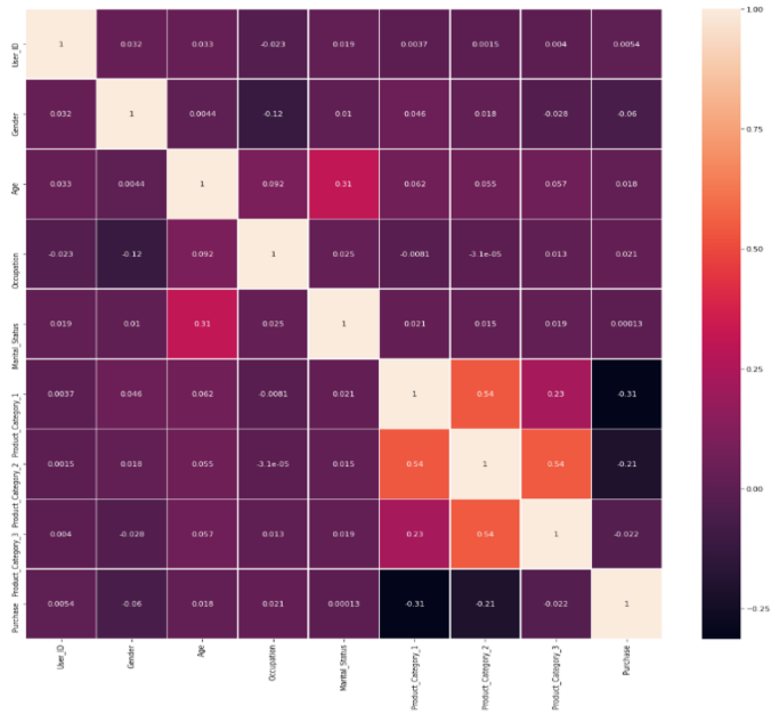
Correlations between features



Clustering (K-Means)

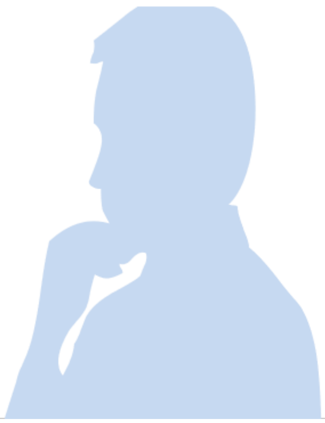
Based on my initial assumption, we would like to cluster our customers based on the products they have purchased and other features

We need to change our features a little bit to be more suitable for clustering. Firstly, we are going to cluster users so we should group our transactions by User_ID and create our features based on that. Now, what we know about each user, is his/her gender, stayed-city, stay in current city years, age, marital status, occupation and products he/she has bought. Let's exclude products he has bought and their categories for now, we will get to them later:

```
train = bf.drop(['Product_ID', 'Product_Category_1', 'Product_Category_2', 'Product_Category_3', 'Purchase'], axis=1).groupby('User_ID')
```

In terms of other features (gender, city, stay in city years, age, marital status and occupation) their values should all be the same in all rows for a particular User_ID but we assume there might be small noises and some of them may have wrong values in different rows, so we will take the mode value (one which is repeated the most) for each of them.

On the other hand, they are all nominal features so we will one hot encode them since there are not that much unique values:



Clustering (K-Means)

```
train = train.agg(lambda x: x.value_counts().index[-1])
feaures_list = list(train.columns.values)
encoder = OneHotEncoder().fit(train[feaures_list])
train = pd.concat([train, pd.DataFrame(encoder.transform(train[feaures_list]).toarray(), index=train.index, columns=encoder.get_feature_names(feaures_list)
train.drop(feaures_list, axis=1, inplace=True)
```

Now in regard to products, we can't add a column for each product and make it 1 if the user has bought it or 0 otherwise (due to huge number of products) so we'll do it for only top 100 products (by number of transactions).
Note that not only we know which user has bought which product, but we can also use purchase amounts as a metric for how much does this user likes/needs this product. We'll do the same thing for product categories but with all of them as features since there are no more than 18 categories.
So, in conclusion, we are going to find 100 most selling products and 18 categories (by number of transactions) and for each user, put purchase amount of this product/product category as a new feature for him, adding totally 118 new features to our data:

Clustering (K-Means)

```
columns = ['Product_ID', 'Product_Category_1']
for column in columns:
    top_100 = bf[column].value_counts().index[:100]
    user_purchase = pd.pivot_table(
        bf[['User_ID', column, 'Purchase']],
        values='Purchase',
        index='User_ID',
        columns=column,
        aggfunc=np.sum
    ).fillna(0)[top_100]
    train = train.join(user_purchase)
```

add total purchase amount for each user as a new feature:

```
train = train.join(bf[['User_ID', 'Purchase']].groupby('User_ID').agg('sum'))
```

standardize data in columns so features will have same scales in order to get clustered:

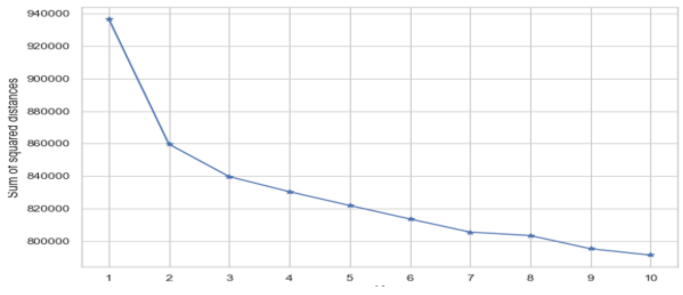
```
train_scaled = StandardScaler().fit_transform(train)
```

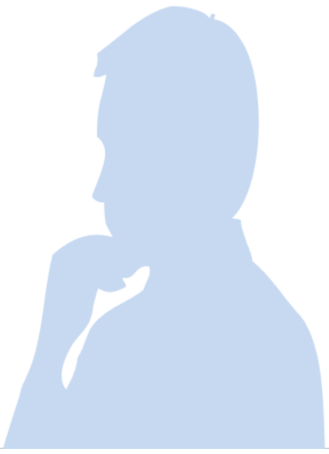
Clustering (K-Means)

we should choose number of clusters (K). We will use elbow method to choose one. So let's plot different distance sums for different number of clusters:

```
k_values = np.arange(1, 11)
models = []
dists = []
for k in k_values:
    model = KMeans(k).fit(train_scaled)
    models.append(model)
    dists.append(model.inertia_)

plt.figure(figsize=(9, 6))
plt.plot(k_values, dists, 'o-')
plt.ylabel('Sum of squared distances', size=13)
plt.xlabel('K', size=13)
plt.xticks(k_values)
plt.show()
```





Clustering (K-Means)

```
for k in k_values:
    model = models[k]
    print("{} model Silhouette score: {:.2f}".format(k,silhouette_score(train_scaled, model.predict(train_scaled))) )
```

results are as followed:

```
1 model Silhouette score: 0.20
2 model Silhouette score: 0.09
3 model Silhouette score: 0.04
4 model Silhouette score: 0.04
5 model Silhouette score: -0.01
6 model Silhouette score: -0.01
7 model Silhouette score: 0.01
8 model Silhouette score: -0.01
9 model Silhouette score: -0.01
```

We clustered our customers using K-Means algorithm with (1,11) clusters and the silhouettes score which shows maybe we can't group users confidently