Fashion
Final Project
Data Analysis

Emma Trivini Bellini

Introduction

Imagine working for a second-hand clothing marketplace, a platform promoting sustainable fashion.

The marketplace offers people an easy and safe way to resell their used clothes, giving clothes a second life and promoting a more sustainable lifestyle.

Here, anyone can offer clothes, accessories and footwear for sale, allowing others to find unique pieces at affordable prices.

The platform is designed for people who love fashion but also want to reduce their environmental impact, offering an ethical alternative to fast fashion.

Every purchase helps reduce waste and promotes a circular economy.

Dataset

To visualize the entire dataset, click on this link.

user_uuid	Identifier of the selling user
category	Category of clothing
designer_id	Identifier of the designer
language	Language of the selling user
level	Price range
country	Nationality of the selling user
purchase_date	The date on which the user sold the product
platform	Platform from which the payment was made
item_id	Product identifier
stars	Average stars assigned to the product
subscription_date	Day on which the selling user subscribed

EDA with Python

Exploratory Data Analysis

Exploratory Data Analysis (EDA) is the fundamental process of understanding and becoming familiar with a dataset before carrying out further statistical analyses. It consists of exploring and visualizing the data to uncover patterns, trends, anomalies, and relationships among variables. EDA with Python leverages the power and flexibility of the Python programming language to perform analysis efficiently and intuitively. One of the main Python libraries used for EDA is Pandas, which is specialized in managing and analyzing tabular data.

In the non-graphical approach, I use functions such as shape, summary, describe, isnull, info, dtypes, and others.

In the graphical approach, I use plots such as scatter plots, box plots, bar charts, density plots, and correlation heatmaps.

Exploratory Data Analysis in Python

Data Loading: the dataset is loaded into the Python working environment using Pandas. Subsequently, the data can be read from CSV, Excel, databases, or other sources. (In the present analysis, we worked with a CSV file).

Library Importing: the fundamental libraries required for data analysis are imported, including Pandas, NumPy, and Matplotlib. These libraries provide advanced functionalities for data manipulation, processing, and visualization.

Initial Data Exploration: the first step consists of exploratory data analysis, which examines the dataset rows in order to understand the structure and format of the data. (In the present analysis, we employed the methods head(), info(), and describe(), which are particularly useful for obtaining a general overview of the dataset).

Exploratory Data Analysis in Python

Data Cleaning: this step involves identifying and handling missing values, duplicate records, or potential errors within the dataset. During this phase of the analysis, we employed methods such as isnull(), drop_duplicates(), and fillna(), which are essential for ensuring proper dataset cleaning.

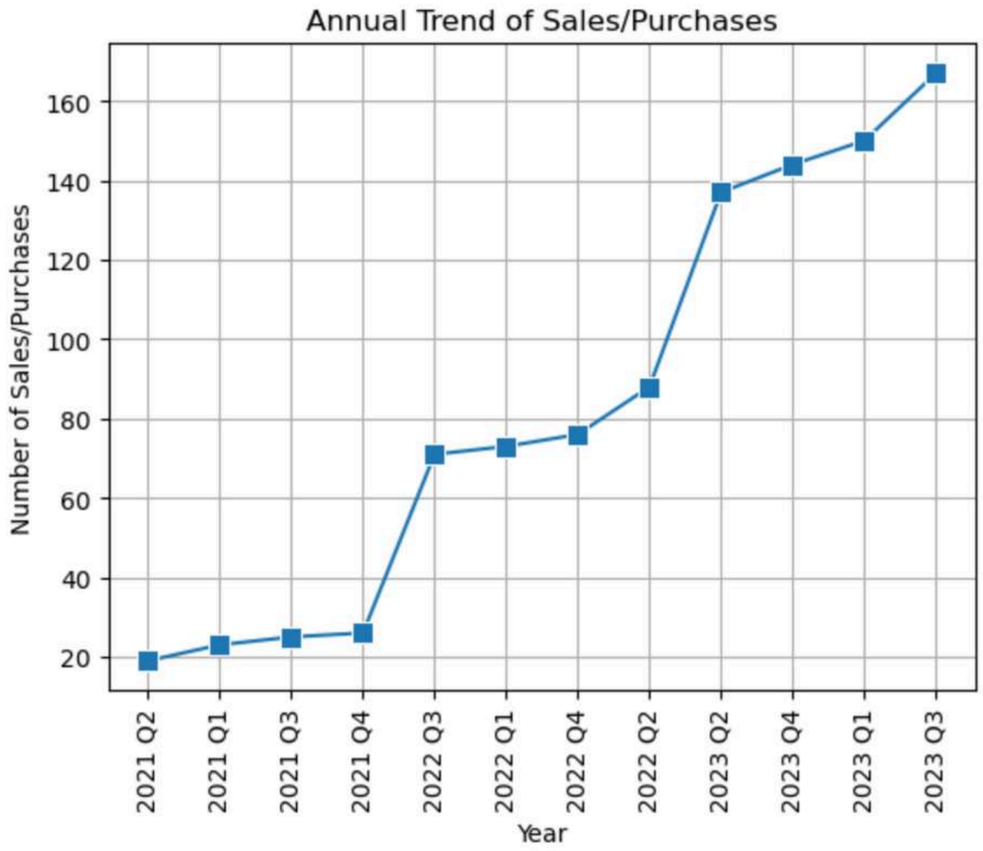
Variable Analysis: the different variables within the dataset are explored, and statistical measures such as mean, median, and standard deviation are computed.

Exploration of Relationships Between Variables: the relationships among the dataset variables are analyzed using visualizations such as scatter plots, correlation matrices, and bar charts.

Sales Evolution

```python
# DATA ANALYSIS AND VISUALIZATION
# calculate the number of sales for each year-quarter
sales_per_yearquarter = dtfashion['yquarter_purchase'].value_counts()
# sort the results in ascending order
sales_per_yearquarter_asc = sales_per_yearquarter.sort_values(ascending=True)
sales_per_yearquarter_asc
```

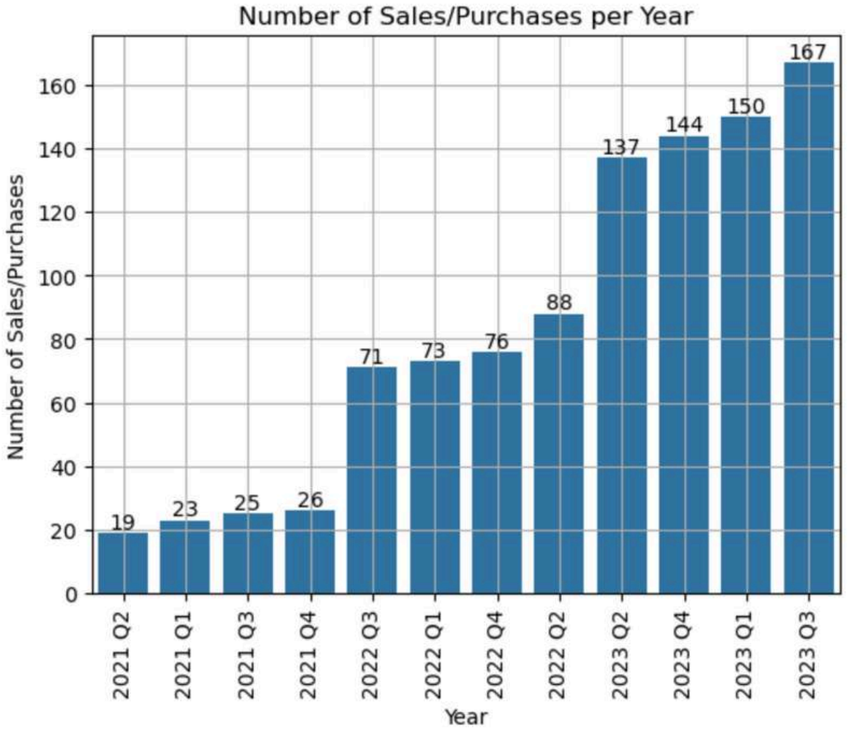
```python
# create a line plot to show the annual trend of sales
line_plot = sns.lineplot(
    x=sales_per_yearquarter_asc.index,
    y=sales_per_yearquarter_asc.values,
    marker='s',
    markersize=8
)

plt.xlabel('Year')
plt.ylabel('Number of Sales/Purchases')
plt.title('Annual Trend of Sales/Purchases')

plt.xticks(rotation=90)
plt.grid(True)
plt.show()
```



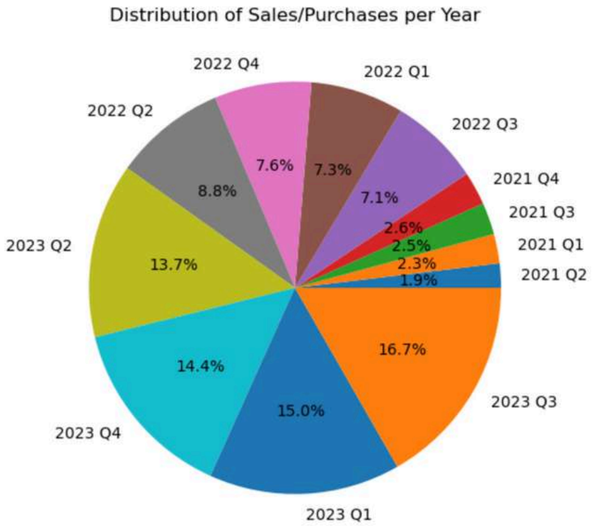

```
yquarter_purchase
2021 Q2     19
2021 Q1     23
2021 Q3     25
2021 Q4     26
2022 Q3     71
2022 Q1     73
2022 Q4     76
2022 Q2     88
2023 Q2    137
2023 Q4    144
2023 Q1    150
2023 Q3    167
```

Sales Evolution

```python
# visualize sales grouped by year using a bar plot
bar_plot = sns.barplot(x=sales_per_yearquarter_asc.index, y=sales_per_yearquarter_asc.values)

plt.xlabel('Year')
plt.ylabel('Number of Sales/Purchases')
plt.title('Number of Sales/Purchases per Year')
plt.grid(True)
plt.xticks(rotation=90)

# add value labels on top of each bar
for index, value in enumerate(sales_per_yearquarter_asc.values):
    bar_plot.text(index, value, str(value), ha='center', va='bottom')

plt.show()
```




```python
# create a pie chart for sales grouped by year
plt.figure(figsize=(10, 6))
plt.pie(sales_per_yearquarter_asc.values, labels=sales_per_yearquarter_asc.index, autopct='%1.1f%%')
plt.title('Distribution of Sales/Purchases per Year')
plt.show()
```



Sales Trend by Platform

```python
# calculate the number of purchases for each platform
purchases_per_platform = dtfashion['platform'].value_counts()
purchases_per_platform
```

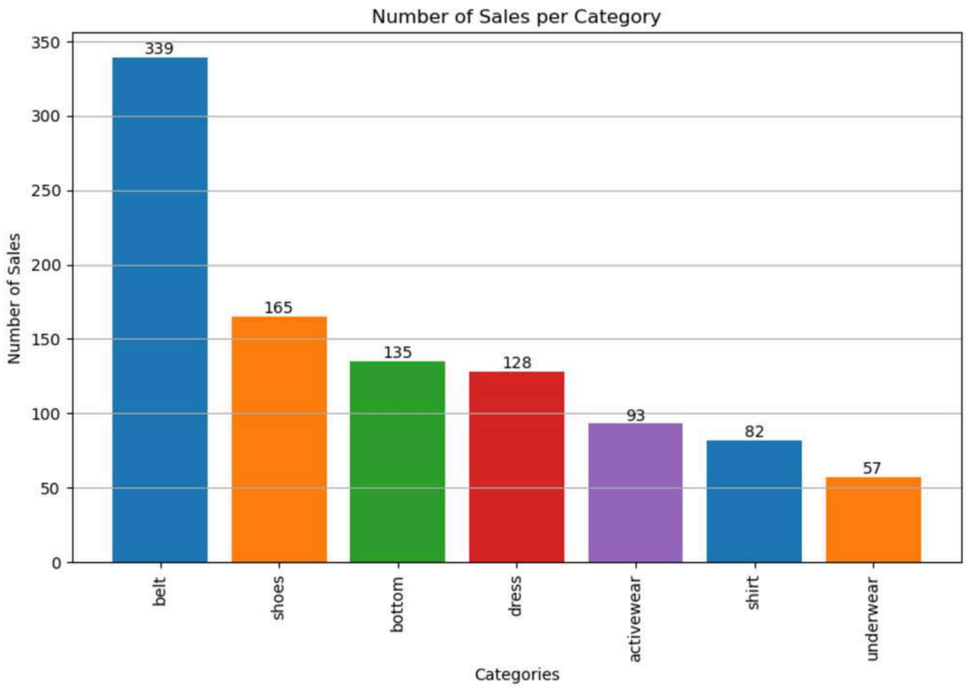
```python
# calculate the number of purchases for each platform and plot a bar chart
bar_plot = sns.barplot(
    x=purchases_per_platform.index,
    y=purchases_per_platform.values,
    hue=purchases_per_platform.index,  # assign x values to hue
    palette="Set2",
    dodge=False
)

plt.xlabel('Platform')
plt.ylabel('Number of Sales/Purchases')
plt.title('Number of Sales/Purchases per Platform')
plt.grid(True)
plt.xticks(rotation=90)
plt.legend([],[], frameon=False)  # remove legend

for index, value in enumerate(purchases_per_platform.values):
    bar_plot.text(index, value, str(value), ha='center', va='bottom')

plt.show()
```

```
platform
android    458
pc         330
apple      211
Name: count, dtype: int64
```

Sales Trend by Category

```python
# calculate the number of sales for each product category
sales_per_category = dtfashion['category'].value_counts()
sales_per_category
```

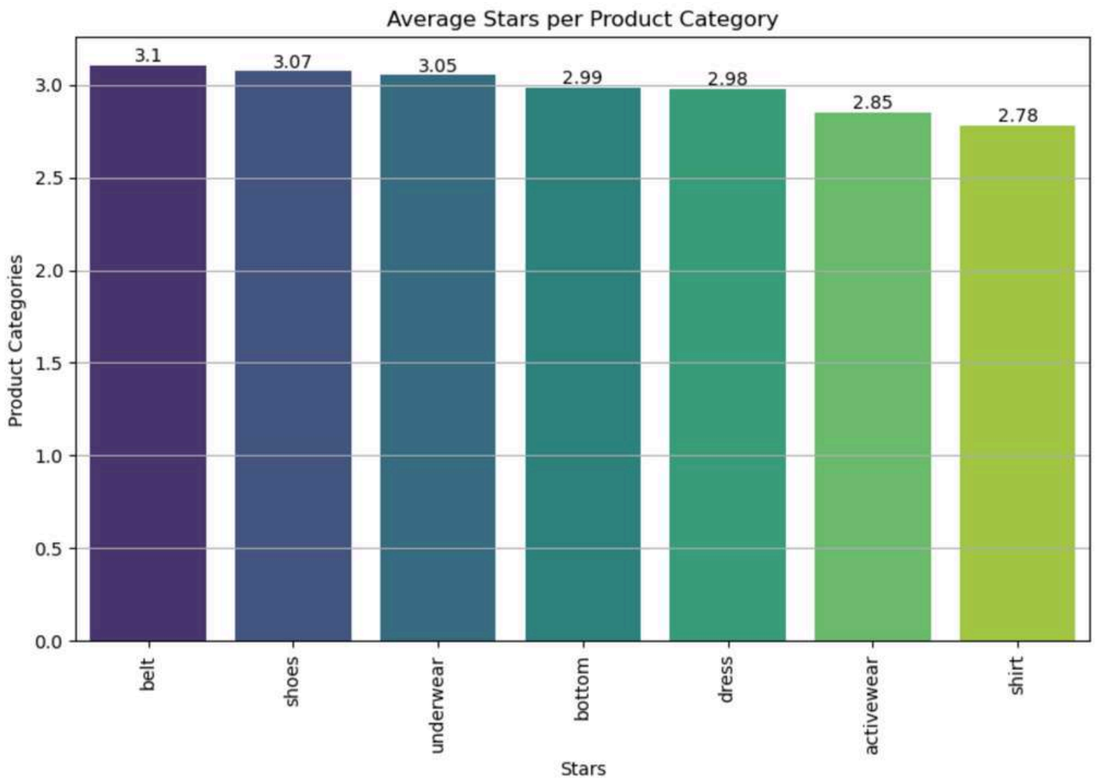
```python
# visualize these sales using a bar chart
colors = ["#1f77b4", "#ff7f0e", "#2ca02c", "#d62728", "#9467bd"]

plt.figure(figsize=(10,6))
plt.bar(sales_per_category.index, sales_per_category.values, color=colors[:len(sales_per_category)])
plt.xlabel('Categories')
plt.ylabel('Number of Sales')
plt.title('Number of Sales per Category')
plt.xticks(rotation=90)
plt.grid(True, axis='y')

for index, value in enumerate(sales_per_category.values):
    plt.text(index, value, str(value), ha='center', va='bottom')

plt.show()
```




```
category
belt          339
shoes         165
bottom        135
dress         128
activewear     93
shirt          82
underwear      57
Name: count, dtype: int64
```

Product Satisfaction Analysis by Category

```python
# calculate the average stars assigned to products grouped by product category
average_stars_per_category = dtfashion.groupby('category')['stars'].mean()
average_stars_per_category_desc = average_stars_per_category.sort_values(ascending=False)
average_stars_per_category_desc

# create a bar chart to show the average stars per product category
plt.figure(figsize=(10,6))
bar_plot = sns.barplot(
    x=average_stars_per_category_desc.index,
    y=average_stars_per_category_desc.values,
    hue=average_stars_per_category_desc.index,  # assign x to hue
    dodge=False,
    palette="viridis",
    legend=False
)

plt.xlabel('Stars')
plt.ylabel('Product Categories')
plt.title('Average Stars per Product Category')
plt.xticks(rotation=90)
plt.grid(True, axis='y')

for index, value in enumerate(average_stars_per_category_desc.values):
    plt.text(index, value, s=round(value, 2), ha='center', va='bottom')

plt.show()
```



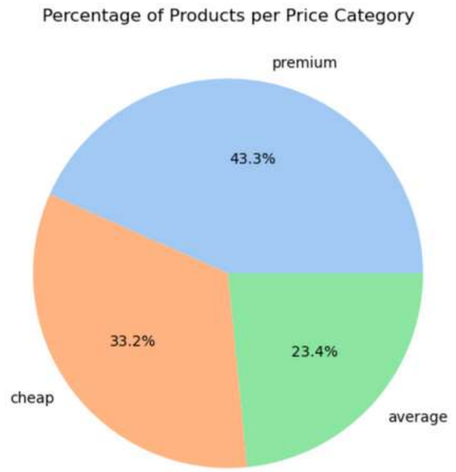

```
category
belt          3.103245
shoes         3.072727
underwear     3.052632
bottom        2.985185
dress         2.976562
activewear    2.849462
shirt         2.780488
Name: stars, dtype: float64
```

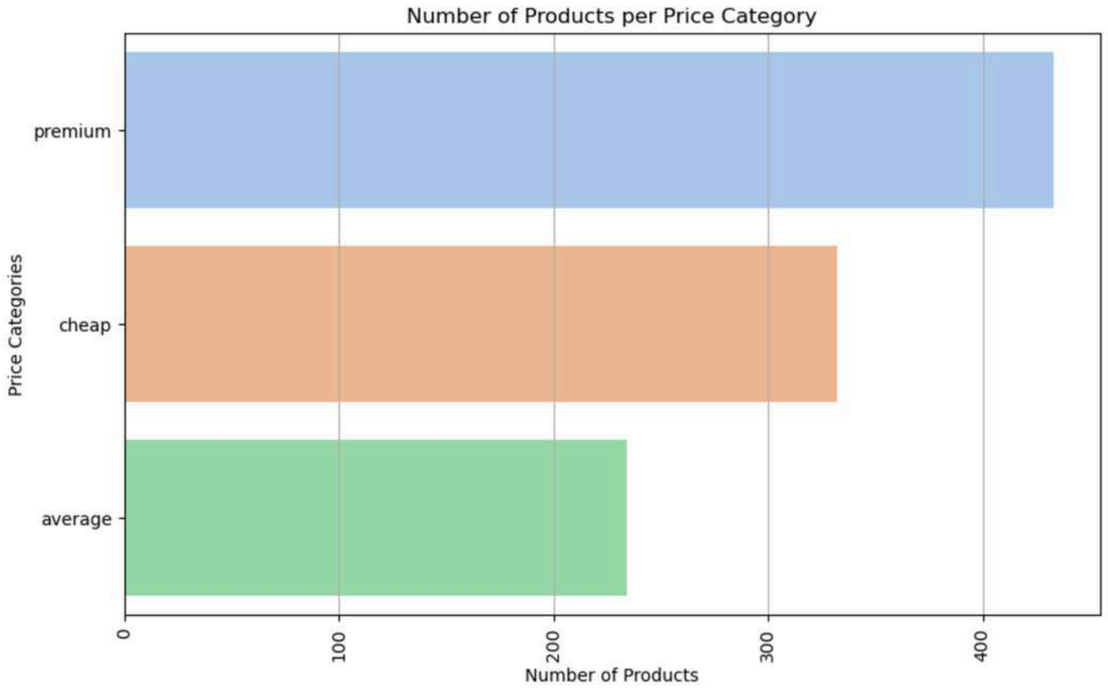
Product Analysis by Price Category

```python
# calculate the number of products grouped by their corresponding price level
products_per_price_level = dtfashion['level'].value_counts()
products_per_price_level
```

```python
# create a bar chart for products grouped by price level
colors = sns.color_palette("pastel", n_colors=len(products_per_price_level))

plt.figure(figsize=(10,6))
bar_plot = sns.barplot(
    x=products_per_price_level.values,
    y=products_per_price_level.index,
    palette=colors,
    hue=products_per_price_level.index,
    dodge=False,
    legend=False
)
plt.xlabel('Number of Products')
plt.ylabel('Price Categories')
plt.title('Number of Products per Price Category')
plt.xticks(rotation=90)
plt.grid(True, axis='x')
plt.show()
```

```
level
premium    433
cheap      332
average    234
Name: count, dtype: int64
```

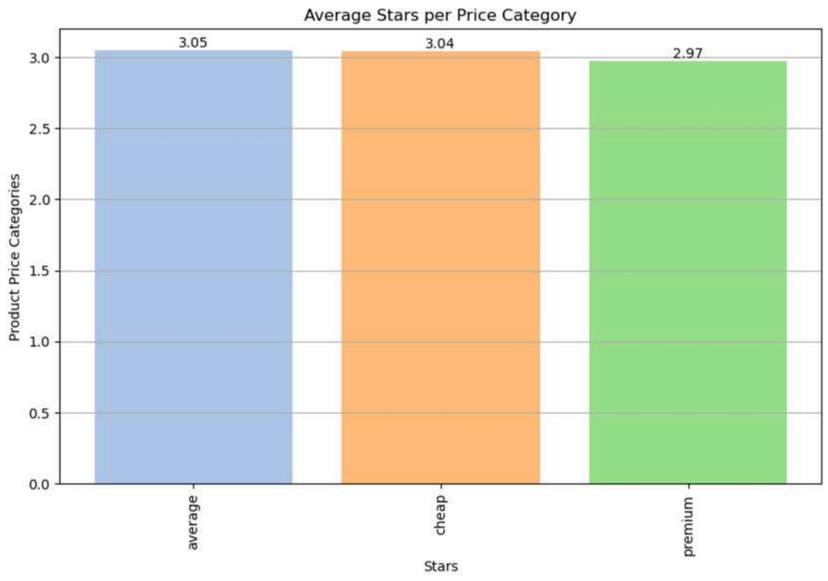




Product Satisfaction Analysis by Price Category

```python
# calculate the average stars assigned to products grouped by price category
avg_stars_per_price_level = dtfashion.groupby('level')['stars'].mean()
avg_stars_per_price_level_desc = avg_stars_per_price_level.sort_values(ascending=False)
avg_stars_per_price_level_desc
```

```python
# create a line chart to show the average stars assigned to each price category
colors = ["#aec7e8", "#ffbb78", "#98df8a", "#ff9896", "#c5b0d5"]

plt.figure(figsize=(10, 6))
for i, (category, value) in enumerate(avg_stars_per_price_level_desc.items()):
    plt.bar(category, value, color=colors[i % len(colors)])

plt.xlabel('Stars')
plt.ylabel('Product Price Categories')
plt.title('Average Stars per Price Category')
plt.xticks(rotation=90)
plt.grid(True, axis='y')

for index, value in enumerate(avg_stars_per_price_level_desc.values):
    plt.text(index, value, s=round(value, 2), ha='center', va='bottom')

plt.show()
```

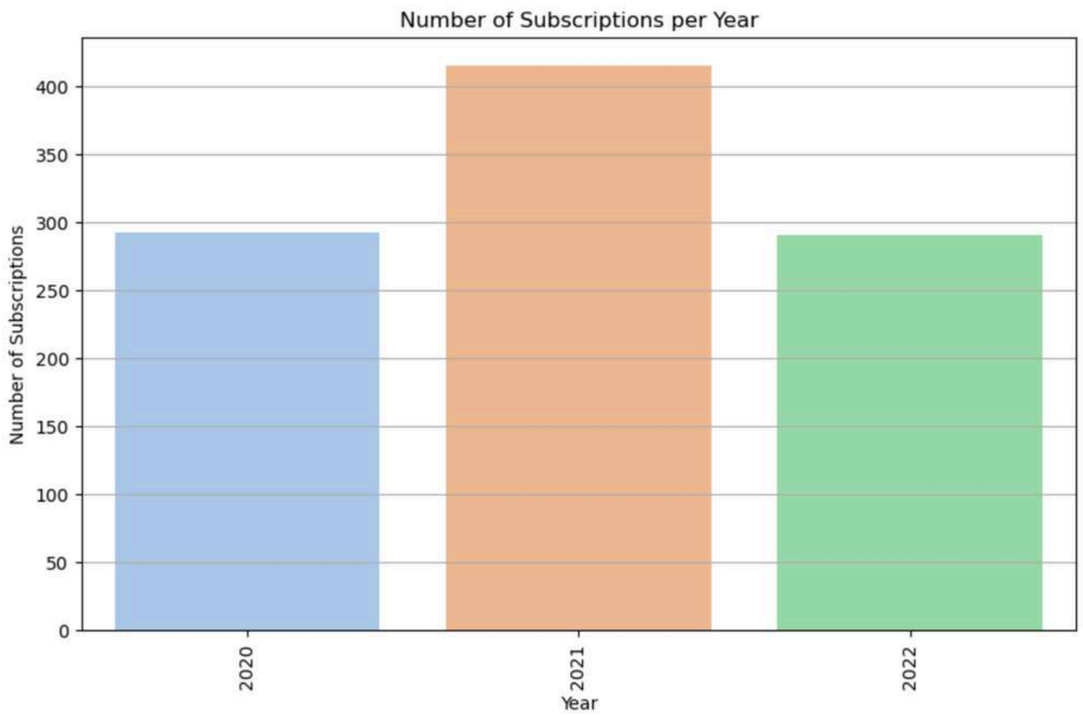



```
level
average    3.047009
cheap      3.042169
premium    2.972286
Name: stars, dtype: float64
```

Analysis of Annual Registrations

```
# calculate the number of subscriptions per year
subscriptions_per_year = dtfashion['subscription_year'].value_counts()
subscriptions_per_year
```

```
# visualize the subscriptions grouped by year using a bar chart
plt.figure(figsize=(10, 6))
bar_plot = sns.barplot(
    x=subscriptions_per_year.index,
    y=subscriptions_per_year.values,
    hue=subscriptions_per_year.index,
    palette="pastel",
    dodge=False,
    legend=False
)

plt.xlabel('Year')
plt.ylabel('Number of Subscriptions')
plt.title('Number of Subscriptions per Year')
plt.grid(True, axis='y')
plt.xticks(rotation=90)

plt.show()
```



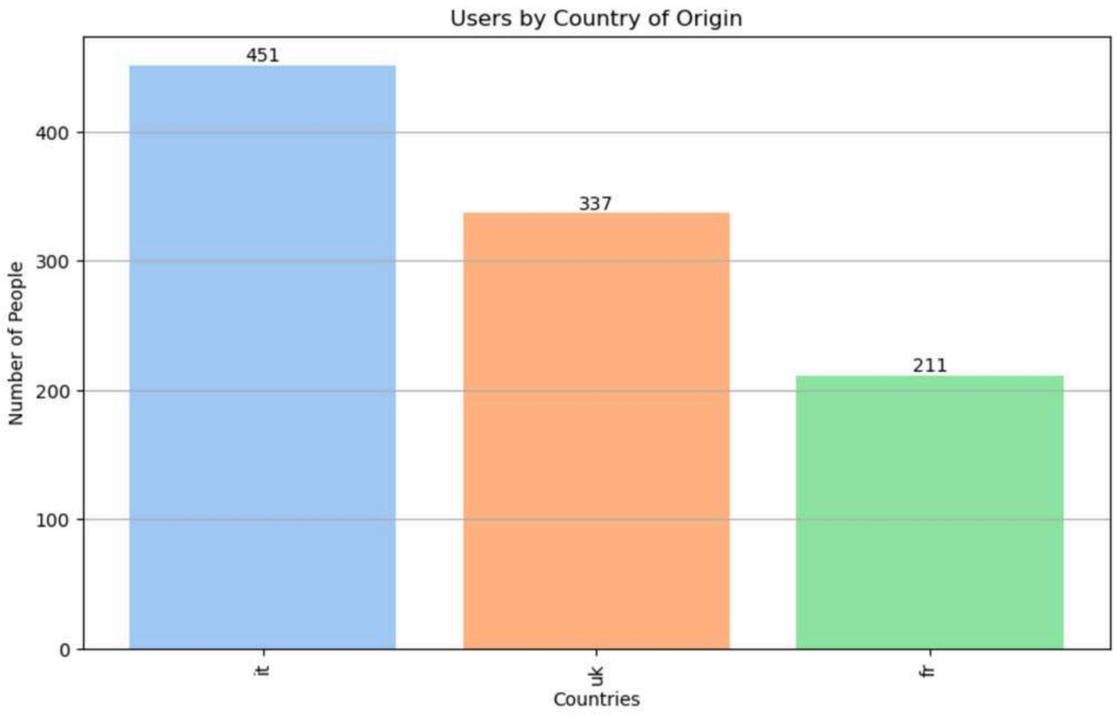

```
subscription_year
2021    415
2020    293
2022    291
Name: count, dtype: int64
```

Nationality Analysis of Subscribers

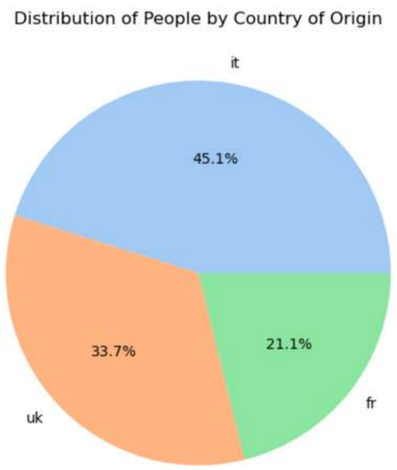
```python
# calculate the number of people grouped by country of origin
people_per_country = dtfashion['country'].value_counts()
people_per_country
```

```python
# create a bar chart showing the distribution of people by country of origin
colors = sns.color_palette("pastel", len(people_per_country))

plt.figure(figsize=(10, 6))
bars = plt.bar(people_per_country.index, people_per_country.values, color=colors)
plt.xlabel('Countries')
plt.ylabel('Number of People')
plt.title('Users by Country of Origin')
plt.grid(True, axis='y')
plt.xticks(rotation=90)

# add numbers above bars
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, height, str(height), ha='center', va='bottom')

plt.show()
```

```python
# create a pie chart showing the distribution of people by country of origin
colors = sns.color_palette("pastel", len(people_per_country))

plt.figure(figsize=(10, 6))
plt.pie(people_per_country.values, labels=people_per_country.index, autopct='%1.1f%%', colors=colors)
plt.title('Distribution of People by Country of Origin')
plt.show()
```



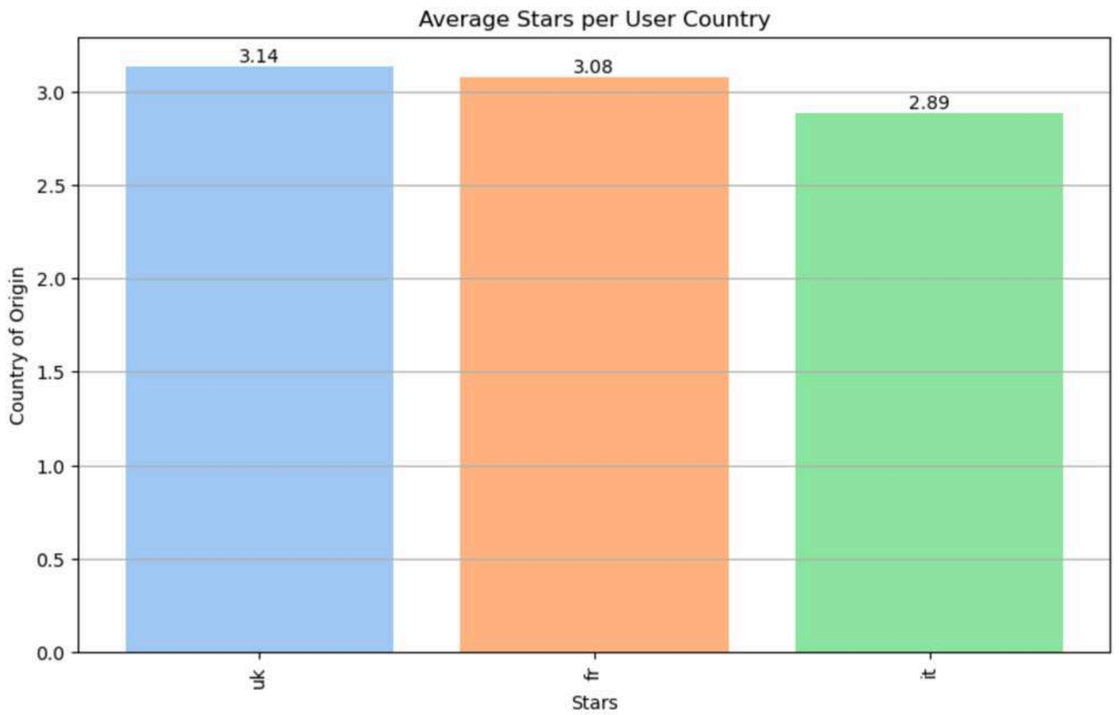

```
country
it    451
uk    337
fr    211
Name: count, dtype: int64
```



Purchase Satisfaction Analysis by User Country of Origin

```python
# calculate the average stars assigned to sales grouped by the user's country of origin
avg_stars_per_country = dtfashion.groupby('country')['stars'].mean()
avg_stars_per_country_desc = avg_stars_per_country.sort_values(ascending=False)
avg_stars_per_country_desc
```

```python
colors = sns.color_palette("pastel", len(avg_stars_per_country_desc))

plt.figure(figsize=(10, 6))
plt.bar(
    avg_stars_per_country_desc.index,
    avg_stars_per_country_desc.values,
    color=colors
)
plt.xlabel('Stars')
plt.ylabel('Country of Origin')
plt.title('Average Stars per User Country')
plt.xticks(rotation=90)
plt.grid(True, axis='y')

for index, value in enumerate(avg_stars_per_country_desc.values):
    plt.text(index, value, s=round(value, 2), ha='center', va='bottom')
```




```
country
uk    3.136499
fr    3.080569
it    2.889135
Name: stars, dtype: float64
```

EDA Conclusions – Exploratory Data Analysis

After conducting EDA, we can state that:

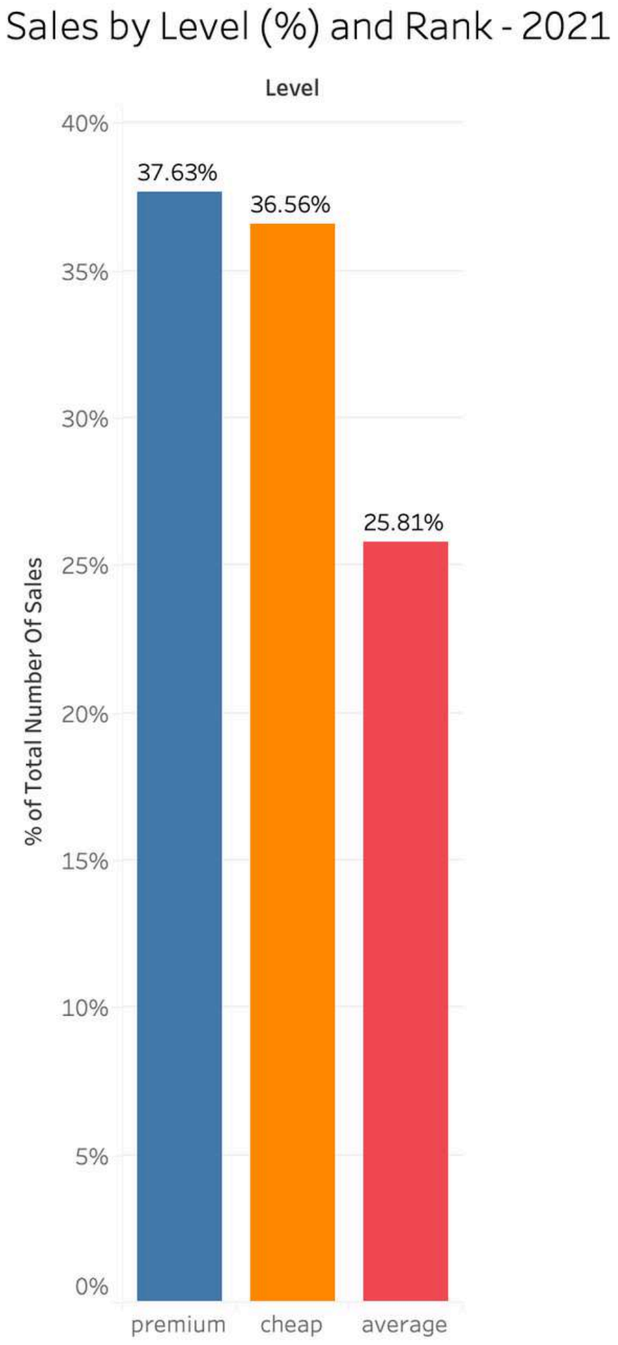
- the category with the highest number of sales is Belts;
- most purchases fall within the expensive price range, meaning they belong to Premium users;
- the most appreciated category among users is Belts;
- the users in the dataset come from Italy, England, and France, with the majority from Italy;
- most purchases are made through the Android platform;
- there are 241 unique users;
- sales per user type range between 2.4 and 2.9;
- the most reliable sellers, who are assigned the highest ratings, belong to the Premium category.

Data Exploration 2021

For the 2021 analysis, I focused exclusively on SQL queries to structure and explore the dataset. First, I created the main table (**fashion_data**) containing all relevant fields such as user identifiers, product categories, designer information, country, platform, and purchase details. Then, I prepared a dedicated view (**fashion_data_2021**) by filtering purchases made between January and December 2021, while also enriching the dataset with derived attributes such as year, quarter, and month of purchase and subscription. Finally, I performed several analytical queries: (1) calculating sales distribution by user level with percentages and ranking, (2) identifying the top three countries per user level based on distinct users, (3) breaking down sales by platform and category while measuring each category's share within the platform, (4) determining the most used platform per level including relative percentage usage, and (5) identifying the top-selling designer for each category in 2021.

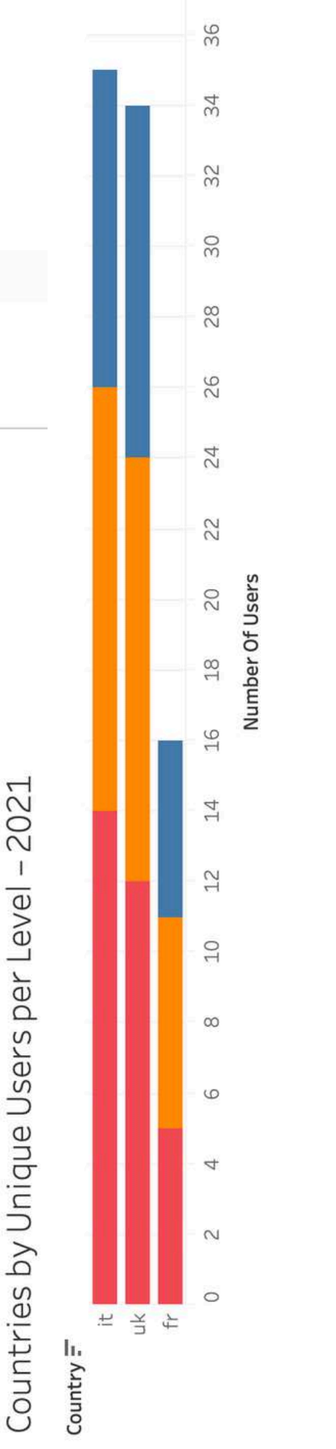
Sales distribution by user level with percentages and ranking

```sql
-- SECTION 3: Analytical Queries for 2021

-- 1. Sales by 'level' with percentage and ranking
WITH sales_per_level AS (
    SELECT
        level,
        COUNT(item_id) AS number_of_sales
    FROM fashion_data_2021
    GROUP BY level
)
SELECT
    level,
    number_of_sales,
    ROUND(100.0 * number_of_sales / SUM(number_of_sales) OVER (), 2) AS percentage_of_total,
    RANK() OVER (ORDER BY number_of_sales DESC) AS sales_rank
FROM sales_per_level
ORDER BY sales_rank;
```



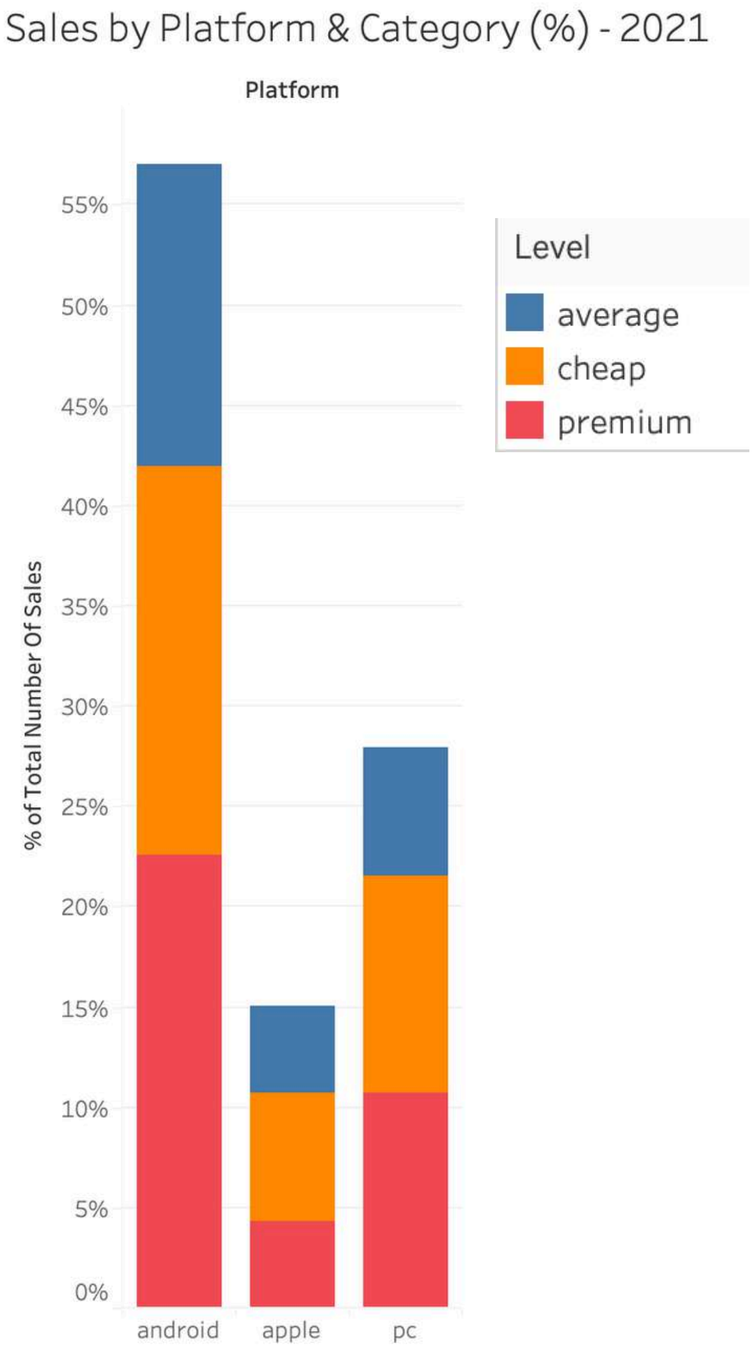
Top three countries per user level based on distinct users

```sql
-- 2. Top 3 countries per 'level' by distinct users
WITH country_level_users AS (
    SELECT
        country,
        level,
        COUNT(DISTINCT user_uuid) AS number_of_users
    FROM fashion_data_2021
    GROUP BY country, level
)
SELECT *
FROM (
    SELECT
        country,
        level,
        number_of_users,
        RANK() OVER (PARTITION BY level ORDER BY number_of_users DESC) AS country_rank
    FROM country_level_users
) ranked
WHERE country_rank <= 3
ORDER BY level, country_rank;
```



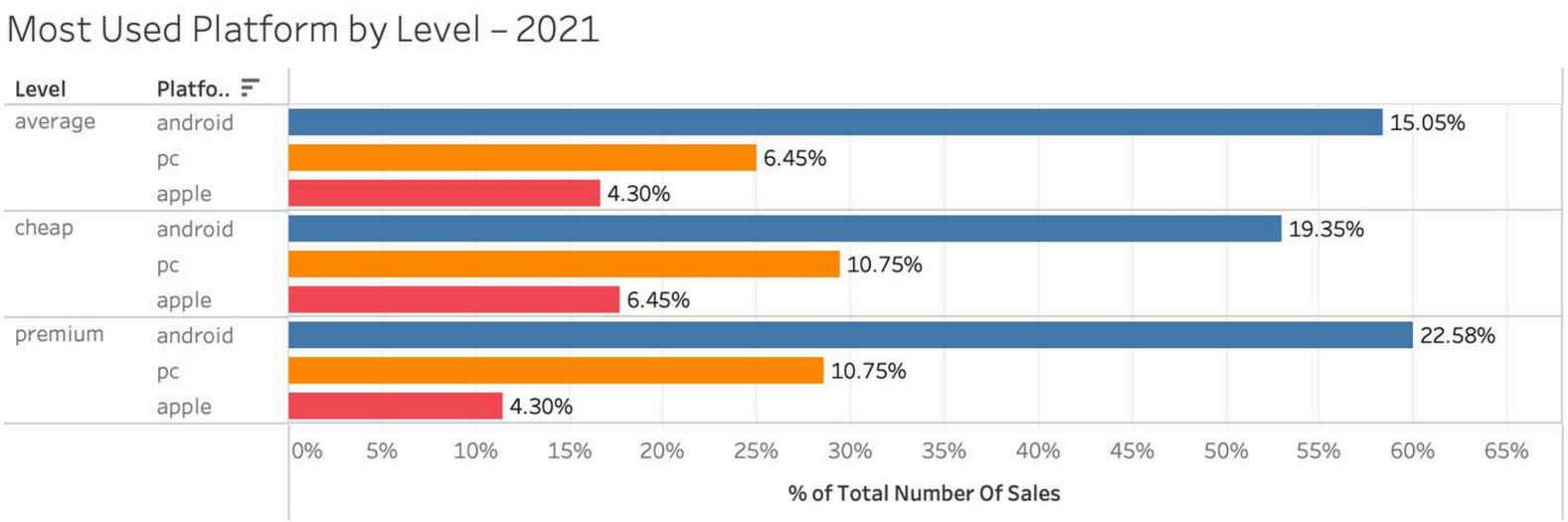
Sales breakdown by platform and category with share per platform

```sql
-- 3. Sales breakdown by platform and category with share per platform
WITH platform_sales AS (
    SELECT
        platform,
        category,
        level,
        COUNT(item_id) AS number_of_sales
    FROM fashion_data_2021
    GROUP BY platform, category, level
)
SELECT
    platform,
    category,
    level,
    number_of_sales,
    ROUND(100.0 * number_of_sales / SUM(number_of_sales) OVER (PARTITION BY platform), 2) AS share_within_platform
FROM platform_sales
ORDER BY platform, number_of_sales DESC;
```



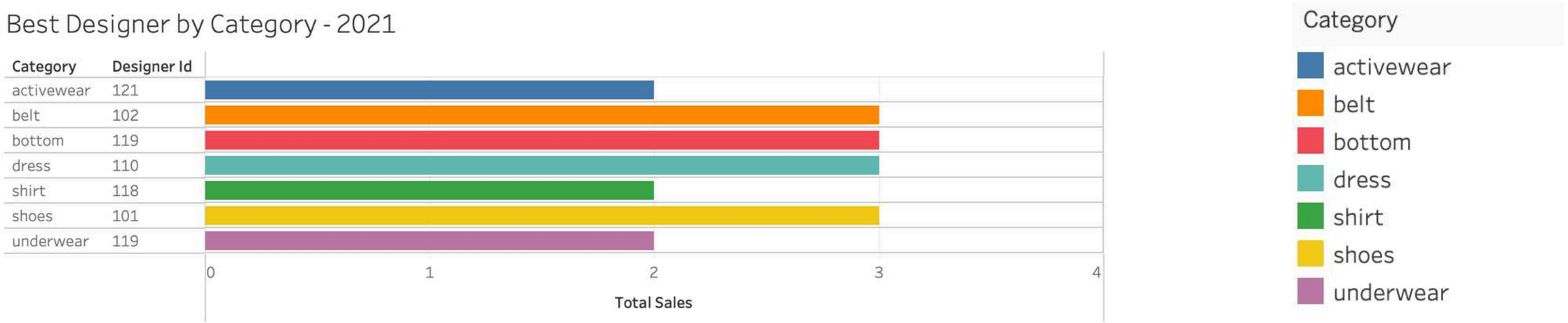
The most used platform by level

```sql
-- 4. Most used platform per 'level' including percentage usage
WITH platform_level_sales AS (
    SELECT
        platform,
        level,
        COUNT(item_id) AS number_of_sales
    FROM fashion_data_2021
    GROUP BY platform, level
)
SELECT
    platform,
    level,
    number_of_sales,
    ROUND(100.0 * number_of_sales / SUM(number_of_sales) OVER (PARTITION BY level), 2) AS share_within_level
FROM platform_level_sales
ORDER BY level, number_of_sales DESC;
```

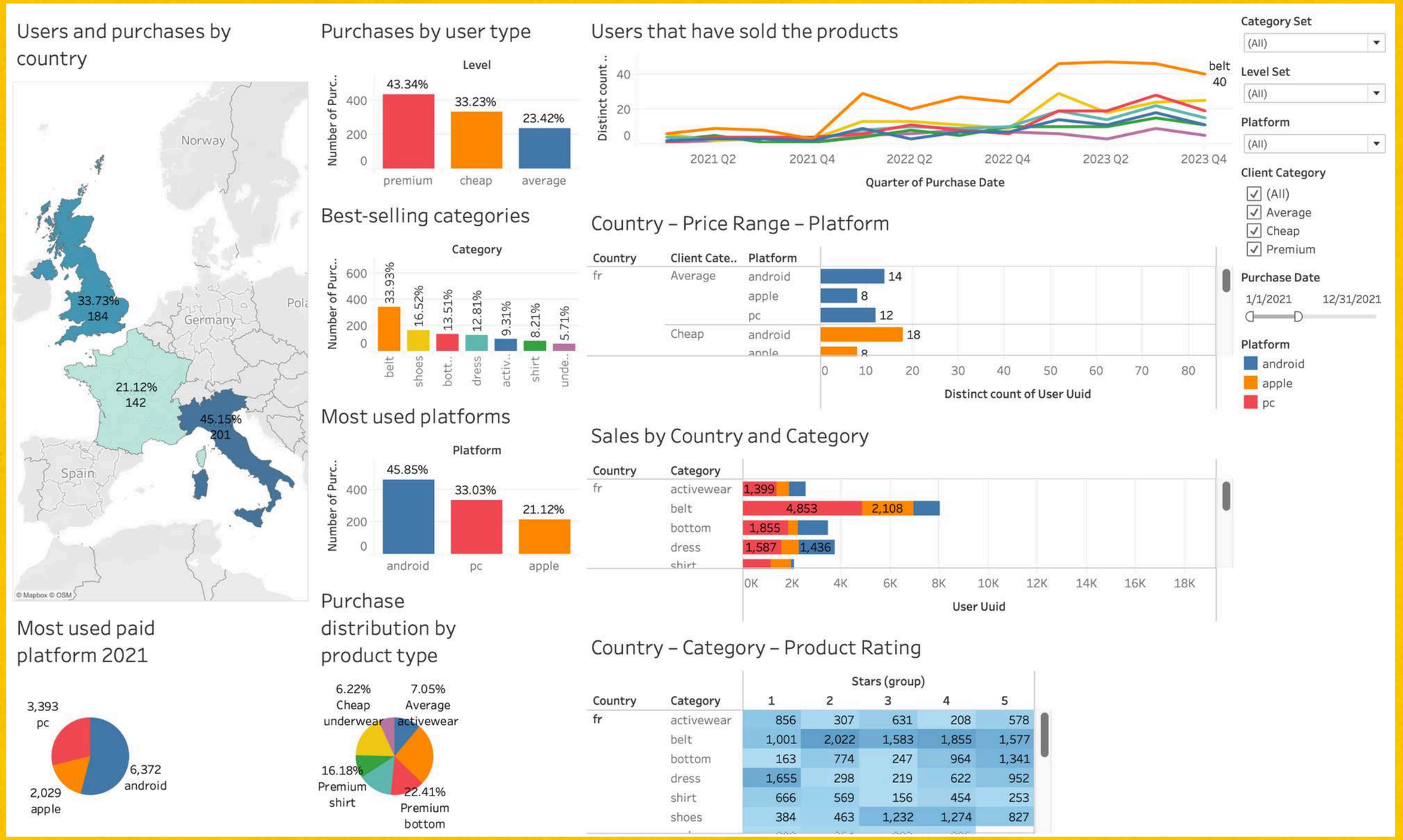


The top-selling designer for each category in 2021

```sql
-- 5. Top selling designer per category in 2021
WITH designer_sales AS (
    SELECT
        category,
        designer_id,
        COUNT(item_id) AS total_sales
    FROM fashion_data_2021
    GROUP BY category, designer_id
)
SELECT *
FROM (
    SELECT
        category,
        designer_id,
        total_sales,
        ROW_NUMBER() OVER (PARTITION BY category ORDER BY total_sales DESC) AS rn
    FROM designer_sales
) ranked
WHERE rn = 1
ORDER BY category;
```

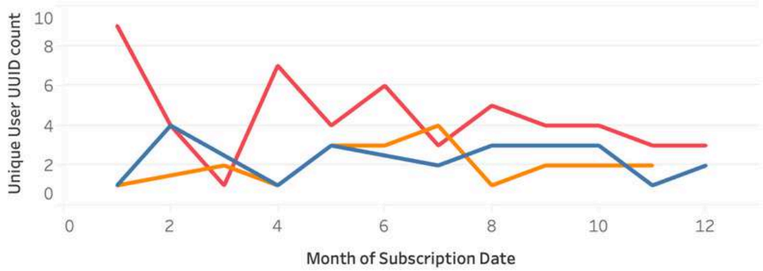


General Dashboard in TABLEAU

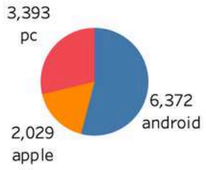


2021 Dashboard in TABLEAU

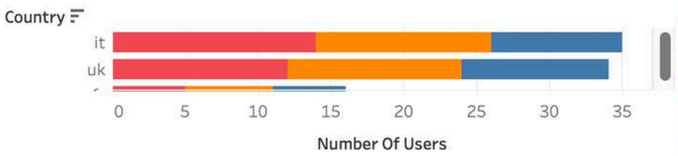
Users registered in the months of 2021



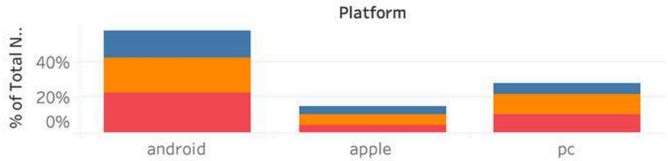
Most used paid platform 2021



Countries by Unique Users per Level – 2021



Sales by Platform & Category (%) - 2021



Sales by Level (%) and Rank - 2021

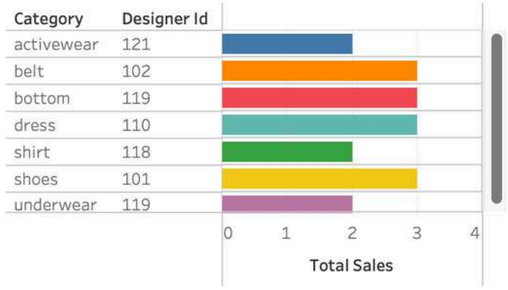
Category
- activewear
- belt
- bottom
- dress
- shirt
- shoes
- underwear

Country
- (All)
- fr
- it
- uk

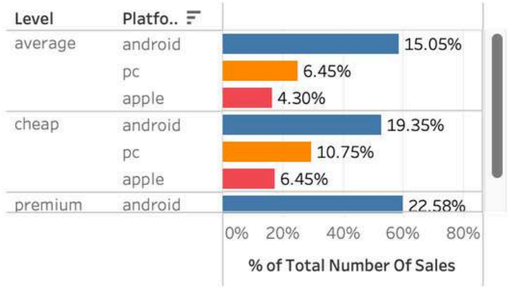
Client Category
- (All)
- Average
- Cheap
- Premium

Platform
- android
- apple
- pc

Best Designer by Category - 2021



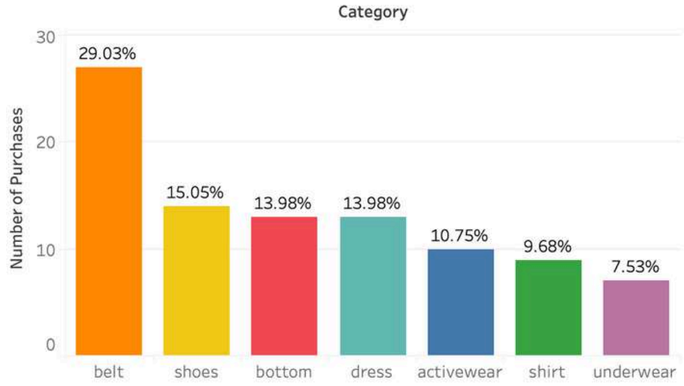
Most Used Platform by Level – 2021



Country - Category - Stars assigned to the product in 2021

Country	Client ..	Stars (group) 1	2	3	4	5
fr	Premium	697	1,140	946	1,064	1,513
	Cheap	489	797	180	429	230
	Average	335	78	262	768	740
it	Premium	2,258	3,476	2,221	2,584	2,344
	Cheap	1,193	1,435	1,834	964	1,608
	Average	897	858	1,217	1,285	1,679
uk	Premium	1,681	887	1,819	1,850	2,864
	Cheap	976	794	1,063	1,119	851
	Average	735	558	917	807	1,022

Best-selling categories 2021



Sales trend 2021

Clustering

This SQL query executes a behavioral clustering of users leveraging two primary metrics: the total number of purchases completed and the average rating received (stars). The analysis segments users into four groups, whose labels are inspired by characters from the TV series Emily in Paris, making the results more relatable.

```sql
-- SECTION 4: Clustering

-- Calculate user-level metrics: total sales and average stars (rounded to 1 decimal)
WITH user_metrics AS (
    SELECT
        user_uuid,
        COUNT(*) AS total_sales,  -- Total number of sales per user
        ROUND(AVG(stars)::numeric, 1) AS avg_stars  -- Average rating (stars) per user, rounded to 1 decimal place
    FROM fashion_data
    GROUP BY user_uuid
),

-- Calculate median values for total_sales and avg_stars across all users
stats AS (
    SELECT
        PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY total_sales) AS median_sales,  -- Median of total sales
        PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY avg_stars) AS median_stars    -- Median of average stars
    FROM user_metrics
),

-- Assign each user to a cluster based on their total_sales and avg_stars relative to the medians
clustered AS (
    SELECT
        um.user_uuid,
        um.total_sales,
        um.avg_stars,
        CASE
            WHEN um.total_sales < s.median_sales AND um.avg_stars < s.median_stars THEN 'Gabriel'   -- Low sales, low rating
            WHEN um.total_sales < s.median_sales AND um.avg_stars >= s.median_stars THEN 'Emily'    -- Low sales, high rating
            WHEN um.total_sales >= s.median_sales AND um.avg_stars < s.median_stars THEN 'Camille'  -- High sales, low rating
            WHEN um.total_sales >= s.median_sales AND um.avg_stars >= s.median_stars THEN 'Sylvie'  -- High sales, high rating
        END AS cluster_name
    FROM user_metrics um
    CROSS JOIN stats s  -- Combine metrics with median stats for comparison
)

-- Aggregate results: number of users and percentage per cluster
SELECT
    cluster_name,
    COUNT(*) AS user_count,  -- Total number of users in each cluster
    ROUND(100.0 * COUNT(*) / SUM(COUNT(*)) OVER (), 1) AS pct_users  -- Percentage of users in each cluster
FROM clustered
GROUP BY cluster_name
ORDER BY cluster_name;
```

User Clustering

Gabriel
Low sales, low ratings
Users who are not very active and whose feedback is not particularly positive.

Camille
High sales, low ratings
Very active sellers, but with a perceived quality below average.

Emily
Low sales, high ratings
They sell little, but they do it well. Users appreciate their items.

Sylvie
High sales, high ratings
They sell a lot and with taste, with excellent ratings

Key Strategic Insights

1. Seasonal Growth
   - Sales growth over three years closely follows user registrations, primarily driven by Premium and Cheap segments.
   - Strategies: Plan targeted seasonal promotions by category (e.g., Dresses in spring, Shoes in November) and strengthen onboarding campaigns with product suggestions ahead of peak periods.
2. Android Dominance & Cross-Platform Opportunities
   - Android is the leading platform across all clusters and countries; Apple usage remains marginal. Ratings are consistent across devices.
   - Strategies: Optimize UX and performance for all platforms to boost acquisition, conversion, and loyalty. Enhance desktop navigation and implement platform-native campaigns (push notifications) during peak periods.

Key Strategic Insights

3. Italy as Leading Market; France Requires Stimulation

- Italy remains the top-performing market, while France shows potential for growth.
- Strategies: Implement localized promotions in France tailored to category and price preferences. Explore pan-European campaigns for categories with converging trends, leveraging the standardization of preferences observed after 2022.

4. Polarized Market: Growth Driven by Premium and Cheap Segments

- Mid-range products underperform, while sales are concentrated in Premium and Cheap segments. Ratings remain stable across all price points.
- Strategy: Enhance the perceived value of Premium products, targeting high-rated clusters, to further drive growth.

User Clustering
Strategy

Gabriel
Low-volume sellers with low ratings
Not very relevant in terms of volume, but present in core categories.
🎯 Activation potential with training tools, or quality and sales incentives

Camille
High-volume sellers with low ratings
Clusters that are important for revenue, but can compromise the experience of other users
🎯 Activation of quality incentives, or quality control system

Emily
Low-volume sellers with high ratings
Valuable segment for the platform's reputation.
🎯 Activate loyalty programs or targeted up-selling campaigns.

Sylvie (Dominant Category)
High-volume sellers with high ratings
Ideal cluster: quality + volume, essential for revenue
🎯 Aim for retention and loyalty through priority access to promotions or premium tools.

Conclusion

In today's increasingly competitive market, simply selling is no longer sufficient; it is essential to understand who sells, how they sell, and when. This analysis highlights that behind every resold dress lies a strategic opportunity: a seller to engage, a behavior to influence, and a value to capture. Optimizing the experience for each user cluster, guiding seasonal trends, and tailoring marketing levers are critical to transforming the platform from a conventional marketplace into a sophisticated engine of value creation.

The future does not materialize on its own—it must be deliberately built, cluster by cluster.

Thank you for your attention

Google Drive

Click on this link to view the folder containing the Notebook and SQL queries

Tableau Public

Click on this link to view the dashboards