



Troisième année data science

Semestre 5

Rapport

Titre de projet : Visualisation de l'Analyse de la personnalité des clients

Titre de matière : Visualisation des données

Préparée par : Nourhane Jneid

Dr.ING Nicole Challita Al Bouty


Plan :

1-Introduction.....	3
2-Description de projet	3
3-Importer les modules et Lire DataDrame.....	4
4-Problematiques.....	4
5-Distribution de la population.....	5
6-Etude de La Relation entre quelque attribut.....	9
7- Etudes sur salaire	11
8-Division de la population par catégorie de salaire.....	13
9-Conclusion	
10-References.....	
11-Annexe	

1-Introduction :

‘Customer Personality Analysis’ est une analyse détaillée des clients idéaux d'une entreprise. L'analyse de la personnalité des clients aide les entreprises à mieux comprendre leurs clients et à améliorer leur activité en adaptant les produits, les services et les messages marketing aux différents types de personnalité. Cela peut conduire à une expérience client plus personnalisée, à une augmentation des ventes, à une réduction du taux de désabonnement et à un meilleur développement de produits.

2-Description du Projet:

En début, Je prends un ensemble de données  marketing_campaign, je l'ai pris de 'Kaggle', cet ensemble de données contient des données sur les clients dont on veut faire l'analyse, j'ai nettoyé les données en supprimant les lignes qui contiennent des informations manquantes jusqu'à ce que j'aie 2216 lignes et 29 attributs.

Définition pour chaque attribut avec le type de l'attribut :

- ID: Identifiant unique du client – **valeur numérique**
- Year_Birth: Année de naissance du client - **valeur numérique**
- Education: Niveau d'éducation du client - **valeur numérique**
- Marital_Status: Situation matrimoniale du client - **valeur numérique**
- Income: Revenu annuel du ménage du client - **valeur numérique**
- Kidhome: Nombre d'enfants dans le ménage du client - **valeur numérique**
- Teenhome: Nombre d'adolescents dans le ménage du client - **valeur numérique**
- Dt_Customer: Date d'inscription du client à l'entreprise - **valeur numérique(date)**
- Recency: Nombre de jours depuis le dernier achat du client - **valeur numérique**
- Complain: 1 si le client s'est plaint au cours des 2 dernières années, 0 sinon – **attribut Binaire**
- MntWines: dépensé en vin au cours des 2 dernières années - **valeur numérique**
- MntFruits: Montant dépensé en fruits au cours des 2 dernières années - **valeur numérique**
- MntMeatProducts: Montant dépensé en viande au cours des 2 dernières années - **valeur numérique.**
- MntFishProducts: Montant dépensé en poisson au cours des 2 dernières années - **valeur numérique.**
- MntSweetProducts: Montant dépensé en sucreries au cours des 2 dernières années - **valeur numérique.**
- MntGoldProds: Montant dépensé en or au cours des 2 dernières années - **valeur numérique.**
- NumDealsPurchases: Nombre d'achats effectués avec une remise - **valeur numérique.**
- AcceptedCmp1: 1 si le client a accepté l'offre de la 1ère campagne, 0 sinon – **attribut binaire**
- AcceptedCmp2: 1 si le client a accepté l'offre de la 2ème campagne, 0 sinon – **attribut binaire**
- AcceptedCmp3: 1 si le client a accepté l'offre de la 3ème campagne, 0 sinon – **attribut binaire**
- AcceptedCmp4: 1 si le client a accepté l'offre de la 4ème campagne, 0 sinon – **attribut binaire**
- AcceptedCmp5: 1 si le client a accepté l'offre de la 5ème campagne, 0 sinon – **attribut binaire**

- Response: 1 si le client a accepté l'offre de la dernière campagne, 0 sinon – **attribut binaire**

3-Importer les librairies nécessaires, lire le fichier :

On commence par importer les librairies nécessaires pour la visualisation, on a utilisé les

librairies suivantes chacune pour sa fonction précise.

1. **Matplotlib (matplotlib.pyplot)** : Matplotlib est une bibliothèque Python largement utilisée pour créer des visualisations statiques, animées et interactives.

2. **Seaborn (seaborn)** : Seaborn est construit sur Matplotlib et fournit une interface de haut niveau pour créer des graphiques statistiques esthétiquement plaisants et informatifs. On a utilisé Seaborn pour créer des graphiques plus avancés tels que des graphiques radar. Seaborn améliore les capacités de visualisation de Matplotlib avec ses styles attrayants et sa syntaxe simplifiée.

3. **Pandas (pandas)** : Pandas est une bibliothèque puissante pour la manipulation et l'analyse des données.

4. **NumPy (numpy)** : NumPy est une bibliothèque fondamentale pour les opérations numériques en Python.

5. **Datetime et Date (datetime, date)** : Ces bibliothèques offrent des fonctionnalités pour travailler avec des dates et des heures, elles sont utiles pour la manipulation de données temporelles dans votre analyse. On les a utilisées pour gérer des données liées aux dates, effectuer des filtrages basés sur le temps et calculer des intervalles de temps comme « l'âge » de la population.

6. **Squarify (squarify)** : Squarify est une bibliothèque pour créer des cartes arborescentes (treemaps) qui représentent des données hiérarchiques sous forme de rectangles imbriqués. On l'a utilisé pour visualiser la distribution de la population par différents niveaux d'éducation.

7. **Mplot3d et Axes3D (mpl_toolkits.mplot3d, mpl_toolkits)** : Ces bibliothèques font partie de Matplotlib et sont utilisées pour créer des graphiques et des visualisations en 3D. On les a importés pour activer la visualisation en 3D.

8. **Plotly Express (plotly.express)** : Plotly est une bibliothèque puissante pour créer des visualisations interactives et basées sur le web. On a importé Plotly Express, qui fournit une interface de haut niveau pour créer une variété de graphiques interactifs comme les barcharts dans les trois parties de la population, afin d'améliorer l'interactivité des représentations visuelles.

Après avoir importé les bibliothèques nécessaires, nous allons lire les données à partir d'un fichier CSV en utilisant la fonction 'read_csv' de la bibliothèque pandas. Ensuite, nous souhaitons afficher les cinq premières lignes de données.

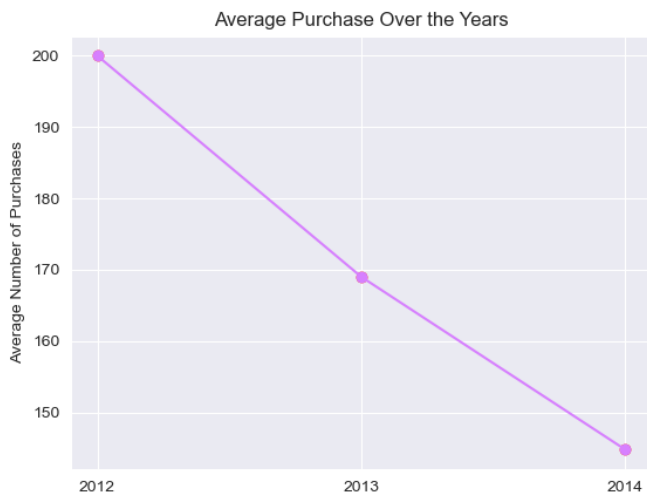
4-Problématique :

Pour effectuer une analyse de la personnalité du client pour ces données, nous avons d'abord souhaité évaluer la performance de l'entreprise qui fournit ces données. Cette démarche nous permet de détecter d'éventuels problèmes avant de procéder à la visualisation basée sur ces données.

Nous souhaitons d'abord observer comment le montant total des achats a varié au cours des trois années de collecte de données, de 2012 à 2014.

Pour cela, on a fait un **Linechart** pour observer comment le montant total d'achat a varié au cours de trois années.

Ce « Line Chart », On peut constater que le montant total des achats a diminué au cours des 3 années, en fonction de la date d'inscription ce qui est logique. Plusieurs hypothèses sont à envisager. La première est que les opérations de l'entreprise ne ciblent pas le bon public. La deuxième hypothèse est que des raisons externes influencent le comportement de nos clients qui ne sont pas en relations l'opérations de l'entreprise.



On a utilisé la fonction **plt.plot()** pour faire le graphique après qu'on a calculé la moyenne du montant total de l'achat pour chaque année en utilisant la méthode suivante :

```
df.groupby('Year')['Total_Purchases'].mean()/2,
```

pour calculer la moyenne en groupant par année, on a divisé la valeur par deux parce que le montant total est durant deux années d'achat.

Figure 1

Ce graphique ne suffit pas pour évaluer la performance de l'entreprise. Nous allons utiliser plusieurs types de visualisations et différentes combinaisons d'attributs pour mieux comprendre la performance de l'entreprise et la réaction des clients aux différentes campagnes et canaux.

5-Distribution de la population :

A-Histogramme pour la distribution de l'âge de la population :

Pour mettre en évidence la barre représentant la valeur maximale dans l'histogramme. Tout d'abord, on récupère l'objet "patches" de l'histogramme à l'aide de la fonction `plt.gca().patches`, qui contient des informations sur chaque barre de l'histogramme. Ensuite, on utilise une boucle `for` pour parcourir les barres et déterminer la valeur maximale à l'aide de la fonction `max([p.get_height() for p in patches])`, qui extrait la hauteur de chaque barre et identifie la valeur maximale. Enfin, on parcourt à nouveau les barres à l'aide d'une boucle `for` et, grâce à la fonction `p.set_facecolor('#f7e6ff')`, on change la couleur de la barre correspondant à la valeur maximale en la colorant en ('#f7e6ff').

Tous nos clients sont plus grands que 18 ans. L'âge la plus fréquente dans la population est entre 40 et 50, en utilisant. Les données sont distribuées suivant la loi normal

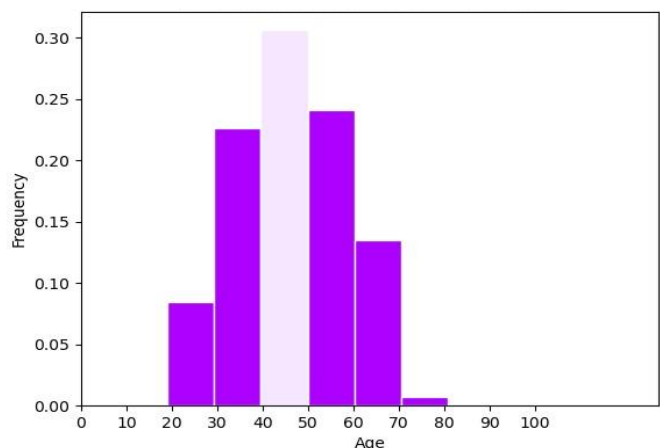


Figure 2

B-Pie Chart du 'Marital Status' : état civil :

On a utilisé 'plt.pie()', on a mis les valeurs moins que 5 % ensembles dans une nouvelle catégorie autres.

On peut conclure que la majorité de notre population sont des couples mariés avec un pourcentage de 38.7%.

On veut voir après comment l'état civil affecte le Comportement des clients.

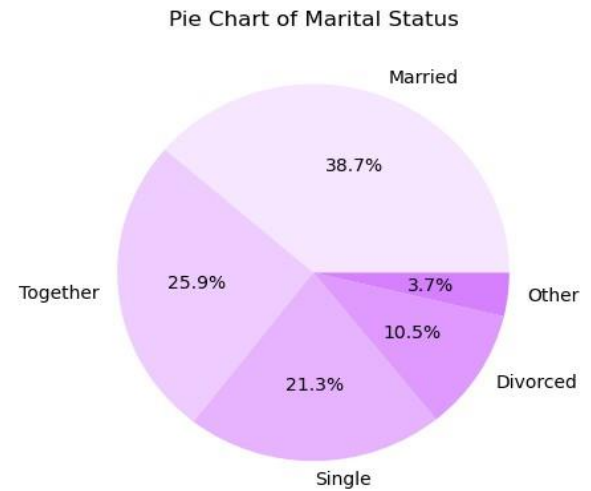
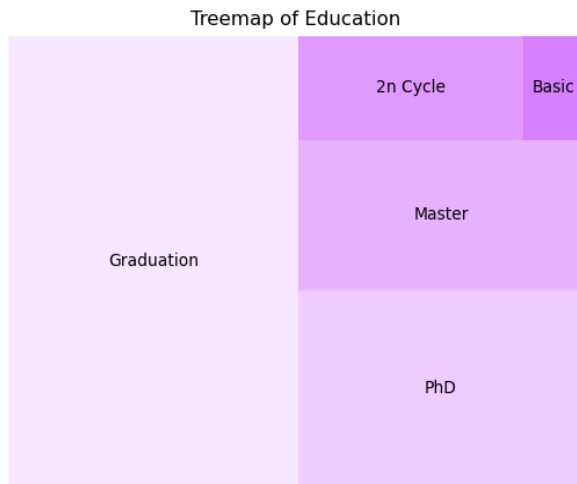


Figure 3

C-Distribution des Individus selon leur niveau d'éducation :



J'ai utilisé la librairie 'Squarify' et la fonction **squarify.plot** pour faire cette 'Treemap' et puis visualiser la distribution de niveau d'éducation sur la population

Le niveau d'éducation le plus commun entre les individus est la catégorie : 'Graduate', on a à peu près la même distribution pour les master et PHD doctorat, on peut conclure que la majorité de notre population sont avec spécialités

Figure 4

D-Histogramme de distribution de la population selon le salaire des individus :

Cette visualisation utilise un histogramme pour représenter la distribution des revenus de la population. Tout d'abord, la fonction 'plt.figure()' crée une nouvelle figure avec une taille personnalisée de 10 pouces par 6 pouces et une résolution de 100 points par pouce.

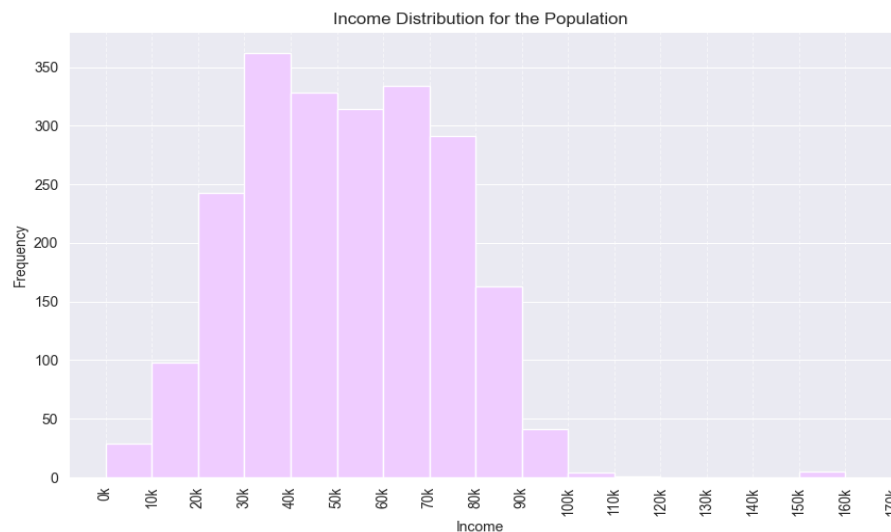


Figure 5

Ensuite, les données sont filtrées avec la fonction `df = df[(df['Income'] <= 170000) & (df['Income'] >= 1000)]`, éliminant ainsi les valeurs. L'histogramme est généré avec `plt.hist()`, utilisant des tranches de 10 000 dollars. Pour une meilleure lisibilité, grâce à `tick_spacing` et `plt.xticks()` importe de `matplotlib.ticker as ticker`. De plus, la fonction de formatage `plt.gca().xaxis.set_major_formatter()` est utilisée pour afficher les valeurs de l'axe x en unités de milliers (k). En conclusion, cette visualisation met en évidence la concentration des revenus entre 30 000 et 40 000 dollars, présentée à travers des barres d'histogramme et bénéficiant d'ajustements visuels pour une compréhension optimale.

F-Carte therique de la moyenne de dépense par produit en fonction de la composition du ménage :

Cette visualisation utilise la bibliothèque **seaborn** pour créer une grille de sous-graphiques représentant la moyenne des dépenses par produit en fonction de la composition du ménage. La fonction principale ici est `sns.heatmap()`, qui génère des cartes thermiques avec des annotations pour chaque produit. La boucle `for` itère à travers les différents produits ('MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts', 'MntGoldProds') pour créer des cartes thermiques distinctes. La disposition en grille est gérée par `plt.subplots()`. L'ensemble du processus vise à offrir une représentation visuelle claire des tendances de dépenses en fonction de la composition du ménage et du produit spécifique.

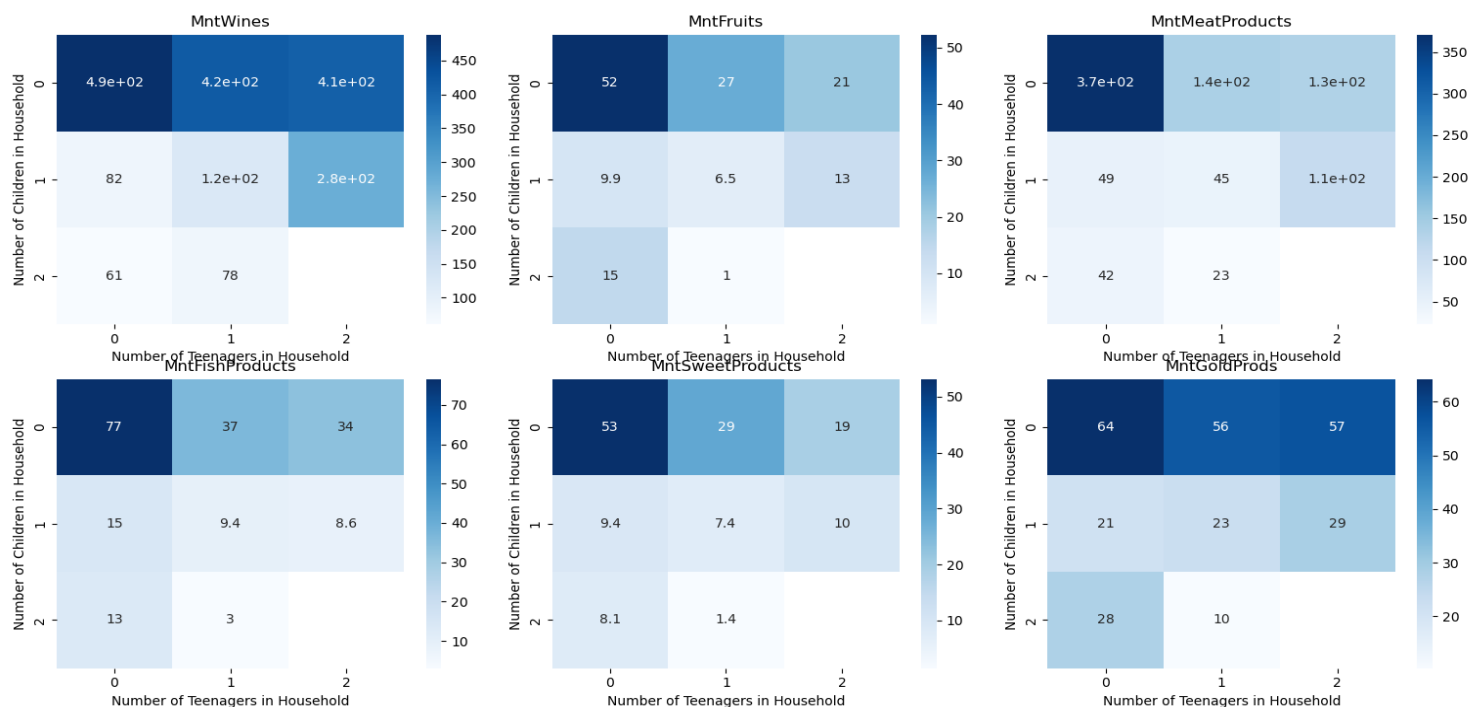


Figure 6

On peut remarquer que les familles sans enfants ni adolescents ont une moyenne de dépenses plus élevée que celles avec deux enfants et deux adolescents nécessite une interprétation prudente. La heatmap ne prend pas en compte le nombre total de ménages dans chaque catégorie, ce qui pourrait influencer les moyennes. Des variations dans le nombre de ménages et la présence de dépenses exceptionnelles peuvent contribuer à cette observation.

E-Doughnut Chart' Pourcentage D'acceptation de chaque campagne :

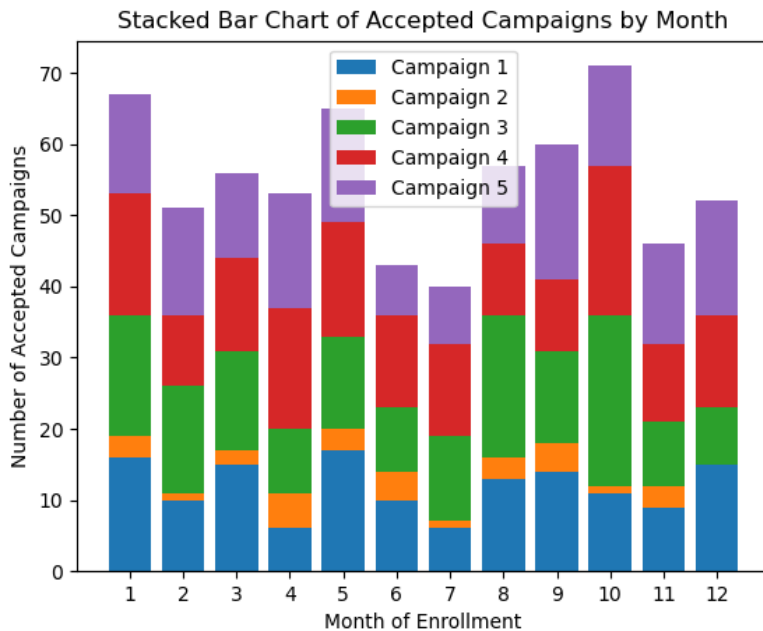


Figure 7

Pour visualiser ces données, un graphique à barres empilées a été créé avec la fonction `plt.bar()`. Chaque campagne acceptée est représentée par une barre empilée sur les mois d'inscription. Les différentes sections empilées ont été gérées en utilisant l'argument `bottom` de la fonction `plt.bar()` pour spécifier la hauteur de la section précédente. **Cependant, il est important de noter que la visualisation de ce type de données sous forme de graphique empilé peut être complexe et difficile à interpréter, en particulier avec plusieurs campagnes. Pour simplifier l'analyse, une alternative plus claire, comme un graphique en anneau (donut chart), sera créée pour représenter le taux d'acceptation global de toutes les campagnes.**

G-Donut Chart pour pourcentage d'acceptation des campagnes :

La conception du graphique en anneau, réalisée avec l'aide des bibliothèques Matplotlib et NumPy, visait à simplifier l'évaluation du taux d'acceptation des campagnes. Tout d'abord, une liste des campagnes, allant de AcceptedCmp1 à AcceptedCmp5, a été établie. Ensuite, une liste en

Pour créer ce graphique à barres empilées tout d'abord, la colonne 'Dt_Customer' a été convertie au format datetime à l'aide de la fonction `pd.to_datetime()` pour extraire le mois d'inscription. Ensuite, les données ont été regroupées par mois en utilisant la fonction `groupby()` et la somme des campagnes acceptées (AcceptedCmp1 à Response) a été calculée à l'aide de la fonction `sum()`.

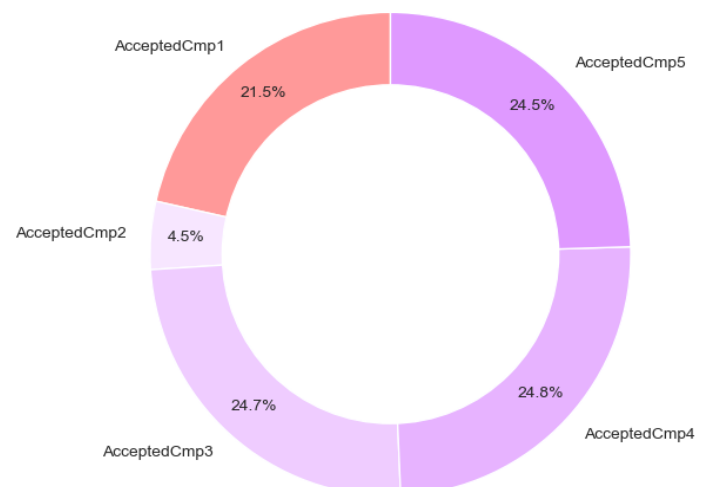


Figure 8

compréhension de la bibliothèque NumPy a été utilisée pour calculer la somme des participants acceptés pour chaque campagne. Une palette de couleurs de la bibliothèque Matplotlib a été définie pour différencier visuellement chaque campagne dans le graphique. La fonction `ax.pie()` de Matplotlib a été employée pour créer le graphique en anneau, avec les sommes des campagnes comme données, des étiquettes correspondantes, et la palette de couleurs définie. L'ajout d'un cercle blanc au centre a été réalisé avec la fonction `plt.Circle()` de Matplotlib. En ajustant l'aspect de l'axe pour garantir un dessin circulaire, le graphique offre une vue claire du taux d'acceptation global pour chaque campagne publicitaire. On note que les campagnes sont distribuées à peu près également sauf la deuxième qui a un pourcentage très faible en comparaison avec les autres qui sont entre 21 et 25 %. Ce qui pose un problème dans la deuxième campagne offerte par l'entreprise.

6-Etude de La Relation entre quelque attribut :

A-Nuage de points 3D pour la relation entre l'âge des clients, le total des achats et la quantité spécifique de produits achetés (Fruits, Viandes, Poissons) :

Ici on a créé un graphique en nuage de points 3D avec les bibliothèques Matplotlib et Pandas, pour analyser la relation entre l'âge des clients, le total des achats et la quantité spécifique de produits achetés (Fruits, Viandes, Poissons). Le code utilise les fonctions suivantes : calcul du Total des Achats (`df['Total_Purchases']`), initialisation de la Figure 3D (`fig = plt.figure(figsize=(10, 8))`), création du Sous-Graphique 3D (`ax = fig.add_subplot(111, projection='3d')`), nuage de Points (`ax.scatter(...)`), ajustement de la Perspective (`ax.view_init(elev=30, azim=45)`) et affichage du Graphique (`plt.show()`). Le code ajoute également une nouvelle colonne au DataFrame pour représenter le total des achats, calculé en

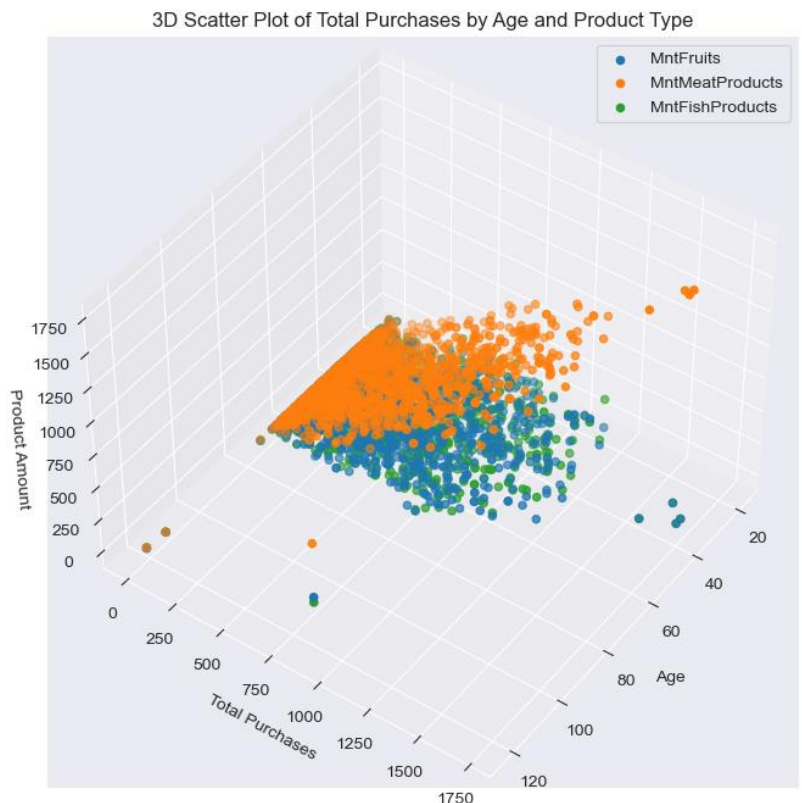


Figure 9

combinant les montants dépensés pour les fruits, les produits de viande et de poisson. Le graphique résultant présente toutefois un inconvénient : la superposition des trois types de produits dans le même graphique peut rendre l'interprétation difficile.

B- Boxplot représentant montant total d'achat pour chaque catégorie d'état civil :

L'élaboration de ce graphique en boîte (box plot) s'est déroulée en utilisant les bibliothèques Seaborn et Matplotlib, mettant en lumière la répartition du montant total des achats en fonction du statut matrimonial des clients. Les étapes comprenaient le calcul du total des achats en agrégeant les montants dépensés pour différents produits : `df['Total_Purchases'] = df['MntWines'] + df['MntFruits'] + df['MntMeatProducts'] + df['MntFishProducts'] + df['MntSweetProducts'] + df['MntGoldProds']`, la création d'un graphique en boîte avec Seaborn à l'aide de `sns.boxplot()`, l'ajout d'une grille grise en pointillés avec `plt.grid()`.

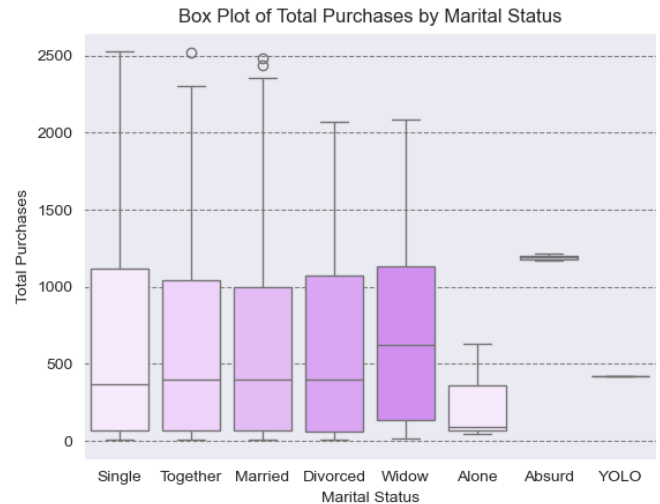


Figure 10

L'alignement des moyennes à travers les catégories dans le diagramme en boîte suggère qu'en moyenne, le montant total des achats ne varie pas significativement entre les catégories de clients célibataires, mariés, vivant ensemble et divorcés. Cependant, les différences observées dans les valeurs maximales et minimales, ainsi que le léger désalignement dans les quartiles supérieurs (Q3), peuvent indiquer une certaine variabilité et des valeurs aberrantes potentielles dans les données.

C- Etude de la corrélation entre les caractéristiques des clients et le montant total de l'achat :

On a importé la librairie « Seaborn » pour faire la carte thermique, on utilisant la fonction `Sns.heatmap()`, et on a défini une palette de couleur « purple » spécifiquement pour notre visualisation. On peut voir que le salaire des individus a le plus grand effet sur le montant total de l'achat.

Pour cela on veut maintenant faire différentes visualisations pour voir comment le salaire affecte l'achat et comment le salaire est distribué selon différentes caractéristique des individus.

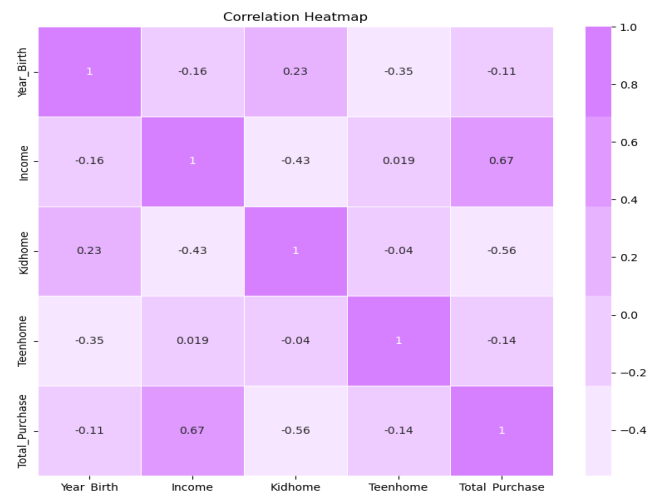


Figure 11

D- Area charts pour les trois locaux d'achats par rapport au montant total d'achat :

Ce graphique en aires est créé en utilisant la bibliothèque Matplotlib. On a utilise la colonne d'achat total déjà calcule, puis on a converti la date d'achat en objets datetime avec la fonction `pd.datetime()`, on a extraire les

année d'achat, puis calcules la moyenne d'achat pour chaque année d'inscription, et calculer la moyenne d'achat pour chaque local d'achat, puis création de l'aire pour les achats moyens pour chaque canal et le total des achats en utilisant la fonction plt.fill_between()

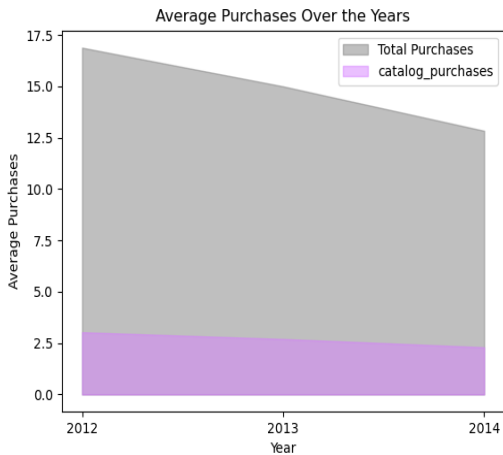


Figure 14

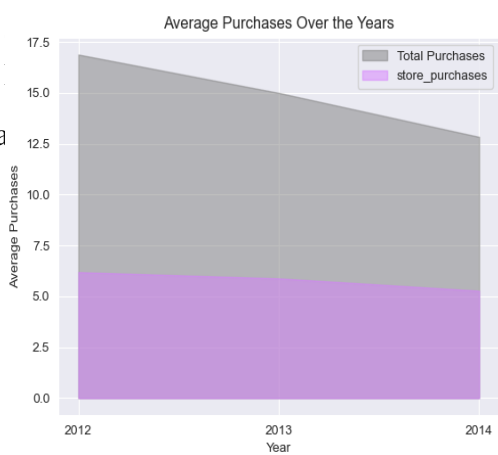


Figure 13

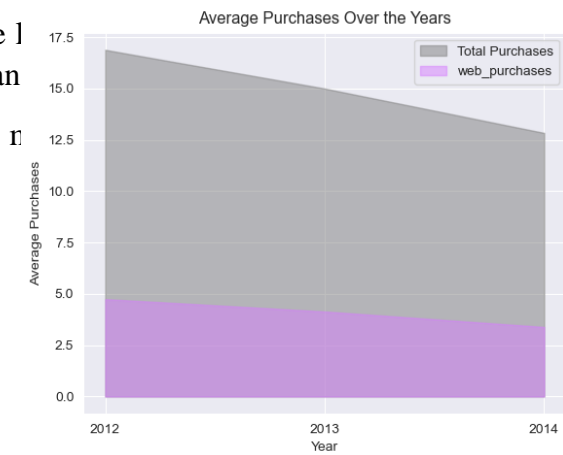


Figure 12

La tendance observée, avec une consommation plus élevée en magasin par rapport aux achats par catalogue et en ligne, peut être attribuée à divers facteurs. Les préférences démographiques des clients et leurs choix personnels jouent un rôle essentiel. Certains clients peuvent préférer l'expérience en magasin pour des produits spécifiques, tandis que d'autres peuvent être attirés par des offres exclusives en magasin.

De plus, la nature des produits offerts à travers chaque canal et les stratégies promotionnelles de l'entreprise peuvent influencer ces habitudes d'achat. Des promotions spéciales en magasin ou des offres exclusives peuvent inciter les clients à privilégier les achats en magasin.

En conclusion, comprendre les préférences des clients, les caractéristiques des produits et les stratégies promotionnelles de l'entreprise est crucial. Cela permet d'optimiser chaque canal de vente pour répondre aux attentes des clients et améliorer leur expérience d'achat.

En ce qui concerne les années on ne peut pas prendre ça en considération parce que cette date est la date d'enroulement des clients donc c'est bien logique qu'un client qui est depuis 2012, a un nombre plus grand d'achat que ceux qui sont seulement inscrites depuis une année.

7- Etude sur salaire :

A-Bar Chart de revenu moyen en fonction du niveau d'éducation :

La différence de revenu moyen en fonction du niveau d'éducation est évidente dans ce graphique à barres. En analysant les données démographiques, on constate que les individus ayant un diplôme universitaire et un doctorat ont les revenus moyens les plus élevés dans la population étudiée. Cette tendance suggère une corrélation entre le niveau d'éducation et le revenu, avec des implications potentielles pour la stratégie de segmentation du marché et les initiatives de marketing ciblées.

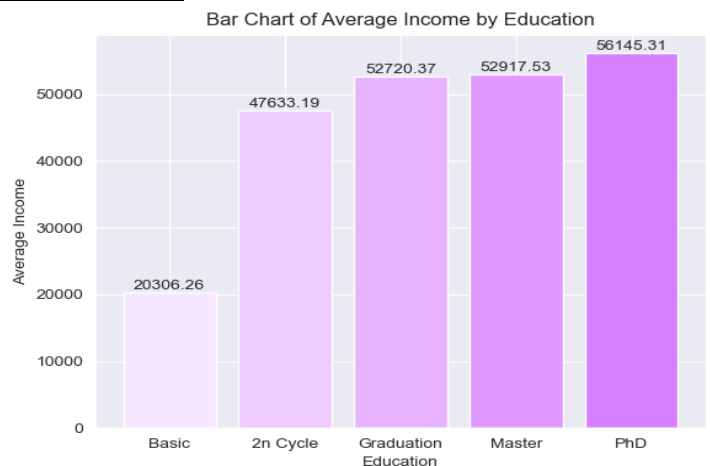


Figure 15

B-Bar Chart pour la distribution du salaire par niveau d'éducation :

Cette analyse approfondie des catégories de statut marital révèle des nuances significatives dans le revenu moyen. La catégorie "Veuve" se distingue particulièrement avec un revenu moyen supérieur par rapport aux autres catégories, même après l'exclusion de la catégorie "Absurde". Cette observation souligne l'importance de considérer chaque catégorie de manière distincte pour une compréhension plus approfondie des comportements financiers et des préférences de consommation. Ces informations peuvent orienter les stratégies marketing en mettant l'accent sur des approches spécifiques pour répondre aux besoins uniques de chaque catégorie de statut marital.

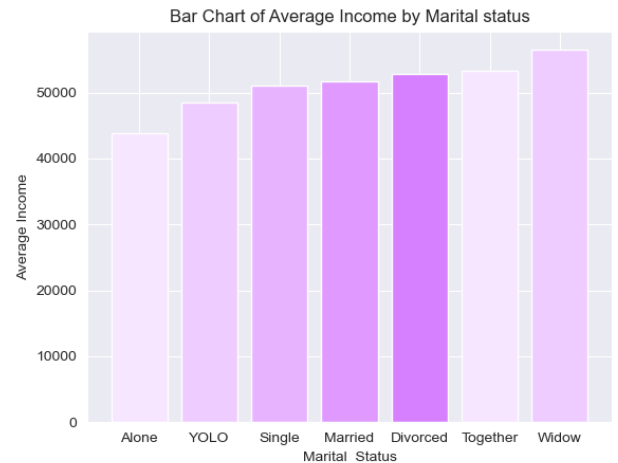


Figure 16

C-Bar Chart pour la distribution du salaire par age :

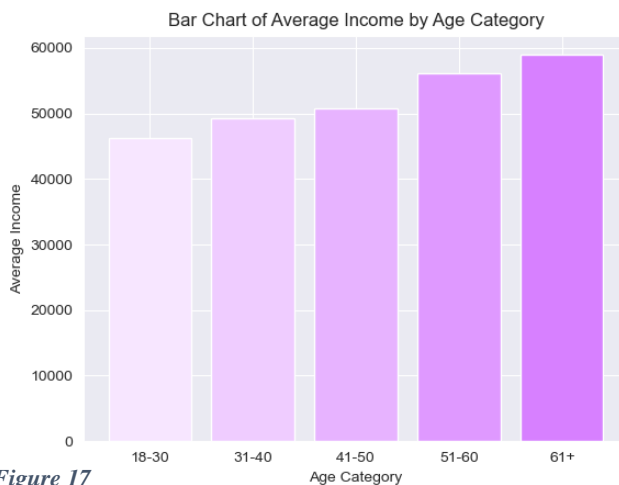


Figure 17

Cette analyse explore la relation entre l'âge des clients et leur revenu moyen. En divisant les clients en catégories d'âge, nous observons des tendances distinctes. Par exemple, la catégorie "61+" affiche un revenu moyen plus élevé, indiquant que les personnes plus âgées ont tendance à avoir des revenus supérieurs. Cependant, il est essentiel de noter que cette analyse offre une vue d'ensemble, et des variations individuelles peuvent exister au sein de chaque catégorie d'âge.

D-Scatter plot pour visualiser la relation entre salaire et achat total :

Tout d'abord on a crée un nuage de points en utilisant la bibliothèque matplotlib. On a trace la ligne de régression en utilisant `sns.regplot()` du Seaborn, on peut conclure la relation forte entre le salaire et le montant d'achat total finalement on veut prendre le salaire comme paramètres pour segmenter la population.



Figure 18

Nous pouvons clairement observer une relation directe entre les niveaux de revenu et la quantité de produits achetés. Les clients ayant des revenus plus élevés sont également plus enclins à réagir positivement aux campagnes marketing. De plus, les clients qui ont accepté un plus grand nombre de campagnes ont davantage de chances de réaliser des achats de produits plus importants.

Il serait judicieux de concentrer davantage les efforts marketing sur les clients à revenu élevé, car ils sont à la fois plus réceptifs aux campagnes et ont tendance à dépenser davantage. De plus, l'entreprise pourrait envisager d'élaborer des stratégies visant à encourager l'acceptation des campagnes auprès d'une clientèle plus large pour potentiellement stimuler les ventes globales.

8-Division de la population par categorie de salaire :

A-Lollipop de la distribution du categories du salaires :

On a divise la population en catégories de revenus (Very Low, Low, Medium, High) en utilisant la moyenne et l'écart-type comme références. Cela offre une vue segmentée de la population en fonction de son niveau de revenu, ce qui peut être utile pour une analyse démographique plus approfondie. Une nouvelle colonne « Income_Category » a été créée en faisant cette division. Puis on a fait un Lollipop Chart pour visualiser les 4 différentes categories de salaires.

Le code pour créer le graphique en forme de lollipop, associé à la constatation que la catégorie "Low_Income" présente la fréquence la plus élevée, peut être expliqué comme suit :

Pour la création du graphique lollipop on a utilisé la fonction `plt.stem()` de la bibliothèque Matplotlib, puis on a calculé la fréquence des catégories de revenus en utilisant la fonction `value_counts()` sur la colonne "Income_Category" pour obtenir le décompte de chaque catégorie. On a trié les données par ordre de fréquence décroissante en utilisant la fonction `sort_values()`.

La majorité de la population appartenant à la catégorie "Low_Income" offre à l'entreprise plusieurs opportunités de se démarquer sur le marché et de créer un impact social positif. En adaptant ses stratégies marketing, son développement de produits, son expansion du marché, ses initiatives de responsabilité sociale, sa planification financière et son engagement client aux besoins et aux réalités de ce segment démographique, l'entreprise peut attirer et fidéliser les individus à faible revenu, tout en s'adaptant aux tendances économiques plus larges. Ainsi, l'entreprise peut tirer parti de ces informations pour optimiser ses performances commerciales et sa responsabilité sociale.

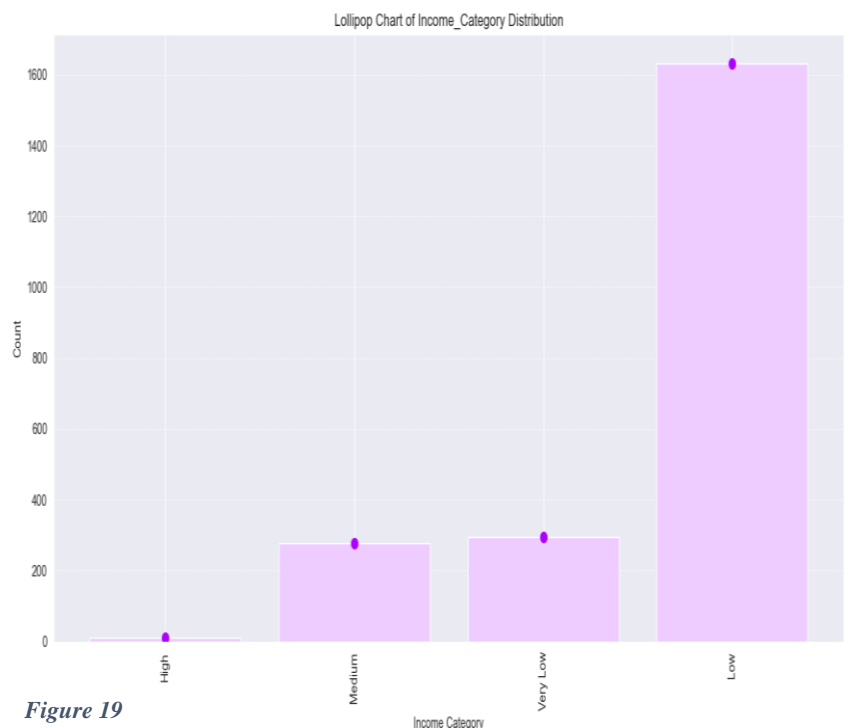


Figure 19

B-Distribution des 4 catégories par âge :



Figure 21

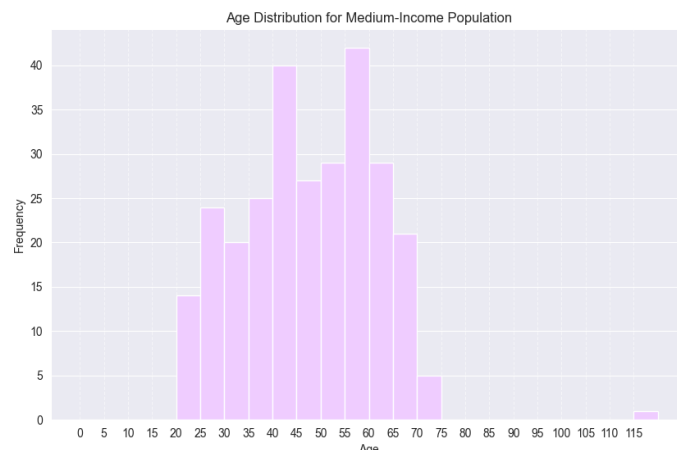


Figure 20



Figure 23



Figure 22

Des Bar charts pour la distribution des 4 population selon l'âge des individus, qui montre que la population avec « High Income » a l'âge de fréquence plus élevée entre 40-45

Population avec « Medium Income » a l'âge de fréquence plus élevée entre 60-65, la population de « low Income » a l'âge de fréquence plus élevée entre 40-45, la population avec « Very_Low Income » a l'âge de fréquence plus élevée entre 35-40.

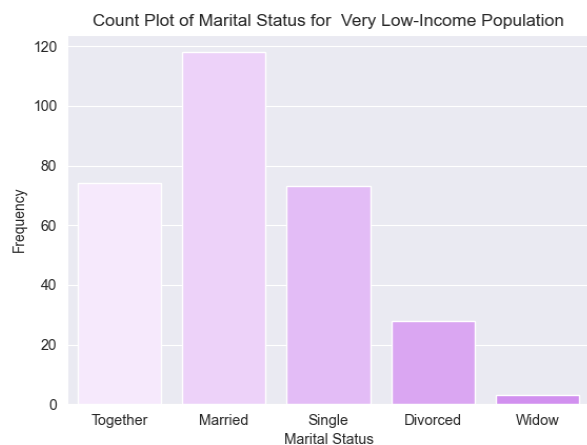


Figure 25

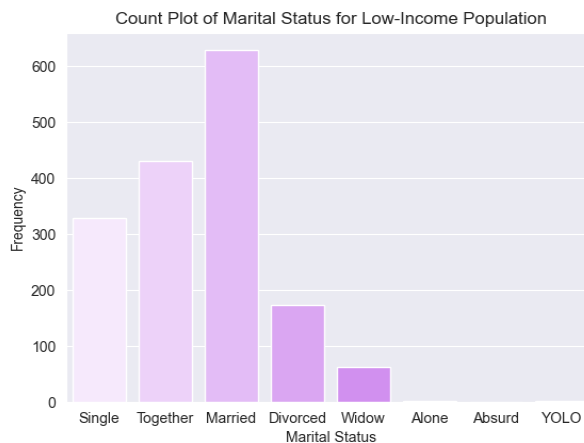


Figure 24

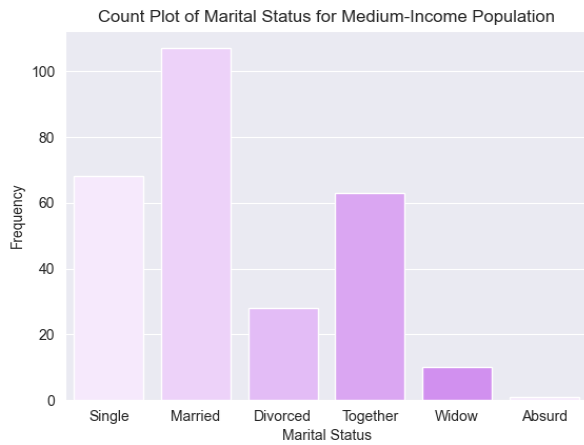


Figure 27

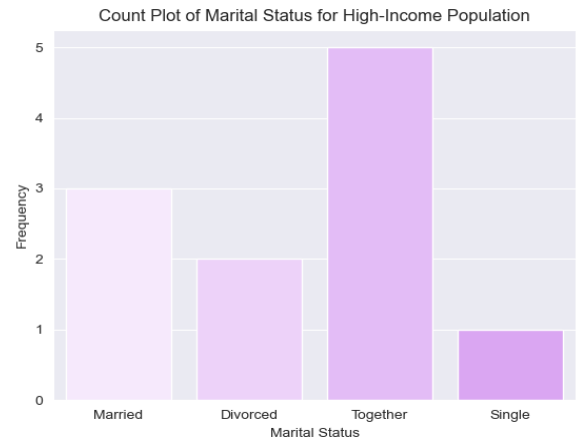


Figure 26

La fonction `countplot` de Seaborn est utilisée pour créer un graphique qui compte le nombre d'occurrences de chaque catégorie du statut matrimonial pour la population à revenu élevé.

Étiquettes et titres : Des étiquettes pour les axes x et y sont ajoutées, ainsi qu'un titre au graphique.

Affichage du graphique : La fonction `show()` est utilisée pour afficher le graphique.

Ce countplot permet de comparer la distribution du statut matrimonial au sein de la population à revenu élevé, fournissant une représentation visuelle du nombre de personnes dans chaque catégorie. Il est important de noter que, contrairement à un histogramme, un countplot utilise des barres discrètes plutôt que des intervalles continus, car il s'agit d'une représentation du décompte des occurrences de chaque catégorie.

La difference entre bar chart et

D-Distribution selon le niveau d'éducation :

Distribution of Education Level for High Income Population

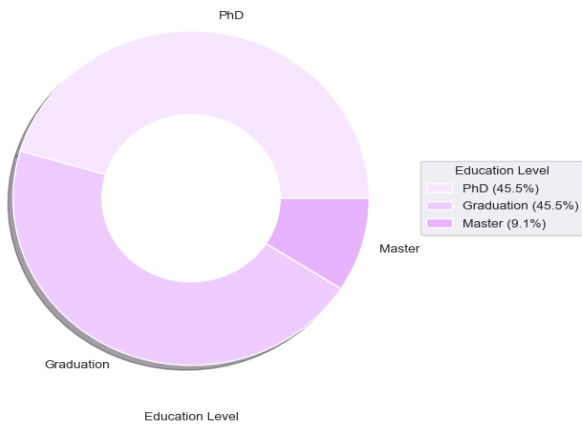


Figure 28

Distribution of Education Level for Medium Income Population

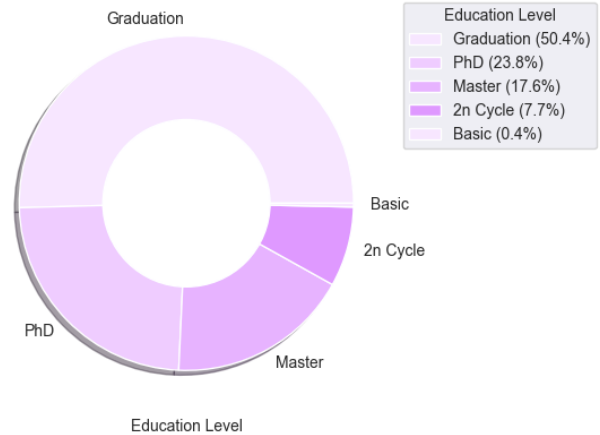


Figure 29

Distribution of Education Level for Very Low Income Population

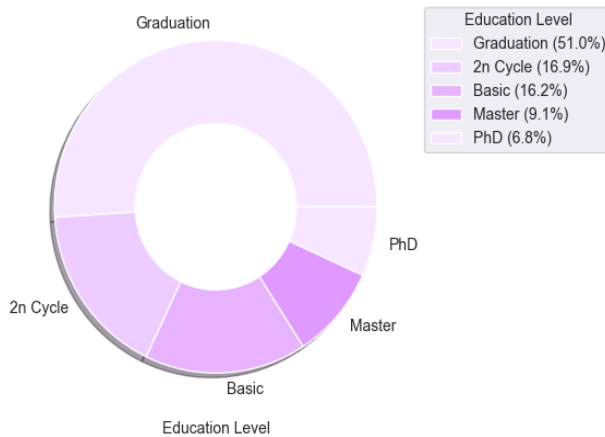


Figure 30

Distribution of Education Level for low Income Population

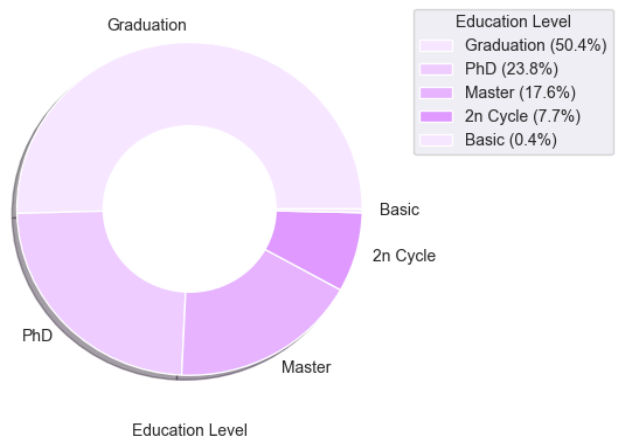


Figure 31

Les données sont filtrées pour inclure uniquement la catégorie de revenus correspondante puis on a fait comptage des occurrences : Les occurrences de chaque niveau d'éducation dans les données filtrées sont comptées. Pour faire le donut chart on fait un pie chart avec `plt.pie()` puis on ajoute le cercle blanc au centre à l'aide de la fonction `plt.Circle()`, et il est ensuite ajouté à l'axe du graphique avec `ax.add_artist(cercle)`

E-Analyse pour High salaire groupe :

E1-Etude pour les types des produits :

On a utilisé les bibliothèques **pandas** et **matplotlib** pour créer un graphique radar de la distribution des produits pour la population à revenu élevé.

Pour créer un graphique polaire, qui est un graphique avec un système de coordonnées radial on a utilisé la fonction **plt.polar()** puis pour remplir des régions avec une couleur on utilise la fonction **plt.fill()** , puis pour définir les étiquettes et les positions des lignes de grille angulaires on a utilisé **plt.thetagrids()**.

En conclusion, l'analyse de la distribution moyenne de la consommation de différents produits révèle des tendances significatives. Le graphique radar démontre clairement que ce groupe démographique à revenu élevé a une consommation particulièrement élevée de viande (Meat),

étant le produit le plus consommé au sein de cette catégorie spécifique. Cette observation souligne l'importance de cibler stratégiquement les campagnes marketing et les promotions liées aux viandes pour ce segment de clientèle, ce qui pourrait potentiellement stimuler les ventes et renforcer la relation avec cette clientèle à fort pouvoir d'achat.

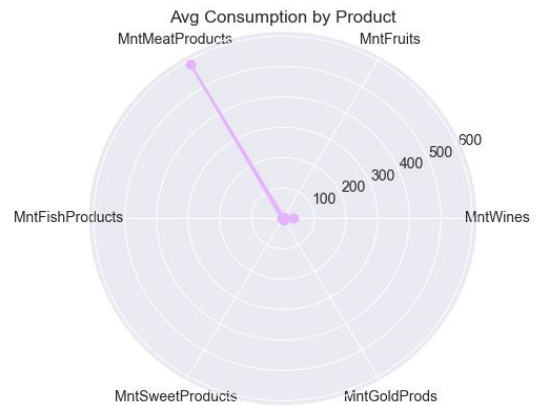
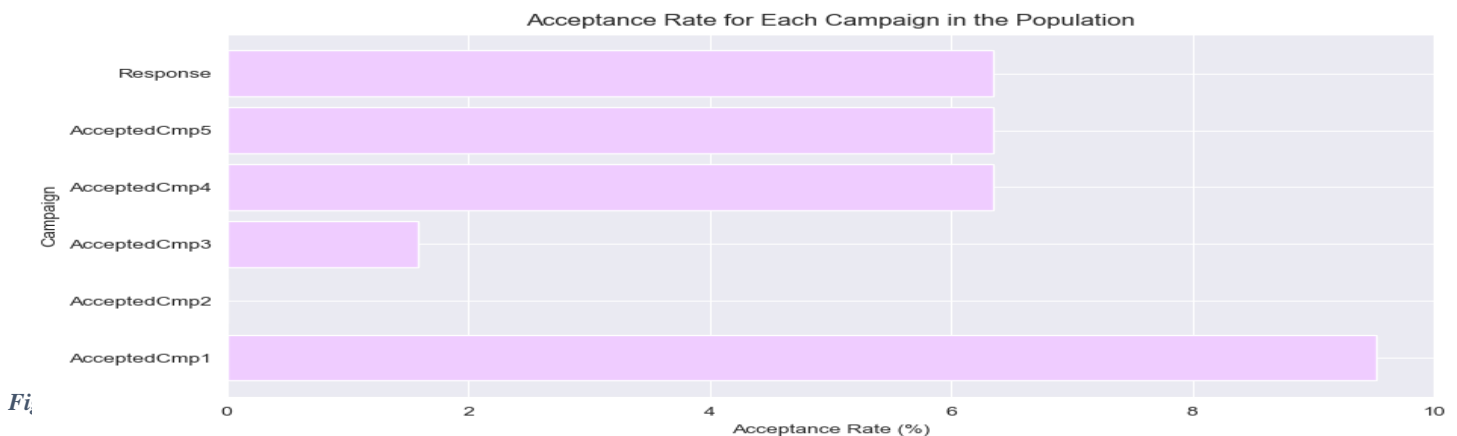


Figure 32

E2-Bar Chart horizontal pour la percentage d'acceptation des campagnes :

Dans ce bar chart on peut voir que toutes les campagnes ont un pourcentage d'acceptation très faible ce qui impose sur l'entreprise une nouvelle méthode pour adapter ses campagnes et avoir et que leurs clients cibles interagissent avec ces campagnes.



Fi

E3-Etude sur les canaux d'achats :

Cette population, prefere acheter par catalogue, on a fait un oie-chart avec la fonction **plt.pie** , on a ajouter la caracteristique explode, et shadow=True pur une visualisation mieux claire et un élaboration sur le local le plus élevée comme nombre de visiteurs,

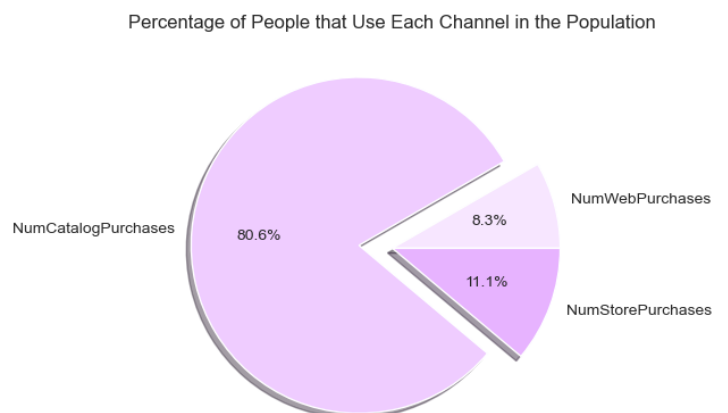


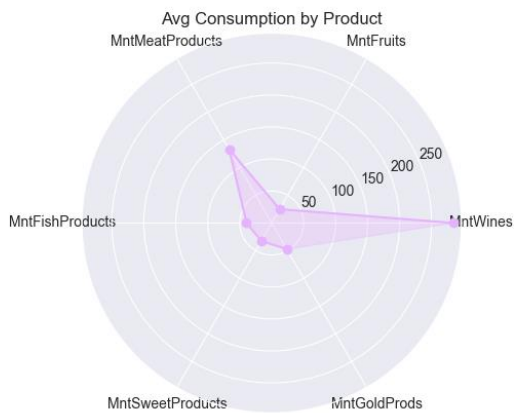
Figure 34

F-Comparaison entre les population :

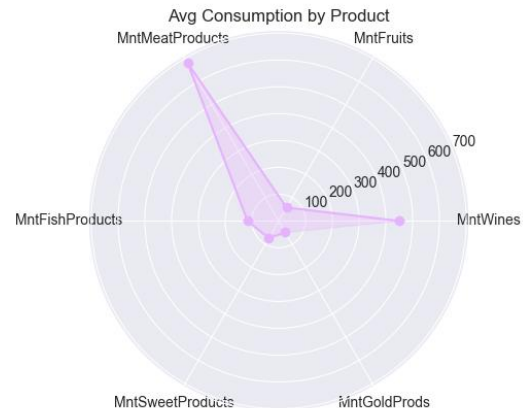
On a fait les mêmes études pour les 3 groupes restants en utilisant les mêmes graphiques et même variabilité, on peut voir des différences de comportement des clients pour chaque groupe qu'on a déjà segmenté.

F1- Radar Charts pour Medium-Low-VeryLow groupes :

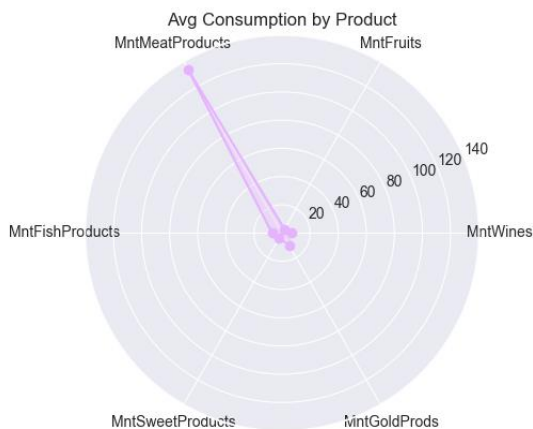
Low-Income radar chart:



Medium-Income radar Chart :



Very-Low Income radar Chart:



Population de salaire élevé a une consommation grande en Meat(Viande).

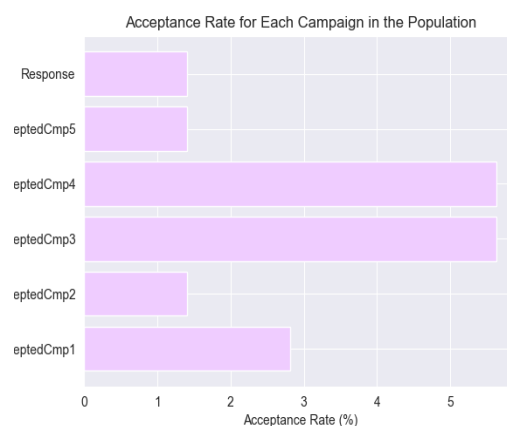
Population de salaire moyenne consommation dans viandes et vins et un peu des autres produits.

Population de faible revenu consommation grande en wines vins mais moins que celle de revenu moyen.

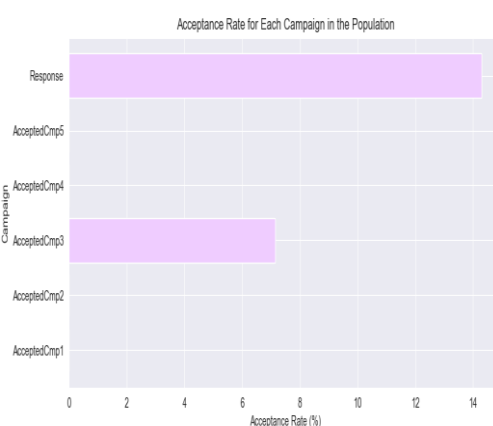
Population de très faible revenu, consommation de viandes mais très petite en comparaison avec la population de salaire élevé.

F2-Horizantal Bar chart des trois groupes

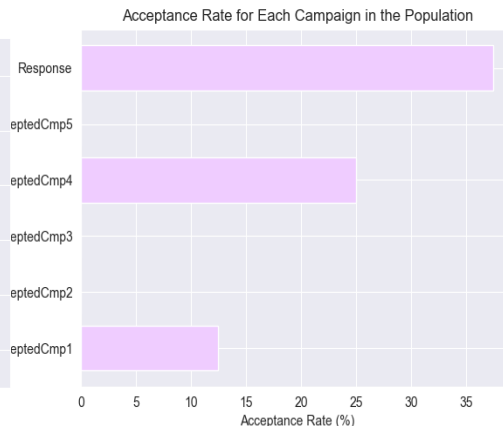
Low Income :



very-Low income :



Medium Income:



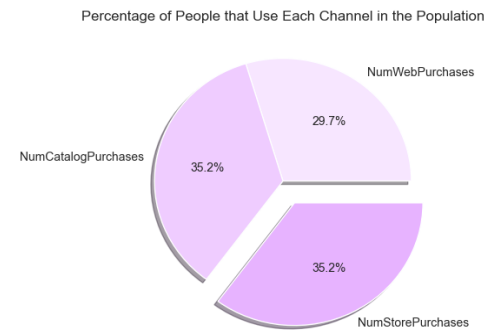
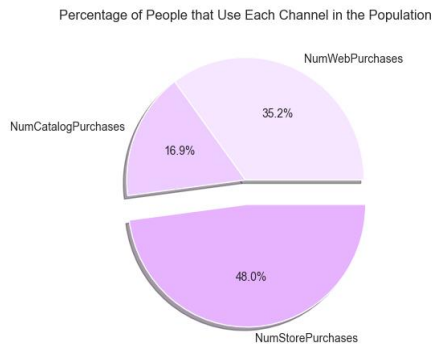
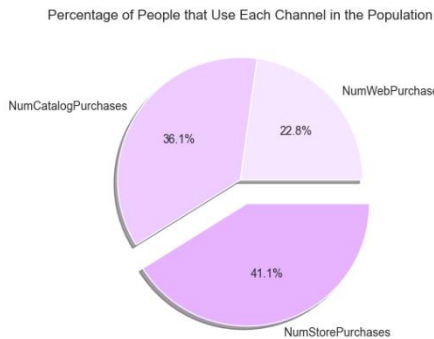
Le pourcentage d'acceptation des campagnes est très faible chez la population entière. Mais le pourcentage est le plus grand chez la population de salaire moyenne pour la dernière campagne, ce qui consiste de l'entreprise d'investir dans les autres campagnes pour les améliorer pour les groupes qui les ciblent.

F3-Pie Chart pour les différents locaux d'achat :

Medium Income :

Low Income :

Very-Low Income:



Le « Store » est le local le plus préféré pour l'achat chez la population du Medium et du Low salaire, même pour le groupe de « Very low » salaire, pour cela il faut essayer d'améliorer le local web qui est très importante pour le succès des entreprises. Il faut prendre en considération que les données sont prises en 2014.

Conclusion :

Division de la population d'après leur salaire, pour mieux comprendre le comportement des consommateurs en se basant sur différentes caractéristiques.

Une conclusion qui est en plus des solutions pour résoudre des problèmes dans les opérations et des méthodes pour améliorer les opérations déjà présentes.

Créez un programme de fidélité à plusieurs niveaux pour récompenser les clients à revenus élevés et moyens qui investissent dans les vins et les viandes, nos catégories les plus rentables.

Développez des offres groupées combinant les achats de viande et de vin pour tirer parti de leur corrélation avec des dépenses plus élevées.

Les clients sans partenaire ont tendance à dépenser plus dans la plupart des catégories, sauf pour les vins, où la différence est mineure. Cela est probablement dû à l'inclusion de clients veufs dans ce groupe, qui sont généralement âgés de plus de 60 ans et font partie de la tranche démographique la plus dépensière.

Pour les segments de clientèle qui préfèrent les achats par catalogue, concevez des campagnes marketing ciblées mettant en valeur la commodité des achats par catalogue, en mettant particulièrement l'accent sur la disponibilité de produits populaires comme la viande et les vins.

Employer du personnel compétent en magasin pour fournir une expertise sur les accords vins et viandes. Leurs conseils peuvent orienter les clients vers l'exploration de produits complémentaires et l'amélioration de leur expérience d'achat globale.

Références :

blog.hubspot.com

tableau.com

datylon.com visme.co

[Free 3D Infographics for Google Slides and PowerPoint \(slidesgo.com\)](http://slidesgo.com)

[3d Infographic Images - Free Download on Freepik](#) [seaborn.set style](#) — [seaborn](#)

[0.13.0 documentation \(pydata.org\)](#)

[Create a 3D Plot Using Seaborn and Matplotlib | Delft Stack](#)

[12 Data Plot Types for Visualization from Concept to Code - \(analyticsvidhya.com\)](#)

Annexes :

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import datetime
from datetime import date
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import squarify
# squarify fo the treemap
import pandas as pd
from math import pi
from mpl_toolkits.mplot3d import Axes3D
from mpl_toolkits import mplot3d
import plotly.express as px
sns.set_style("darkgrid")
import matplotlib.ticker as ticker

df = pd.read_csv(r"C:\Users\jneid\OneDrive\Desktop\Data Visualizatiomn\DataVizProject\marketing_campaign
- marketing_campaign.csv (1).csv")

#0-Problem :

# Convert the date to datetime objects with the correct format
df['Dt_Customer'] = pd.to_datetime(df['Dt_Customer'], format='%d-%m-%Y')

df['Total_Purchases'] = df['MntFruits'] + df['MntMeatProducts'] +
df['MntFishProducts'] + df["MntFishProducts"] + df["MntSweetProducts"] + df["MntGoldProds"]

# Extract the year from the purchase date
df['Year'] = df['Dt_Customer'].dt.year

# Group the data by year and calculate the average purchase for each year
average_purchase_by_year = df.groupby('Year')['Total_Purchases'].mean()/2

# Create a line chart of the average purchase over time
plt.plot(average_purchase_by_year.index, average_purchase_by_year.values, marker='o', linestyle='-' , color
="#d780ff")

# Add data points for the average of each year
```

```
plt.scatter(average_purchase_by_year.index, average_purchase_by_year.values, color='red', label='Average Purchase')
```

```
# Set the x-axis label and title  
plt.ylabel('Average Number of Purchases')  
plt.title('Average Purchase Over the Years')
```

```
# Customize the x-axis ticks and labels  
years = [2012, 2013, 2014]  
plt.xticks(years, years)
```

```
# Show the plot  
plt.show()
```

A - Distribution de la population : done

1 distribution by age

```
# Calculate the age of each customer based on their year of birth  
df['Age'] = 2015 - df['Year_Birth']
```

```
# Create a histogram of the age column using matplotlib  
plt.hist(df['Age'], bins=10, weights=np.ones(len(df['Age']))/len(df['Age']), color='#ae00ff', edgecolor='white')  
plt.xlabel('Age')  
plt.ylabel('Frequency')  
plt.title('Histogram of Customer Age Distribution')  
plt.xticks(np.arange(0, 110, 10)) # Set the ticks on the x axis from 0 to 110 with an interval of 10
```

```
# Get the patches object from the histogram  
patches = plt.gca().patches
```

```
# Get the max value from the histogram  
max_value = max([p.get_height() for p in patches])
```

```
# Loop through the patches and change the color of the max value  
for p in patches:
```



```
if p.get_height() == max_value:
    p.set_facecolor('#f7e6ff')
plt.show()
```

#1.2 marital status

```
# Define the color palette
colors = ['#f7e6ff', '#efccff', '#e7b3ff', '#df99ff', '#d780ff']

# Group the smaller values together
data = df["Marital_Status"].value_counts()
labels = [x if data[x] >= 5 else "others" for x in data.index]
data = data.groupby(labels).sum()

# Set the explode values
explode = [0.1 if x < max(data) else 0 for x in data]

# Create a pie chart with matplotlib
plt.pie(data, labels=labels, explode=explode, shadow=True, autopct="%1.1f%%", colors=colors)

# Set the title and the equal aspect ratio
plt.title("Marital Status Distribution")
plt.axis("equal")

# Show the chart
plt.show()
```

#1.3 treemap for education level : using squarify library

```
counts = df['Education'].value_counts()
labels = counts.index
sizes = counts.values
colors = ['#f7e6ff', '#efccff', '#e7b3ff', '#df99ff', '#d780ff']
squarify.plot(sizes=sizes, label=labels, color=colors)
plt.title("Treemap of Education")
plt.axis('off')
```

```
plt.show()
```

#1-4 Income Distribution

```
# Create a figure with a custom size and resolution
```

```
plt.figure(figsize=(10, 6), dpi=100)
```

```
df = df[(df['Income'] <= 170000) & (df['Income'] >= 1000)]
```

```
# Create a histogram of age for the high-income population
```

```
plt.hist(df["Income"], bins=range(0, max(df["Income"]) + 1000, 10000), color="#efccff", edgecolor="white")
```

```
# Add labels and a title to the plot
```

```
plt.xlabel("Income")
```

```
plt.ylabel("Frequency")
```

```
plt.title("Income Distribution for the Population")
```

```
tick_spacing = 10000
```

```
plt.xticks(range(0, max(df["Income"]) + 10000, tick_spacing), rotation='vertical')
```

```
# Format ticks to display in k units
```

```
plt.gca().xaxis.set_major_formatter(ticker.FuncFormatter(lambda x, _: '{:,.0f}k'.format(x / 1000)))
```

```
# Add vertical grid lines for better visibility
```

```
plt.grid(axis='x', linestyle='--', alpha=0.6)
```

```
# Show the plot
```

```
plt.show()
```

1.5 heatmap of avg spent on product by household composition

```
products = ['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',  
'MntGoldProds']
```

```
fig, axes = plt.subplots(2, 3, figsize=(15, 10))
```

```
for i, product in enumerate(products):
```

```
    row = i // 3
```

```
    col = i % 3
```

```

grouped = df.groupby(['Kidhome', 'Teenhome'])[product].mean().reset_index()

matrix = grouped.pivot(index='Kidhome', columns='Teenhome', values=product)

sns.heatmap(matrix, annot=True, cmap='Blues', ax=axes[row, col])
axes[row, col].set_xlabel('Number of Teenagers in Household')
axes[row, col].set_ylabel('Number of Children in Household')
axes[row, col].set_title(f'{product}')

plt.tight_layout()

plt.show()

# 1.6 stacked bar chart for if you want to create a stacked bar chart of the AcceptedCmp1, AcceptedCmp2,
AcceptedCmp3,
#AcceptedCmp4, AcceptedCmp5, and Response attributes by month of enrollment,

# Convert the Dt_Customer attribute to datetime format
df['Dt_Customer'] = pd.to_datetime(df['Dt_Customer'])

# Extract the month of enrollment from the Dt_Customer attribute
df['Month'] = df['Dt_Customer'].dt.month

# Group the data by month and calculate the sum of accepted campaigns
grouped = df.groupby('Month')[['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'AcceptedCmp4',
'AcceptedCmp5', 'Response']].sum()

# Get the labels, heights, and positions for each month
labels = grouped.index
heights1 = grouped['AcceptedCmp1'].values
heights2 = grouped['AcceptedCmp2'].values
heights3 = grouped['AcceptedCmp3'].values
heights4 = grouped['AcceptedCmp4'].values
heights5 = grouped['AcceptedCmp5'].values
heights6 = grouped['Response'].values
positions = range(len(labels))

# Create a stacked bar chart using matplotlib
plt.bar(positions, heights1, label='Campaign 1')
plt.bar(positions, heights2, bottom=heights1, label='Campaign 2')
plt.bar(positions, heights3, bottom=heights1+heights2, label='Campaign 3')
plt.bar(positions, heights4, bottom=heights1+heights2+heights3, label='Campaign 4')
plt.bar(positions, heights5, bottom=heights1+heights2+heights3+heights4, label='Campaign 5')
plt.bar(positions, heights6, bottom=heights1+heights2+heights3+heights4+heights5, label='Last Campaign')

```

```

plt.xticks(positions, labels)
plt.xlabel('Month of Enrollment')
plt.ylabel('Number of Accepted Campaigns')
plt.title('Stacked Bar Chart of Accepted Campaigns by Month')
plt.legend()
plt.show()

```

#1.7 Campaigns

```

# List of campaigns
campaigns = ['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5']

# Calculate the sum of each campaign
campaign_sums = [df[campaign].sum() for campaign in campaigns]

# Define a color palette
colors = ['#ff9999', '#f7e6ff', '#efccff', '#e7b3ff', '#df99ff', '#d780ff']

# Create a doughnut chart
fig, ax = plt.subplots()
ax.pie(campaign_sums, labels = campaigns, colors = colors, autopct='%1.1f%%', startangle=90,
pctdistance=0.85)

# Draw a white circle at the center to create the doughnut hole
centre_circle = plt.Circle((0,0),0.70,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

# Equal aspect ratio ensures that pie is drawn as a circle.
ax.axis('equal')
plt.tight_layout()
plt.show()

```

#####

#B - Etude de la relation entre les attributs :

1 3d Scatter Plot nb of total purchases by type of purchases and by age : non sense

```

# Assuming 'df' is your DataFrame with the dataset
# Calculate Total Purchases for each customer

```

```

df['Total_Purchases'] = df['MntFruits'] + df['MntMeatProducts'] + df['MntFishProducts']

```

```
df['Age'] = 2015 - df['Year_Birth']
```

```
# Create a 3D scatter plot
```

```
fig = plt.figure(figsize=(10, 8))
```

```
ax = fig.add_subplot(111, projection='3d')
```

```
# Scatter plot with Age, Total Purchases, and a marker for each product type  
for product in ['MntFruits', 'MntMeatProducts', 'MntFishProducts']:
```

```
    ax.scatter(df['Age'], df['Total_Purchases'], df[product], label=product)
```

```
# Set labels and title
```

```
ax.set_xlabel('Age')
```

```
ax.set_ylabel('Total Purchases')
```

```
ax.set_zlabel('Product Amount')
```

```
ax.set_title('3D Scatter Plot of Total Purchases by Age and Product Type')
```

```
# Add a legend
```

```
ax.legend()
```

```
ax.view_init(elev=30, azim=45)
```

```
plt.show()
```

```
# 1.2 nb of total purchases amount over marital status :
```

```
# Defining color palette :
```

```
colors = ['#f7e6ff', '#efccff', '#e7b3ff', '#df99ff', '#d780ff']
```

```
# Calculate the total amount of purchases for each customer by summing up the amount spent on different  
products
```

```
df['Total_Purchases'] = df['MntWines'] + df['MntFruits'] + df['MntMeatProducts'] + df['MntFishProducts'] +  
df['MntSweetProducts'] + df['MntGoldProds']
```

```
# Create a box plot of total purchases by marital status using seaborn
```

```
sns.boxplot(x='Marital_Status', y='Total_Purchases', data=df, palette=colors)
```

```
# Add a gray grid to the plot
```

```
plt.grid(axis='y', color='gray', linestyle='--')
```

```
# Set the labels and title for the axes and the plot
```

```
plt.xlabel('Marital Status')
```

```
plt.ylabel('Total Purchases')
```

```
plt.title('Box Plot of Total Purchases by Marital Status')
```

```
plt.xlabel('Marital Status')
plt.ylabel('Total Purchases')
plt.title('Box Plot of Total Purchases by Marital Status')
plt.show()
```

3 heatmap : for people features and total purchase amount

we have imported seaborn to plot the heatmap :

```
# Calculate total purchase amount
df["Total_Purchase"] = df["MntWines"] + df["MntFruits"] + df["MntMeatProducts"] + df["MntFishProducts"]
+ df["MntSweetProducts"] + df["MntGoldProds"]
```

```
# Calculate correlation matrix
corr = df[["Year_Birth", "Education", "Marital_Status", "Income", "Kidhome", "Teenhome",
"Total_Purchase"]].corr()
```

```
# Plot heatmap
plt.figure(figsize=(10, 10))
purple_palette = sns.color_palette(['#f7e6ff', '#efccff', '#e7b3ff', '#df99ff', '#d780ff'])
sns.heatmap(corr, cmap=purple_palette, annot=True, linewidths=0.5)
plt.title("Correlation Heatmap")
plt.show()
```

#5 Catalogue purchase and Total Purchase over 3 years :

```
# Calculate the total purchases for all channels for each row
df['TotalPurchases'] = df[['NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases',
'NumDealsPurchases']].sum(axis=1)
```

```
# Convert the date to datetime objects with the correct format
df['Dt_Customer'] = pd.to_datetime(df['Dt_Customer'], format='%d-%m-%Y')
```

```
# Extract the year from the purchase date
df['Year'] = df['Dt_Customer'].dt.year
```

```
# Group the data by year and calculate the average total purchases for each year
average_purchase_by_year = df.groupby('Year')['TotalPurchases'].mean()
```

```

# Get the individual purchase channels for each year
web_purchases = df.groupby('Year')['NumWebPurchases'].mean()
catalog_purchases = df.groupby('Year')['NumCatalogPurchases'].mean()
store_purchases = df.groupby('Year')['NumStorePurchases'].mean()

# Create an area plot for total purchases and each purchase channel with distinct colors
plt.fill_between(average_purchase_by_year.index, 0, average_purchase_by_year.values, label='Total Purchases',
color='grey', alpha=0.5)
plt.fill_between(catalog_purchases.index, 0, catalog_purchases.values, label='catalog_purchases',
color='#d780ff', alpha=0.5)

years = [2012, 2013, 2014]
plt.xticks(years, years)

# Set the x-axis label and title
plt.xlabel('Year')
plt.ylabel('Average Purchases')
plt.title('Average Purchases Over the Years')

# Show a legend
plt.legend()

# Show the plot
plt.show()

# 6 Store and total purchase over three years :
# Calculate the total purchases for all channels for each row
df['TotalPurchases'] = df[['NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases',
'NumDealsPurchases']].sum(axis=1)

# Convert the date to datetime objects with the correct format
df['Dt_Customer'] = pd.to_datetime(df['Dt_Customer'], format='%d-%m-%Y')

# Extract the year from the purchase date
df['Year'] = df['Dt_Customer'].dt.year

# Group the data by year and calculate the average total purchases for each year
average_purchase_by_year = df.groupby('Year')['TotalPurchases'].mean()

# Get the individual purchase channels for each year
web_purchases = df.groupby('Year')['NumWebPurchases'].mean()
catalog_purchases = df.groupby('Year')['NumCatalogPurchases'].mean()
store_purchases = df.groupby('Year')['NumStorePurchases'].mean()

```

```
# Create an area plot for total purchases and each purchase channel with distinct colors
plt.fill_between(average_purchase_by_year.index, 0, average_purchase_by_year.values, label='Total Purchases',
color='grey', alpha=0.5)
plt.fill_between(store_purchases.index, 0, store_purchases.values, label='store_purchases', color='#d780ff',
alpha=0.5)
```

```
years = [2012, 2013, 2014]
plt.xticks(years, years)
```

```
# Set the x-axis label and title
plt.xlabel('Year')
plt.ylabel('Average Purchases')
plt.title('Average Purchases Over the Years')
```

```
# Show a legend
plt.legend()
```

```
# Show the plot
plt.show()
```

```
# 7 web and total purchase over three years :
# Calculate the total purchases for all channels for each row
df['TotalPurchases'] = df[['NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases',
'NumDealsPurchases']].sum(axis=1)
```

```
# Convert the date to datetime objects with the correct format
df['Dt_Customer'] = pd.to_datetime(df['Dt_Customer'], format='%d-%m-%Y')
```

```
# Extract the year from the purchase date
df['Year'] = df['Dt_Customer'].dt.year
```

```
# Group the data by year and calculate the average total purchases for each year
average_purchase_by_year = df.groupby('Year')['TotalPurchases'].mean()
```

```
# Get the individual purchase channels for each year
web_purchases = df.groupby('Year')['NumWebPurchases'].mean()
catalog_purchases = df.groupby('Year')['NumCatalogPurchases'].mean()
store_purchases = df.groupby('Year')['NumStorePurchases'].mean()
```

```
# Create an area plot for total purchases and each purchase channel with distinct colors
plt.fill_between(average_purchase_by_year.index, 0, average_purchase_by_year.values, label='Total Purchases',
color='grey', alpha=0.5)
```



```
plt.fill_between(web_purchases.index, 0, web_purchases.values, label='web_purchases', color='#d780ff', alpha=0.5)
```

```
years = [2012, 2013, 2014]  
plt.xticks(years, years)
```

```
# Set the x-axis label and title  
plt.xlabel('Year')  
plt.ylabel('Average Purchases')  
plt.title('Average Purchases Over the Years')
```

```
# Show a legend  
plt.legend()
```

```
# Show the plot  
plt.show()
```

```
#CCCCCCCCCCC- Etude du Salaire
```

```
#1 avg income by education  
sns.set_style('darkgrid')  
# Group data by education and calculate mean income  
grouped = df.groupby("Education")["Income"].mean()  
  
# Sort grouped data by income values in ascending order  
grouped = grouped.sort_values()
```

```
# Create labels, heights, and colors for bar chart  
labels = grouped.index  
heights = grouped.values  
colors = ["#f7e6ff", "#efccff", "#e7b3ff", "#df99ff", "#d780ff"]
```

```
# Plot bar chart  
plt.bar(labels, heights, color=colors)  
plt.xlabel("Education")  
plt.ylabel("Average Income")  
plt.title("Bar Chart of Average Income by Education")
```

```
# Loop over the bars and add text  
for i, bar in enumerate(plt.gca().patches):  
    # Get the x and y coordinates of the bar  
    x = bar.get_x() + bar.get_width() / 2  
    y = bar.get_height()  
    # Format the height as a string with two decimal places
```

```

height = f"{y:.2f}"
# Annotate the bar with the height
plt.text(x, y, height, ha="center", va="bottom")

plt.show()

```

#2.2 avg of income distributed on diff marital status for people without the absurd category:

```

# Filter data by year of birth
df = df.query("Year_Birth <= 1997")

# Filter data by marital status
df = df.query("Marital_Status != 'Absurd'")

# Group data by marital status and calculate mean income
grouped = df.groupby("Marital_Status")["Income"].mean()

# Sort grouped data by income values in ascending order
grouped = grouped.sort_values()

# Create labels, heights, and colors for bar chart
labels = grouped.index
heights = grouped.values
colors = ["#f7e6ff", "#efccff", "#e7b3ff", "#df99ff", "#d780ff"]

# Plot bar chart
plt.bar(labels, heights, color=colors)
plt.xlabel("Marital_Status")
plt.ylabel("Average Income")
plt.title("Bar Chart of Average Income by Marital status")
plt.show()

```

#2.3 avg of income distributed on by age

```

# Calculate the age of each customer based on their year of birth
df['Age'] = 2015 - df['Year_Birth']

# Define age categories and labels
age_bins = [0, 30, 40, 50, 60, 100] # Define age bins
age_labels = ["18-30", "31-40", "41-50", "51-60", "61+"]
df['Age_Category'] = pd.cut(df['Age'], bins=age_bins, labels=age_labels)

# Group data by age category and calculate the mean income

```

```
grouped = df.groupby("Age_Category")["Income"].mean()
```

```
# Create labels, heights, and colors for the bar chart
```

```
labels = grouped.index
```

```
heights = grouped.values
```

```
colors = ["#f7e6ff", "#efccff", "#e7b3ff", "#df99ff", "#d780ff"]
```

```
# Plot bar chart
```

```
plt.bar(labels, heights, color=colors)
```

```
plt.xlabel("Age Category")
```

```
plt.ylabel("Average Income")
```

```
plt.title("Bar Chart of Average Income by Age Category")
```

```
plt.show()
```

```
# 2.4 nb of total purchases amount over income :
```

```
# Filter the data
```

```
df = df[(df['Income'] <= 170000) & (df['Income'] >= 1000)]
```

```
# Calculate the total amount of purchases for each customer by summing up the amount spent on different products
```

```
df['Total_Purchases'] = df['MntWines'] + df['MntFruits'] + df['MntMeatProducts'] + df['MntFishProducts'] +  
df['MntSweetProducts'] + df['MntGoldProds']
```

```
# Create a scatter plot of income vs total purchases using matplotlib
```

```
plt.scatter(df['Income'], df['Total_Purchases'])
```

```
plt.xlabel('Income')
```

```
plt.ylabel('Total Purchases')
```

```
plt.title('Scatter Plot of Income vs Total Purchases')
```

```
sns.regplot(x='Income', y='Total_Purchases', data=df, color='#d780ff')
```

```
plt.show()
```

```
#D Disecting the population :
```

```
"""
```

```
# Calculate the mean and standard deviation
```

```
mean_income = np.mean(df['Income'])
```

```
std_dev = np.std(df['Income'])
```

```
# Define the cutoffs for low, medium, and high based on standard deviation
```

```
low_cutoff = mean_income - std_dev
```

```

medium_cutoff = mean_income + std_dev
high_cutoff = mean_income + (2 * std_dev)

# Define the bins for categorizing income
bins = [-np.inf, low_cutoff, medium_cutoff, high_cutoff, np.inf]

# Define labels for the categories
labels = ['Very Low', 'Low', 'Medium', 'High']

# Create the "Income Category" column
df['Income_Category'] = pd.cut(df['Income'], bins=bins, labels=labels)

df['Age'] = 2015 - df['Year_Birth']

"""

# Income_Ctageory Distribution :

# Create a figure with a custom size and resolution
plt.figure(figsize=(10, 6), dpi=100)

# Filter the data to remove outliers
df = df[(df['Income'] <= 170000) & (df['Income'] >= 1000)]

# Count the occurrences of each Income_Category
income_counts = df['Income_Category'].value_counts().sort_values()

# Create a bar chart of the count distribution for each Income_Category
plt.bar(income_counts.index, income_counts, color="#efccff", edgecolor="white")

# Add lollipops as scatter points
plt.scatter(income_counts.index, income_counts, color="#ae00ff", marker='o', s=50)

# Add labels and a title to the plot
plt.xlabel("Income Category")
plt.ylabel("Count")
plt.title("Lollipop Chart of Income_Category Distribution")

# Set the x-axis ticks for a clearer result
plt.xticks(rotation='vertical')

# Add vertical grid lines for better visibility
plt.grid(axis='y', linestyle='--', alpha=0.6)

# Show the plot

```

```
plt.show()
```

#E1: Distribution of population for high_income :

#1- Age Distributon

Filter the data to keep only the rows where the income category is high

```
df = df.loc[df["Income_Category"] == "High"]
```

Create a figure with a custom size and resolution

```
plt.figure(figsize=(10, 6), dpi=100)
```

Create a histogram of age for the high-income population

```
plt.hist(df["Age"], bins=range(0, max(df["Age"]) + 5, 5), color="#ae00ff", edgecolor="white")
```

Add labels and a title to the plot

```
plt.xlabel("Age")
```

```
plt.ylabel("Frequency")
```

```
plt.title("Age Distribution for High-Income Population")
```

Set the x-axis ticks for a clearer result

```
plt.xticks(range(0, max(df["Age"]) + 1, 5))
```

Add vertical grid lines for better visibility

```
plt.grid(axis='x', linestyle='--', alpha=0.6)
```

Show the plot

```
plt.show()
```

#2= Donut Chart for education distribution :

education for high income population

```
sns.set_style("darkgrid")
```

Filter the DataFrame for "High" income category

```
df = df.loc[df["Income_Category"] == "High"]
```

Count the frequency of education level

```
counts = df["Education"].value_counts()
```

Create a pie chart of education level

```
plt.pie(counts, labels=counts.index, colors=["#f7e6ff", "#efccff", "#e7b3ff", "#df99ff"], shadow=True)
```

```

# Add a circle at the center of the pie chart
circle = plt.Circle((0, 0), 0.5, color="white")
ax = plt.gca()
ax.add_artist(circle)

# Add percentage labels and a title to the donut chart
plt.xlabel("Education Level")
plt.title("Distribution of Education Level for Medium Income Population")

# Calculate and annotate with percentage labels
total = sum(counts)
percentages = [(count / total) * 100 for count in counts]
labels = [f"{edu} ({percent:.1f}%)" for edu, percent in zip(counts.index, percentages)]
plt.legend(labels, title="Education Level", loc="center left", bbox_to_anchor=(1, 0, 0.5, 1))

plt.show()

```

#3 Countplot for Marital status for high income category (state the difference between a counterplot bar and histo

```

df = df.loc[df["Income_Category"] == "High"]
colors = ["#f7e6ff", "#efccff", "#e7b3ff", "#df99ff", "#d780ff"]

# Create a count plot of marital status for high-income population
sns.countplot(x=df["Marital_Status"], palette=colors)
plt.xlabel("Marital Status")
plt.ylabel("Frequency")
plt.title("Count Plot of Marital Status for High-Income Population")
plt.show()

```

4 Radar Chart of products distribution for High Income Population
Filter the data

```

df = df.loc[(df["Marital_Status"] == "Together") & (df["Age"] >= 35) & (df["Age"] <= 40) &
(df["Education"].isin(["PhD", "Graduation"])&(df["Income_Category"] == "High"))]

# Select the columns for the products and calculate the mean by row
products = df[["MntWines", "MntFruits", "MntMeatProducts", "MntFishProducts", "MntSweetProducts",
"MntGoldProds"]].mean(axis=0)

```

```

# Create a radar chart for the average of product consumption
theta = np.linspace(0, 2*np.pi, len(products), endpoint=False)
plt.polar(theta, products, color="#e7b3ff", marker='o') # Use specified color and add markers
plt.fill(theta, products, color="#e7b3ff", alpha=0.3) # Fill the radar area with alpha for transparency

# Add labels and a title to the plot
plt.thetagrids(np.degrees(theta), products.index)
plt.title("Avg Consumption by Product")

# Show or save the plot
plt.show()

```

5 Distribution of acceptance rate

```

df = df.loc[(df["Marital_Status"] == "Together") & (df["Age"] >= 35) & (df["Age"] <= 40) &
(df["Education"].isin(["PhD", "Graduation"])&(df["Income_Category"] == "High"))]

```

Calculate the acceptance rate for each campaign by dividing the mean of accepted customers by the total number of customers

```

acceptance_rate = df[["AcceptedCmp1", "AcceptedCmp2", "AcceptedCmp3", "AcceptedCmp4",
"AcceptedCmp5", "Response"]].mean()

```

Plot the horizontal bar chart with labels and title

```

plt.barh(acceptance_rate.index, acceptance_rate.values * 100 , color = "#efccff")
plt.ylabel("Campaign")
plt.xlabel("Acceptance Rate (%)")
plt.title("Acceptance Rate for Each Campaign in the Population")
plt.show()

```

#6 Channels for high income population

```

df = df.loc[(df["Marital_Status"] == "Together") & (df["Age"] >= 35) & (df["Age"] <= 40) &
(df["Education"].isin(["PhD", "Graduation"])) & (df["Income_Category"] == "High")]

```

```

colors = ["#f7e6ff", "#efccff", "#e7b3ff"]

```

Calculate the total number of purchases for each channel by summing up the columns

```

channels = df[["NumWebPurchases", "NumCatalogPurchases", "NumStorePurchases"]].sum()

```

```
# Plot the pie chart with labels and title
plt.pie(channels.values, labels=channels.index, autopct="%1.1f%%" , shadow = True , colors = colors ,
explode=(0,0.2,0))
plt.title("Percentage of People that Use Each Channel in the Population")
plt.show()
```

E2 : Population with medium income :

#1- Age Distributon

```
# Filter the data to keep only the rows where the income category is high
df = df.loc[df["Income_Category"] == "Medium"]
```

```
# Create a figure with a custom size and resolution
plt.figure(figsize=(10, 6), dpi=100)
```

```
# Create a histogram of age for the high-income population
plt.hist(df["Age"], bins=range(0, max(df["Age"]) + 5, 5), color="#efccff", edgecolor="white")
```

```
# Add labels and a title to the plot
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.title("Age Distribution for Medium-Income Population")
```

```
# Set the x-axis ticks for a clearer result
plt.xticks(range(0, max(df["Age"]) + 1, 5))
```

```
# Add vertical grid lines for better visibility
plt.grid(axis='x', linestyle='--', alpha=0.6)
```

```
# Show the plot
plt.show()
```

#2= Donut Chart for education distribution :

education for high income population

```
sns.set_style("darkgrid")
```

```
# Filter the DataFrame for "High" income category
```



```

df = df.loc[df["Income_Category"] == "Medium"]

# Count the frequency of education level
counts = df["Education"].value_counts()

# Create a pie chart of education level
plt.pie(counts, labels=counts.index, colors=["#f7e6ff", "#efccff", "#e7b3ff", "#df99ff"] , shadow=True)

# Add a circle at the center of the pie chart
circle = plt.Circle((0, 0), 0.5, color="white")
ax = plt.gca()
ax.add_artist(circle)

# Add percentage labels and a title to the donut chart
plt.xlabel("Education Level")
plt.title("Distribution of Education Level for Medium Income Population")

# Calculate and annotate with percentage labels
total = sum(counts)
percentages = [(count / total) * 100 for count in counts]
labels = [f"{edu} ({percent:.1f}%)" for edu, percent in zip(counts.index, percentages)]
plt.legend(labels, title="Education Level", loc="center left", bbox_to_anchor=(1, 0, 0.5, 1))

plt.show()

```

#3 Countplot for Marital status for Medium income category (state the difference between a counterplot bar and histo

```

df = df.loc[df["Income_Category"] == "Medium"]
colors = ["#f7e6ff", "#efccff", "#e7b3ff", "#df99ff", "#d780ff"]

# Create a count plot of marital status for high-income population
sns.countplot(x=df["Marital_Status"], palette=colors)
plt.xlabel("Marital Status")
plt.ylabel("Frequency")
plt.title("Count Plot of Marital Status for Medium-Income Population")
plt.show()

```

4 Radar Chart of products distribution for Medium Income Population

Filter the data

```
df = df.loc[df["Income_Category"] == "Medium"]
```

```
df = df.loc[(df["Marital_Status"] == "Married") & (df["Age"] >= 55) & (df["Age"] <= 60) & (df["Education"].isin(["Graduation"]))]
```

Select the columns for the products and calculate the mean by row

```
products = df[["MntWines", "MntFruits", "MntMeatProducts", "MntFishProducts", "MntSweetProducts", "MntGoldProds"]].mean(axis=0)
```

Create a radar chart for the average of product consumption

```
theta = np.linspace(0, 2*np.pi, len(products), endpoint=False)
```

```
plt.polar(theta, products, color="#e7b3ff", marker='o') # Use specified color and add markers
```

```
plt.fill(theta, products, color="#e7b3ff", alpha=0.3) # Fill the radar area with alpha for transparency
```

Add labels and a title to the plot

```
plt.thetagrids(np.degrees(theta), products.index)
```

```
plt.title("Avg Consumption by Product")
```

Show or save the plot

```
plt.show()
```

5 Distribution of acceptance rate

```
df = df.loc[df["Income_Category"] == "Medium"]
```

```
df = df.loc[(df["Marital_Status"] == "Married") & (df["Age"] >= 55) & (df["Age"] <= 60) & (df["Education"].isin(["Graduation"]))]
```

Calculate the acceptance rate for each campaign by dividing the mean of accepted customers by the total number of customers

```
acceptance_rate = df[["AcceptedCmp1", "AcceptedCmp2", "AcceptedCmp3", "AcceptedCmp4", "AcceptedCmp5", "Response"]].mean()
```

Plot the horizontal bar chart with labels and title

```
plt.barh(acceptance_rate.index, acceptance_rate.values * 100, color="#efccff")
```

```
plt.ylabel("Campaign")
```

```
plt.xlabel("Acceptance Rate (%)")
```

```
plt.title("Acceptance Rate for Each Campaign in the Population")
```

```
plt.show()
```

#6 Channels for Medium income population

```
df = df.loc[df["Income_Category"] == "Medium"]
df = df.loc[(df["Marital_Status"] == "Married") & (df["Age"] >= 55) & (df["Age"] <= 60) &
(df["Education"].isin(["Graduation"]))]
```

```
colors = ["#f7e6ff", "#efccff", "#e7b3ff"]
```

```
# Calculate the total number of purchases for each channel by summing up the columns
```

```
channels = df[["NumWebPurchases", "NumCatalogPurchases", "NumStorePurchases"]].sum()
```

```
# Plot the pie chart with labels and title
```

```
plt.pie(channels.values, labels=channels.index, autopct="%1.1f%%" , shadow = True , colors = colors ,
explode=(0,0,0.2))
```

```
plt.title("Percentage of People that Use Each Channel in the Population")
```

```
plt.show()
```

#E3 - Low Income Study :

#1- Age Distributon

```
# Filter the data to keep only the rows where the income category is high
```

```
df = df.loc[df["Income_Category"] == "Low"]
```

```
# Create a figure with a custom size and resolution
```

```
plt.figure(figsize=(10, 6), dpi=100)
```

```
# Create a histogram of age for the high-income population
```

```
plt.hist(df["Age"], bins=range(0, max(df["Age"]) + 5, 5), color="#efccff", edgecolor="white")
```

```
# Add labels and a title to the plot
```

```
plt.xlabel("Age")
```

```
plt.ylabel("Frequency")
```

```
plt.title("Age Distribution for Low-Income Population")
```

```
# Set the x-axis ticks for a clearer result
```

```
plt.xticks(range(0, max(df["Age"]) + 1, 5))
```

```
# Add vertical grid lines for better visibility
```

```
plt.grid(axis='x', linestyle='--', alpha=0.6)
```

```
# Show the plot
plt.show()
```

#2= Donut Chart for education distribution :

```
# education for high income population
```

```
sns.set_style("darkgrid")
df = df.loc[df["Income_Category"] == "Low"]
```

```
# Count the frequency of education level
counts = df["Education"].value_counts()
```

```
# Create a pie chart of education level
plt.pie(counts, labels=counts.index, colors=["#f7e6ff", "#efccff", "#e7b3ff", "#df99ff"], shadow=True)
```

```
# Add a circle at the center of the pie chart
circle = plt.Circle((0, 0), 0.5, color="white")
ax = plt.gca()
ax.add_artist(circle)
plt.xlabel("Education Level")
plt.title("Distribution of Education Level for low Income Population")
total = sum(counts)
percentages = [(count / total) * 100 for count in counts]
labels = [f"{edu} ({percent:.1f}%)" for edu, percent in zip(counts.index, percentages)]
plt.legend(labels, title="Education Level", loc="best", bbox_to_anchor=(1, 0, 0.5, 1))
```

```
plt.show()
```

#3 Countplot for Marital status for Medium income category (state the difference between a counterplot bar and histo

```
df = df.loc[df["Income_Category"] == "Low"]
colors = ["#f7e6ff", "#efccff", "#e7b3ff", "#df99ff", "#d780ff"]
sns.countplot(x=df["Marital_Status"], palette=colors)
plt.xlabel("Marital Status")
plt.ylabel("Frequency")
```

```
plt.title("Count Plot of Marital Status for Low-Income Population")
plt.show()
```

```
# 4 Radar Chart of products distribution for low Income Population
```

```
# Filter the data
```

```
df = df.loc[df["Income_Category"] == "Low"]
```

```
df = df.loc[(df["Marital_Status"] == "Married") & (df["Age"] >= 40) & (df["Age"] <= 45) & (df["Education"].isin(["Graduation"]))]
```

```
products = df[["MntWines", "MntFruits", "MntMeatProducts", "MntFishProducts", "MntSweetProducts", "MntGoldProds"]].mean(axis=0)
```

```
theta = np.linspace(0, 2*np.pi, len(products), endpoint=False)
```

```
plt.polar(theta, products, color="#e7b3ff", marker='o') # Use specified color and add markers
```

```
plt.fill(theta, products, color="#e7b3ff", alpha=0.3) # Fill the radar area with alpha for transparency
```

```
plt.thetagrids(np.degrees(theta), products.index)
```

```
plt.title("Avg Consumption by Product")
```

```
# Show or save the plot
```

```
plt.show()
```

```
# 5 Distribution of acceptance rate
```

```
df = df.loc[df["Income_Category"] == "Low"]
```

```
df = df.loc[(df["Marital_Status"] == "Married") & (df["Age"] >= 40) & (df["Age"] <= 45) & (df["Education"].isin(["Graduation"]))]
```

```
acceptance_rate = df[["AcceptedCmp1", "AcceptedCmp2", "AcceptedCmp3", "AcceptedCmp4", "AcceptedCmp5", "Response"]].mean()
```

```
plt.barh(acceptance_rate.index, acceptance_rate.values * 100, color = "#efccff")
```

```
plt.ylabel("Campaign")
```

```
plt.xlabel("Acceptance Rate (%)")
```

```
plt.title("Acceptance Rate for Each Campaign in the Population")
```

```
plt.show()
```

```
#6 Channels for high income population
```

```
df = df.loc[df["Income_Category"] == "Low"]
```

```
df = df.loc[(df["Marital_Status"] == "Married") & (df["Age"] >= 40) & (df["Age"] <= 45) &
(df["Education"].isin(["Graduation"]))]
```

```
colors = ["#f7e6ff", "#efccff", "#e7b3ff"]
channels = df[["NumWebPurchases", "NumCatalogPurchases", "NumStorePurchases"]].sum()
plt.pie(channels.values, labels=channels.index, autopct="%1.1f%%" , shadow = True , colors = colors ,
explode=(0,0,0.2))
plt.title("Percentage of People that Use Each Channel in the Population")
plt.show()
```

#E4 - Population with Very-Low Income :

#1- Age Distributon

```
df = df.loc[df["Income_Category"] == "Very Low"]
plt.figure(figsize=(10, 6), dpi=100)
plt.hist(df["Age"], bins=range(0, max(df["Age"]) + 5, 5), color="#efccff", edgecolor="white")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.title("Age Distribution for Very Low-Income Population")
plt.xticks(range(0, max(df["Age"]) + 1, 5))
plt.grid(axis='x', linestyle='--', alpha=0.6)
plt.show()
```

#2= Donut Chart for education distribution :

```
sns.set_style("darkgrid")
df = df.loc[df["Income_Category"] == "Very Low"]
counts = df["Education"].value_counts()
plt.pie(counts, labels=counts.index, colors=["#f7e6ff", "#efccff", "#e7b3ff", "#df99ff"] , shadow=True)
circle = plt.Circle((0, 0), 0.5, color="white")
ax = plt.gca()
ax.add_artist(circle)
plt.xlabel("Education Level")
plt.title("Distribution of Education Level for Very Low Income Population")
total = sum(counts)
percentages = [(count / total) * 100 for count in counts]
labels = [f"{edu} ({percent:.1f}%" for edu, percent in zip(counts.index, percentages)]
plt.legend(labels, title="Education Level", loc="best", bbox_to_anchor=(1, 0, 0.5, 1))
plt.show()
```

#3 Countplot for Marital status for Medium income category (state the difference between a counterplot bar and histo

```

df = df.loc[df["Income_Category"] == "Very Low"]
colors = ["#f7e6ff", "#efccff", "#e7b3ff", "#df99ff", "#d780ff"]
sns.countplot(x=df["Marital_Status"], palette=colors)
plt.xlabel("Marital Status")
plt.ylabel("Frequency")
plt.title("Count Plot of Marital Status for Very Low-Income Population")
plt.show()

```

4 Radar Chart of products distribution for High Income Population

Filter the data

```

df = df.loc[df["Income_Category"] == "Very Low"]
df = df.loc[(df["Marital_Status"] == "Married") & (df["Age"] >= 35) & (df["Age"] <= 40) &
(df["Education"].isin(["Graduation"]))]
products = df[["MntWines", "MntFruits", "MntMeatProducts", "MntFishProducts", "MntSweetProducts",
"MntGoldProds"]].mean(axis=0)
theta = np.linspace(0, 2*np.pi, len(products), endpoint=False)
plt.polar(theta, products, color="#e7b3ff", marker='o') # Use specified color and add markers
plt.fill(theta, products, color="#e7b3ff", alpha=0.3) # Fill the radar area with alpha for transparency
plt.thetagrids(np.degrees(theta), products.index)
plt.title("Avg Consumption by Product")

```

Show or save the plot

```
plt.show()
```

5 Distribution of acceptance rate

```

df = df.loc[df["Income_Category"] == "Very Low"]
df = df.loc[(df["Marital_Status"] == "Married") & (df["Age"] >= 35) & (df["Age"] <= 40) &
(df["Education"].isin(["Graduation"]))]
acceptance_rate = df[["AcceptedCmp1", "AcceptedCmp2", "AcceptedCmp3", "AcceptedCmp4",
"AcceptedCmp5", "Response"]].mean()
plt.barh(acceptance_rate.index, acceptance_rate.values * 100, color = "#efccff")
plt.ylabel("Campaign")
plt.xlabel("Acceptance Rate (%)")
plt.title("Acceptance Rate for Each Campaign in the Population")
plt.show()

```

#6 Channels for high income population

```
df = df.loc[df["Income_Category"] == "Very Low"]
```

```
df = df.loc[(df["Marital_Status"] == "Married") & (df["Age"] >= 35) & (df["Age"] <= 40) &
(df["Education"].isin(["Graduation"]))]
colors = ["#f7e6ff", "#efccff", "#e7b3ff"]
channels = df[["NumWebPurchases", "NumCatalogPurchases", "NumStorePurchases"]].sum()
plt.pie(channels.values, labels=channels.index, autopct="%1.1f%%" , shadow = True , colors = colors ,
explode=(0,0,0.2))
plt.title("Percentage of People that Use Each Channel in the Population")
plt.show()
```

""""