Data Analytics

Second Hand Luxury Market : Men’s Watch Analysis

To Van CAO

October, 2024

Table of content

Introduction

In recent years, the second-hand luxury market has experienced unprecedented growth, tapping into a rising consumer interest in sustainable fashion and economically-savvy purchasing. This market segment caters to a new breed of consumers who seek the allure of high-end fashion without the traditionally high price tags. By offering luxury goods at a fraction of their original prices, the second-hand market provides accessibility to luxury while promoting circular fashion—a practice mindful of the environmental impacts of fast fashion.

As the world grapples with sustainability challenges, consumers are increasingly favoring the second-hand market. Digital platforms have revolutionized this market, providing seamless access to a wide array of authentic luxury items with the promise of authenticity. With versatile offerings and reduced prices, pre-owned luxury has become a booming sector in the broader fashion industry.

Market Valuation and Growth

The global second-hand luxury market is valued at approximately USD 100-120 billion and is expected to grow at a compound annual growth rate of around 20-30% over the next few years, according to industry reports. This growth is fueled by increasing consumer demand for sustainable fashion and the proliferation of online resale platforms.

Vestiaire Collective: Background and Business Value

Founded in 2009, Vestiaire Collective has emerged as a leading global online marketplace for pre-owned luxury fashion. With a dedicated focus on quality and authenticity, the platform has cultivated a strong reputation among fashion enthusiasts and eco-conscious shoppers alike.

Background: Vestiaire Collective was born out of a desire to create a trusted platform for individuals to buy and sell pre-loved luxury items. Its founders saw an opportunity to address growing consumer demand for affordable luxury and sustainable practices. Today, it operates as a highly curated marketplace that carefully vets sellers and authenticates products, ensuring a top-notch shopping experience.

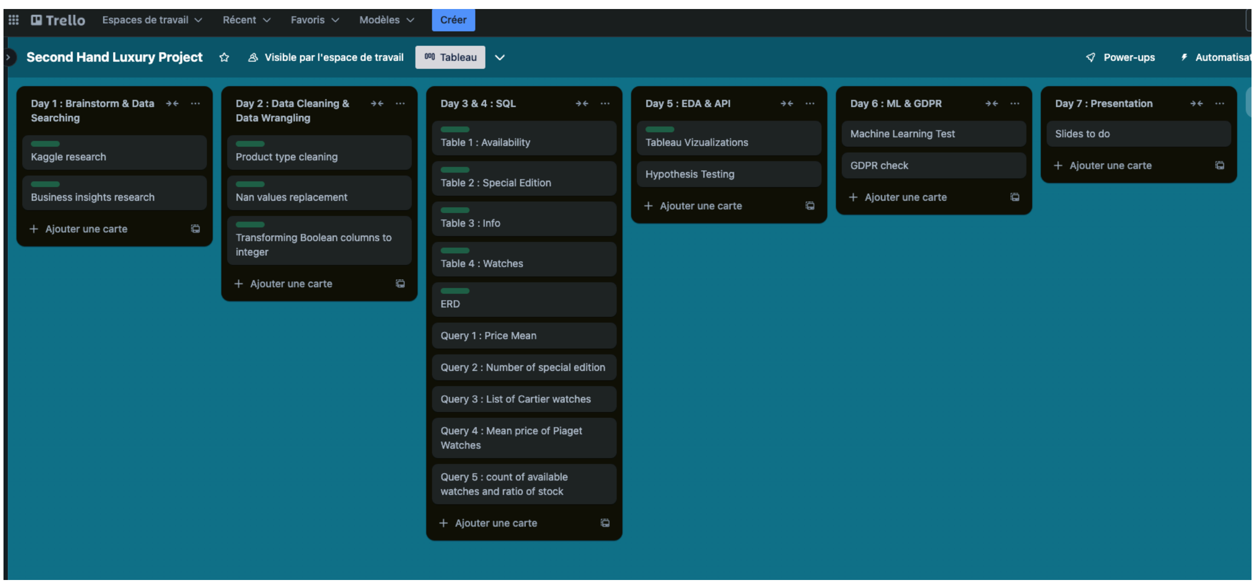
Business Value: Vestiaire Collective has established itself as a noteworthy player with an estimated valuation of over USD 1,7 billion, reflecting its significant role in the luxury resale market. The company adds value by offering a diverse catalog of luxury items ranging from clothing and accessories to jewelry and watches. Its unique business model combines fashion-tech innovation with environmentally sustainable solutions. By facilitating a circular economy, Vestiaire Collective allows buyers to find rare, high-quality items at competitive prices and sellers to monetize their wardrobe, ensuring both sides benefit from the transaction.

Moreover, Vestiaire Collective leverages a global network to reach consumers in major fashion capitals, catering to a diverse clientele with a shared passion for luxury and sustainability. Its commitment to providing a seamless buying and selling experience with emphasis on due diligence has set a new benchmark in the second-hand luxury industry.

Throughout our findings, we will focus on the men's market that seems promising and on the second hand luxury watches market.

Project Planning

Kanban was done on Trello.



Data and data sources

Data for this project were sources in 2 different ways : Kaggle flat csv file and web scraping.

Flat file

Kaggle : The dataset found on Kaggle provided an extensive picture of the Vestiaire collective offer with 900K samples :
https://www.kaggle.com/datasets/justinpakzad/vestiaire-fashion-dataset

Web scraping

A focus on luxury second-hand watches has been made by web scraping with the beautifulsoup library from the following website :
https://www.bucherer.com/fr/en/buy-certifiedpreowned?srule=Ranking+by+Category+Position&start=0&sz=72
This data has been used mainly for the SQL part of the project.

We get the following dataframe :

```
df
```

	Brand	Model	Price	Availability	Special Edition
0	Chopard	L.U.C. Certified Pre-Owned	13 800 €	In Stock	Not Special Edition
1	Chopard	Happy Sport Certified Pre-Owned	20 700 €	In Stock	Not Special Edition
2	Zenith	Chronomaster Certified Pre-Owned	15 100 €	In Stock	Not Special Edition
3	Blancpain	Fifty Fathoms Certified Pre-Owned	11 400 €	In Stock	Not Special Edition
4	Girard-Perregaux	1966 Certified Pre-Owned	8 800 €	In Stock	Not Special Edition
...	...	...	...	...	...
65	Breguet	Héritage Certified Pre Owned	29 700 €	In Stock	Not Special Edition
66	Breguet	Type XXII Certified Pre-Owned	10 500 €	In Stock	Special Edition
67	IWC Schaffhausen	Ingenieur Ingenieur Certified Pre-Owned	35 000 €	In Stock	Not Special Edition
68	IWC Schaffhausen	Pilot Certified Pre-Owned	11 200 €	In Stock	Not Special Edition
69	Cartier	Ballon Bleu de Cartier Certified Pre-Owned	18 000 €	In Stock	Not Special Edition

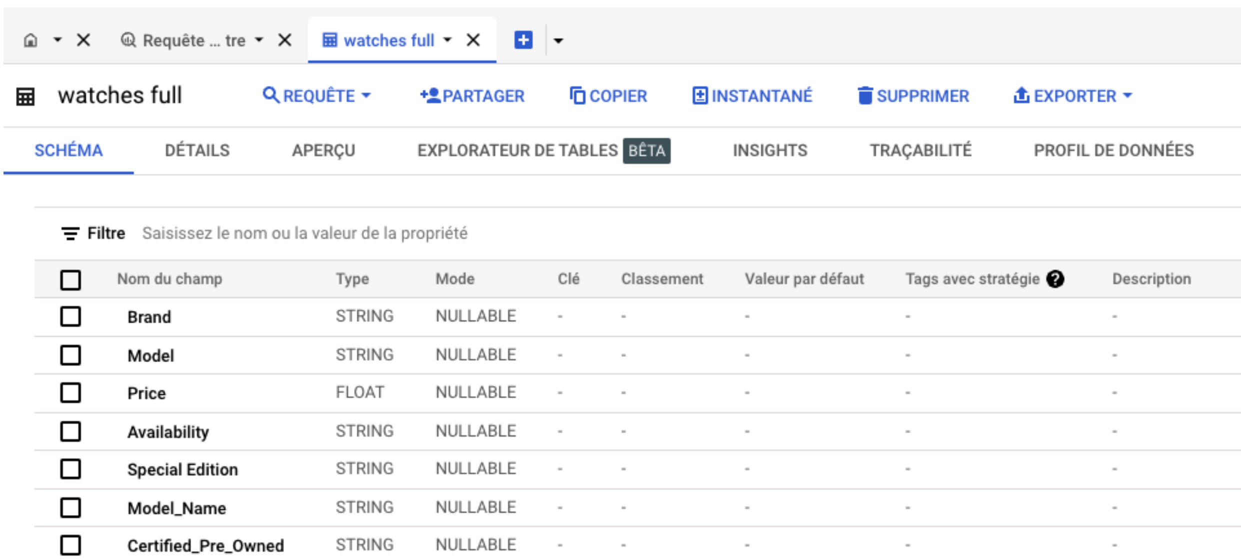
70 rows × 5 columns

Web scraping has also been used to scrap 2 images from report such as the following on Second Hand Market and Watches luxury market done by the Boston Consulting Group for the presentation :

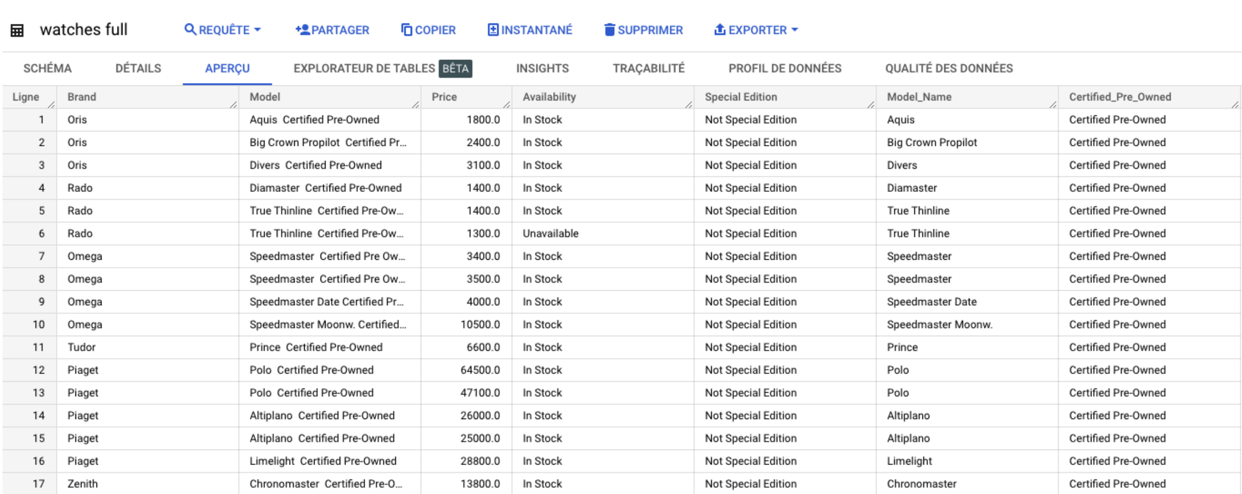
1. https://www.bcg.com/publications/2022/the-impact-of-secondhand-market-on-fashion-retailers
2. https://www.bcg.com/publications/2023/luxury-watch-market-trends#:~:text=Preowned%20watch%20sales%20reached%20%2422,trend%20is%20likely%20to%20continue.
3. https://usa.watchpro.com/how-gen-z-is-reshaping-the-watch-market/

Big Query

The database web scraped had been imported into Big Query : ferrous-coda-438507-g5 https://console.cloud.google.com/bigquery?ws=!1m4!1m3!3m2!1sferrous-coda-438507-g5!2swatches



Nom du champ	Type	Mode	Clé	Classement	Valeur par défaut	Tags avec stratégie	Description
Brand	STRING	NULLABLE	-	-	-	-	-
Model	STRING	NULLABLE	-	-	-	-	-
Price	FLOAT	NULLABLE	-	-	-	-	-
Availability	STRING	NULLABLE	-	-	-	-	-
Special Edition	STRING	NULLABLE	-	-	-	-	-
Model_Name	STRING	NULLABLE	-	-	-	-	-
Certified_Pre_Owned	STRING	NULLABLE	-	-	-	-	-

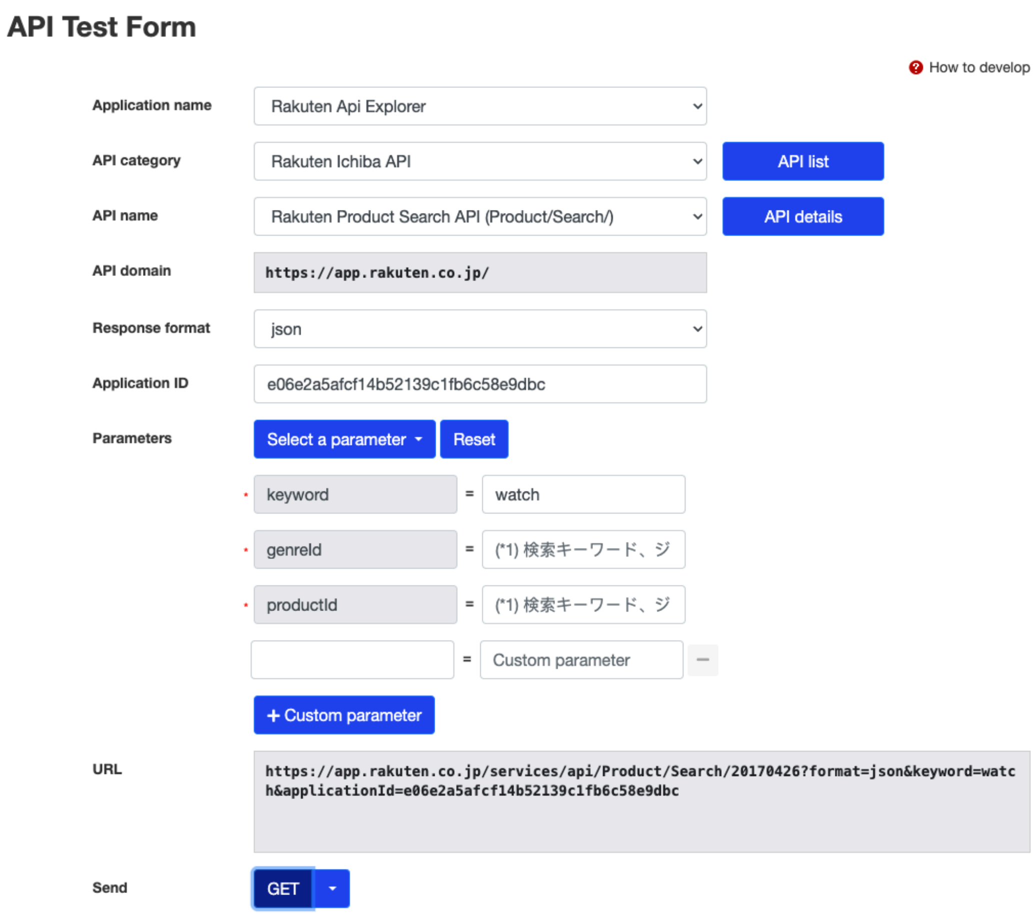


Ligne	Brand	Model	Price	Availability	Special Edition	Model_Name	Certified_Pre_Owned
1	Oris	Aquis Certified Pre-Owned	1800.0	In Stock	Not Special Edition	Aquis	Certified Pre-Owned
2	Oris	Big Crown Propilot Certified Pr...	2400.0	In Stock	Not Special Edition	Big Crown Propilot	Certified Pre-Owned
3	Oris	Divers Certified Pre-Owned	3100.0	In Stock	Not Special Edition	Divers	Certified Pre-Owned
4	Rado	Diamaster Certified Pre-Owned	1400.0	In Stock	Not Special Edition	Diamaster	Certified Pre-Owned
5	Rado	True Thinline Certified Pre-Ow...	1400.0	In Stock	Not Special Edition	True Thinline	Certified Pre-Owned
6	Rado	True Thinline Certified Pre-Ow...	1300.0	Unavailable	Not Special Edition	True Thinline	Certified Pre-Owned
7	Omega	Speedmaster Certified Pre Ow...	3400.0	In Stock	Not Special Edition	Speedmaster	Certified Pre-Owned
8	Omega	Speedmaster Certified Pre Ow...	3500.0	In Stock	Not Special Edition	Speedmaster	Certified Pre-Owned
9	Omega	Speedmaster Date Certified Pr...	4000.0	In Stock	Not Special Edition	Speedmaster Date	Certified Pre-Owned
10	Omega	Speedmaster Moonw. Certified...	10500.0	In Stock	Not Special Edition	Speedmaster Moonw.	Certified Pre-Owned
11	Tudor	Prince Certified Pre-Owned	6600.0	In Stock	Not Special Edition	Prince	Certified Pre-Owned
12	Piaget	Polo Certified Pre-Owned	64500.0	In Stock	Not Special Edition	Polo	Certified Pre-Owned
13	Piaget	Polo Certified Pre-Owned	47100.0	In Stock	Not Special Edition	Polo	Certified Pre-Owned
14	Piaget	Altiplano Certified Pre-Owned	26000.0	In Stock	Not Special Edition	Altiplano	Certified Pre-Owned
15	Piaget	Altiplano Certified Pre-Owned	25000.0	In Stock	Not Special Edition	Altiplano	Certified Pre-Owned
16	Piaget	Limelight Certified Pre-Owned	28800.0	In Stock	Not Special Edition	Limelight	Certified Pre-Owned
17	Zenith	Chronomaster Certified Pre-O...	13800.0	In Stock	Not Special Edition	Chronomaster	Certified Pre-Owned
18	Zenith	Star Certified Pre-Owned	25300.0	In Stock	Not Special Edition	Star	Certified Pre-Owned

API

There wasn’t relevant and free of access API for my topic but I found out that Vestiaire collective is using the Google cloud translation API to translate their product description almost in real time https://cloud.google.com/customers/vestiaire-collective

And I retrieved from the Rakuten API some information about watches but I couldn’t use the data as it was in Japanese :



The request got some results though :

Send GET


```
{
  "GenreInformation": {
    "children": [],
    "current": [],
    "parent": []
  },
  "Products": [
    {
      "Product": {
        "ProductDetails": [],
        "affiliateUrl": null,
        "averagePrice": 22079,
        "brandName": "",
        "genreId": "554973",
        "genreName": "GPSナビ",
        "itemCount": 28,
        "makerCode": "10458027582",
        "makerName": "テクタイト",
        "makerNameFormal": "テクタイト株式会社",
        "makerNameKana": "テクタイト",
```

Data cleaning

Vestiaire Collective data set cleaning

```
[99]: df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 900514 entries, 0 to 900513
Data columns (total 36 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   product_id                  900514 non-null  int64  
 1   product_type                900514 non-null  object 
 2   product_name                900514 non-null  object 
 3   product_description         900507 non-null  object 
 4   product_keywords            899331 non-null  object 
 5   product_gender_target       900514 non-null  object 
 6   product_category            899331 non-null  object 
 7   product_season              900512 non-null  object 
 8   product_condition           900514 non-null  object 
 9   product_like_count          900514 non-null  float64
 10  sold                        900514 non-null  bool   
 11  reserved                    900514 non-null  bool   
 12  available                   900514 non-null  bool   
 13  in_stock                    900514 non-null  bool   
 14  should_be_gone              900514 non-null  bool   
 15  brand_id                    900514 non-null  int64  
 16  brand_name                  900514 non-null  object 
 17  brand_url                   900514 non-null  object 
 18  product_material            900510 non-null  object 
 19  product_color               900513 non-null  object 
 20  price_usd                   900514 non-null  float64
 21  seller_price                900514 non-null  float64
 22  seller_earning              900514 non-null  float64
 23  seller_badge                900514 non-null  object 
 24  has_cross_border_fees       886778 non-null  object 
 25  buyers_fees                 886778 non-null  float64
 26  warehouse_name              900514 non-null  object 
 27  seller_id                   900514 non-null  int64  
 28  seller_username             900475 non-null  object 
 29  usually_ships_within        745723 non-null  object 
 30  seller_country              900514 non-null  object 
 31  seller_products_sold        900514 non-null  float64
 32  seller_num_products_listed  900514 non-null  float64
 33  seller_community_rank       900514 non-null  float64
 34  seller_num_followers        900514 non-null  float64
 35  seller_pass_rate            900514 non-null  float64
dtypes: bool(5), float64(10), int64(3), object(18)
memory usage: 217.3+ MB
```

```
[100]: (df1.isna().mean() * 100).sum()
```

```
[100]: 20.50850958452617
```

20.5% of data is missing from the dataset.

There are 20,5% of data missing in this data set

```
[102]: df1.isna().mean() * 100
```

```
[102]: product_id                    0.000000
       product_type                  0.000000
       product_name                  0.000000
       product_description           0.000777
       product_keywords              0.131369
       product_gender_target         0.000000
       product_category              0.131369
       product_season                0.000222
       product_condition             0.000000
       product_like_count            0.000000
       sold                          0.000000
       reserved                      0.000000
       available                     0.000000
       in_stock                      0.000000
       should_be_gone                0.000000
       brand_id                      0.000000
       brand_name                    0.000000
       brand_url                     0.000000
       product_material              0.000444
       product_color                 0.000111
       price_usd                     0.000000
       seller_price                  0.000000
       seller_earning                0.000000
       seller_badge                  0.000000
       has_cross_border_fees         1.525351
       buyers_fees                   1.525351
       warehouse_name                0.000000
       seller_id                     0.000000
       seller_username               0.004331
       usually_ships_within         17.189183
       seller_country                0.000000
       seller_products_sold          0.000000
       seller_num_products_listed    0.000000
       seller_community_rank         0.000000
       seller_num_followers          0.000000
       seller_pass_rate              0.000000
       dtype: float64
```

The column usually_ships_within has the highest number of missing values (17.2%). Given that this column is of type object, we will replace these missing values with the most common one using the mode. As for the remaining 3.3% missing values, we will identify columns with categorical data and replace them using the mode or median for numerical columns.
Any remaining missing values will be removed from the dataset.
Doing so we will make sure to keep the other columns informations that can still be interesting for our analysis as I wanted to orient the analysis on the offer on the website.

```
# Filling missing values with the mode for categorical columns
df1['usually_ships_within'] = df1['usually_ships_within'].fillna(df1['usually_ships_within'].mode()[0])
df1['product_category'] = df1['product_category'].fillna(df1['product_category'].mode()[0])
df1['product_keywords'] = df1['product_keywords'].fillna(df1['product_keywords'].mode()[0])

# Filling missing values with the median for numerical columns
df1['buyers_fees'] = df1['buyers_fees'].fillna(df1['buyers_fees'].median())
```

```
df1[['has_cross_border_fees']].value_counts()
```

```
has_cross_border_fees
True                     886778
Name: count, dtype: int64
```

In the `has_cross_border_fees` column, all existing values are True. Therefore, for any missing values in this column, we will replace them with 'False'.

```
df1['has_cross_border_fees'].fillna(False, inplace=True)
```

```
print(f'Percentage of missing values: {(df1.isna().mean() * 100).sum().round(4)}')
```

```
Percentage of missing values: 0.0059
```

```
df1.dropna(inplace=True)
```

```
print(f'Percentage of missing values: {(df1.isna().mean() * 100).sum()}')
```

```
Percentage of missing values: 0.0
```

```
df1.shape
```

```
(900461, 36)
```

```
print(f'Amount of duplicates: {df1.duplicated().sum()}')
```

```
Amount of duplicates: 0
```

Product categories as they are now are too precise to do grouping as shown below in the product_type columns with about 11K values :

```
df1[['product_type']].nunique()
```

```
product_type    10983
dtype: int64
```

```python
df1.head()
```

	product_id	product_type	product_name	product_description	product_keywords	product_gender_target
0	43247626	Wool mini skirt	Wool mini skirt Miu Miu Grey size S Internatio...	Miu Miu – Pleated mini skirt Size: 36 (S) Wai...	Miu Miu Wool Skirts	Women
1	43247441	Jacket	Jacket Barbara Bui Navy size 42 FR in Cotton	For selling nice women's suit Barbara Bui size...	Barbara Bui Cotton Jackets	Women
2	43246517	Wool coat	Wool coat Comme Des Garcons White size S Inter...	Magnificent boiled wool coat. I bought it in t...	Comme Des Garcons Wool Coats	Women
3	43246507	Mini skirt	Mini skirt MSGM Black size 38 IT in Polyester	MSGM Skirt Black Printed Raw-Edge & Embroidere...	MSGM Polyester Skirts	Women
4	43246417	Vegan leather trousers	Vegan leather trousers LVIR Black size 36 FR i...	LVIR black grained faux leather trousers size ...	LVIR Vegan leather Trousers	Women

So we will add a new column that keeps only the last word of the product_type column

```python
#creating new columns to have a more precise product type

# Split the column by whitespace and keep only the last word
df1['last_word_product_type'] = df1['product_type'].apply(lambda x: x.split()[-1])
```

```python
df1.head()
```

seller_id	seller_username	usually_ships_within	seller_country	seller_products_sold	seller_num_products_listed	seller_community_rank	seller_num_followers	seller_pass_rate	last_word_product_type
25775970	vitalii25775970	1-2 days	Germany	3.0	14.0	0.0	13.0	0.0	skirt
13698770	olivia13698770	1-2 days	Belgium	0.0	0.0	0.0	8.0	0.0	Jacket
6042365	cecilia6042365	1-2 days	Spain	58.0	69.0	0.0	62.0	96.0	coat
13172949	gretchen13172949	1-2 days	United States	63.0	274.0	126346.0	131.0	96.0	skirt

Then convert this column into lower case for consistency :

```
#convert to lowercase
df1['last_word_product_type'] = df1['last_word_product_type'].apply(lambda x: x.lower())

df1.head()
```

seller_id	seller_username	usually_ships_within	seller_country	seller_products_sold	seller_num_products_listed	seller_community_rank	seller_num_followers	seller_pass_rate	last_word_product_type
25775970	vitalii25775970	1-2 days	Germany	3.0	14.0	0.0	13.0	0.0	skirt
13698770	olivia13698770	1-2 days	Belgium	0.0	0.0	0.0	8.0	0.0	jacket
6042365	cecilia6042365	1-2 days	Spain	58.0	69.0	0.0	62.0	96.0	coat
13172949	gretchen13172949	1-2 days	United States	63.0	274.0	126346.0	131.0	96.0	skirt
2578605	crunchykat	3-5 days	United Kingdom	19.0	14.0	102821.0	40.0	89.0	trousers

Now products categories are narrowed down from 11K to 100.

```
df1['last_word_product_type'].unique()
```

```
array(['skirt', 'jacket', 'coat', 'trousers', 'dress', 'top', 'swimsuit',
       'blazer', 'suit', 'shirt', 'jeans', 'blouse', 'shorts', 'set',
       'vest', 'pants', 'tunic', 't-shirt', 'jumper', 'caban', 'bermuda',
       'corset', 'knitwear', 'jumpsuit', 'tight', 'polo', 'short',
       'sweatshirt', 'lingerie', 'bra', 'parka', 'cardigan', 'peacoat',
       'slip', 'leggings', 'camisole', 'twin-set', 'harem', 'puffer',
       'cape', 'dufflecoat', 'combishort', 'swimwear', 'pareo', 'poncho',
       'string', 'bustier', 'overall', 'accessory', 'pull', 'trench',
       'trenchcoat', 'jean', 'laine', 'sunglasses', 'tie', 'watch', 'hat',
       'square', 'jewellery', 'belt', 'cufflinks', 'gloves', 'lifestyle',
       'neckerchief', 'scarf', 'beanie', 'wallet', 'handkerchief', 'cap',
       'purse', 'stole', 'clutch', 'glasses', 'ring', 'mittens', 'panama',
       'beret', 'choker', 'cheche', 'diary', 'case', 'accessories',
       'embellishments', 'trainers', 'flats', 'ups', 'sandals', 'boots',
       'espadrilles', 'page/nom_singulier/31', 'leather', 'heels',
       'clogs', 'sandal', 'mules', 'heel', 'flops', 'flat', 'paillettes'],
      dtype=object)
```

```
df1['last_word_product_type'].nunique()
```

```
100
```

Web-scrapped Bucherer.com data set cleaning

For the web scraped data set :

A first part of cleaning has been done earlier in the web scraping step for the availability and special edition column, replacing missing values.

Then, I created 2 extra columns to separate the model and the fact that they are "Certified Pre-Owned" watches

```
df
```

	Brand	Model	Price	Availability	Special Edition
0	Chopard	L.U.C. Certified Pre-Owned	13 800 €	In Stock	Not Special Edition
1	Chopard	Happy Sport Certified Pre-Owned	20 700 €	In Stock	Not Special Edition
2	Zenith	Chronomaster Certified Pre-Owned	15 100 €	In Stock	Not Special Edition
3	Blancpain	Fifty Fathoms Certified Pre-Owned	11 400 €	In Stock	Not Special Edition
4	Girard-Perregaux	1966 Certified Pre-Owned	8 800 €	In Stock	Not Special Edition
...	...	...	...	...	...
65	Breguet	Héritage Certified Pre Owned	29 700 €	In Stock	Not Special Edition
66	Breguet	Type XXII Certified Pre-Owned	10 500 €	In Stock	Special Edition
67	IWC Schaffhausen	Ingenieur Ingenieur Certified Pre-Owned	35 000 €	In Stock	Not Special Edition
68	IWC Schaffhausen	Pilot Certified Pre-Owned	11 200 €	In Stock	Not Special Edition
69	Cartier	Ballon Bleu de Cartier Certified Pre-Owned	18 000 €	In Stock	Not Special Edition

70 rows × 5 columns

```
# Check which rows contain "Certified Pre-Owned"
contains_certified = df['Model'].str.contains('Certified Pre-Owned')

# Create the new columns using a conditional split
df['Model_Name'] = df['Model'].where(~contains_certified, df['Model'].str.split('Certified Pre-Owned', expand=True)[0])
df['Certified_Pre_Owned'] = df['Model'].apply(lambda x: 'Certified Pre-Owned' if 'Certified Pre-Owned' in x else '')

# Clean up by removing unnecessary whitespace
df['Model_Name'] = df['Model_Name'].str.strip()
```

```
df
```

	Brand	Model	Price	Availability	Special Edition	Model_Name	Certified_Pre_Owned
0	Chopard	L.U.C. Certified Pre-Owned	13 800 €	In Stock	Not Special Edition	L.U.C.	Certified Pre-Owned
1	Chopard	Happy Sport Certified Pre-Owned	20 700 €	In Stock	Not Special Edition	Happy Sport	Certified Pre-Owned
2	Zenith	Chronomaster Certified Pre-Owned	15 100 €	In Stock	Not Special Edition	Chronomaster	Certified Pre-Owned
3	Blancpain	Fifty Fathoms Certified Pre-Owned	11 400 €	In Stock	Not Special Edition	Fifty Fathoms	Certified Pre-Owned
4	Girard-Perregaux	1966 Certified Pre-Owned	8 800 €	In Stock	Not Special Edition	1966	Certified Pre-Owned
...	...	...	...	...	...	...	...
65	Breguet	Héritage Certified Pre Owned	29 700 €	In Stock	Not Special Edition	Héritage Certified Pre Owned	
66	Breguet	Type XXII Certified Pre-Owned	10 500 €	In Stock	Special Edition	Type XXII	Certified Pre-Owned
67	IWC Schaffhausen	Ingenieur Ingenieur Certified Pre-Owned	35 000 €	In Stock	Not Special Edition	Ingenieur Ingenieur	Certified Pre-Owned
68	IWC Schaffhausen	Pilot Certified Pre-Owned	11 200 €	In Stock	Not Special Edition	Pilot	Certified Pre-Owned
69	Cartier	Ballon Bleu de Cartier Certified Pre-Owned	18 000 €	In Stock	Not Special Edition	Ballon Bleu de Cartier	Certified Pre-Owned

Then transform the Certified Pre-Owned, Special Edition and Availability as integer for upcoming SQL uses.

```python
# Create a copy of the original DataFrame
df_transformed = df.copy()

# Transform the 'Certified_Pre_Owned' column
df_transformed['Certified_Pre_Owned'] = np.where(df_transformed['Certified_Pre_Owned'] == 'Certified Pre-Owned', 1, 0)

# Transform the 'Special Edition' column
df_transformed['Special Edition'] = np.where(df_transformed['Special Edition'] == 'Special Edition', 1, 0)

# Transform the 'Availability' column
df_transformed['Availability'] = np.where(df_transformed['Availability'] == 'In Stock', 1, 0)
```

```python
df_transformed
```

	Brand	Model	Price	Availability	Special Edition	Model_Name	Certified_Pre_Owned
0	Chopard	L.U.C. Certified Pre-Owned	13 800 €	1	0	L.U.C.	1
1	Chopard	Happy Sport Certified Pre-Owned	20 700 €	1	0	Happy Sport	1
2	Zenith	Chronomaster Certified Pre-Owned	15 100 €	1	0	Chronomaster	1
3	Blancpain	Fifty Fathoms Certified Pre-Owned	11 400 €	1	0	Fifty Fathoms	1
4	Girard-Perregaux	1966 Certified Pre-Owned	8 800 €	1	0	1966	1
...	...	...	...	...	...	...	...
65	Breguet	Héritage Certified Pre Owned	29 700 €	1	0	Héritage Certified Pre Owned	0
66	Breguet	Type XXII Certified Pre-Owned	10 500 €	1	1	Type XXII	1
67	IWC Schaffhausen	Ingenieur Ingenieur Certified Pre-Owned	35 000 €	1	0	Ingenieur Ingenieur	1
68	IWC Schaffhausen	Pilot Certified Pre-Owned	11 200 €	1	0	Pilot	1
69	Cartier	Ballon Bleu de Cartier Certified Pre-Owned	18 000 €	1	0	Ballon Bleu de Cartier	1

70 rows × 7 columns

And finally changed the Price format to float for further uses :

```python
df_transformed.dtypes
```

```
Brand                     object
Model                     object
Price                     object
Availability_ID            int64
Special_Edition_ID         int64
Model_Name                object
Certified_Pre_Owned_ID     int64
dtype: object
```

```python
# Remove non-numeric characters and convert to float
df_transformed['Price'] = df_transformed['Price'].replace({'€': '', ' ': ''}, regex=True).astype(float)

# Verify the conversion
print(df_transformed['Price'].head())
```

```
0    13800.0
1    20700.0
2    15100.0
3    11400.0
4     8800.0
Name: Price, dtype: float64
```

Database type selection

I chose an SQL-based approach to create my database from Bucherer because it effectively handles structured data. SQL databases are relational, organizing data into tables with rows and columns. This structure allows for linking tables through foreign keys, which is ideal for my data's organized tables and predefined schema.

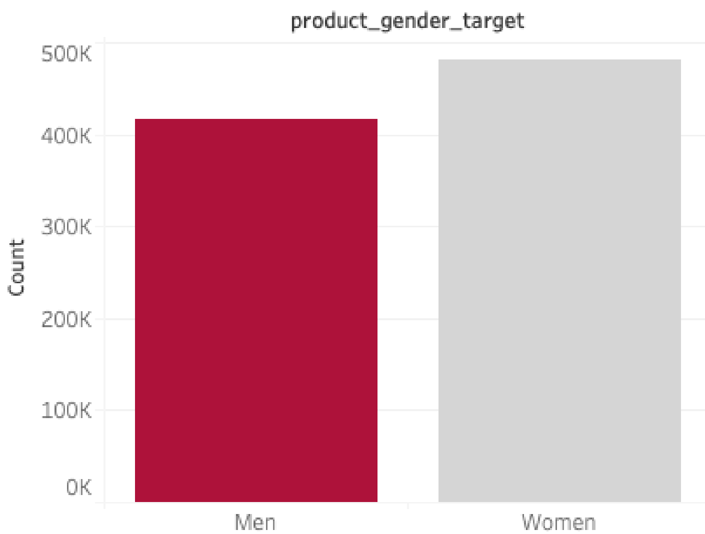
Reasons for Choosing SQL over NoSQL:

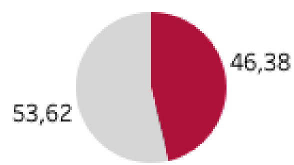
Structure: SQL uses structured query language with a predefined schema, while NoSQL supports dynamic schemas for unstructured data.
Scalability: SQL offers vertical scalability, whereas NoSQL is horizontally scalable.
Data Model: SQL is table-based; NoSQL can be document, key-value, graph, or wide-column stores.
Transactions: SQL excels at multi-row transactions, whereas NoSQL handles unstructured data like JSON better.
Interrelation: Relational databases reduce data redundancy and enhance integrity.
Relational databases, such as MySQL Workbench, are a fitting choice for managing structured, interrelated data efficiently and performing complex queries across multiple tables.

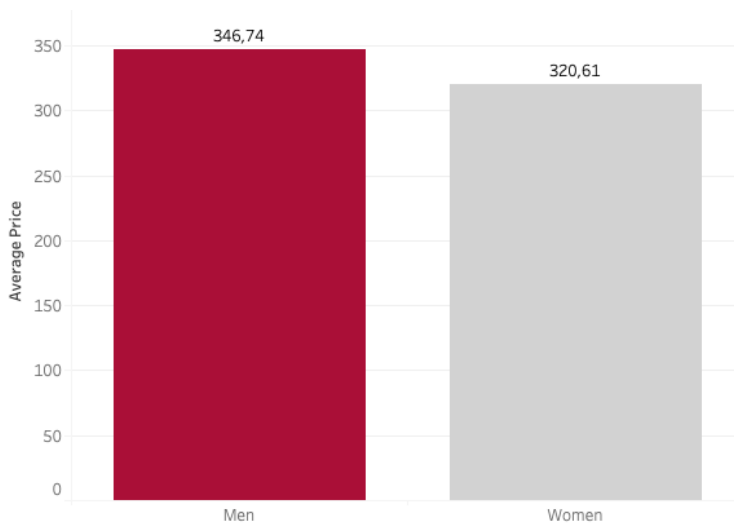
Exploratory data analysis

Please note all findings in Python have been exported in order to be visualized in Tableau.

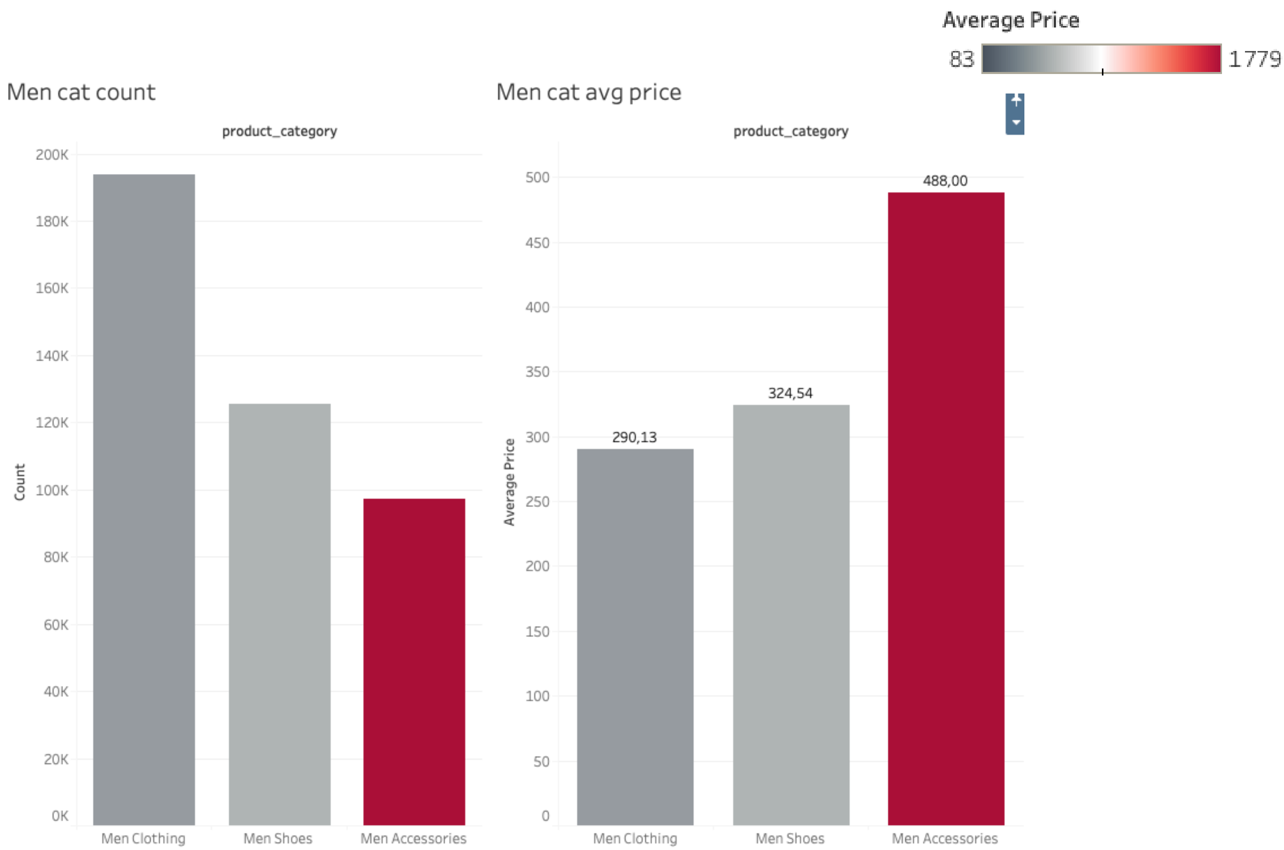
Analysis by category :

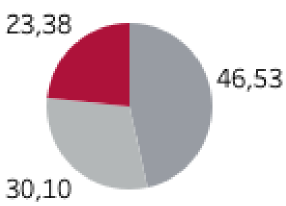




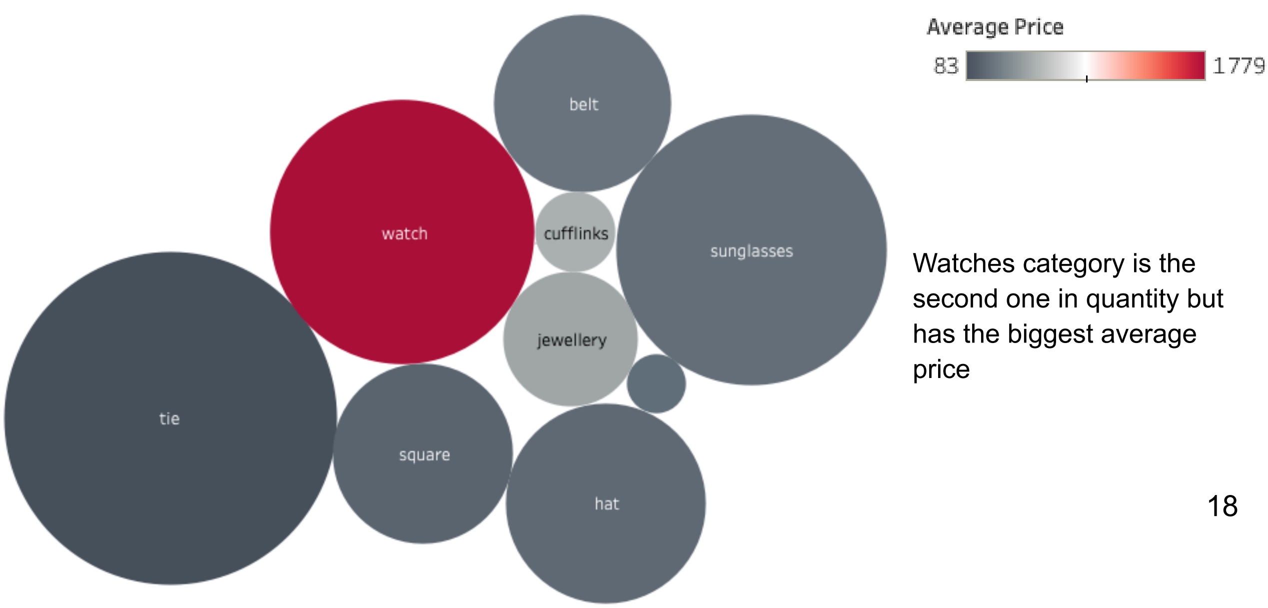


Men average price is + 8% vs Women average price





Men accessories have the smallest share but has bigger average price +68% vs Men's clothing even if it accounts for 23 % of the offer, so we think it would be interesting to deep dive into this category



Watches category is the second one in quantity but has the biggest average price

ANOVA hypothesis testing shows us that there is a correlation between prices and watches :

Hypothesis Testing

ANOVA

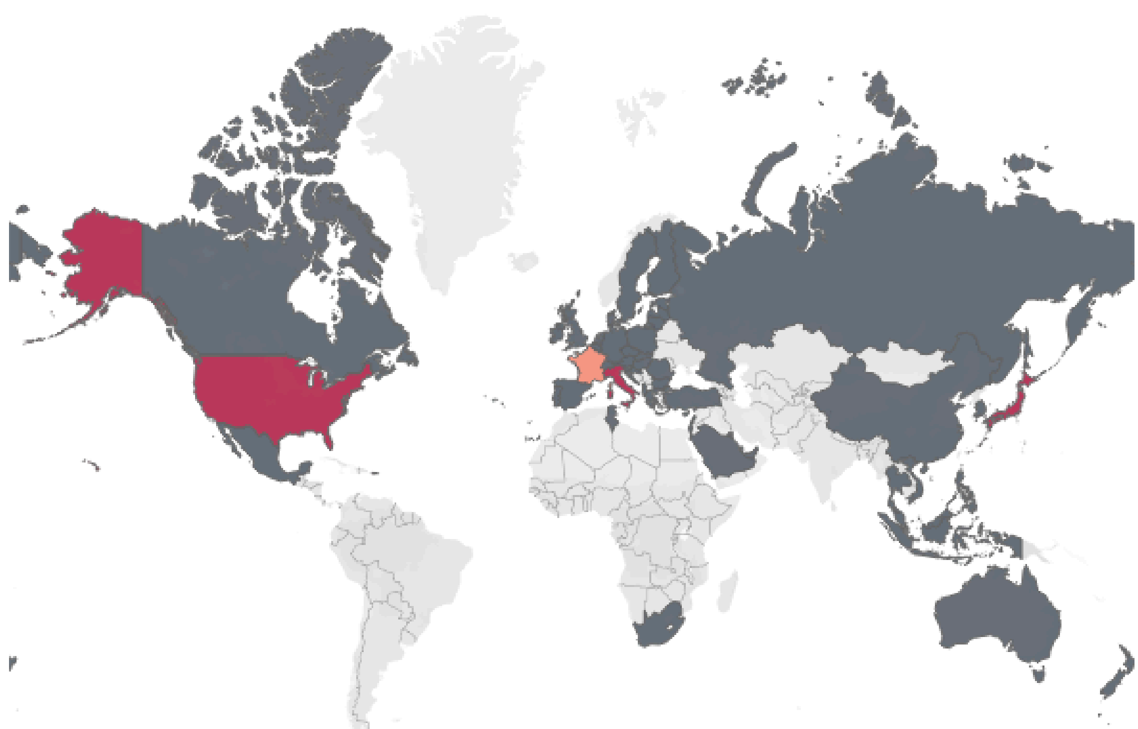
```
df_watch = df2[(df2["last_word_product_type"] == "watch")]["price_usd"]
df_non_watch = df2[(df2["last_word_product_type"] != "watch")]["price_usd"]
alpha = 0.10
st.f_oneway(df_watch, df_non_watch)
```

```
F_onewayResult(statistic=128562.54672620428, pvalue=0.0)
```

```
alpha = 0.10
st.f_oneway(df_watch, df_non_watch)
```

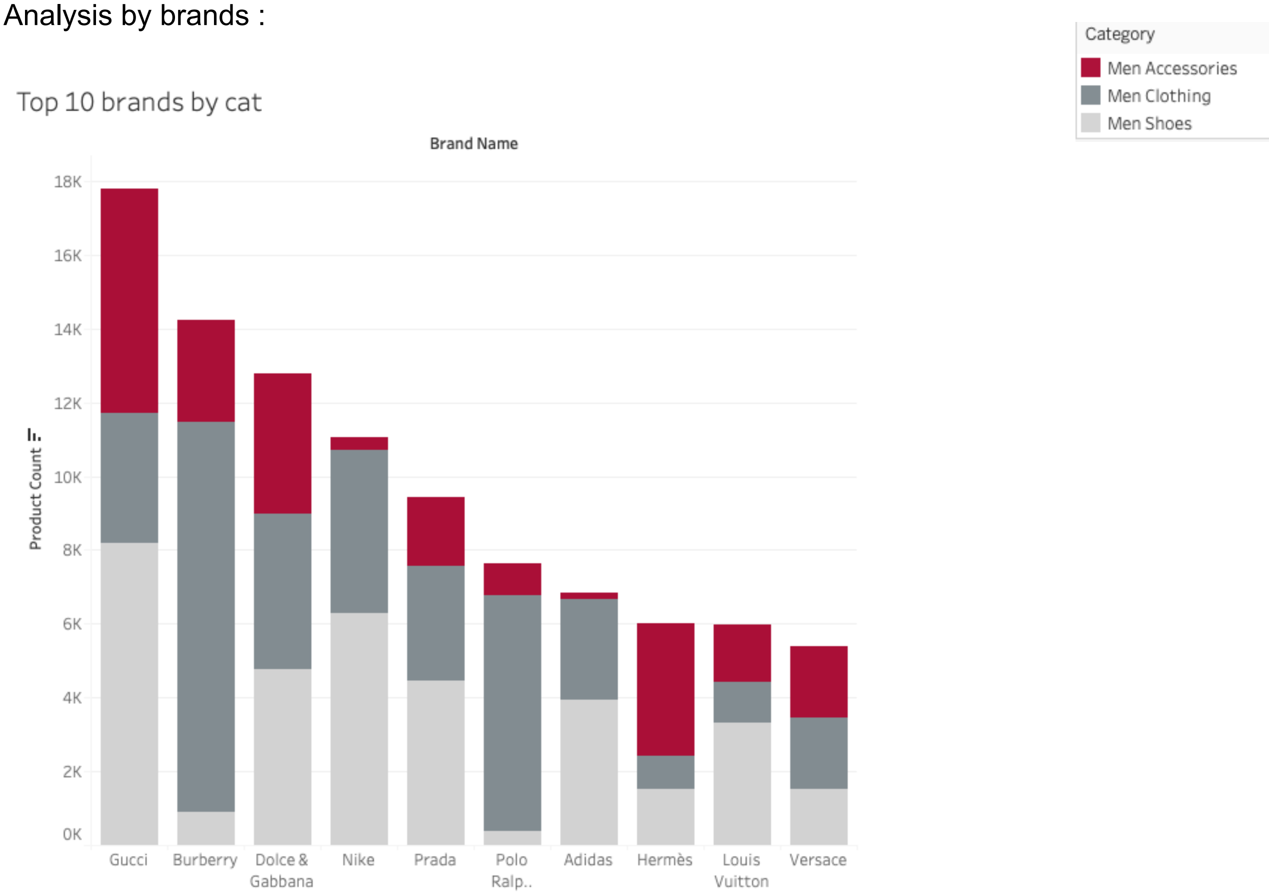
```
F_onewayResult(statistic=128562.54672620428, pvalue=0.0)
```

If we have a look of the distribution of the watches offer by country :

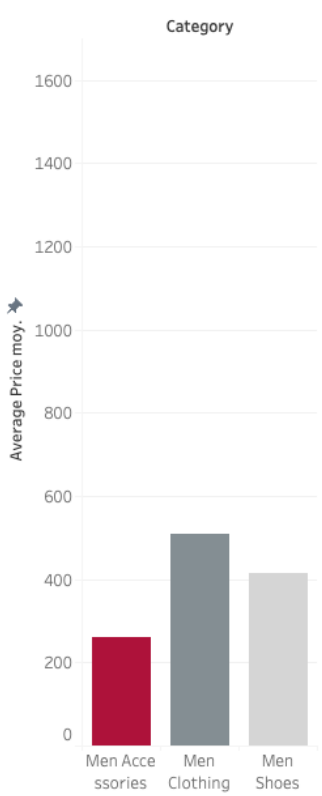


Japan is the top watch sellers on the platform

Analysis by brands :

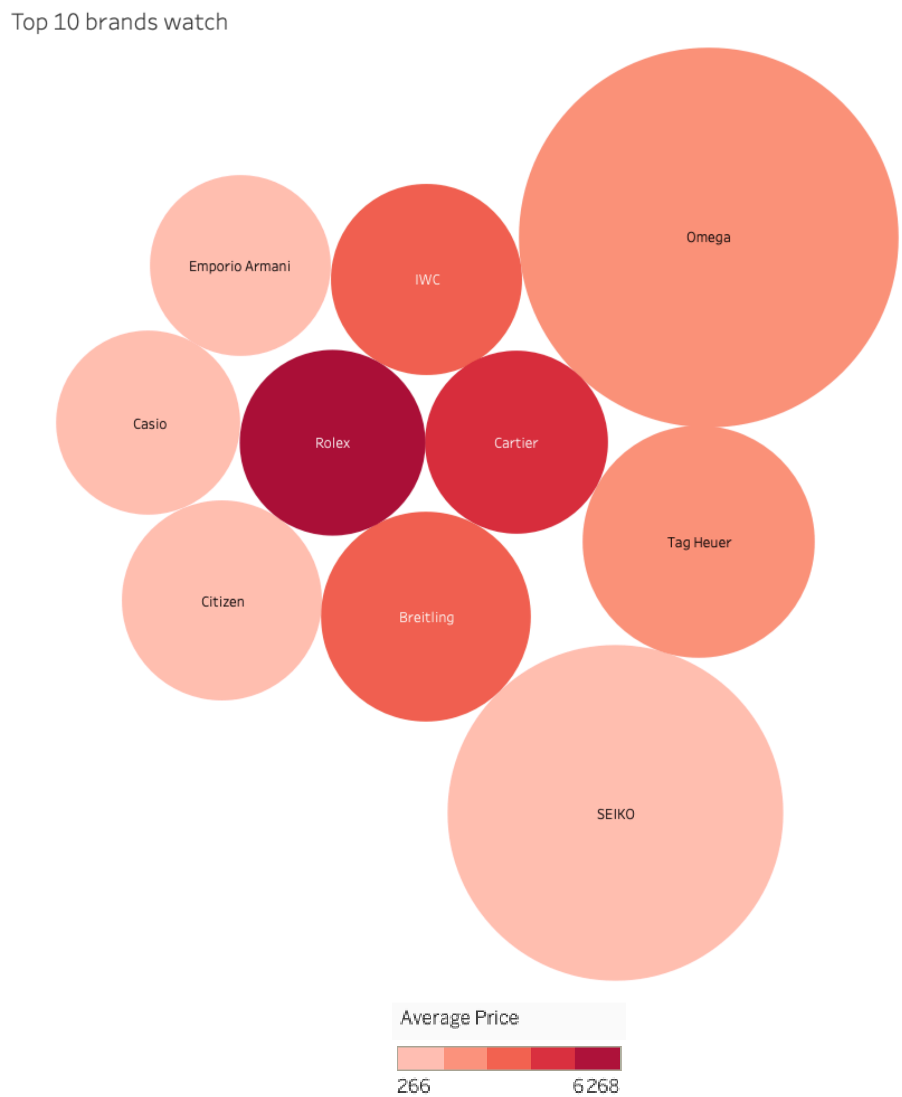


Leading brand in Vestiaire collective offer are not specialized luxury watches brands :



Watches average price is + 107% higher than the prices of the top 10 brands all category

If we have a look on the Men’s watches by brand and their related average prices :



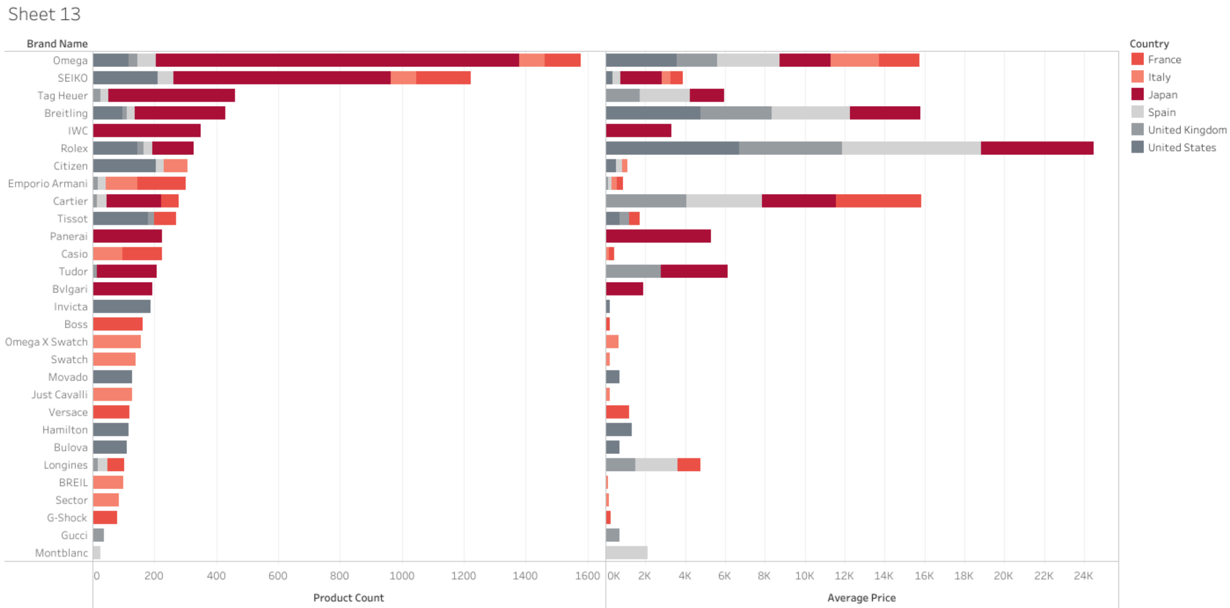
Omega and Seiko are the biggest brands in terms of quantity but Rolex has the highest average price.

With the same approach I did a focus on the top 10 watches brands :

	Brand Name	Product Count	Product Ratio (%)	Average Price
0	Omega	1781	10.09	2661.15
1	SEIKO	1401	7.94	1271.24
2	Tag Heuer	649	3.68	1876.72
3	Breitling	521	2.95	3845.65
4	Citizen	495	2.81	584.38
5	Rolex	484	2.74	6390.50
6	Cartier	436	2.47	4281.01
7	Tissot	422	2.39	618.94
8	Casio	421	2.39	336.04
9	IWC	417	2.36	3536.26

60% of the top 10 watch brands is made of Swiss brands

And deep dived in to countries to see which brands were the most offered by countries :



Hypothesis Testing :

```
from scipy.stats import chi2_contingency

# Step 1: Filter the DataFrame for "watch" products
watch_df = df2[df2['last_word_product_type'] == 'watch']
crosstab_result = pd.crosstab(watch_df['seller_country'], watch_df['brand_name'])

chi2_statistic, chi2_p_value, _, _ = chi2_contingency(crosstab_result)

chi2_statistic, chi2_p_value
```

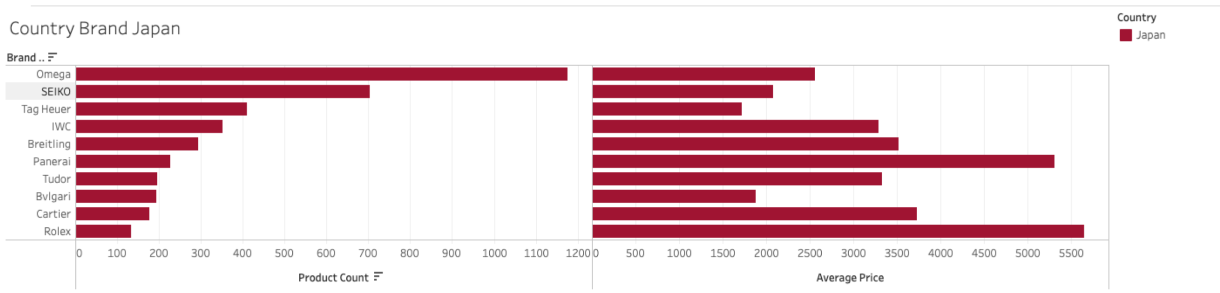
```
(105564.11349108192, 0.0)
```

```
##ANOVA

from scipy.stats import f_oneway

# Assuming df2 is your DataFrame

# Step 1: Filter the DataFrame for "watch" products
watch_df = df2[df2['last_word_product_type'] == 'watch']

# Step 2: Group the data by brand and extract price data
brands = watch_df['brand_name'].unique()
price_groups = [watch_df[watch_df['brand_name'] == brand]['price_usd'] for brand in brands]

# Step 3: Perform the ANOVA test
f_statistic, p_value = f_oneway(*price_groups)

# Print the results
print("ANOVA F-statistic:", f_statistic)
print("p-value:", p_value)
```

```
ANOVA F-statistic: 65.14796058884477
p-value: 0.0
```

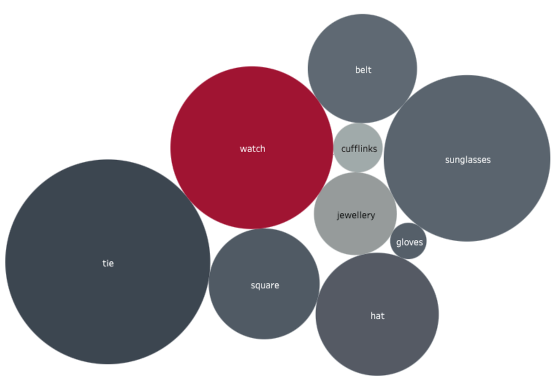
The null hypothesis of the Chi-square test is that the two categorical variables are independent. Given the extremely low (p)-value, we reject the null hypothesis. This implies that there's a significant association between country and brand in the dataset.

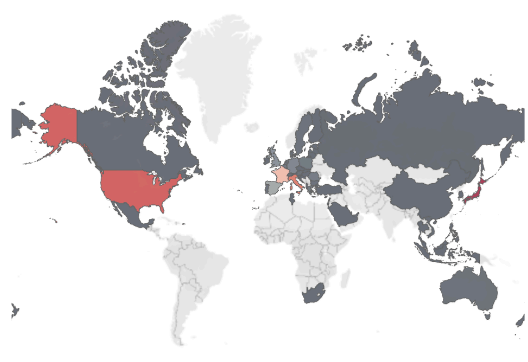
And finally I did a focus on top brands in Japan and average prices.



Tableau

Most of my visualizations used in my presentation have been done on Tableau :
https://public.tableau.com/views/Final_projectPresentation/VestiaireCollectiveCase?:language=fr-FR&publish=yes&:sid=&:redirect=auth&:display_count=n&:origin=viz_share_link
Here are some examples :





Entities & SQL Queries

Entities

The dataset scrapped from the website bucherer.com consists of four primary entities: `Watches`, `Certified_Pre_Owned`, `Special_Edition`, and `Availability`. These entities are organized to capture the detailed attributes and statuses of watches, while maintaining efficient data structure through relational database principles.

1. Watches Entity:
   - This is the central entity of the dataset and contains detailed information about each watch, including attributes such as `WatchID`, `Brand`, `Model_Name`, `Model`, and `Price`.
   - The `Watches` table also includes foreign keys – `Certified_Pre_Owned_ID`, `Special_Edition_ID`, and `Availability_ID` – which link each watch to its respective status across other entities. It has been imported via the wizard import tool from the web scraped data set done before.

2. Certified_Pre_Owned Entity:
   - Contains the `ID` and `Description` fields to indicate whether a watch is certified pre-owned. This entity helps to ensure that descriptions of certified status are consistent across the dataset.

3. Special_Edition Entity:
   - Includes `ID` and `Description` fields that detail whether a watch is a special edition. This entity provides standardized information about the special edition status, contributing to data normalization.

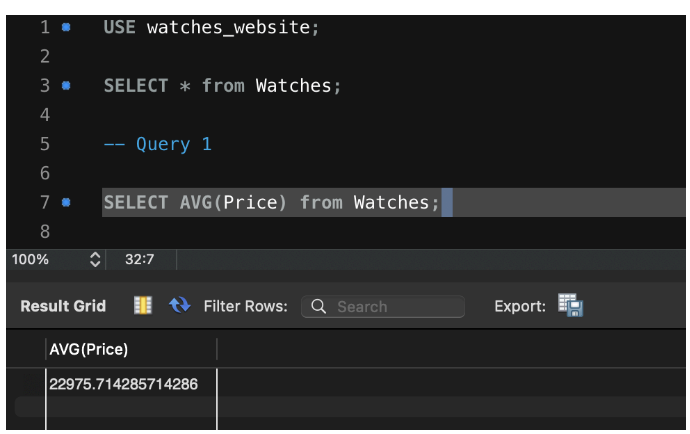
4. Availability Entity:
   - Consists of `ID` and `Description` fields describing the availability of each watch. This entity standardizes the description of availability status across the dataset.

Relationships:
- The `Watches` entity is connected to the `Certified_Pre_Owned`, `Special_Edition`, and `Availability` entities through foreign key constraints. These keys ensure referential integrity, such that each watch's status (whether certified, special edition, or available) is correctly referenced and maintained according to the related entity descriptions.
- The use of foreign keys not only maintains data consistency but also simplifies queries for aggregated or detailed data analysis.

Entity	Attributes	Description
Watches	`WatchID`, `Brand`, `Model_Name`, `Model`, `Price`, `Certified_Pre_Owned_ID`, `Special_Edition_ID`, `Availability_ID`	Main table containing the watch details and foreign keys linking to other attribute tables.
Certified_Pre_Owned	`ID`, `Description`	Contains information about whether a watch is certified pre-owned (0 or 1 with descriptions such as 'Certified Pre-Owned' and 'Not Certified').
Special_Edition	`ID`, `Description`	Stores information about special edition status (0 or 1) with associated descriptions (e.g., 'Special Edition', 'Not Special Edition').
Availability	`ID`, `Description`	Records availability status (0 or 1) with descriptions like 'In Stock', 'Not Available'.

SQL Queries

Query 1 :
Watches global price mean



Query 2 : Finding which watch is a “Special Edition”

```sql
-- Query 2

SELECT w.Brand, w.Model_Name, w.Model, w.Price
FROM Watches w
JOIN Special_Edition se ON w.Special_Edition_ID = se.ID
WHERE se.Description = 'Special Edition';
```

100% 1:16

Result Grid Filter Rows: Search Export:

Brand	Model_Name	Model	Price
Breguet	Type XXII	Type XXII Certified Pre-Owned	10500.0

Query 3 : Getting the list of watches from the brand “Cartier”

```sql
SELECT
    w.Brand,
    w.Model_Name,
    w.Model,
    w.Price,
    se.Description AS Special_Edition_Status,
    av.Description AS Availability_Status,
    cpo.Description AS Certified_Pre_Owned_Status
FROM Watches w
JOIN Special_Edition se ON w.Special_Edition_ID = se.ID
JOIN Availability av ON w.Availability_ID = av.ID
JOIN Certified_Pre_Owned cpo ON w.Certified_Pre_Owned_ID = cpo.ID
WHERE w.Brand = 'Cartier';
```

100% 27:30

Result Grid Filter Rows: Search Export:

Brand	Model_Name	Model	Price	Special_Edition_Stat...	Availability_Stat...	Certified_Pre_Owned_Sta...
Cartier	Pasha Certified Pre Owned	Pasha Certified Pre Owned	3500.0	Not Special Edition	In Stock	Not Certified Pre-Owned
Cartier	Panthere Certified Pre Owned	Panthere Certified Pre Owned	3800.0	Not Special Edition	In Stock	Not Certified Pre-Owned
Cartier	Tank	Tank Certified Pre-Owned	7800.0	Not Special Edition	In Stock	Certified Pre-Owned
Cartier	Captive	Captive Certified Pre-Owned	11550.0	Not Special Edition	In Stock	Certified Pre-Owned
Cartier	Ballon Bleu de Cartier	Ballon Bleu de Cartier Certified Pre-Owned	18000.0	Not Special Edition	In Stock	Certified Pre-Owned

Query 4 : Mean price on Cartier watches

```sql
-- Query 4
SELECT AVG(w.Price) AS Mean_Price
FROM Watches w
WHERE w.Brand = 'Cartier';
```

100% 27:35

Result Grid Filter Rows: Search Export:

Mean_Price
8930

Query 5 : Knowing the stock available for the Cartier watches

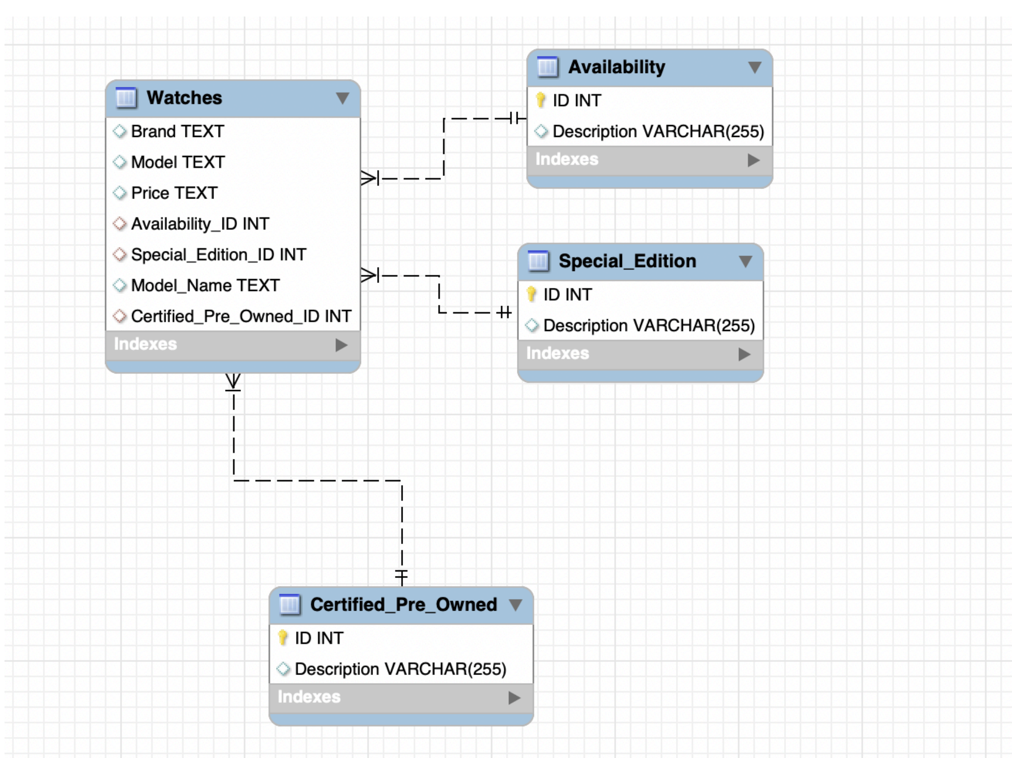
```sql
-- Query 5
SELECT
    w.Brand,
    w.Model_Name,
    w.Model,
    w.Price,
    av.Description AS Availability_Status,
    COUNT(w.Brand) AS NumberOfAvailableWatches
FROM
    Watches AS w
JOIN
    Availability av ON w.Availability_ID = av.ID
WHERE
    av.Description = 'In Stock' AND w.Brand = 'Cartier'
GROUP BY
    w.Brand, w.Model_Name, w.Model, w.Price, av.Description
```

100% 1:55

Result Grid Filter Rows: Search Export:

Brand	Model_Name	Model	Price	Availability_Stat...	NumberOfAvailableWatch...
Cartier	Pasha Certified Pre Owned	Pasha Certified Pre Owned	3500.0	In Stock	1
Cartier	Panthere Certified Pre Owned	Panthere Certified Pre Owned	3800.0	In Stock	1
Cartier	Tank	Tank Certified Pre-Owned	7800.0	In Stock	1
Cartier	Captive	Captive Certified Pre-Owned	11550.0	In Stock	1
Cartier	Ballon Bleu de Cartier	Ballon Bleu de Cartier Certified Pre-Owned	18000.0	In Stock	1

ERD

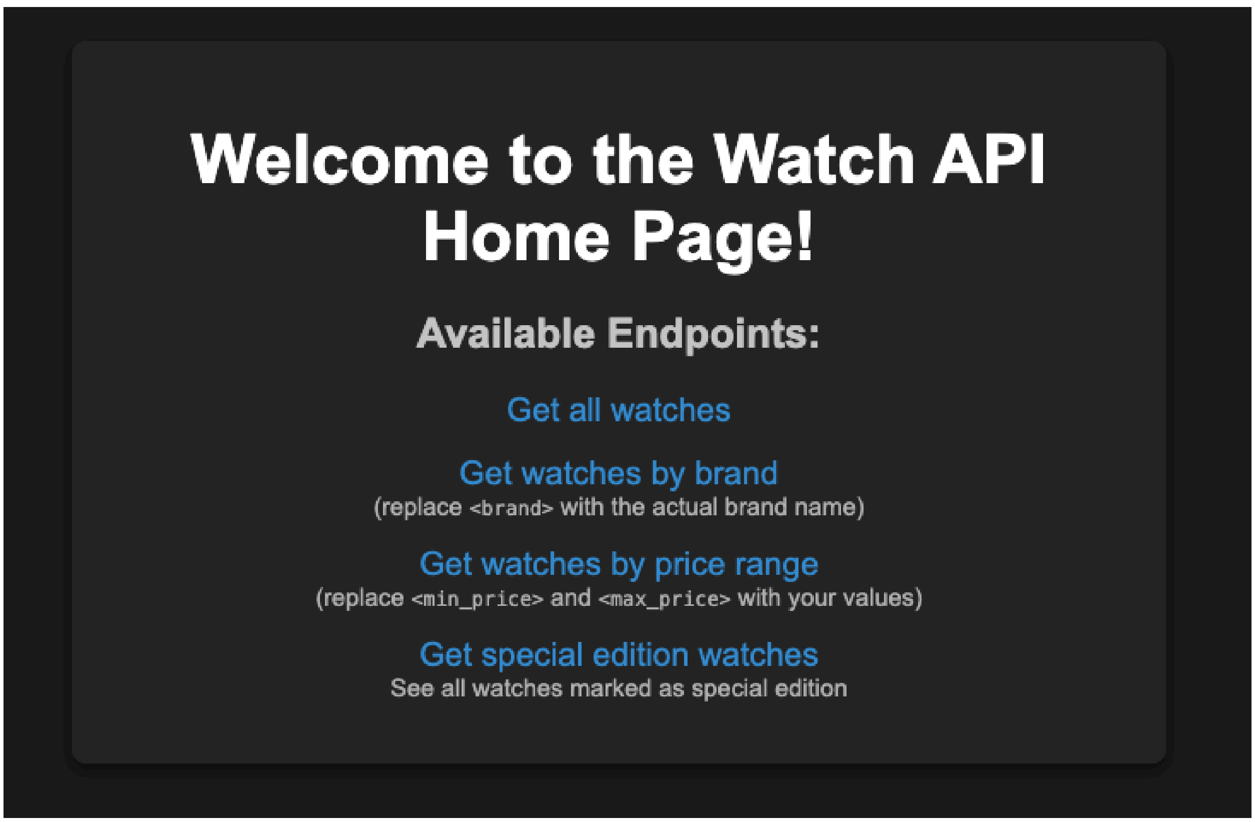


API

The Watch API provides a structured interface to access information about various watches available in our database (the one web scraped on bucherer.com). It allows users to retrieve details about different timepieces based on specific criteria such as brand, price range, and special editions. Below is a description of the main features and available endpoints.

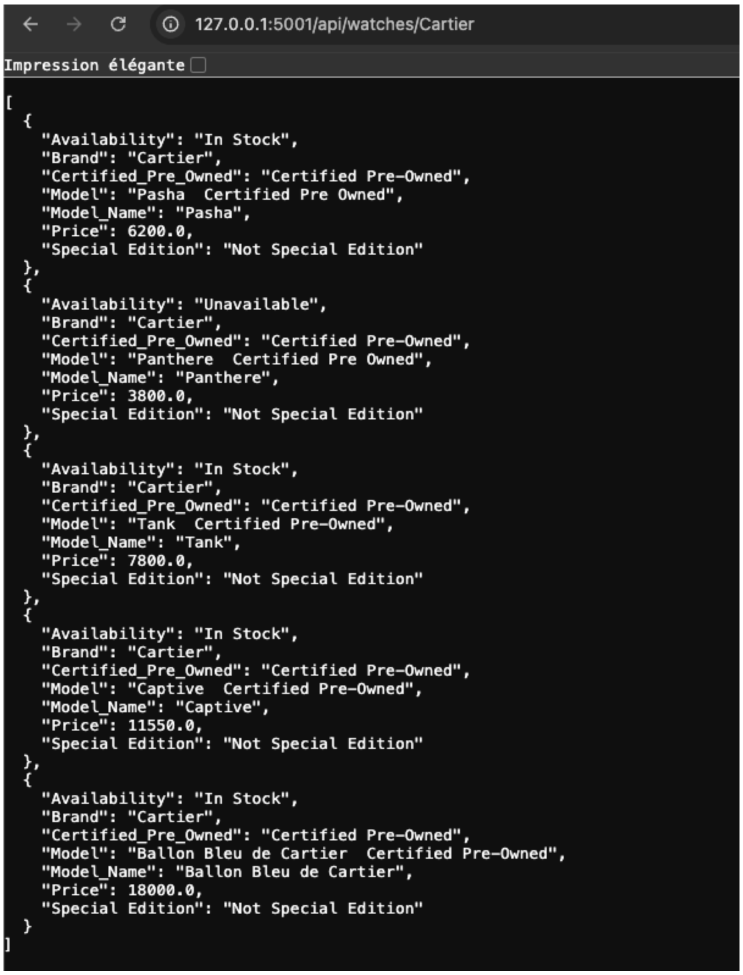
1. Home Endpoint (`/`) : http://127.0.0.1:5001/

The home endpoint serves as the introductory page of the Watch API. It outlines the available functionalities and provides links to access various data points regarding watches.



2. Get All Watches (`/api/watches`)

This endpoint allows users to retrieve a complete list of all watches stored in the database. It returns detailed information about each watch, including attributes such as brand, model, price, and whether it is a special edition. This is useful for users who want an overview of the available watches without filtering.



3. Get Watches by Brand (`/api/watches/<brand>`)

Users can filter the list of watches by brand through this endpoint. By replacing `<brand>` with the desired brand name (e.g : Cartier), the API returns only the watches that match the specified brand. This functionality is particularly beneficial for brand enthusiasts looking for specific watch models.

127.0.0.1:5001/api/watches

Impression élégante


```
[
  {
    "Availability": "In Stock",
    "Brand": "Zenith",
    "Certified_Pre_Owned": "Certified Pre-Owned",
    "Model": "Chronomaster  Certified Pre-Owned",
    "Model_Name": "Chronomaster",
    "Price": 13800.0,
    "Special Edition": "Not Special Edition"
  },
  {
    "Availability": "In Stock",
    "Brand": "Chopard",
    "Certified_Pre_Owned": "Certified Pre-Owned",
    "Model": "L.U.C.  Certified Pre-Owned",
    "Model_Name": "L.U.C.",
    "Price": 20700.0,
    "Special Edition": "Not Special Edition"
  },
  {
```


4. Get Watches by Price Range (`/api/watches/price?min=<min_price>&max=<max_price>`)

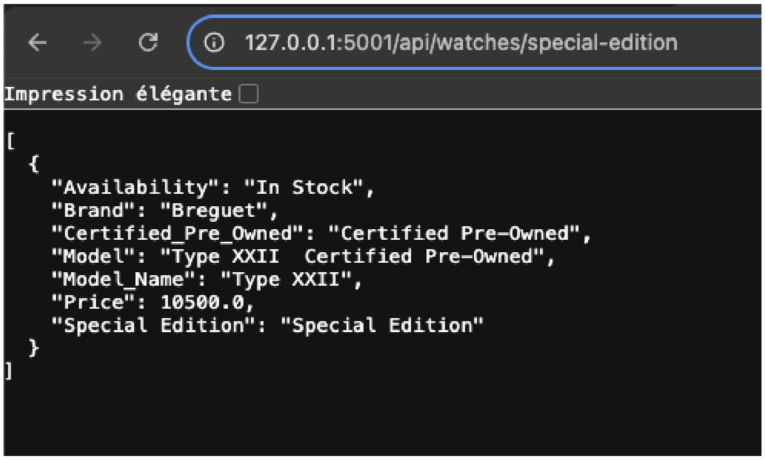
This endpoint enables users to query watches based on their price range. By specifying minimum (`min_price`) and maximum (`max_price`) values, users can obtain a list of watches that fall within their budget.

127.0.0.1:5001/api/watches/price?min=5000&max=10000

Impression élégante


```
[
  {
    "Availability": "In Stock",
    "Brand": "Girard-Perregaux",
    "Certified_Pre_Owned": "Certified Pre-Owned",
    "Model": "1966  Certified Pre-Owned",
    "Model_Name": "1966",
    "Price": 8800.0,
    "Special Edition": "Not Special Edition"
  },
  {
    "Availability": "In Stock",
    "Brand": "Longines",
    "Certified_Pre_Owned": "Certified Pre-Owned",
    "Model": "Master  Certified Pre-Owned",
    "Model_Name": "Master",
    "Price": 5600.0,
    "Special Edition": "Not Special Edition"
  },
  {
    "Availability": "In Stock",
    "Brand": "Cartier",
    "Certified_Pre_Owned": "Certified Pre-Owned",
    "Model": "Pasha  Certified Pre Owned",
    "Model_Name": "Pasha",
    "Price": 6200.0,
    "Special Edition": "Not Special Edition"
  },
  {
    "Availability": "In Stock",
    "Brand": "Carl F. Bucherer",
    "Certified_Pre_Owned": "Certified Pre-Owned",
    "Model": "Patravi Traveltec II Certified Pre-Owned",
    "Model_Name": "Patravi Traveltec II",
    "Price": 7100.0,
    "Special Edition": "Not Special Edition"
  },
  {
```

5. Get Special Edition Watches (`/api/watches/special-edition`)
This endpoint provides access to watches that are categorized as special editions. Users can retrieve a list of unique timepieces that stand out due to their limited availability or exclusive features.



GDPR

To determine if this dataset is compliant with the GDPR (General Data Protection Regulation), we need to examine its content and how the data was collected and processed.
Regarding the Vestiaire Fashion dataset from Kagglet: it contains information about fashion items, including brands, prices, descriptions, and categories. This data does not appear to include personal data as defined by the GDPR, besides , the information comes from Vestiaire Collective, a platform for reselling luxury clothing. Therefore, it consists of publicly accessible data.
In addition, the dataset on Kaggle is under a CC0: Public Domain license, which means that its creator has waived all copyright and related rights on this data.

Conclusion

Analysis Findings:
Our different analysis compared the prices of watches against non-watch products, identifying significant price differences. The statistically validated higher average prices for watches suggest that watches, especially premium brands, form a lucrative category within our product lineup.

Business Implications:

1. Sourcing Top Brands:
   - Focus on Key Luxury Watch Brands: Given the significant price disparity, investing in top-tier brands like Rolex, Omega, Tag Heuer, and Cartier could capture market demand for luxury items. These brands command strong pricing power and consumer recognition, enhancing our brand portfolio's prestige and profitability.

2. Investigate the Japanese Second-Hand Market:
   - Leverage the Quality and Prestige: Japan's second-hand market is renowned for high-quality, meticulously-kept luxury items, including watches. By sourcing from this market, we can diversify our inventory with well-maintained, premium watches at competitive prices.
   - Access to Rare and Vintage Pieces: Engaging with this market may provide access to rare, sought-after timepieces, attracting collectors and enthusiasts while potentially yielding high margins.

3. Deep Dive into the Secondary Market in Europe:
   - Explore Mature Markets: Europe's secondary watch market is mature, offering robust opportunities for finding both contemporary and vintage high-value watches.
   - Currency and Pricing Strategies: A focused analysis on pricing dynamics and exchange rates in Europe could enhance competitive pricing strategies, ensuring attractive options for both buyers and sellers.

4. Set Partnerships with Luxury Watch Brands:
   - Mutually Beneficial Collaborations: Establish exclusive partnerships or consignment arrangements with luxury watch brands to ensure authenticated, top-quality inventory and potentially secure better pricing terms.
   - Enhanced Brand Loyalty and Marketing: Collaboration with prestigious watch brands can fortify customer trust, enhance brand identity, and offer joint marketing opportunities to reach affluent consumer demographics.

Strategic Recommendations:

To capitalize on these insights, our approach should include targeted sourcing strategies, focused market exploration efforts, and strategic brand partnerships. Expanding inventory through these channels allows us to leverage the robust luxury market, meet consumer demands effectively, and position our company as a leading purveyor of luxury watches in the global marketplace.

REFERENCES

Flat file :
https://www.kaggle.com/datasets/justinpakzad/vestiaire-fashion-dataset

Web Scraping :

https://www.bucherer.com/fr/en/buy-certifiedpreowned?srule=Ranking+by+Category+Position&start=0&sz=72
https://www.bcg.com/publications/2022/the-impact-of-secondhand-market-on-fashion-retailers

https://www.bcg.com/publications/2023/luxury-watch-market-trends#:~:text=Preowned%20watch%20sales%20reached%20%2422,trend%20is%20likely%20to%20continue.

https://usa.watchpro.com/how-gen-z-is-reshaping-the-watch-market/

Tableau Story :
https://public.tableau.com/app/profile/tvc94/viz/Final_projectPresentation/VestiaireCollectiveCase

Github Repository :
https://github.com/tvc94/rncp_project