RESTER LIVRE

ANALYSE DES DONNEES ECOMMERCE

Décembre 2018, par Nicolas Deschamps de Boishebert pour le projet n°4 de la formation « Data Analyst » d'OpenClassrooms

1. Nettoyage des données

Products Table

Une valeur aberrante sur le produit « T_0 ».

Le prix est négatif pour cette référence produit.

	id_prod	price	categ
0	0_1421	19.99	0
1	0_1368	5.13	0
2	0_731	17.99	0
3	1_587	4.99	1
4	0_1507	3.99	0
5	0_1163	9.99	0
6	1_463	36.99	1
7	0_2157	34.99	0
8	0_1915	16.99	0
9	0_389	18.99	0

```
Lower price: -1.0
Max price: 300.0
```

```
Rows with an empty value:
```

id_prod	price	categ

```
Zero and negative prices:
```

	id_prod	price	categ
731	T_0	-1.0	0

Customers Table

Rien à signaler, les dates de naissance sont toutes valides, le sexe également.

	client_id	sex	birth
0	c_4410	f	1967
1	c_7839	f	1975
2	c_1699	f	1984
3	c_5961	f	1962
4	c_5320	m	1943
5	c_415	m	1993
6	c_285	f	1967
7	c_160	f	1978
8	c_6446	m	1971
9	c_6866	m	1993

```
Oldest customer: 1929
Youngest customer: 2004

Customers with unknowned sex:
```

client_id	sex	birth

```
Rows with an empty value:
```

id_prod	price	categ

Transactions Table

200 transactions ont une date qui commence par « test_ ».

Valeurs rattachées concernées :

- Session « s_0 »
- Clients « ct_0 » et « ct_1 »
- Produit « T_0 »

	id_prod	date	session_id	client_id
0	0_1483	2021-04-10 18:37:28.723910	s_18746	c_4450
1	2_226	2022-02-03 01:55:53.276402	s_159142	c_277
2	1_374	2021-09-23 15:13:46.938559	s_94290	c_4270
3	0_2186	2021-10-17 03:27:18.783634	s_105936	c_4597
4	0_1351	2021-07-17 20:34:25.800563	s_63642	c_1242

```
Oldest date: 2021-03-01 00:02:26.047414
Newest date: test_2021-03-01 02:30:02.237450


Wrong dates:
200 rows

Wrong dates Sessions ID:

1431    s_0
Name: session_id, dtype: object

Wrong dates Client ID:

1431    ct_1
5955    ct_0
Name: client_id, dtype: object

Wrong dates Product ID:

1431    T_0
Name: id_prod, dtype: object


Rows with an empty value:
```

id_prod	price	categ

Main Dataframe

103 transactions n'ont pas obtenu de fiches produits. Il s'agit des transactions sur le produit « 0_2245 ».

Au vu du faible nombre de lignes concernées, nous les supprimerons.

Après avoir réalisé les différentes analyses univariées et bivariées, il aurait également été possible d'imputer le prix de la fiche manquante par une mesure de tendance centrale, comme la moyenne par exemple.

Main dataframe

```python
main_df = (
            # The main df is based on the transactions dataframe
            transactions_df

                # Merge products dataframe
                .merge(products_df, how='left', on='id_prod')

                # Merge customers dataframe
                .merge(customers_df, how='left', on='client_id')
)

# Check if some transactions have no join
len_null_rows = len(main_df[main_df.isnull().any(axis=1)])
print(f"Number of rows with at least an empty value: {len_null_rows}")

# Check which transaction(s) it s
len_null_products = len(main_df[main_df.isnull().any(axis=1)]['id_prod'].drop_duplicates())
print(f"Number of products with at least an empty value: {len_null_products}")
the_product = main_df[main_df.isnull().any(axis=1)]['id_prod'].drop_duplicates().iloc[0]
print(f"The product is: {the_product}")
```

```
Number of rows with at least an empty value: 103
Number of products with at least an empty value: 1
The product is: 0_2245
```

Main Dataframe

- Exclusion des ventes test
- Conservation des lignes valides et complètes
- Transactions avec fiche produit

~~None~~ ~~negative~~

~~NaN~~

~~Test~~

```python
main_df = (
            # The main df is based on the transactions dataframe
            transactions_df

                # Merge products dataframe
                .merge(products_df, how='left', on='id_prod')

                # Merge customers dataframe
                .merge(customers_df, how='left', on='client_id')

                # Clean date (remove "test" dates)
                .query("date.str.contains('test') == False")

                # Clean prices (remove 0 and negative prices)
                .query("price > 0")

                # Convert data column to datetime type
                .assign(
                    date = lambda x: pd.to_datetime(x.date),
                    categ = lambda x: x.categ.astype(int)
                )
           )

# Create a year column
main_df['year'] = main_df.date.apply(lambda x: x.year)

# Create a month column
main_df['month'] = main_df.date.apply(lambda x: x.month)

# Create a yearmonth column
main_df['yearmonth'] = main_df.date.apply(lambda x: int(str(x.year) + str(x.month).zfill(2)))

# Create a age column
main_df['age'] = 2022 - main_df.birth.astype(int)

# Main DF length
print(f"Len main_df: {len(main_df):,}")

display(main_df.head(10))
```

```
Len main_df: 336,713
```

Main Dataframe

A ce stade, le dataframe semble nettoyé.

Nous n'avons aucune cellule avec des valeurs aberrantes ou manquantes.

Il reste néanmoins un dernier point à traiter.

	id_prod	date	session_id	client_id	price	categ	sex	birth	year	month	yearmonth	age
0	0_1483	2021-04-10 18:37:28.723910	s_18746	c_4450	4.99	0	f	1977	2021	4	202104	45
1	2_226	2022-02-03 01:55:53.276402	s_159142	c_277	65.75	2	f	2000	2022	2	202202	22
2	1_374	2021-09-23 15:13:46.938559	s_94290	c_4270	10.71	1	f	1979	2021	9	202109	43
3	0_2186	2021-10-17 03:27:18.783634	s_105936	c_4597	4.20	0	m	1963	2021	10	202110	59
4	0_1351	2021-07-17 20:34:25.800563	s_63642	c_1242	8.99	0	f	1980	2021	7	202107	42
5	0_1085	2021-09-15 05:47:48.215162	s_90139	c_2526	3.99	0	f	1982	2021	9	202109	40
6	0_1508	2021-08-29 05:39:01.055455	s_82100	c_5799	8.03	0	f	1962	2021	8	202108	60
7	0_1627	2021-09-19 14:54:52.410362	s_92294	c_1422	3.99	0	f	1980	2021	9	202109	42
8	0_1469	2022-02-03 07:13:22.559995	s_159252	c_2207	14.99	0	f	1970	2022	2	202202	52
9	0_1453	2022-02-26 09:03:10.571665	s_171098	c_5433	7.99	0	f	1981	2022	2	202202	41

```python
print("Rows with an empty cell:")
display(empty_val(main_df))
print("Rows with a null cell:")
display(main_df.isnull().any())
```

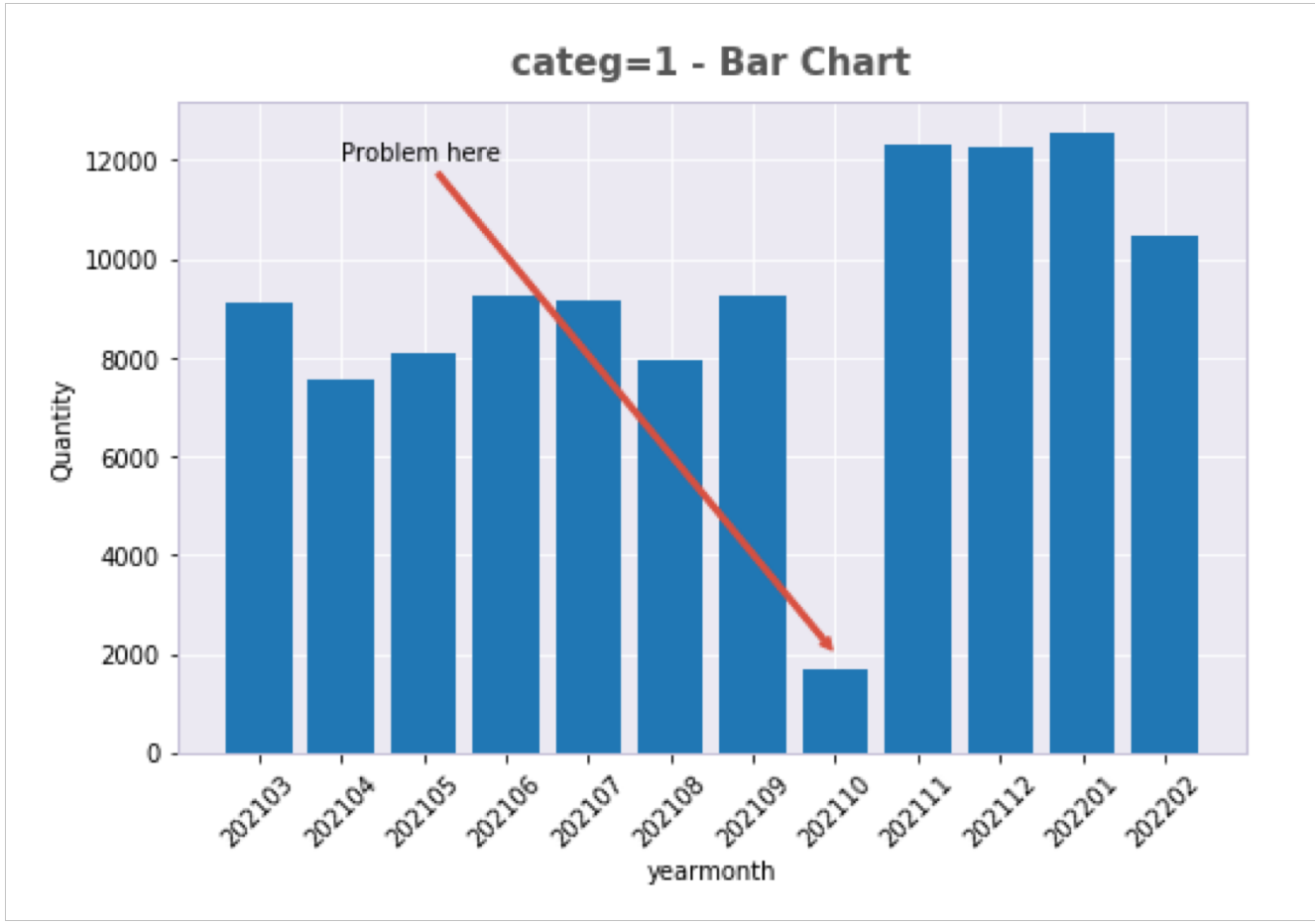
```
Rows with an empty cell:
```

id_prod	date	session_id	client_id	price	categ	sex	birth	year	month	yearmonth	age

```
Columns with a null cell:

id_prod        False
date           False
session_id     False
client_id      False
price          False
categ          False
sex            False
birth          False
year           False
month          False
yearmonth      False
age            False
dtype: bool
```

Category Variable

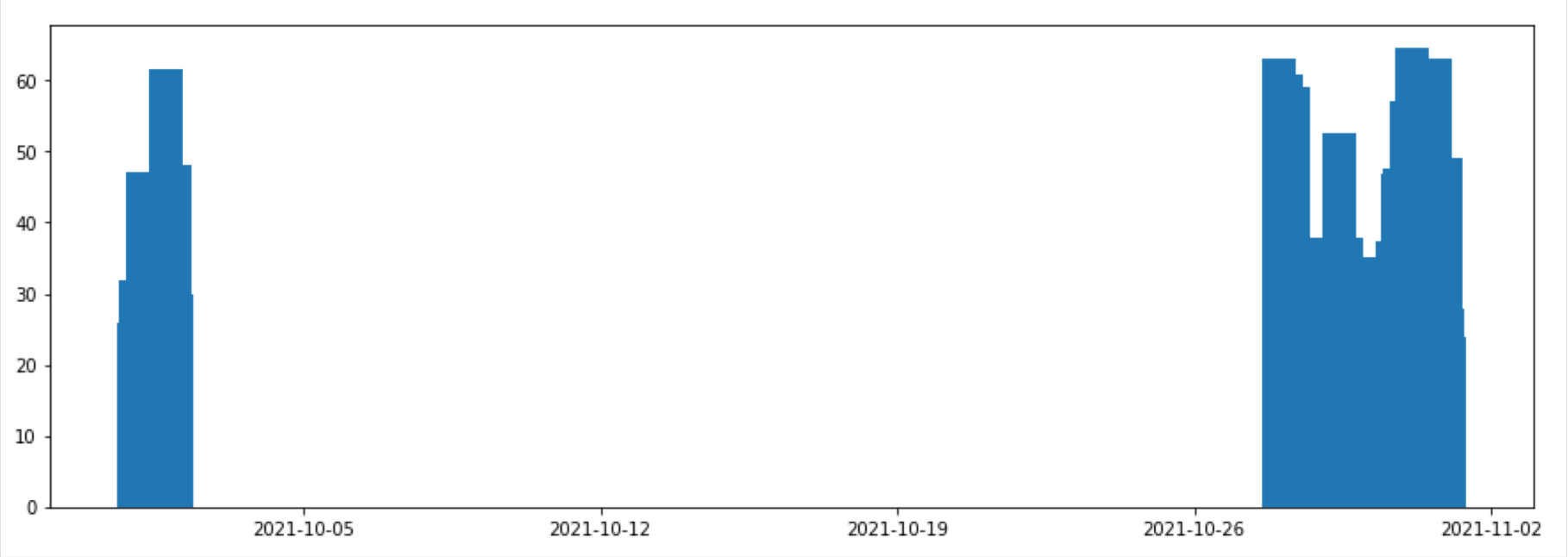
Lors de l'analyse de la distribution des « mois / années », j'ai pu constater un trou important sur le couple mois / année « 202110 » en filtrant sur chaque catégorie.



Category Variable

Le problème s'étale du 01/10/2021 à 23:50 jusqu'au 28/10/2021 à 02:54

```
id_prod                              1_312
date          2021-10-01 23:50:02.475137
session_id                         s_98670
client_id                           c_6741
price                               24.56
categ                                   1
sex                                     f
birth                                1972
year                                 2021
month                                  10
yearmonth                          202110
age                                    50
Name: 311542, dtype: object

id_prod                              1_564
date          2021-10-28 00:02:54.903656
session_id                        s_111093
client_id                            c_650
price                               62.99
categ                                   1
sex                                     f
birth                                1952
year                                 2021
month                                  10
yearmonth                          202110
age                                    70
Name: 161858, dtype: object
```

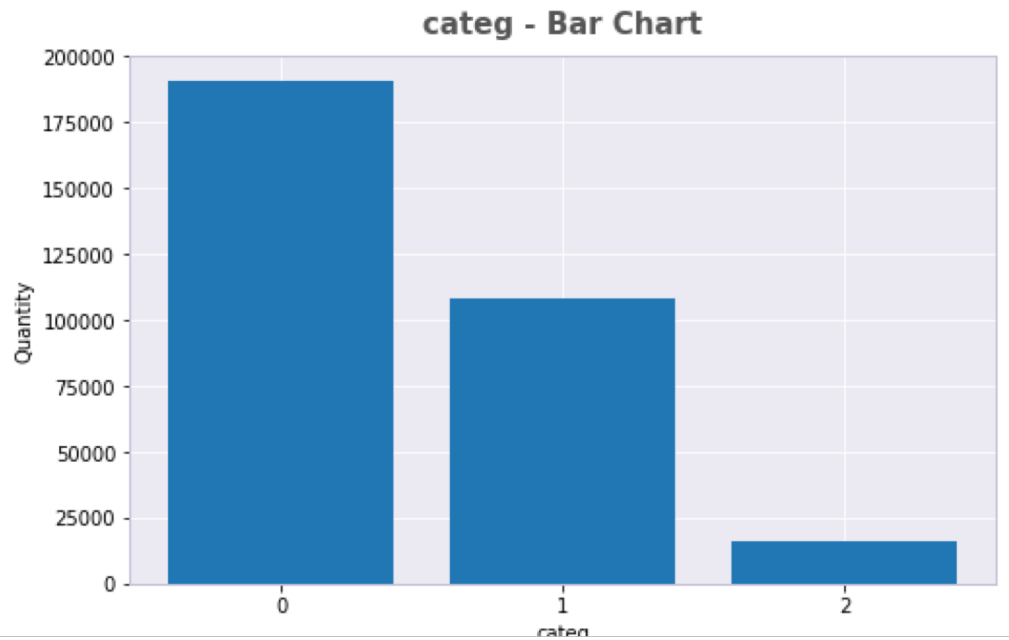


Category Variable

Il s'agissait en réalité d'une rupture de stock. J'ai décidé de conserver les données, exceptées pour :

- L'analyse du volume par catégorie
- L'analyse univariée de la variable « price » pour ne pas fausser la comparaison du C.A. total entre les catégories.

```
data = main_df[main_df['yearmonth'] != 202110].copy()['categ']

categ_univa = Univa(data, quali=True)
categ_univa.distribution()
categ_univa.piechart()
```




```
temp_df = main_df[main_df.yearmonth != 202110]

price_by_cat_univa = Univa(temp_df['price'], filter_var=temp_df['categ'], unit='€')
price_by_cat_univa.describe()
price_by_cat_univa.distribution()
price_by_cat_univa.boxplot()
price_by_cat_univa.lorenz()
```

2. Analyses préliminaires

Category Variable

distribution simple

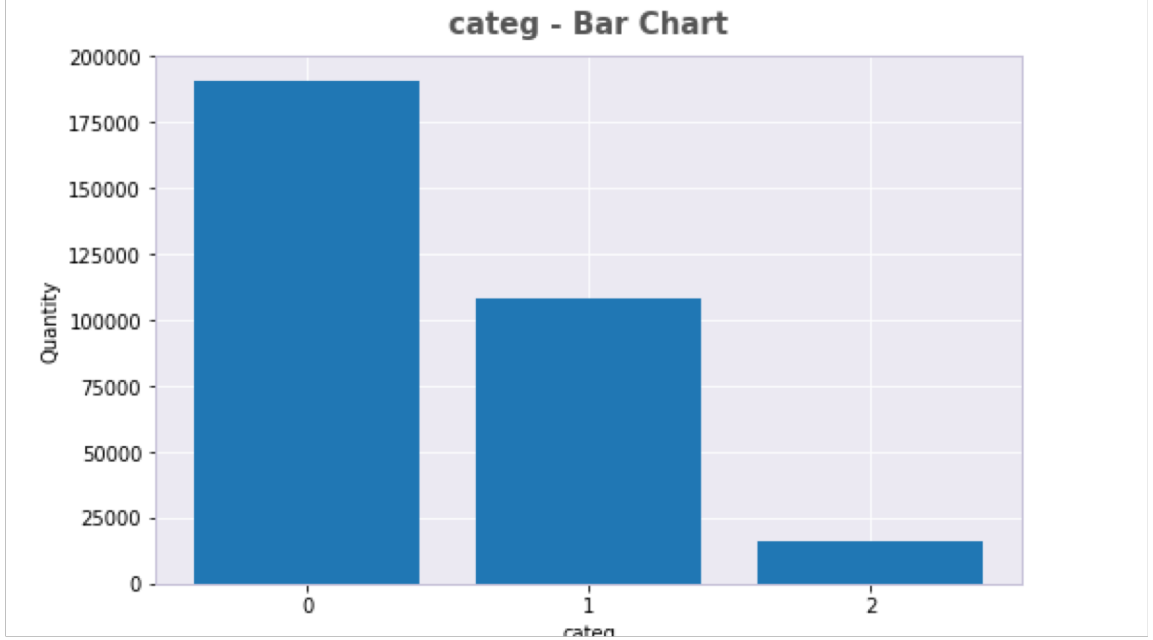
Nombre de transactions par catégorie :

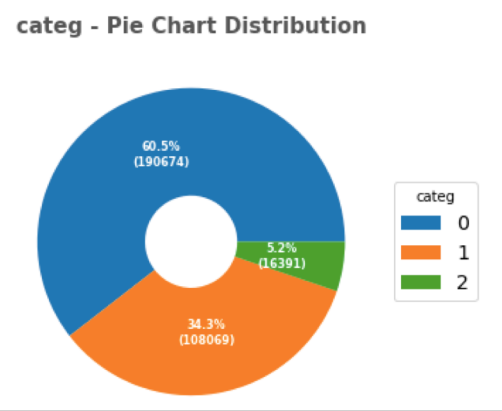
0 : 191k

1 : 109k

2 : 16k

```python
data = main_df[main_df['yearmonth'] != 202110].copy()['categ']

categ_univa = Univa(data, quali=True)
categ_univa.distribution()
categ_univa.piechart()
```





Category Variable

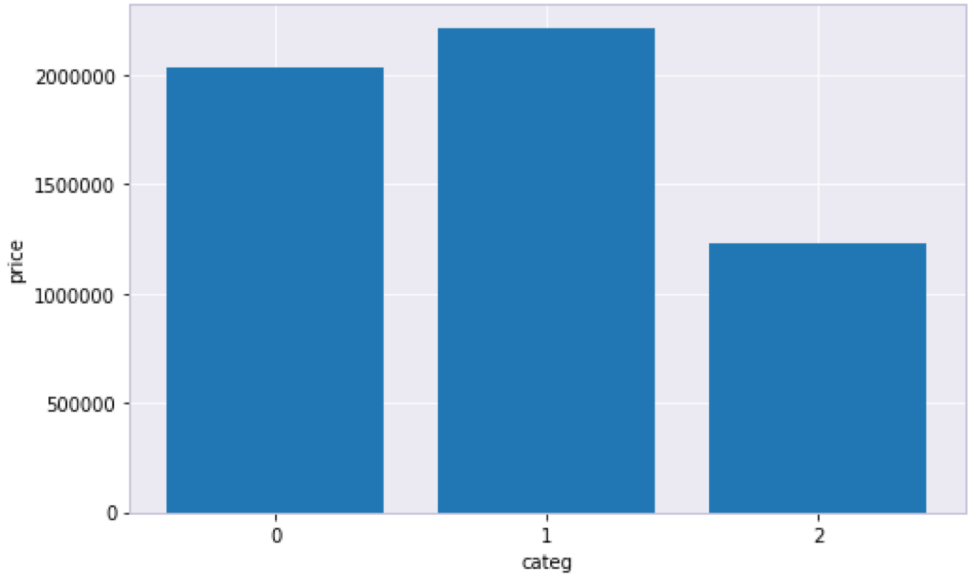
en valeur monétaire

La distribution en valeur ne reflète pas la distribution en volume.

Les prix moyens sont donc dépendants de la catégorie.

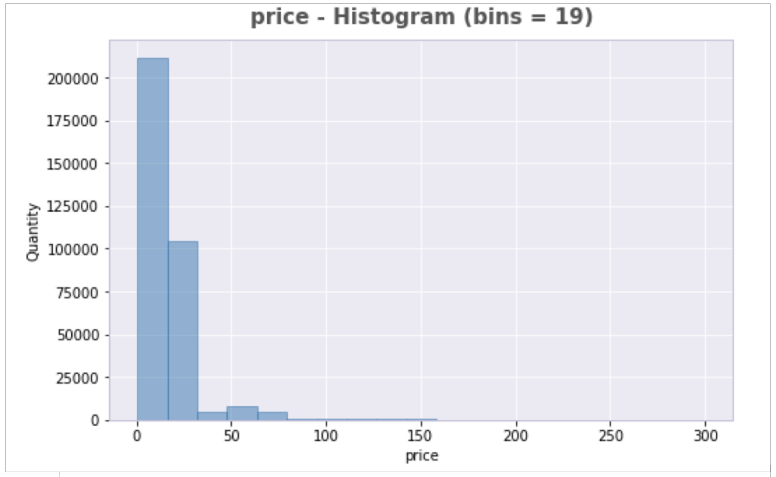
Category Variable - Value

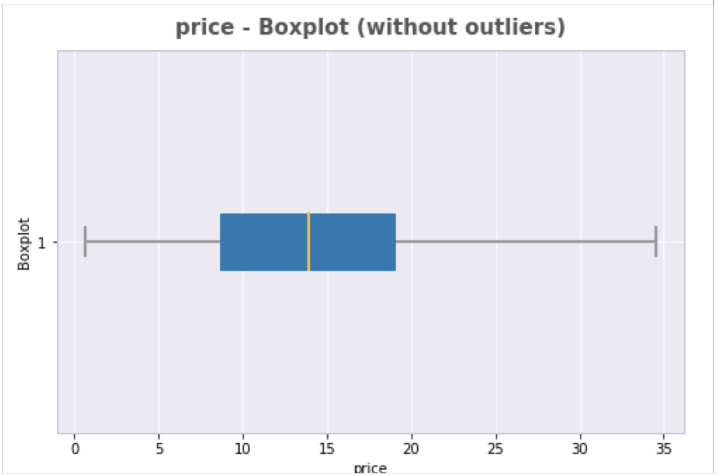
```
data = (main_df[main_df['yearmonth'] != 202110]
            .copy()
            .groupby('categ')
            .sum()
            .reset_index()
        )

# Distribution
fig, ax = MyPlots.new_plot()
MyPlots.bar(ax, x=data['categ'], y=data['price'])
#repere
#data
```

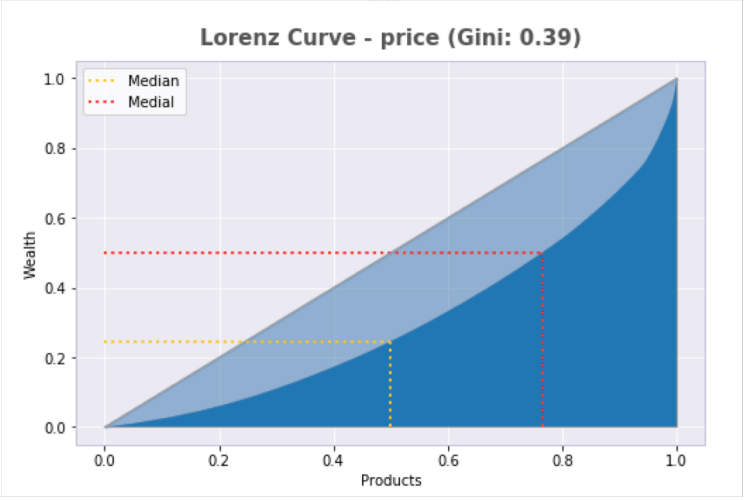


Price Variable

	price
Sample Size	336,713
Total	5,796,578€
Min	0.62€
Max	300.0€
Mode	16€
Mean	17€
Median	14€
Variance (σ^2)	318.82€
Standard Deviation (σ)	17.86€
Coefficient of variation (CV)	1.04
Skewness	5.48
Kurtosis	45.43



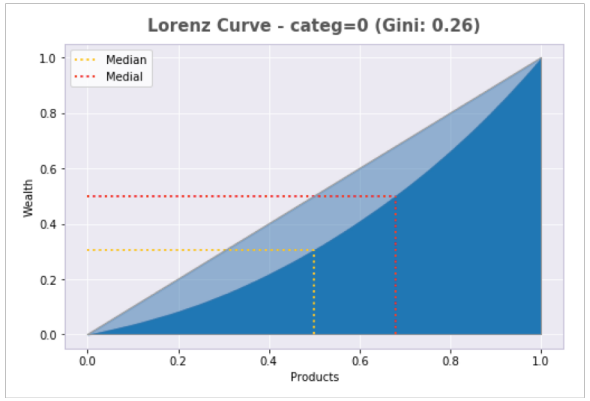


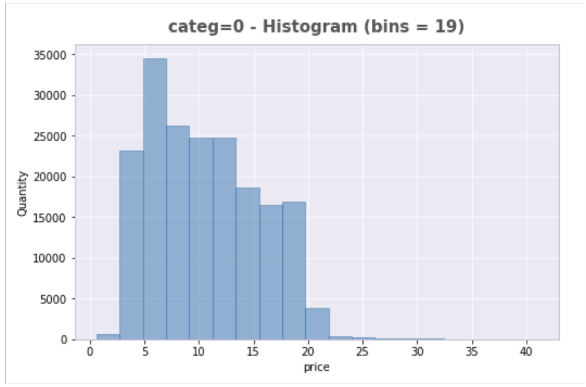


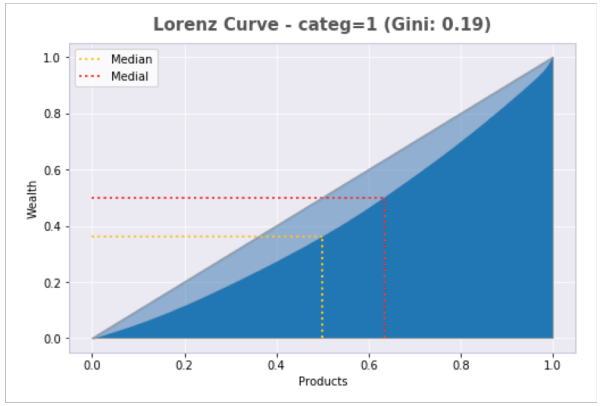
Price Variable

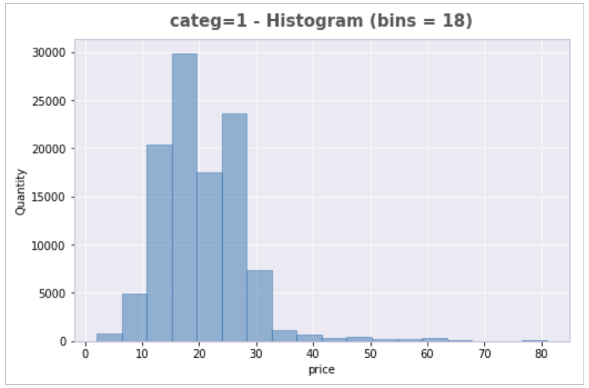
filtrée par catégorie

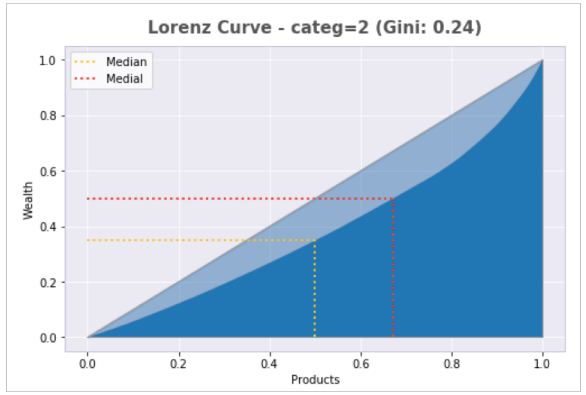
	price	categ=0	categ=1	categ=2
Sample Size	315,136	190,675	108,069	16,392
Total	5,477,418€	2,030,505€	2,213,622€	1,233,291€
Min	0.62€	0.62€	2.0€	30.99€
Max	300.0€	40.99€	80.99€	300.0€
Mode	16€	5€	16€	69€
Mean	17€	11€	20€	75€
Median	14€	10€	19€	63€
Variance (σ^2)	319.15€	24.33€	57.41€	1536.34€
Standard Deviation (σ)	17.86€	4.93€	7.58€	39.20€
Coefficient of variation (CV)	1.03	0.28	0.44	2.26
Skewness	5.47	0.43	1.73	2.31
Kurtosis	45.43	-0.38	8.18	5.52

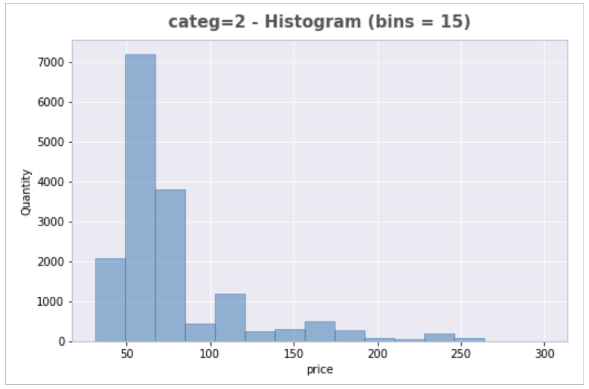








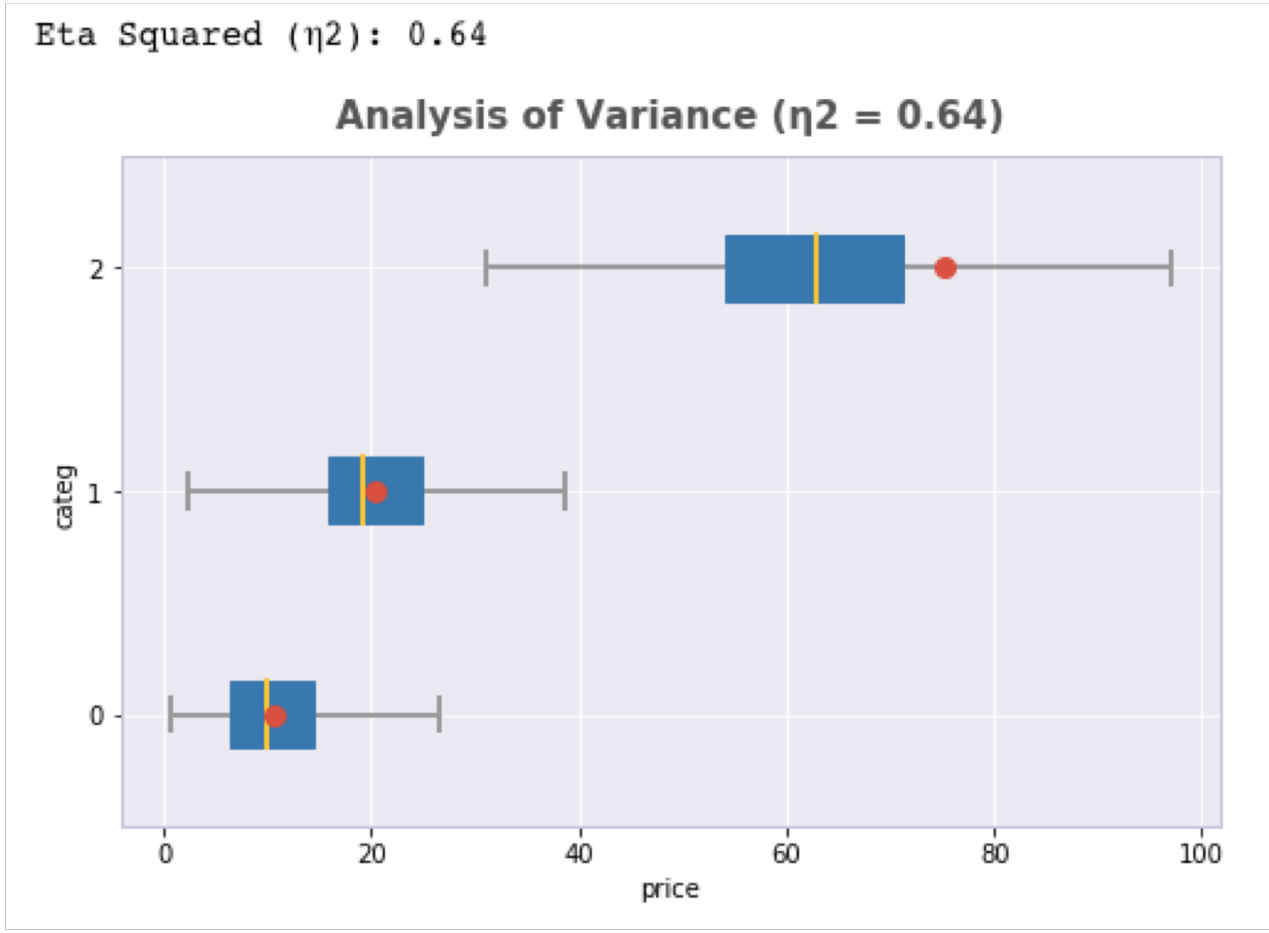




Price x Category

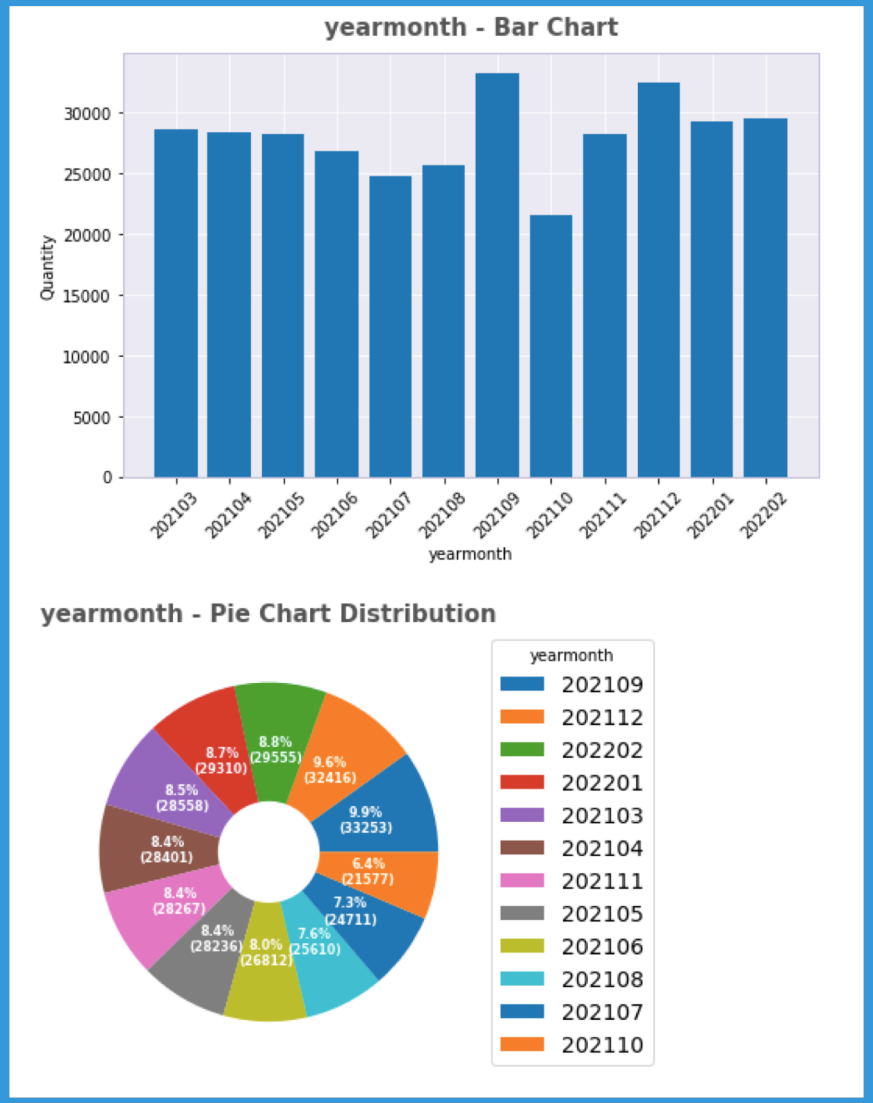
ANOVA

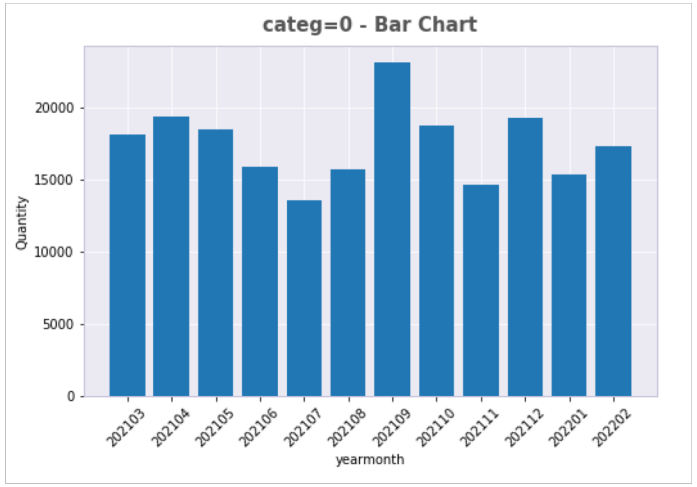
L'Eta carré à 0.64 ainsi que la visualisation des boîtes à moustache permettent de confirmer la dépendance entre la variable prix et la variable catégorie

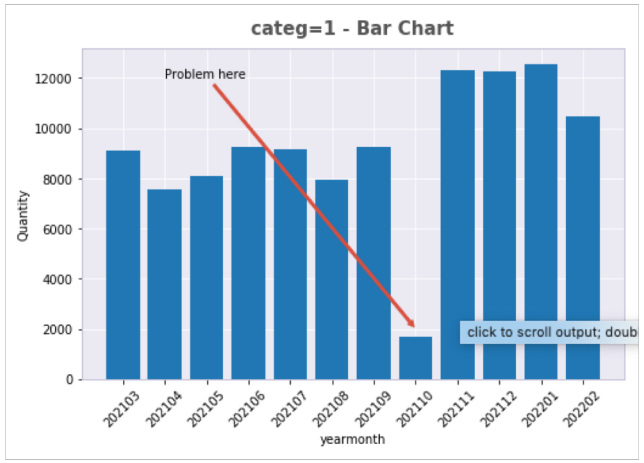


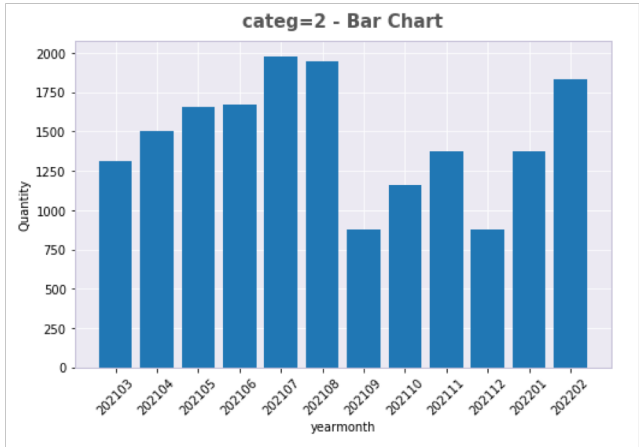
YearMonth Variable

Le volume de transaction est stable.





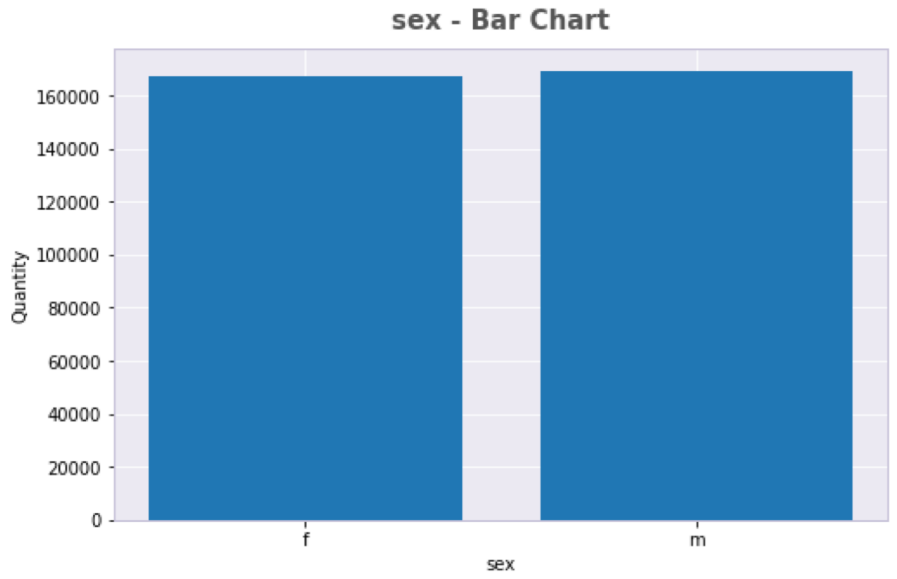




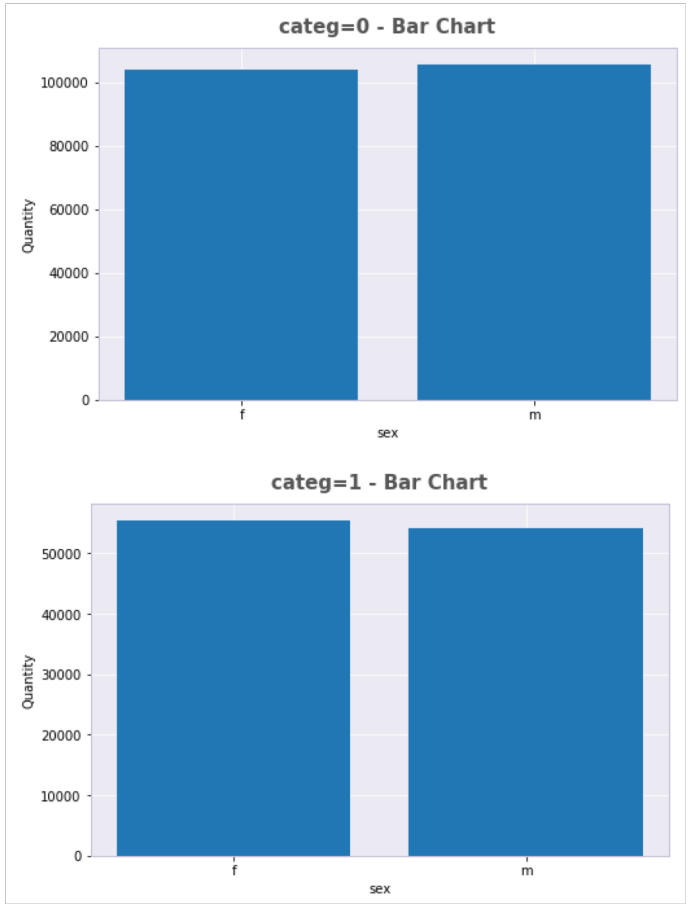
Sex Variable

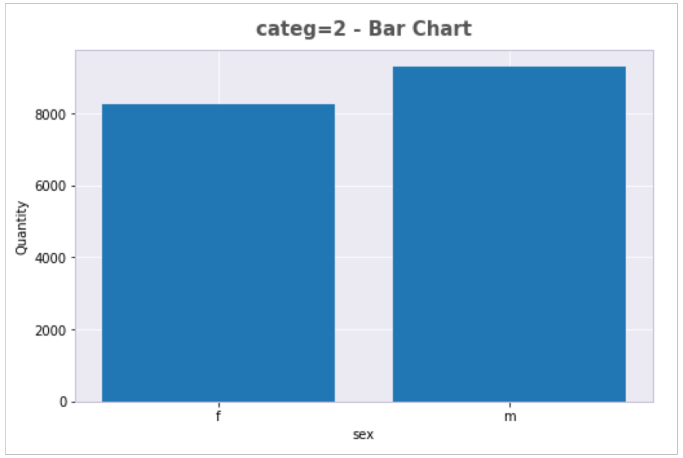
Sex Variable

In [20]:
```
sex_univa = Univa(main_df['sex'], quali=True)
sex_univa.distribution()
```



Légère corrélation avec la catégorie.



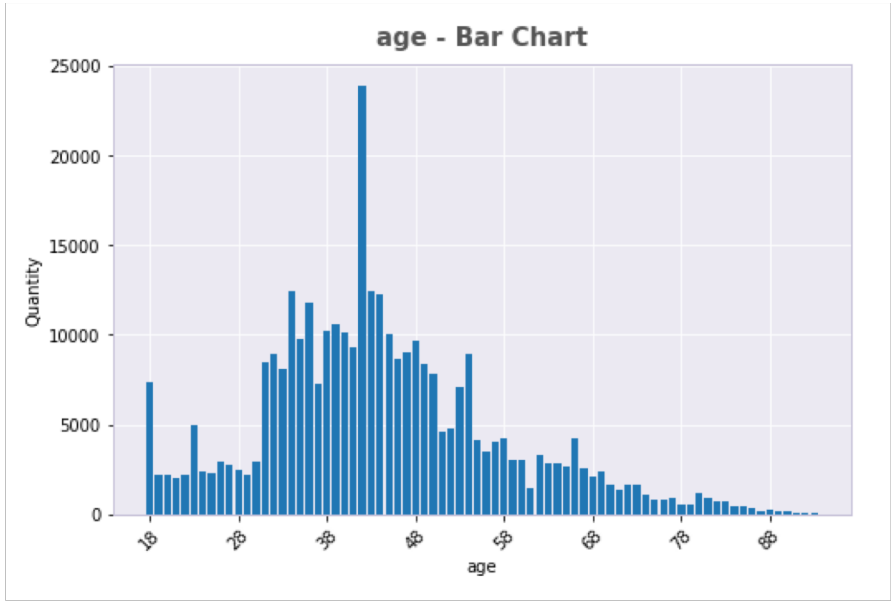


Age Variable

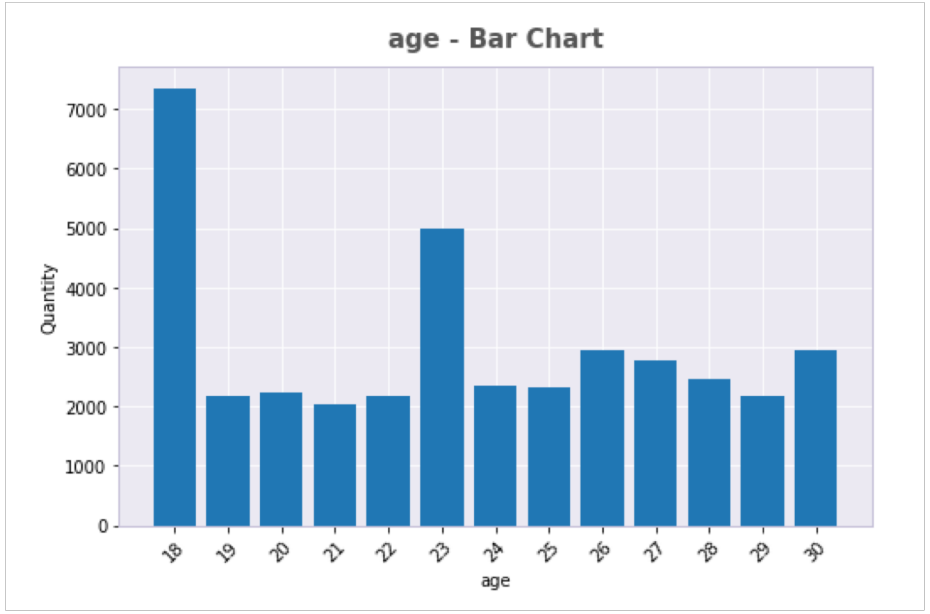
	age
Sample Size	336,713
Total	14,874,779
Min	18
Max	93
Mode	42
Mean	44
Median	42
Variance (σ^2)	182.91
Standard Deviation (σ)	13.52
Coefficient of variation (CV)	0.31
Skewness	0.58
Kurtosis	0.45

Il y a un pic inexpliqué à 42 ans et un pic à 18 ans que l'on peut expliquer par les inscriptions de personnes mineures.

Distribution globale



Focus sur les 18 – 30 ans



3. Analyses complémentaires

Voir le notebook