_with_errors.iloc[8] # e.g. 8th row have a geolocalization error
```

```
Customer ID                                CUST009
Title                                          Mr.
First Name                                   Tammy
Middle Name                                Brandon
Last Name                                      Lowe
Email                          tammy.lowe@gmail.com
Phone Number                             18195558901
Street                4680 Frazier Centers Apt. 032
City                                    East Tanya
Postal Code                                    NaN
Country                                United States
Birthday                                  24/01/1987
Age                                             37.0
Subscription Date                         06/01/2024
Update Date                               15/07/2024
Name: 8, dtype: object
```

```
print(f"They are {len(customers_with_errors.columns)} columns in the customers_with_errors dataset.")
print(f"They are {len(customers_with_errors)} lines in the customers_with_errors dataset.")
```

```
They are 15 columns in the customers_with_errors dataset.
They are 51 lines in the customers_with_errors dataset.
```

After a little investigation, we see that the 11 last elements are empty lines (NaN)! Let's remove them.

```
customers_with_errors.tail(12)
```

	Customer ID	Title	First Name	Middle Name	Last Name	Email	Phone Number	Street	City	Postal Code	Country	Birthday	Age	Subscription Date	Update Date
39	CUST040	NaN	Sherri	Jennifer	Henderson	sherri.henderson@yahoo.com	17 785 554 567	014 Allen Summit	NaN	42483.0	United States	11/12/2003	21.0	01/05/2021	03/11/2023
40	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
41	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
42	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
43	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
44	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
45	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
46	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
47	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
48	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
49	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
50	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
customers_with_errors = customers_with_errors.iloc[:40, :]
customers_with_errors.tail()
```

	Customer ID	Title	First Name	Middle Name	Last Name	Email	Phone Number	Street	City	Postal Code	Country	Birthday	Age	Subscription Date	Update Date
35	CUST036	Ms.	Brett	Curtis	Mccall	brett.mccall@outlook.com	12 265 559 012	75152 Maxwell Green Apt. 805	NaN	NaN	United States	08/06/1995	29.0	09/06/2021	08/05/2024
36	CUST037	Mr.	Jasmine	Nancy	Rodriguez	jasmine.rodriguez@yahoo.com	13 435 551 234	NaN	New Michaelton	48862.0	United States	17/11/1999	25.0	29/06/2020	28/12/2020
37	CUST038	Prof.	Patricia	Mary	Hudson	patricia.hudson@yahoo.com	14 375 552 345	1051 Phillip Ridge Apt. 054	NaN	50958.0	United States	26/03/2001	23.0	09/11/2024	16/11/2024
38	CUST039	Mrs.	Allison	George	Townsend	allison.townsend@outlook.com	14 385 553 456	14801 Danielle Falls Suite 544	Sandrachester	67505.0	United States	16/12/1963	61.0	18/02/2020	22/07/2023
39	CUST040	NaN	Sherri	Jennifer	Henderson	sherri.henderson@yahoo.com	17 785 554 567	014 Allen Summit	NaN	42483.0	United States	11/12/2003	21.0	01/05/2021	03/11/2023

Zoho Analytics Insights for *deco_product:*

```
print(f"They are {len(deco_product.columns)} columns in the deco_product dataset.")
print(f"They are {len(deco_product)} lines in the deco_product dataset.")
```

```
They are 10 columns in the deco_product dataset.
They are 20 lines in the deco_product dataset.
```

Zoho Analytics Insights for *deco_sales:*

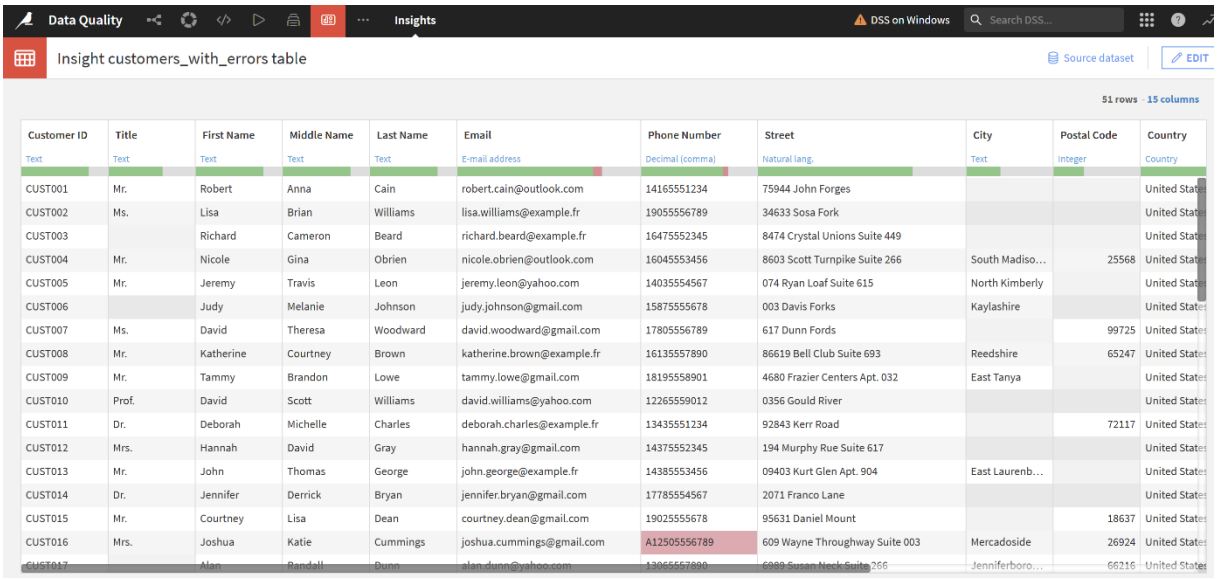

```
print(f"They are {len(deco_sales.columns)} columns in the deco_sales dataset.")
print(f"They are {len(deco_sales)} lines in the deco_sales dataset.")
```

```
They are 7 columns in the deco_sales dataset.
They are 70 lines in the deco_sales dataset.
```

Data Analysis with *Dataiku*

The datasets aren't that big, so we can visually evaluate them.

Dataiku Insights for *customers_with_errors*:



Customer ID	Title	First Name	Middle Name	Last Name	Email	Phone Number	Street	City	Postal Code	Country
Text	Text	Text	Text	Text	E-mail address	Decimal (comma)	Natural lang.	Text	Integer	Country
CUST001	Mr.	Robert	Anna	Cain	robert.cain@outlook.com	14165551234	75944 John Forges			United Stat
CUST002	Ms.	Lisa	Brian	Williams	lisa.williams@example.fr	19055556789	34633 Sosa Fork			United Stat
CUST003		Richard	Cameron	Beard	richard.beard@example.fr	16475552345	8474 Crystal Unions Suite 449			United Stat
CUST004	Mr.	Nicole	Gina	Obrien	nicole.obrien@outlook.com	16045553456	8603 Scott Turnpike Suite 266	South Madiso…	25568	United Stat
CUST005	Mr.	Jeremy	Travis	Leon	jeremy.leon@yahoo.com	14035554567	074 Ryan Loaf Suite 615	North Kimberly		United Stat
CUST006		Judy	Melanie	Johnson	judy.johnson@gmail.com	15875555678	003 Davis Forks	Kaylashire		United Stat
CUST007	Ms.	David	Theresa	Woodward	david.woodward@gmail.com	17805556789	617 Dunn Fords		99725	United Stat
CUST008	Mr.	Katherine	Courtney	Brown	katherine.brown@example.fr	16135557890	86619 Bell Club Suite 693	Reedshire	65247	United Stat
CUST009	Mr.	Tammy	Brandon	Lowe	tammy.lowe@gmail.com	18195558901	4680 Frazier Centers Apt. 032	East Tanya		United Stat
CUST010	Prof.	David	Scott	Williams	david.williams@yahoo.com	12265559012	0356 Gould River			United Stat
CUST011	Dr.	Deborah	Michelle	Charles	deborah.charles@example.fr	13435551234	92843 Kerr Road		72117	United Stat
CUST012	Mrs.	Hannah	David	Gray	hannah.gray@gmail.com	14375552345	194 Murphy Rue Suite 617			United Stat
CUST013	Mr.	John	Thomas	George	john.george@example.fr	14385553456	09403 Kurt Glen Apt. 904	East Laurenb…		United Stat
CUST014	Dr.	Jennifer	Derrick	Bryan	jennifer.bryan@gmail.com	17785554567	2071 Franco Lane			United Stat
CUST015	Mr.	Courtney	Lisa	Dean	courtney.dean@gmail.com	19025555678	95631 Daniel Mount		18637	United Stat
CUST016	Mrs.	Joshua	Katie	Cummings	joshua.cummings@gmail.com	A12505556789	609 Wayne Throughway Suite 003	Mercadoside	26924	United Stat
CUST017		Alan	Randall	Dunn	alan.dunn@yahoo.com	13065557890	6089 Susan Neck Suite 266	Jenniferboro…	66216	United Stat

According to this insight, we have one error in the column *"Email"* and another one in the *"Phone Number"* one.

```
# Ms. MEYER email isn't valid
customers_with_errors.iloc[34]
```

```
Customer ID                    CUST035
Title                              Ms.
First Name                     Michael
Middle Name                     Alicia
Last Name                        Meyer
Email                            Meyer@
Phone Number              18 195 558 901
Street              684 Cody Ferry Apt. 222
City                    Lake Ronaldborough
Postal Code                        NaN
Country                  United States
Birthday                    28/07/1990
Age                               34.0
Subscription Date           17/07/2022
Update Date                 13/04/2023
Name: 34, dtype: object
```

Indeed "Meyer@" isn't a valid format so we will set it up to NaN preferably has we can't guess what is the good one.

```
customers_with_errors.Email.iloc[34] = np.nan
customers_with_errors.iloc[34]
```

```
Customer ID                          CUST035
Title                                    Ms.
First Name                           Michael
Middle Name                           Alicia
Last Name                              Meyer
Email                                    NaN
Phone Number                  18 195 558 901
Street               684 Cody Ferry Apt. 222
City                      Lake Ronaldborough
Postal Code                              NaN
Country                        United States
Birthday                          28/07/1990
Age                                     34.0
Subscription Date                 17/07/2022
Update Date                       13/04/2023
Name: 34, dtype: object
```

```
# Mrs. CUMMINGS phone number isn't valid
customers_with_errors.iloc[15]
```

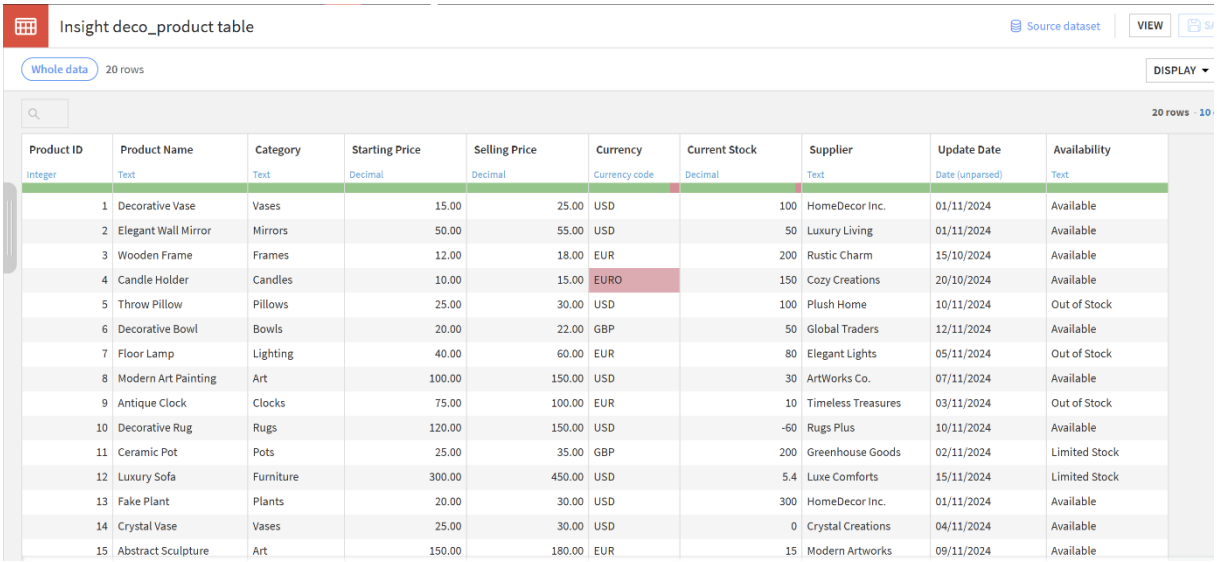
```
Customer ID                                CUST016
Title                                         Mrs.
First Name                                 Joshua
Middle Name                                 Katie
Last Name                                Cummings
Email                    joshua.cummings@gmail.com
Phone Number                       A12505556789
Street               609 Wayne Throughway Suite 003
City                                  Mercadoside
Postal Code                               26924.0
Country                             United States
Birthday                               29/03/1950
Age                                          74.0
Subscription Date                      05/11/2020
Update Date                            13/01/2021
Name: 15, dtype: object
```

Here we just need to remove the invalid character (A at the beginning).

```
customers_with_errors.iloc[15, 6] = customers_with_errors.iloc[15, 6].replace('A', '')
customers_with_errors.iloc[15]
```

```
Customer ID                          CUST016
Title                                   Mrs.
First Name                            Joshua
Middle Name                            Katie
Last Name                           Cummings
Email                  joshua.cummings@gmail.com
Phone Number                      12505556789
Street              609 Wayne Throughway Suite 003
City                                Mercadoside
Postal Code                         26924.0
Country                          United States
Birthday                            29/03/1950
Age                                    74.0
Subscription Date                   05/11/2020
Update Date                         13/01/2021
Name: 15, dtype: object
```

Dataiku Insights for *deco_product:*



Product ID	Product Name	Category	Starting Price	Selling Price	Currency	Current Stock	Supplier	Update Date	Availability
Integer	Text	Text	Decimal	Decimal	Currency code	Decimal	Text	Date (unparsed)	Text
1	Decorative Vase	Vases	15.00	25.00	USD	100	HomeDecor Inc.	01/11/2024	Available
2	Elegant Wall Mirror	Mirrors	50.00	55.00	USD	50	Luxury Living	01/11/2024	Available
3	Wooden Frame	Frames	12.00	18.00	EUR	200	Rustic Charm	15/10/2024	Available
4	Candle Holder	Candles	10.00	15.00	EURO	150	Cozy Creations	20/10/2024	Available
5	Throw Pillow	Pillows	25.00	30.00	USD	100	Plush Home	10/11/2024	Out of Stock
6	Decorative Bowl	Bowls	20.00	22.00	GBP	50	Global Traders	12/11/2024	Available
7	Floor Lamp	Lighting	40.00	60.00	EUR	80	Elegant Lights	05/11/2024	Out of Stock
8	Modern Art Painting	Art	100.00	150.00	USD	30	ArtWorks Co.	07/11/2024	Available
9	Antique Clock	Clocks	75.00	100.00	EUR	10	Timeless Treasures	03/11/2024	Out of Stock
10	Decorative Rug	Rugs	120.00	150.00	USD	-60	Rugs Plus	10/11/2024	Available
11	Ceramic Pot	Pots	25.00	35.00	GBP	200	Greenhouse Goods	02/11/2024	Limited Stock
12	Luxury Sofa	Furniture	300.00	450.00	USD	5.4	Luxe Comforts	15/11/2024	Limited Stock
13	Fake Plant	Plants	20.00	30.00	USD	300	HomeDecor Inc.	01/11/2024	Available
14	Crystal Vase	Vases	25.00	30.00	USD	0	Crystal Creations	04/11/2024	Available
15	Abstract Sculpture	Art	150.00	180.00	EUR	15	Modern Artworks	09/11/2024	Available

```
# We replace EURO by EUR in the currency column
deco_product.Currency = deco_product.Currency.replace('EURO', 'EUR')
deco_product.iloc[3]
```

```
Product ID                    4
Product Name       Candle Holder
Category                 Candles
Starting Price              10.0
Selling Price               15.0
Currency                     EUR
Current Stock                150
Supplier          Cozy Creations
Update Date           20/10/2024
Availability           Available
Name: 3, dtype: object
```

```
# The current stock isn't valid
deco_product.iloc[16]
```

```
Product ID                      17
Product Name      Large Floor Mirror
Category                    Mirrors
Starting Price                 75.0
Selling Price                  85.0
Currency                        EUR
Current Stock                   a20
Supplier                Mirror Magic
Update Date               30/10/2024
Availability               Available
Name: 16, dtype: object
```

The question is:

Is the correct number 20 or 120? (Why 120? Because on the French Keyboard configuration, the touch "a" is very close from "1").

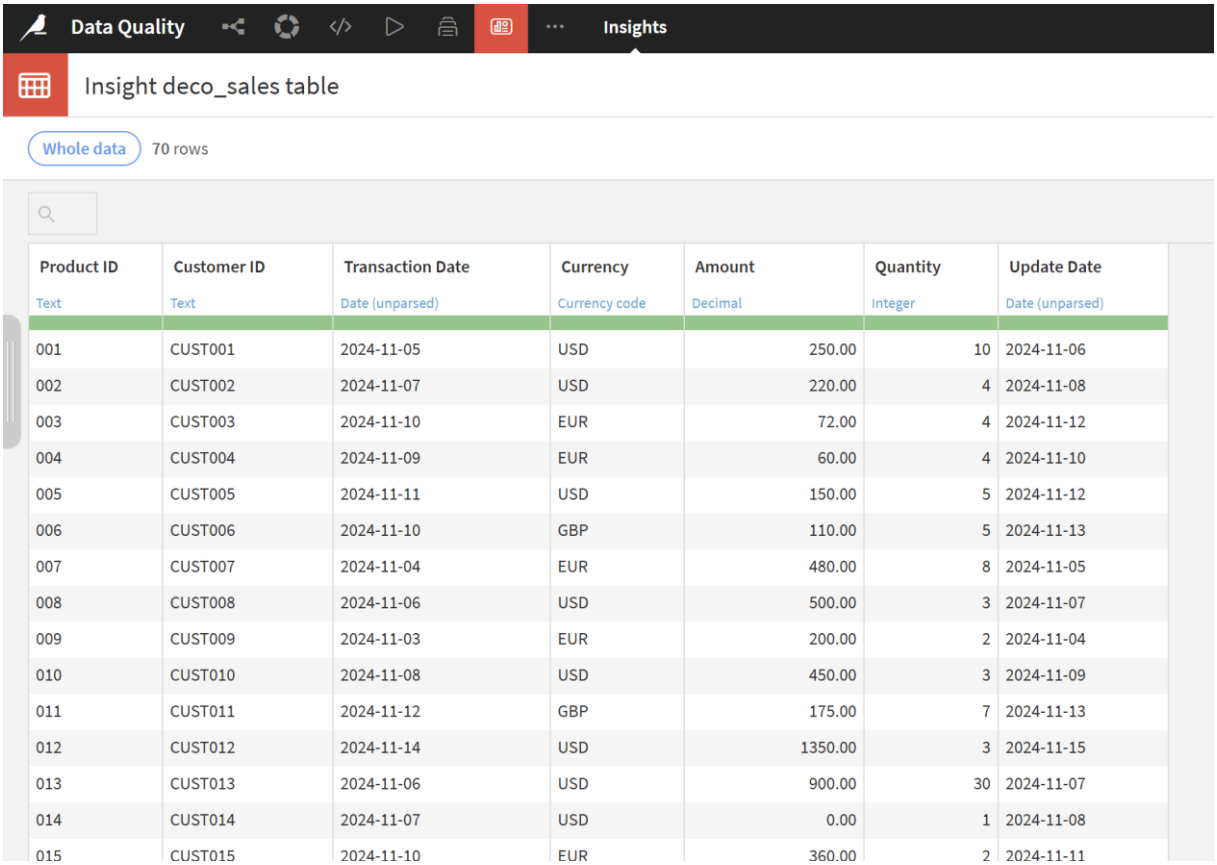
But as we consider that the "a" is in lower case and not in upper case, it probably means that the correct number is 20.

Sometimes in data quality, it's not that bad to have a such degree of overthinking. As we can't ask to the team that provides the dataset, we are by our own.

```
# We remove the a
deco_product.iloc[16, 6] = deco_product.iloc[16, 6].replace('a', '')
deco_product.iloc[16, 6]
```

```
'20'
```

Dataiku Insights for *deco_sales:*



Product ID	Customer ID	Transaction Date	Currency	Amount	Quantity	Update Date
Text	Text	Date (unparsed)	Currency code	Decimal	Integer	Date (unparsed)
001	CUST001	2024-11-05	USD	250.00	10	2024-11-06
002	CUST002	2024-11-07	USD	220.00	4	2024-11-08
003	CUST003	2024-11-10	EUR	72.00	4	2024-11-12
004	CUST004	2024-11-09	EUR	60.00	4	2024-11-10
005	CUST005	2024-11-11	USD	150.00	5	2024-11-12
006	CUST006	2024-11-10	GBP	110.00	5	2024-11-13
007	CUST007	2024-11-04	EUR	480.00	8	2024-11-05
008	CUST008	2024-11-06	USD	500.00	3	2024-11-07
009	CUST009	2024-11-03	EUR	200.00	2	2024-11-04
010	CUST010	2024-11-08	USD	450.00	3	2024-11-09
011	CUST011	2024-11-12	GBP	175.00	7	2024-11-13
012	CUST012	2024-11-14	USD	1350.00	3	2024-11-15
013	CUST013	2024-11-06	USD	900.00	30	2024-11-07
014	CUST014	2024-11-07	USD	0.00	1	2024-11-08
015	CUST015	2024-11-10	EUR	360.00	2	2024-11-11

This is a dataset of dream, everything is good.

Data Cleaning

First dataset: *customers_with_errors.csv*

```
len(customers_with_errors)
```

```
40
```

```
customers_with_errors.head()
```

	Customer ID	Title	First Name	Middle Name	Last Name	Email	Phone Number	Street	City	Postal Code	Country	Birthday	Age	Subscription Date	Update Date
0	CUST001	Mr.	Robert	Anna	Cain	robert.cain@outlook.com	14165551234	75944 John Forges	NaN	NaN	United States	25/10/1961	63.0	17/07/2021	01/06/2022
1	CUST002	Ms.	Lisa	Brian	Williams	lisa.williams@example.fr	19055556789	34633 Sosa Fork	NaN	NaN	United States	11/06/1969	55.0	09/01/2023	22/11/2024
2	CUST003	NaN	Richard	Cameron	Beard	richard.beard@example.fr	16475552345	8474 Crystal Unions Suite 449	NaN	NaN	United States	29/09/1988	36.0	18/04/2021	05/01/2023
3	CUST004	Mr.	Nicole	Gina	Obrien	nicole.obrien@outlook.com	16045553456	8603 Scott Turnpike Suite 266	South Madisonside	25568.0	United States	30/07/1997	27.0	27/02/2020	13/03/2021

```
customers_with_errors.describe()
```

	Postal Code	Age
count	18.000000	40.000000
mean	52985.722222	48.550000
std	27850.812593	19.281917
min	3292.000000	21.000000
25%	28160.750000	29.000000
50%	50987.000000	46.500000
75%	70203.500000	67.750000
max	99725.000000	77.000000

```
# Delete age column
customers_with_errors = customers_with_errors.drop(columns='Age')
customers_with_errors.columns
```

```
Index(['Customer ID', 'Title', 'First Name', 'Middle Name', 'Last Name',
       'Email', 'Phone Number', 'Street', 'City', 'Postal Code', 'Country',
       'Birthday', 'Subscription Date', 'Update Date'],
      dtype='object')
```

Completeness: Identify missing data

```
missing_values=customers_with_errors.isnull().sum()
missing_percent= ((missing_values) / len(customers_with_errors)) * 100

print(round(missing_percent,2))
```

```
Customer ID           0.0
Title                20.0
First Name            0.0
Middle Name           0.0
Last Name             0.0
Email                 2.5
Phone Number          5.0
Street                5.0
City                 50.0
Postal Code          55.0
Country               0.0
Birthday              0.0
Subscription Date     0.0
Update Date           0.0
dtype: float64
```

We can ask ourselves if it's good to remove *"City"* and *"Postal Code"* columns. We will consider here that even they have a lot of missing values; we will keep them because they don't represent a highly determinant information but only something good to know when given.

Identify incomplete data

```
empty_rows= customers_with_errors[customers_with_errors.isnull().all(axis=1)]
print("Number of empty rows:",len(empty_rows))
```

```
Number of empty rows: 0
```

General info of the dataset

```
customers_with_errors.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40 entries, 0 to 39
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Customer ID        40 non-null     object 
 1   Title              32 non-null     object 
 2   First Name         40 non-null     object 
 3   Middle Name        40 non-null     object 
 4   Last Name          40 non-null     object 
 5   Email              39 non-null     object 
 6   Phone Number       38 non-null     object 
 7   Street             38 non-null     object 
 8   City               20 non-null     object 
 9   Postal Code        18 non-null     float64
 10  Country            40 non-null     object 
 11  Birthday           40 non-null     object 
 12  Subscription Date  40 non-null     object 
 13  Update Date        40 non-null     object 
dtypes: float64(1), object(13)
memory usage: 4.5+ KB
```

The change we will make:

Postal Code → into int format (would be a good idea because it uses less stockage space and is more appropriate here; but in Python NaN values are represented as floats so we have to proceed another way)

Birthday / Subscription Date / Update Date → into date format

```
customers_with_errors['Postal Code'] = customers_with_errors['Postal Code'].apply(lambda x: int(x) if pd.notnull(x) else pd.NA)
customers_with_errors['Postal Code'] = customers_with_errors['Postal Code'].astype('Int64')
customers_with_errors['Birthday'] = pd.to_datetime(customers_with_errors['Birthday'], errors='coerce', format='%d/%m/%Y')
customers_with_errors['Subscription Date'] = pd.to_datetime(customers_with_errors['Subscription Date'], errors='coerce', format='%d/%m/%Y')
customers_with_errors['Update Date'] = pd.to_datetime(customers_with_errors['Update Date'], errors='coerce', format='%d/%m/%Y')
customers_with_errors.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40 entries, 0 to 39
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Customer ID        40 non-null     object
 1   Title              32 non-null     object
 2   First Name         40 non-null     object
 3   Middle Name        40 non-null     object
 4   Last Name          40 non-null     object
 5   Email              39 non-null     object
 6   Phone Number       38 non-null     object
 7   Street             38 non-null     object
 8   City               20 non-null     object
 9   Postal Code        18 non-null     Int64
 10  Country            40 non-null     object
 11  Birthday           40 non-null     datetime64[ns]
 12  Subscription Date  40 non-null     datetime64[ns]
 13  Update Date        40 non-null     datetime64[ns]
dtypes: Int64(1), datetime64[ns](3), object(10)
memory usage: 4.5+ KB
```

Encryption Process

We don't want all our customer to view personal data that don't belong to them.

Title: Public

ID, Name, Subscription Date, Update Date: Internal

Email, Phone, Address, Birthday: Confidential

```
customers_with_errors['Email'] = encrypt_column(customers_with_errors['Email'])
customers_with_errors['Phone Number'] = encrypt_column(customers_with_errors['Phone Number'])
customers_with_errors['Birthday'] = encrypt_column(customers_with_errors['Birthday'])
customers_with_errors['Street'] = encrypt_column(customers_with_errors['Street'])
customers_with_errors['City'] = encrypt_column(customers_with_errors['City'])
customers_with_errors['Postal Code'] = encrypt_column(customers_with_errors['Postal Code'])
customers_with_errors['Country'] = encrypt_column(customers_with_errors['Country'])
customers_with_errors['Street'].head()
```

```
0    gAAAAABnis_dJsLiSiH_HHV9XFcvViYc3uyfs9k58txACb...
1    gAAAAABnis_dOT7H8fXTBMXAQBXtbcPl3mL0rrUDaeK1Yo...
2    gAAAAABnis_dD9IoDML3fYtcflZJtV3c9bg6KgA1B1n_Xw...
3    gAAAAABnis_dfRD5GA1yz_15lgQypAopzNNHGEIHe_0RMA...
4    gAAAAABnis_dvxTWjyROl7W4O4_YG8y02PSTg7qA0wegos...
Name: Street, dtype: object
```

```
# Decrypt the 'Street' column for demonstration
customers_with_errors['Street'] = decrypt_column(customers_with_errors['Street'])
customers_with_errors['Street'].head()
```

```
0                75944 John Forges
1                  34633 Sosa Fork
2    8474 Crystal Unions Suite 449
3    8603 Scott Turnpike Suite 266
4          074 Ryan Loaf Suite 615
Name: Street, dtype: object
```

```
# Encrypt the 'Street' column again
customers_with_errors['Street'] = encrypt_column(customers_with_errors['Street'])
customers_with_errors['Street'].head()
```

```
0    gAAAAABnis_dTVLwCxsKIqcHREZ4JhHF2Mnj2WTroMLFdl...
1    gAAAAABnis_d2SvarNyx7byi6k57YUlm1DBUlGDfSvbCO9...
2    gAAAAABnis_dZza3hh8Y7kuUsnA-ULfgvIz1Vm_iuGlJSi...
3    gAAAAABnis_dkTCU6L-97v88GjQ27O9TqTfwoyXPPM6p0a...
4    gAAAAABnis_dSMjFeWBtgdEPmjn8zAuCp9FpwB03S4TlO8...
Name: Street, dtype: object
```

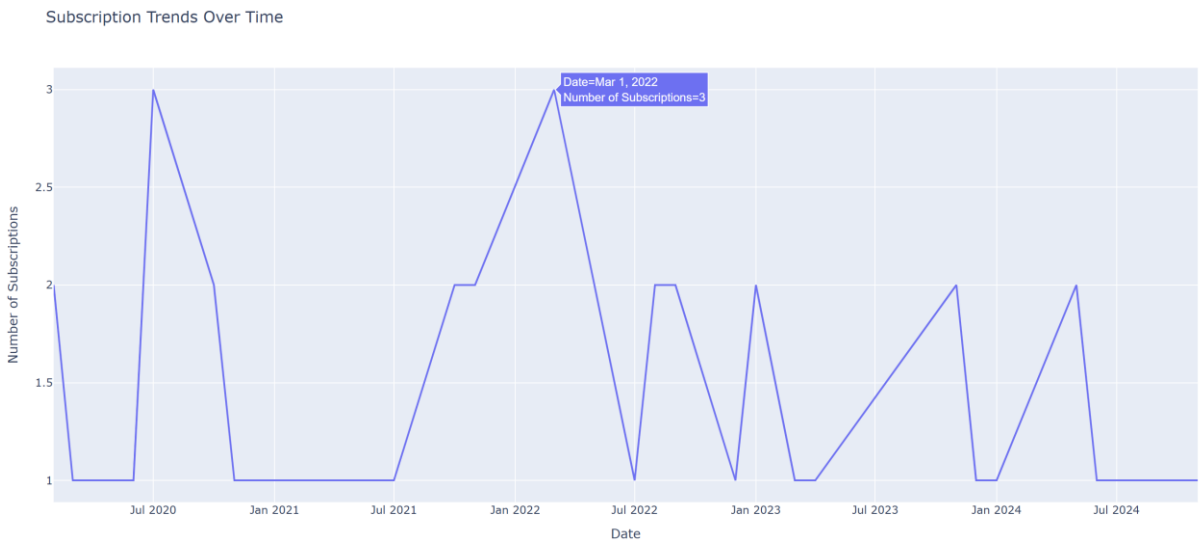
Visualizations

```
import plotly.io as pio

# Configure the renderer
pio.renderers.default = 'vscode'

customers_with_errors['Country'] = decrypt_column(customers_with_errors['Country'])

# Count the number of customers in each country
country_counts = customers_with_errors['Country'].value_counts().reset_index()
country_counts.columns = ['Country', 'Count']

# Plot
customers_by_country = px.bar(
    country_counts, x='Country', y='Count',
    title='Customer Distribution by Country',
    labels={'Country': 'Country', 'Count': 'Number of Customers'}, color='Count')

customers_by_country.show()

# Encrypt the 'Country' column again
customers_with_errors['Country'] = encrypt_column(customers_with_errors['Country'])
```


```
import os

# Create the folder and save the file into it
os.makedirs('visualizations', exist_ok=True)

# Save the figure to an HTML file
pio.write_html(customers_by_country, file='visualizations/customers_by_country.html',
auto_open=True)
```

```
# Convert 'Subscription Date' to datetime if not already
customers_with_errors['Subscription Date'] =
pd.to_datetime(customers_with_errors['Subscription Date'])

# Group by month and count subscriptions
subscription_trends = customers_with_errors.groupby(customers_with_errors['Subscription
Date'].dt.to_period('M')).size().reset_index(name='Count')

# Convert period to datetime for plotting
subscription_trends['Subscription Date'] = subscription_trends['Subscription
Date'].dt.to_timestamp()

# Plot
subscription_over_time = px.line(
    subscription_trends, x='Subscription Date', y='Count',
    title='Subscription Trends Over Time',
    labels={'Subscription Date': 'Date', 'Count': 'Number of Subscriptions'})
```

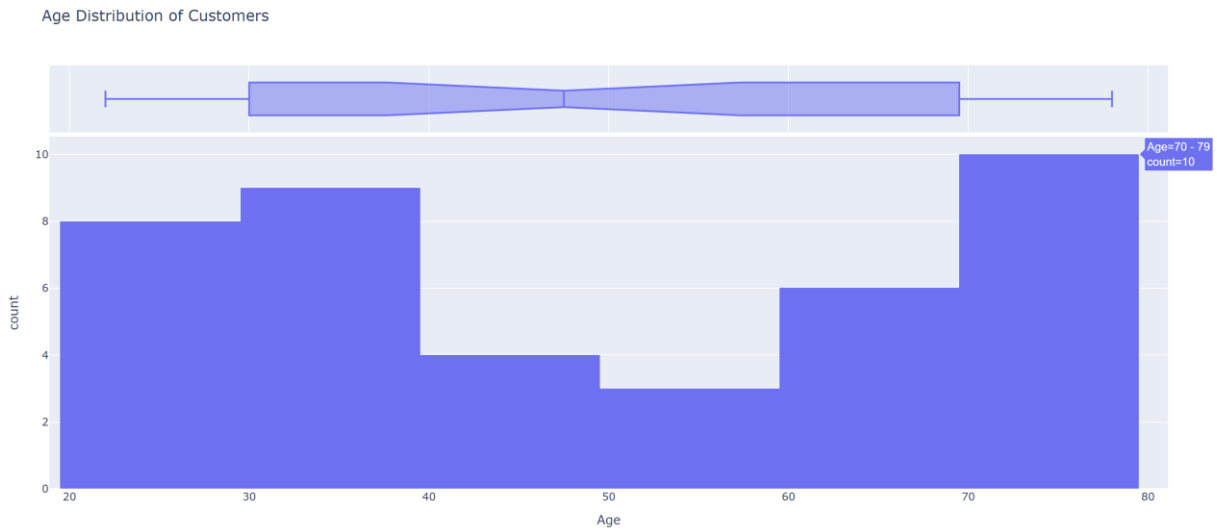
```
subscription_over_time.show()
```




```
# Save the figure to an HTML file
pio.write_html(subscription_over_time, file='visualizations/subscription_over_time.html',
auto_open=True)
```

```
from datetime import datetime

# Calculate age from the 'Birthday' column
customers_with_errors['Birthday'] =
pd.to_datetime(decrypt_column(customers_with_errors['Birthday']))
customers_with_errors['Age'] = customers_with_errors['Birthday'].apply(lambda x:
datetime.now().year - x.year if pd.notna(x) else None)

# Plot age distribution
customers_age_distribution = px.histogram(customers_with_errors, x='Age', nbins=10,
title='Age Distribution of Customers',
                    labels={'Age': 'Age', 'count': 'Number of Customers'}, marginal='box')
customers_age_distribution.show()

# Encrypt the 'Birthday' column again
customers_with_errors['Birthday'] = encrypt_column(customers_with_errors['Birthday'])
```

```
# Delete the 'Age' column to avoid redundancy
customers_with_errors = customers_with_errors.drop(columns='Age')
```




```
# Save the figure to an HTML file
pio.write_html(customers_age_distribution, file='visualizations/customers_age_distribution.html', auto_open=True)
```

We save the cleaned dataset in a new file.

```
# Create the folder and save the file into it
os.makedirs('cleaned_datasets', exist_ok=True)
customers_with_errors.to_csv("cleaned_datasets/cleaned_customers.csv", sep=';', index=False)
```

Second dataset: *deco_sales*

```
deco_sales.head()
```

	Product ID	Customer ID	Transaction Date	Currency	Amount	Quantity	Update Date
0	1	CUST001	2024-11-05	USD	250.0	10	2024-11-06
1	2	CUST002	2024-11-07	USD	220.0	4	2024-11-08
2	3	CUST003	2024-11-10	EUR	72.0	4	2024-11-12
3	4	CUST004	2024-11-09	EUR	60.0	4	2024-11-10
4	5	CUST005	2024-11-11	USD	150.0	5	2024-11-12

```
deco_sales.describe(include='all')
```

	Product ID	Customer ID	Transaction Date	Currency	Amount	Quantity	Update Date
count	70.000000	70	70	70	70.000000	70.000000	70
unique	NaN	50	39	3	NaN	NaN	39
top	NaN	CUST001	2024-11-10	USD	NaN	NaN	2024-11-13
freq	NaN	2	7	35	NaN	NaN	5
mean	9.785714	NaN	NaN	NaN	344.257143	6.257143	NaN
std	5.763184	NaN	NaN	NaN	306.326466	5.827684	NaN
min	1.000000	NaN	NaN	NaN	0.000000	1.000000	NaN
25%	5.000000	NaN	NaN	NaN	120.000000	3.000000	NaN
50%	9.000000	NaN	NaN	NaN	250.000000	4.000000	NaN
75%	14.750000	NaN	NaN	NaN	480.000000	8.000000	NaN
max	20.000000	NaN	NaN	NaN	1350.000000	30.000000	NaN

Completeness: Identify missing data

```
missing_values=deco_sales.isnull().sum()
missing_percent= ((missing_values) / len(deco_sales)) * 100

print(round(missing_percent,2))
```

```
Product ID          0.0
Customer ID         0.0
Transaction Date    0.0
Currency            0.0
Amount              0.0
Quantity            0.0
Update Date         0.0
dtype: float64
```

Identify incomplete data

```
empty_rows= deco_sales[deco_sales.isnull().all(axis=1)]
print("Number of empty rows:",len(empty_rows))
```

```
Number of empty rows: 0
```

General info of the dataset

```
deco_sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70 entries, 0 to 69
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Product ID        70 non-null     int64  
 1   Customer ID       70 non-null     object 
 2   Transaction Date  70 non-null     object 
 3   Currency          70 non-null     object 
 4   Amount            70 non-null     float64
 5   Quantity          70 non-null     int64  
 6   Update Date       70 non-null     object 
dtypes: float64(1), int64(2), object(4)
memory usage: 4.0+ KB
```

The change we will make:

Transaction Date / Update Date → into date format

```
deco_sales['Transaction Date'] = pd.to_datetime(deco_sales['Transaction Date'], errors='coerce', format='%Y-%m-%d')
deco_sales['Update Date'] = pd.to_datetime(deco_sales['Update Date'], errors='coerce', format='%Y-%m-%d')
deco_sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70 entries, 0 to 69
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Product ID        70 non-null     int64
 1   Customer ID       70 non-null     object
 2   Transaction Date  70 non-null     datetime64[ns]
 3   Currency          70 non-null     object
 4   Amount            70 non-null     float64
 5   Quantity          70 non-null     int64
 6   Update Date       70 non-null     datetime64[ns]
dtypes: datetime64[ns](2), float64(1), int64(2), object(2)
memory usage: 4.0+ KB
```

Encryption Process

Personal data related to a transaction between the website and the bank should be restricted as it can be detournated by malveillant people.

IDProd, Currency: Public

IDCustomer, Update Date: Internal

Amount, Quantity: Confidential

Transaction Date: Restricted

```
deco_sales['Amount'] = encrypt_column(deco_sales['Amount'])
deco_sales['Quantity'] = encrypt_column(deco_sales['Quantity'])
deco_sales['Transaction Date'] = encrypt_column(deco_sales['Transaction Date'])
deco_sales['Amount'].head()
```

```
0    gAAAAABnis_fe-BPA8--odrT33LNvTf2R5GSNUkEeKVBie...
1    gAAAAABnis_f0uvRqp_i6CrKk8JNftRsuagv1mMSkexz7d...
2    gAAAAABnis_fzRPNmkCNdFk-jZuD-cN5rklRgFHRmRGPKr...
3    gAAAAABnis_f3v_P1gc7uXnliN7ka5rNwtcBwLtxLB5o_Z...
4    gAAAAABnis_fMiPo5t0Rp9prwnd2_IG7HcAlMZBNOsi-y9...
Name: Amount, dtype: object
```

```
# Decrypt an 'Amount' element for demonstration
deco_sales['Amount'] = decrypt_column(deco_sales['Amount'])
deco_sales['Amount'].head()
```

```
0    250.0
1    220.0
2     72.0
3     60.0
4    150.0
Name: Amount, dtype: object
```

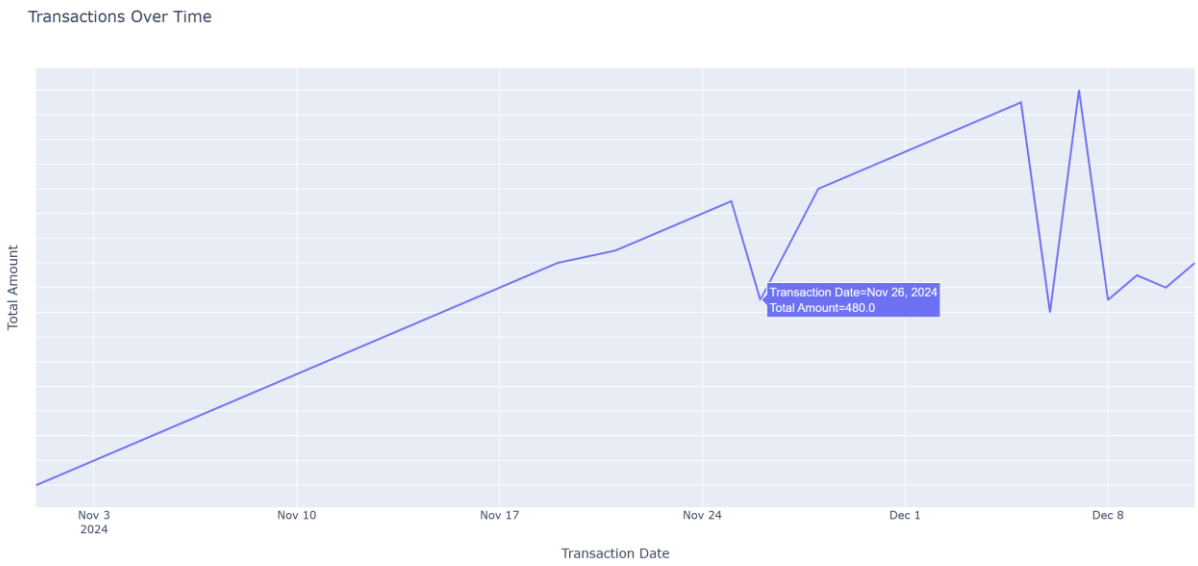
```
# Encrypt the 'Amount' column again
deco_sales['Amount'] = encrypt_column(deco_sales['Amount'])
deco_sales['Amount'].head()
```

```
0    gAAAAABnis_flofGs5KzzHlBn79uFZdsLGRIgpmE_sV6wa...
1    gAAAAABnis_fp6JrUp4knjwrvEt0OZ7p4JsMeXlNCdvP-P...
2    gAAAAABnis_fTPwDUakASzXS-YlLcdYCbb5UUl0MPaOdU4...
3    gAAAAABnis_ftVs4q7KpTnQXtgDrHzIhYHEPLEkqhn_9vt...
4    gAAAAABnis_fmisCYeTv3Frd4r328848e3SQqmpxX0cSo2...
Name: Amount, dtype: object
```

Visualizations

```
# Convert 'Transaction Date' to datetime
deco_sales['Transaction Date'] = pd.to_datetime(decrypt_column(deco_sales['Transaction
Date']))
deco_sales['Amount'] = decrypt_column(deco_sales['Amount'])

# Group by Transaction Date and sum the Amount
daily_transactions = deco_sales.groupby('Transaction Date')['Amount'].sum().reset_index()

# Create line chart
transaction_over_time = px.line(
    daily_transactions, x='Transaction Date', y='Amount',
    title='Transactions Over Time',
    labels={'Transaction Date': 'Transaction Date', 'Amount': 'Total Amount'})
```

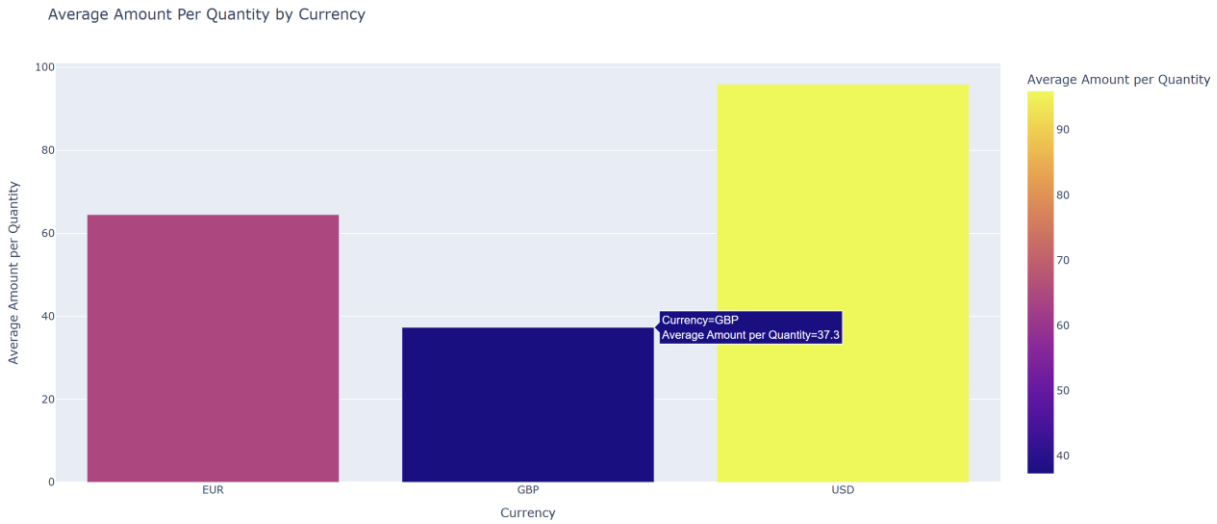
```
# Hide the y-axis numbers
transaction_over_time.update_yaxes(showticklabels=False)
transaction_over_time.show()
# Encrypt the 'Transaction Date' and the 'Amount' columns again
deco_sales['Transaction Date'] = encrypt_column(deco_sales['Transaction Date'])
deco_sales['Amount'] = encrypt_column(deco_sales['Amount'])
```




```
# Save the figure to an HTML file
pio.write_html(transaction_over_time, file='visualizations/transaction_over_time.html',
auto_open=True)
```

```
# Decrypt the 'Amount' and 'Quantity' columns for calculation
deco_sales['Amount'] = decrypt_column(deco_sales['Amount']).astype(float)
deco_sales['Quantity'] = decrypt_column(deco_sales['Quantity']).astype(float)

# Calculate the average amount per quantity for each row
deco_sales['Amount per Quantity'] = deco_sales['Amount'] / deco_sales['Quantity']

# Group by Currency and calculate the mean of 'Amount per Quantity'
currency_avg = deco_sales.groupby('Currency')['Amount per Quantity'].mean().reset_index()

# Create bar chart
avg_amount_per_quantity = px.bar(
```

```
    currency_avg, x='Currency', y='Amount per Quantity',
    title='Average Amount Per Quantity by Currency',
    labels={'Currency': 'Currency', 'Amount per Quantity': 'Average Amount per Quantity'},
    color='Amount per Quantity')

avg_amount_per_quantity.show()
# Encrypt the 'Amount' and 'Quantity' columns again
deco_sales['Amount'] = encrypt_column(deco_sales['Amount'])
deco_sales['Quantity'] = encrypt_column(deco_sales['Quantity'])

# Delete the 'Amount per Quantity' column
deco_sales = deco_sales.drop(columns='Amount per Quantity')
```




```
# Save the figure to an HTML file
pio.write_html(avg_amount_per_quantity, file='visualizations/avg_amount_per_quantity.html',
auto_open=True)
```

```
deco_sales.to_csv("cleaned_datasets/cleaned_deco_sales.csv", sep=';', index=False)
```

Third dataset: *deco_product*

```
deco_product.head()
```

	Product ID	Product Name	Category	Starting Price	Selling Price	Currency	Current Stock	Supplier	Update Date	Availability
0	1	Decorative Vase	Vases	15.0	25.0	USD	100	HomeDecor Inc.	01/11/2024	Available
1	2	Elegant Wall Mirror	Mirrors	50.0	55.0	USD	50	Luxury Living	01/11/2024	Available
2	3	Wooden Frame	Frames	12.0	18.0	EUR	200	Rustic Charm	15/10/2024	Available
3	4	Candle Holder	Candles	10.0	15.0	EUR	150	Cozy Creations	20/10/2024	Available
4	5	Throw Pillow	Pillows	25.0	30.0	USD	100	Plush Home	10/11/2024	Out of Stock

```
deco_product.describe(include='all')
```

	Product ID	Product Name	Category	Starting Price	Selling Price	Currency	Current Stock	Supplier	Update Date	Availability
count	20.00000	20	20	20.000000	20.000000	20	20	20	20	20
unique	NaN	20	16	NaN	NaN	3	15	19	14	3
top	NaN	Decorative Vase	Vases	NaN	NaN	USD	100	HomeDecor Inc.	01/11/2024	Available
freq	NaN	1	2	NaN	NaN	10	2	2	3	14
mean	10.50000	NaN	NaN	59.350000	80.500000	NaN	NaN	NaN	NaN	NaN
std	5.91608	NaN	NaN	68.698636	99.724093	NaN	NaN	NaN	NaN	NaN
min	1.00000	NaN	NaN	10.000000	15.000000	NaN	NaN	NaN	NaN	NaN
25%	5.75000	NaN	NaN	20.000000	28.750000	NaN	NaN	NaN	NaN	NaN
50%	10.50000	NaN	NaN	32.500000	45.000000	NaN	NaN	NaN	NaN	NaN
75%	15.25000	NaN	NaN	75.000000	88.750000	NaN	NaN	NaN	NaN	NaN
max	20.00000	NaN	NaN	300.000000	450.000000	NaN	NaN	NaN	NaN	NaN

Completeness: Identify missing data

```
missing_values=deco_product.isnull().sum()
missing_percent= ((missing_values) / len(deco_product)) * 100
print(round(missing_percent,2))
```

```
Product ID         0.0
Product Name       0.0
Category           0.0
Starting Price     0.0
Selling Price      0.0
Currency           0.0
Current Stock      0.0
Supplier           0.0
Update Date        0.0
Availability       0.0
dtype: float64
```

Identify incomplete data

```
empty_rows= deco_product[deco_product.isnull().all(axis=1)]
print("Number of empty rows:",len(empty_rows))
```

```
Number of empty rows: 0
```

General info of the dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Product ID      20 non-null     int64  
 1   Product Name    20 non-null     object 
 2   Category        20 non-null     object 
 3   Starting Price  20 non-null     float64
 4   Selling Price   20 non-null     float64
 5   Currency        20 non-null     object 
 6   Current Stock   20 non-null     object 
 7   Supplier        20 non-null     object 
 8   Update Date     20 non-null     object 
 9   Availability    20 non-null     object 
dtypes: float64(2), int64(1), object(7)
memory usage: 1.7+ KB
```

The change we will make:

Update Date → into date format

```
deco_product['Update Date'] = pd.to_datetime(deco_sales['Update Date'], errors='coerce', format='%d/%m/%Y')
```

```
deco_product.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Product ID      20 non-null     int64
 1   Product Name    20 non-null     object
 2   Category        20 non-null     object
 3   Starting Price  20 non-null     float64
 4   Selling Price   20 non-null     float64
 5   Currency        20 non-null     object
 6   Current Stock   20 non-null     object
 7   Supplier        20 non-null     object
 8   Update Date     20 non-null     datetime64[ns]
 9   Availability    20 non-null     object
dtypes: datetime64[ns](1), float64(2), int64(1), object(6)
memory usage: 1.7+ KB
```

Encryption Process

The customers must have access to the price and the availability for example to be able to purchase

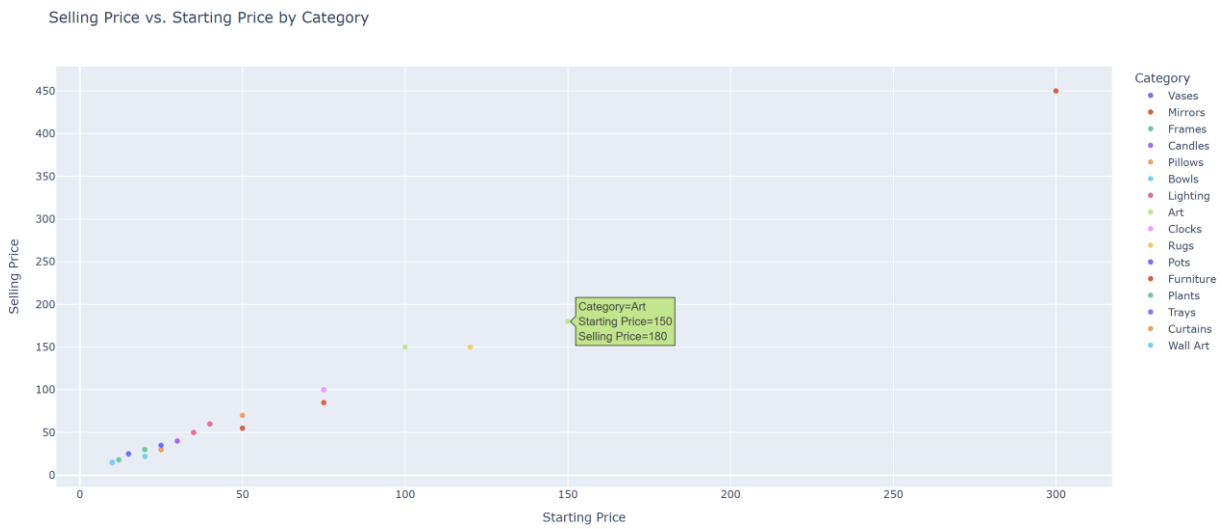
ID, Name, Category, Selling Price, Availability: Public

Starting Price, Current Stock, Supplier, Update Date: Internal

Here we don't need to perform Encryption Process because we don't have any sensitive data (confidential or restricted).

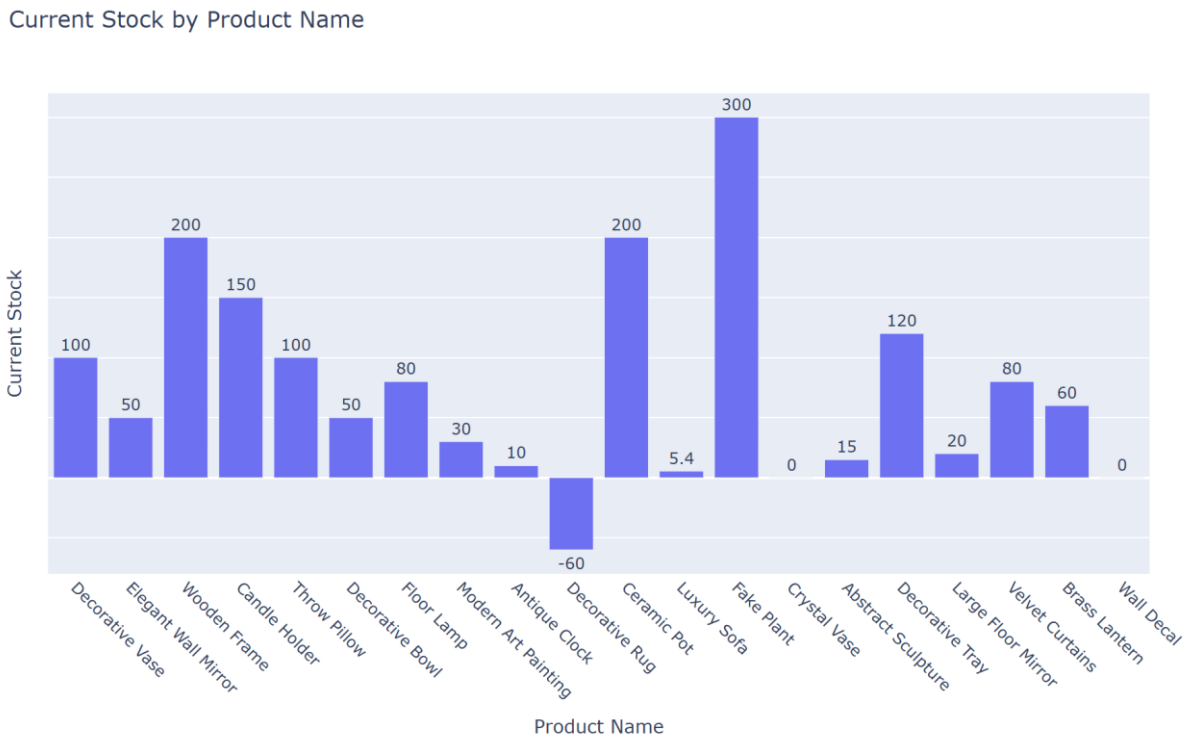
Visualizations

```
# Scatter plot for Selling Price vs. Starting Price by Category
selling_starting_price_category = px.scatter(
    deco_product, x='Starting Price', y='Selling Price', color='Category',
    title='Selling Price vs. Starting Price by Category',
    labels={'Starting Price': 'Starting Price', 'Selling Price': 'Selling Price'})

selling_starting_price_category.show()
```

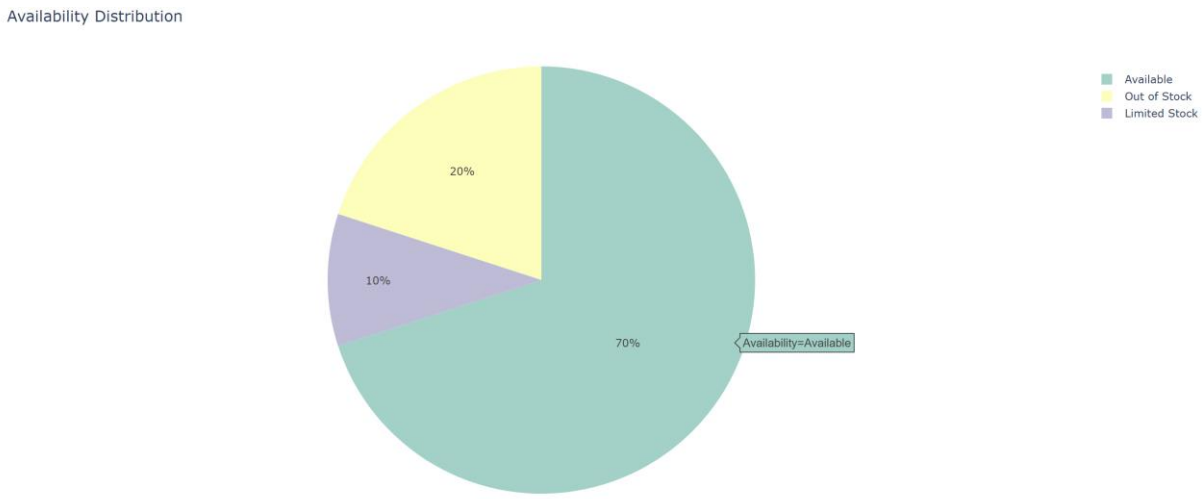



```
# Save the figure to an HTML file
pio.write_html(selling_starting_price_category,
file='visualizations/selling_starting_price_category.html', auto_open=True)
```

```
# Convert 'Current Stock' to numeric
deco_product['Current Stock'] = pd.to_numeric(deco_product['Current Stock'])
# Bar chart for Current Stock by Product Name
current_stock_by_product = px.bar(
    deco_product, x='Product Name', y='Current Stock',
    title='Current Stock by Product Name',
    labels={'Product Name': 'Product Name', 'Current Stock': 'Current Stock'},
    text='Current Stock', width=1000, height=600)

# Hide the y-axis numbers
current_stock_by_product.update_yaxes(showticklabels=False)

# Rotate the x-axis labels
current_stock_by_product.update_xaxes(tickangle=45)

# Show the current stock on top of each bar
current_stock_by_product.update_traces(texttemplate='%{text}', textposition='outside')
current_stock_by_product.show()
```




```
# Save the figure to an HTML file
pio.write_html(current_stock_by_product, file='visualizations/current_stock_by_product.html',
auto_open=True)
# Pie chart for Availability Distribution
availability_distribution = px.pie(
    deco_product, names = 'Availability',
    title = 'Availability Distribution',
    color_discrete_sequence = px.colors.qualitative.Set3)
availability_distribution.show()
```

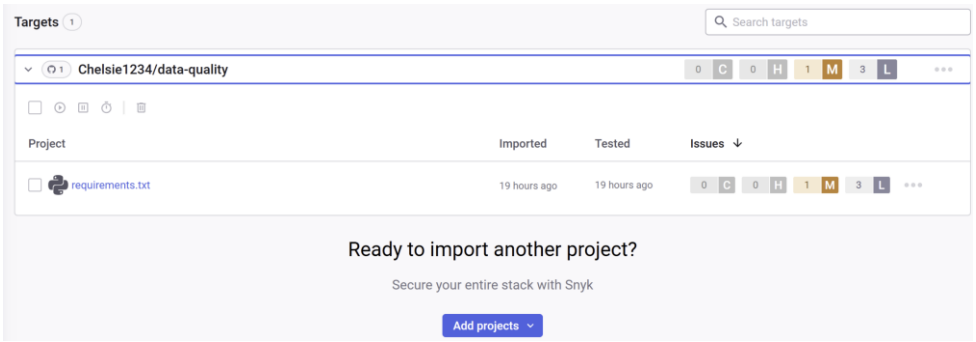


```
# Save the figure to an HTML file
pio.write_html(availability_distribution, file='visualizations/availability_distribution.html',
auto_open=True)
```

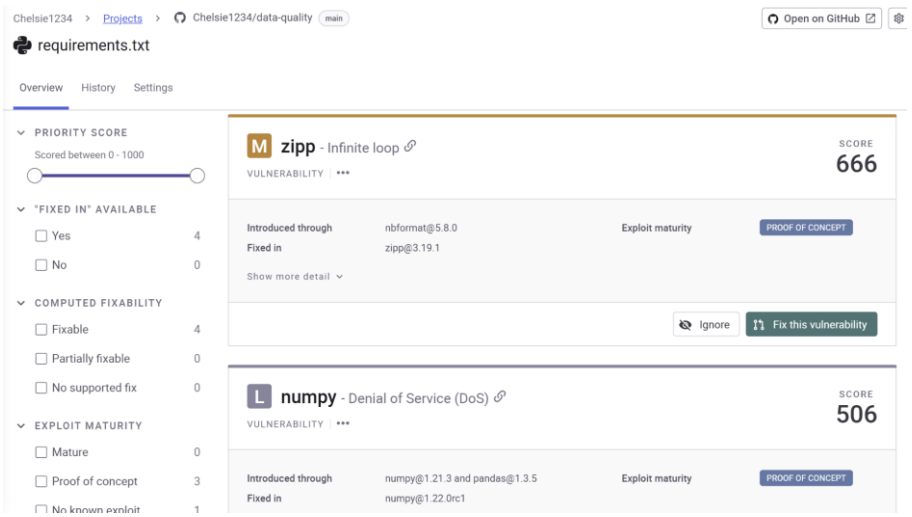
```
deco_product.to_csv("cleaned_datasets/cleaned_deco_product.csv", sep=';', index=False)
```

Last Code Review

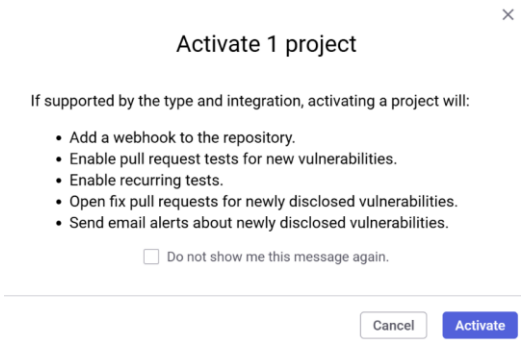
We used *Snyk* for the code review as it is a free online tool and very easy to use and integrate with GitHub.



All the vulnerabilities are in the requirements file and are about the version of 4 library including: zipp and numpy.

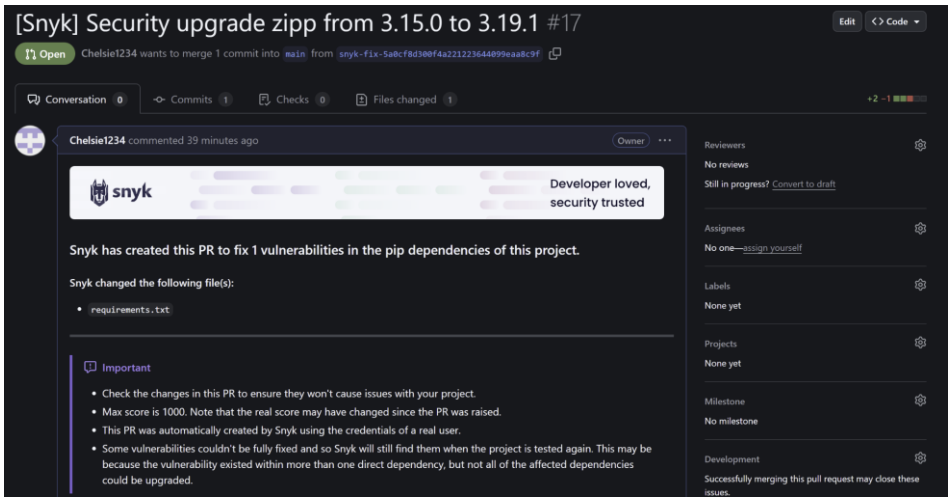


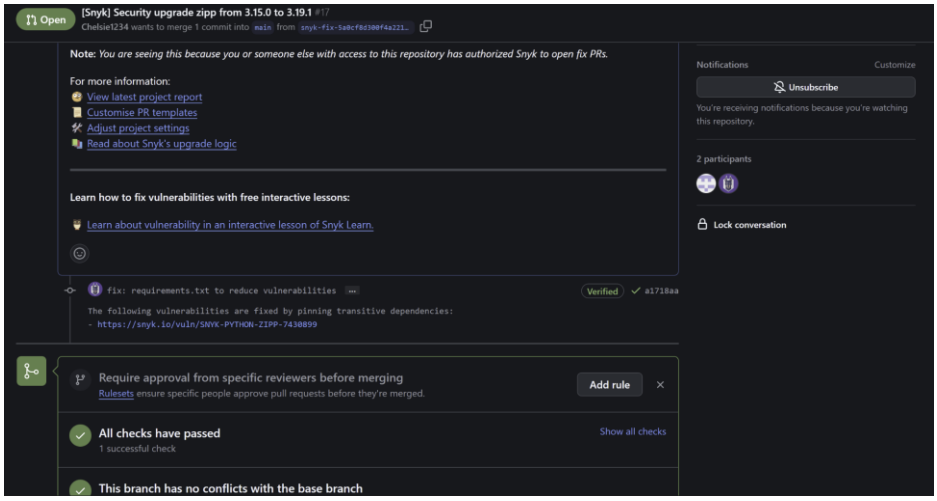
As it isn't big vulnerabilities, we will choose to fix only the first one to test and we activate the Snyk extension in our GitHub Repository to actively enable recurring tests to be up to date from the security point of view.



It has generated a Pull Request in our GitHub Repo and no merge conflict has been detected, so we can securely merge into the main branch.

A good practice would be to click on edit to merge the incoming changes to the "features" branch and not directly into the main branch.





After choosing to resolve only the medium vulnerabilities concerning zipp, we saw now that we have solved all the other one.

JOB DONE!