1.

Name of the data set used in the example: Iris data

And the three species are: Irus Setosa, Iris Versicolor, and Iris Virgina.

Source: https://www.youtube.com/watch?v=a18nd3gwBoA

2.

According to the scatterplot created by Mazen the X-coordinate is sepal length and Y-coordinate is petal width

Source:https://www.youtube.com/watch?v=a18nd3gwBoA

3.

Mazen uses the Elbow Method when we do not known how many clusters are present in the data The "elbow" point of the plot is described as the point at which inertia starts to decrease more gradually, producing a bend like an elbow. This choice represents a sensible trade-off between lowering inertia and not having too many clusters, making it an appropriate one given the amount of clusters.

Source:https://www.youtube.com/watch?v=a18nd3gwBoA

4.

For two-thirds of IT leaders, "cost management and containment" is their top issue when it comes to managing big data cloud technology. A third of senior IT leaders admit to overspending by up to 40% on cloud fees. This problem is exacerbated by complex cloud pricing schemes and dynamic resource provisioning. Effective cost monitoring, resource optimization, and governance measures must be put into place in order to manage cloud costs and prevent unforeseen spending.

Source: - Razzaq, A. (2022). Why real-time cost anomaly detection for your cloud is non-negotiable. Retrieved from: https://www.cfodive.com/news/why-real-time-cost-anomaly-detection-for-your-cloud-is-non-negotiable/621805/

5.

Enterprises can prevent cloud overspending with a comprehensive cost management and anomaly detection platform. It uses machine learning to monitor cloud expenditure, detect anomalies in real-time, and provide cost trend insights. AWS Cost Anomaly Detection, integrated with AWS Cost Explorer, identifies unexpected spending changes and their causes. For complex deployments, third-party cost management tools enhance visibility and optimization across multiple cloud environments (Razzaq, 2022).

Source: - Razzaq, A. (2022). Why real-time cost anomaly detection for your cloud is non-negotiable. Retrieved from: https://www.cfodive.com/news/why-real-time-cost-anomaly-detection-for-your-cloud-is-non-negotiable/621805/

6.

In this article the authors uses the four unspurvised machine learning algorithms:

The researchers in this study employed four prominent machine learning algorithms for anomaly identification, each rooted in distinct principles. Firstly, they utilized an autoencoder (AE) neural network to grasp typical input representations and detect deviations. Secondly, Isolation Forest (IF) was used to isolate anomalies by creating random data partitions. Thirdly, they applied Lightweight On-Line Anomaly Detection (LODA), combining multiple one-dimensional projections to detect outliers. Lastly, Local Outlier Factor (LOF), a proximity-based technique, was utilized to identify anomalies based on data density. The main goal was to enhance the recognition of insider threats using these unsupervised learning techniques for improved cybersecurity measures.

Source: Le, D. C., & Zincir-Heywood, N. (2021). Anomaly detection for insider threats using unsupervised ensembles. IEEE Transactions on Network and Service Management, 18(2), 1152-1164. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9399116&casa_token=doXw042XbSoAAAAA:Vlmb7gxRoPECJn1lqXzLsdHbSbUhoQvbXkiDciOGHRXb9IBRKs1wLTbSfCnISwlgENdVO7w

7.

Threat detection performance of the authors was evaluated using False Positive Rate and Detection Delay. They utilized several unsupervised machine learning techniques, including Autoencoder (AE), to identify insider hazards in the CERT R6.2 dataset. Low FPRs and quick detection times allowed for the quick and simple identification of situations 1, 3, and 4. The FPRs for scenarios 2 and 5 were higher, but they were harder to spot since they involved fewer overtly hostile acts that could be mistaken for normal behavior. The precise method that performed the best in the excerpt's numerous scenarios was not mentioned.

Source:

Le, D. C., & Zincir-Heywood, N. (2021). Anomaly detection for insider threats using unsupervised ensembles. IEEE Transactions on Network and Service Management, 18(2), 1152-1164. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9399116&casa_token=doXw042XbSoAAAAA:Vlmb7gxRoPECJn1lqXzLsdHbSbUhoQvbXkiDciOGHRXb9IBRKs1wLTbSfCnISwlgENdVO7w

VIJAY RAJ PATHANI 7/23/2023

Directions for Unsupervised Machine Learning

K-Means and DBSCAN ¶

Step 1: Import Libraries

In [1]:
```
# Import Python Libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

import plotly as py
import plotly.graph_objs as go
```

In [2]:
```
from scipy import stats
from sklearn.metrics import silhouette_score
from sklearn.cluster import DBSCAN
from sklearn.cluster import KMeans
from itertools import product
from mpl_toolkits.mplot3d import Axes3D
```

In [3]:
```
#filter warnings
import warnings
warnings.filterwarnings("ignore")
```

Step 2: Load Dataset & Get Dataset shape

In [8]:
```
mall_data = pd.read_csv('Mall_Customers.csv')
```

In [9]:
```
mall_data.shape
```

Out[9]: (200, 5)

In [11]:
```
mall_data.head()
```

Out[11]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

In [12]:
```
mall_data.describe()
```

Out[12]:

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

Step 3: Data Cleaning

In [13]:
```
# We see there are no missing data points
mall_data.isnull().sum()
```

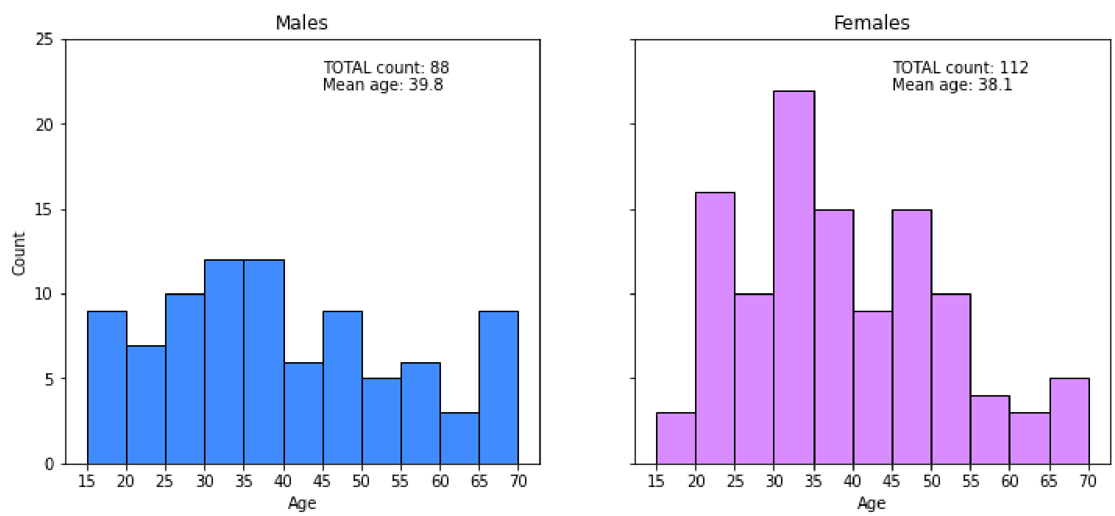
Out[13]:
```
CustomerID                0
Genre                     0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

Step 4: Performing the Exploratory Data Analysis

Distribution - Age

In [14]:

```
males_age = mall_data[mall_data['Genre'] == 'Male']['Age']
# subset with males age
females_age = mall_data[mall_data['Genre'] == 'Female']['Age']
# subset with females age
age_bins = range(15, 75, 5)

# Males histogram
fig2, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5), sharey=True)
sns.histplot(males_age, bins=age_bins, color='#0066ff', ax=ax1)
ax1.set_xticks(age_bins)
ax1.set_ylim(top=25)
ax1.set_title('Males')
ax1.set_ylabel('Count')
ax1.text(45, 23, "TOTAL count: {}".format(males_age.count()))
ax1.text(45, 22, "Mean age: {:.1f}".format(males_age.mean()))

# Females histogram
sns.histplot(females_age, bins=age_bins, color='#cc66ff', ax=ax2)
ax2.set_xticks(age_bins)
ax2.set_title('Females')
ax2.set_ylabel('Count')
ax2.text(45, 23, "TOTAL count: {}".format(females_age.count()))
ax2.text(45, 22, "Mean age: {:.1f}".format(females_age.mean()))

plt.show()

# Kolgomorov-Smirnov test p-value
print('Kolgomorov-Smirnov test p-value: {:.2f}'.format(stats.ks_2samp(male
```



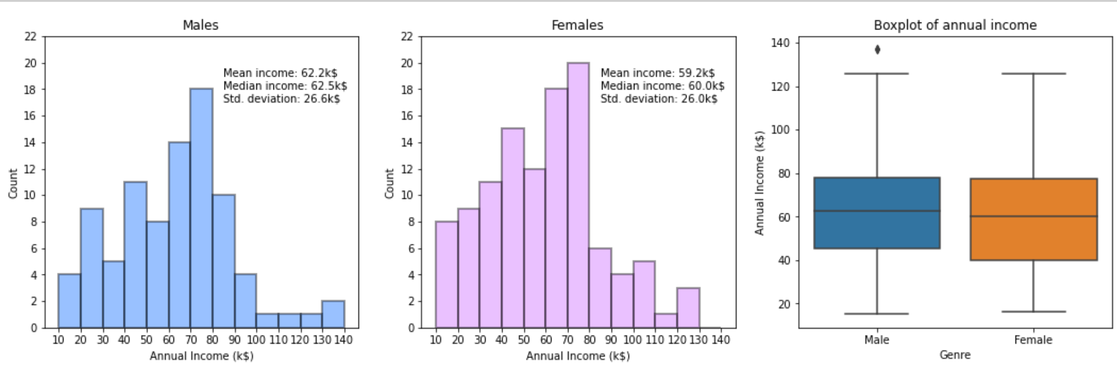

```
Kolgomorov-Smirnov test p-value: 0.49
```

Male age is more evenly distributed than females. The biggest age group for women is between 30-35, and for men, it is between 30-40. The Kolmogorov-Smirnov test shows the difference between the males and females age groups is statistically insignificant as it

Distribution - Annual Income (k$)

In [15]:

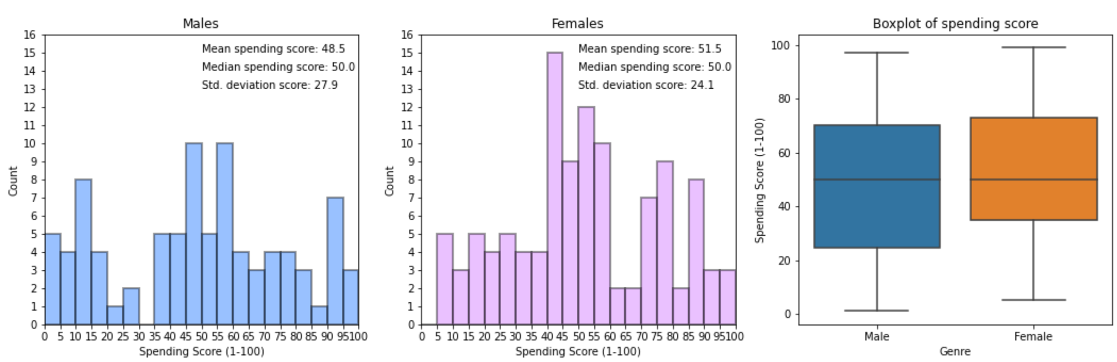
```
males_income = mall_data[mall_data['Genre'] == 'Male']['Annual Income (k$)
# subset with males income
females_income = mall_data[mall_data['Genre'] == 'Female']['Annual Income 
# subset with females income
my_bins = range(10, 150, 10)

# Males histogram
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(18, 5))
sns.distplot(males_income, bins=my_bins, kde=False, color='#0066ff', ax=ax
             hist_kws=dict(edgecolor="k", linewidth=2))
ax1.set_xticks(my_bins)
ax1.set_yticks(range(0, 24, 2))
ax1.set_ylim(0, 22)
ax1.set_title('Males')
ax1.set_ylabel('Count')
ax1.text(85, 19, "Mean income: {:.1f}k$".format(males_income.mean()))
ax1.text(85, 18, "Median income: {:.1f}k$".format(males_income.median()))
ax1.text(85, 17, "Std. deviation: {:.1f}k$".format(males_income.std()))

# Females histogram
sns.distplot(females_income, bins=my_bins, kde=False, color='#cc66ff',
             ax=ax2, hist_kws=dict(edgecolor="k", linewidth=2))
ax2.set_xticks(my_bins)
ax2.set_yticks(range(0, 24, 2))
ax2.set_ylim(0, 22)
ax2.set_title('Females')
ax2.set_ylabel('Count')
ax2.text(85, 19, "Mean income: {:.1f}k$".format(females_income.mean()))
ax2.text(85, 18, "Median income: {:.1f}k$".format(females_income.median()
ax2.text(85, 17, "Std. deviation: {:.1f}k$".format(females_income.std()))

# Boxplot
sns.boxplot(x='Genre', y='Annual Income (k$)', data=mall_data, ax=ax3)
ax3.set_title('Boxplot of annual income')
plt.show()

# Kolmogorov-Smirnov test p-value
print('Kolgomorov-Smirnov test p-value: {:.2f}'.format(stats.ks_2samp(male
```




```
Kolgomorov-Smirnov test p-value: 0.78
```

Distribution - Spending Score (1-100)

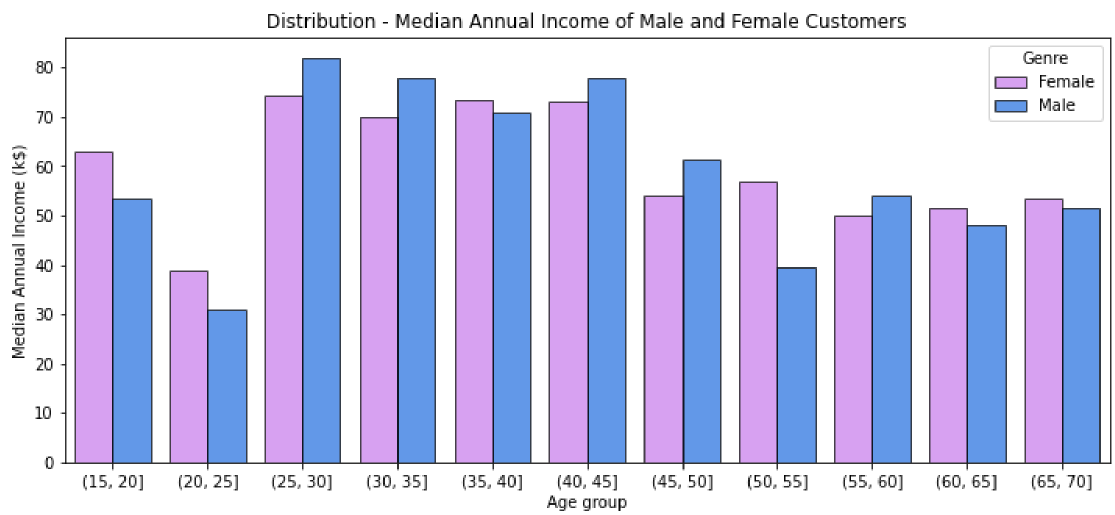
In [16]:
```
males_spending = mall_data[mall_data['Genre'] == 'Male']['Spending Score (
# subset with males spending score
females_spending = mall_data[mall_data['Genre'] == 'Female']['Spending Sco
# subset with females spending score
spending_bins = range(0, 105, 5)

# Males histogram
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(18, 5))
sns.distplot(males_spending, bins=spending_bins, kde=False, color='#0066ff
             ax=ax1, hist_kws=dict(edgecolor="k", linewidth=2))
ax1.set_xticks(spending_bins)
ax1.set_xlim(0, 100)
ax1.set_yticks(range(0, 17, 1))
ax1.set_ylim(0, 16)
ax1.set_title('Males')
ax1.set_ylabel('Count')
ax1.text(50, 15, "Mean spending score: {:.1f}".format(males_spending.mean(
ax1.text(50, 14, "Median spending score: {:.1f}".format(males_spending.med
ax1.text(50, 13, "Std. deviation score: {:.1f}".format(males_spending.std(

# Females histogram
sns.distplot(females_spending, bins=spending_bins, kde=False,
             color='#cc66ff', ax=ax2, hist_kws=dict(edgecolor="k", linewid
ax2.set_xticks(spending_bins)
ax2.set_xlim(0, 100)
ax2.set_yticks(range(0, 17, 1))
ax2.set_ylim(0, 16)
ax2.set_title('Females')
ax2.set_ylabel('Count')
ax2.text(50, 15, "Mean spending score: {:.1f}".format(females_spending.mea
ax2.text(50, 14, "Median spending score: {:.1f}".format(females_spending.m
ax2.text(50, 13, "Std. deviation score: {:.1f}".format(females_spending.st

# Boxplot
sns.boxplot(x='Genre', y='Spending Score (1-100)', data=mall_data, ax=ax3)
ax3.set_title('Boxplot of spending score')
plt.show()

# Kolmogorov-Smirnov test p-value
print('Kolgomorov-Smirnov test p-value: {:.2f}'.format(stats.ks_2samp(male
```




```
Kolgomorov-Smirnov test p-value: 0.29
```

Distribution - Median Annual Income of Male and Female Customers

In [20]:
```
# Group by 'Genre' and 'Age_group' and calculate the median annual income
age_bins = range(15, 75, 5)
mall_data['Age_group'] = pd.cut(mall_data['Age'], bins=age_bins)
medians_by_age_group = mall_data.groupby(['Genre', 'Age_group'])['Annual I

# Create the bar plot
fig, ax = plt.subplots(figsize=(12, 5))
sns.barplot(x='Age_group', y='Annual Income (k$)', hue='Genre',
            data=medians_by_age_group, palette=['#cc66ff', '#0066ff'],
            alpha=0.7, edgecolor='k', ax=ax)

ax.set_title('Distribution - Median Annual Income of Male and Female Custo
ax.set_xlabel('Age group')
ax.set_ylabel('Median Annual Income (k$)')
plt.show()
```



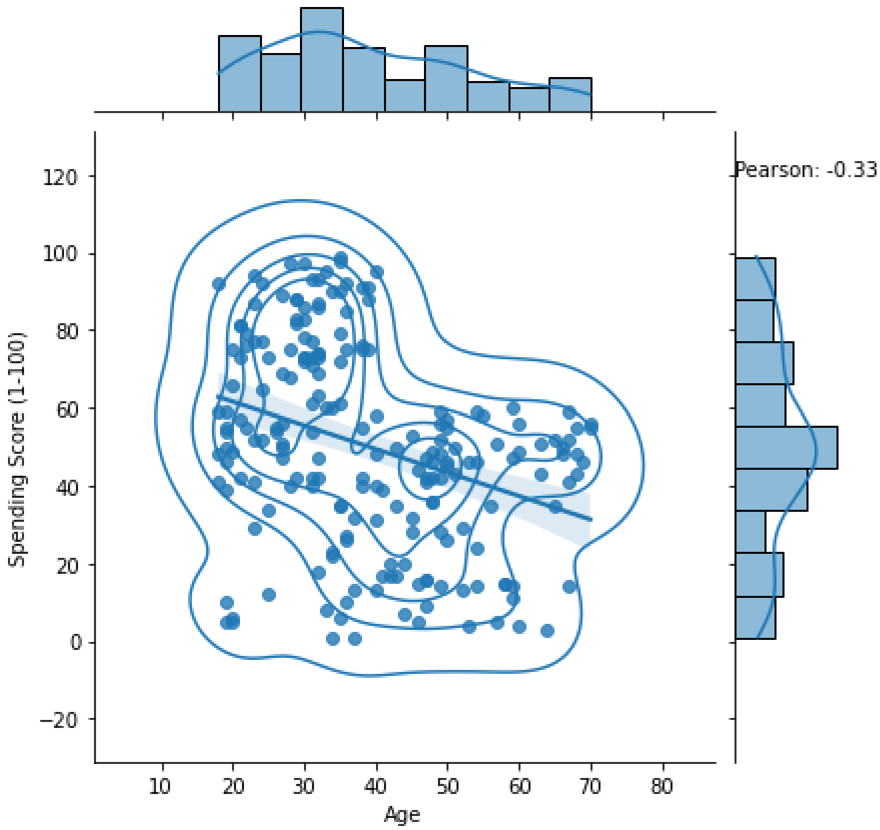
Correlations

Pearson's Correlation for Age & Spending Score (1-100) with jointplot

In [21]:
```
from scipy.stats import pearsonr
```

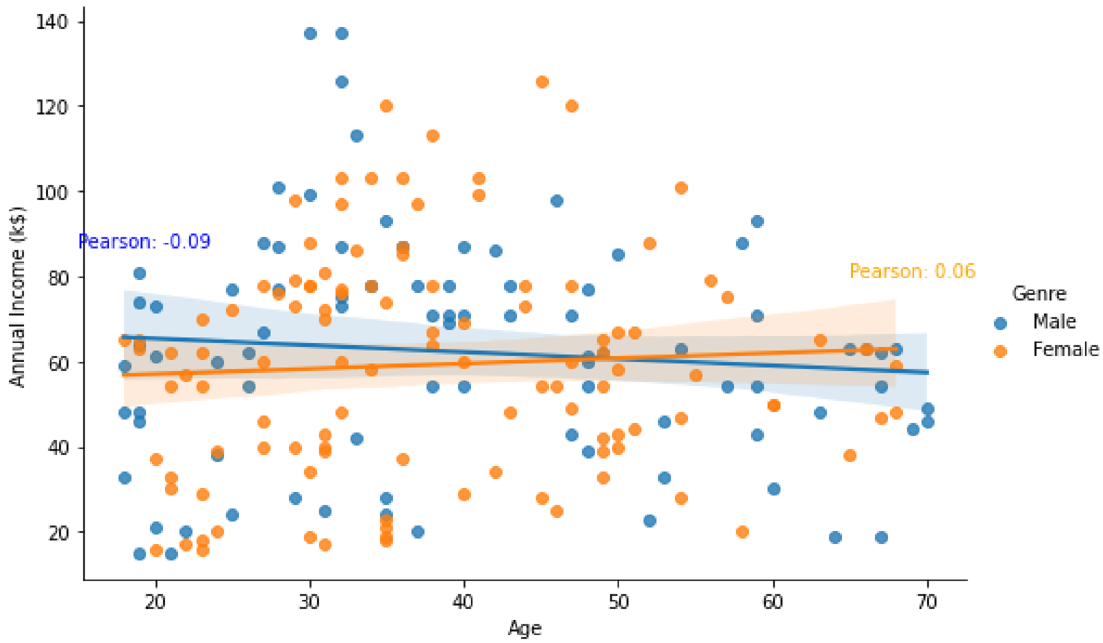
In [22]:
```
# calculating Pearson's correlation
corr, _ = pearsonr(mall_data['Age'], mall_data['Spending Score (1-100)'])
```

In [23]:
```
jp = sns.jointplot(x='Age', y='Spending Score (1-100)', data=mall_data,
 kind='reg').plot_joint(sns.kdeplot, zorder=0, n_levels=6)
plt.text(0, 120, 'Pearson: {:.2f}'.format(corr))
plt.show()
```



Pearson's Correlation for Age & Annual Income with lineplot (lmplot)

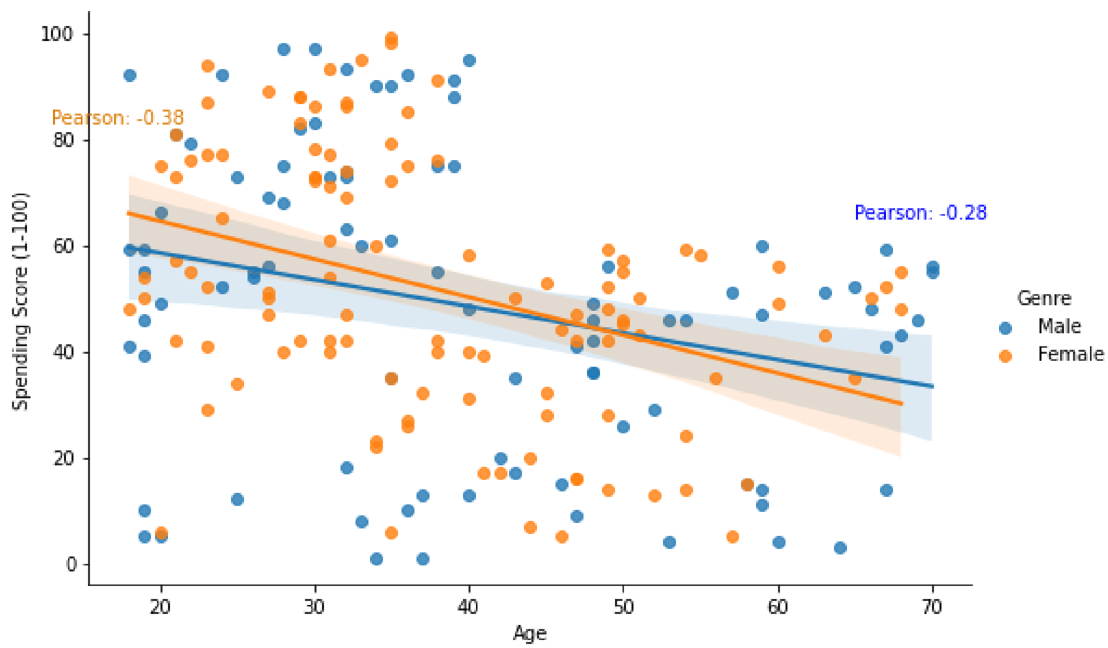
In [24]:
```
# Calculating Pearson's Correlation
corr1, _ = pearsonr(males_age.values, males_income.values)
corr2, _ = pearsonr(females_age.values, females_income.values)
sns.lmplot(x='Age', y='Annual Income (k$)', data=mall_data, hue='Genre', aspect=1.5)
plt.text(15,87, 'Pearson: {:.2f}'.format(corr1), color='blue')
plt.text(65,80, 'Pearson: {:.2f}'.format(corr2), color='orange')
plt.show()
```



Pearson's Correlation for Age & Spending Score (1-100) with line plot (lmplot)

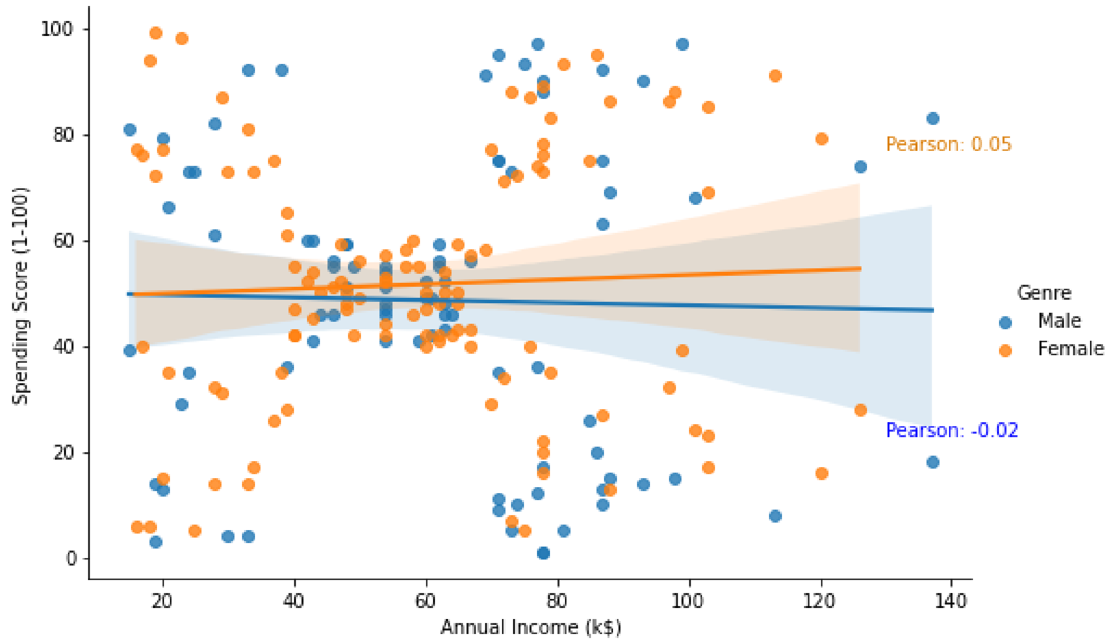
In [25]:

```
# calculating Pearson's correlations
corr1, _ = pearsonr(males_age.values, males_spending.values)
corr2, _ = pearsonr(females_age.values, females_spending.values)
sns.lmplot(x='Age', y='Spending Score (1-100)', data=mall_data, hue='Genre
plt.text(65,65, 'Pearson: {:.2f}'.format(corr1), color='blue')
plt.text(13,83, 'Pearson: {:.2f}'.format(corr2), color='#d97900')
plt.show()
```



Pearson's Correlation for Annual Income & Spending Score (1-100) with line plot (lmplot)

In [26]:
```
# calculating Pearson's correlations
corr1, _ = pearsonr(males_income.values, males_spending.values)
corr2, _ = pearsonr(females_income.values, females_spending.values)
sns.lmplot(x='Annual Income (k$)', y='Spending Score (1-100)', data=mall_d
plt.text(130,23, 'Pearson: {:.2f}'.format(corr1), color='blue')
plt.text(130,77, 'Pearson: {:.2f}'.format(corr2), color='#d97900')
plt.show()
```



Step 5 K-Means

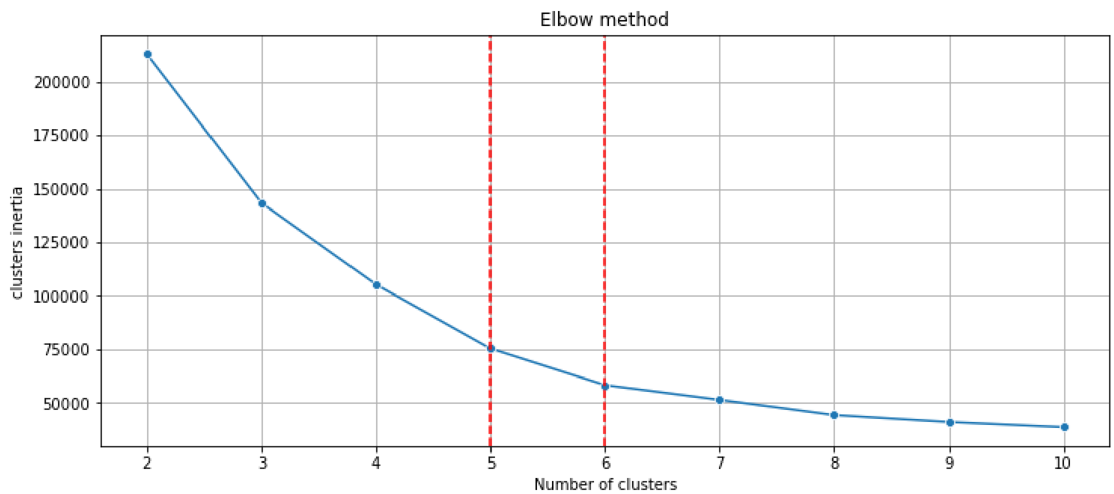
The k-means algorithm is the most used partitional clustering technique. It gained popularity since it is so easy and simple to implement, and it has had many successes in many diverse sectors. As we learned in our lecture for this module, k-means is a greedy algorithm. Remember with the greedy algorithm to be aware of global optimization. Also, remember that the Euclidean distance is implemented, and the number of clusters has to be predefined. In this assignment, you will use the Elbow Method and Silhouette Score to calculate the k-value. For clustering, you will only use numerical columns.

In [28]:
```
X_numerics = mall_data[['Age', 'Annual Income (k$)', 'Spending Score (1-10
# subset with numeric variables only
```

Elbow Method

In [29]:
```
n_clusters = [2,3,4,5,6,7,8,9,10] # number of clusters
clusters_inertia = [] # inertia of clusters
s_scores = [] # silhouette scores
for n in n_clusters:
 KM_est = KMeans(n_clusters=n, init='k-means++').fit(X_numerics)
 clusters_inertia.append(KM_est.inertia_) # data for the elbow method
 silhouette_avg = silhouette_score(X_numerics, KM_est.labels_)
 s_scores.append(silhouette_avg) # data for the silhouette score method
```

In [30]:
```
fig, ax = plt.subplots(figsize=(12, 5))
ax = sns.lineplot(x=n_clusters, y=clusters_inertia, marker='o', ax=ax)
ax.set_title("Elbow method")
ax.set_xlabel("Number of clusters")
ax.set_ylabel("clusters inertia")
ax.axvline(5, ls="--", c="red")
ax.axvline(6, ls="--", c="red")
plt.grid()
plt.show()
```

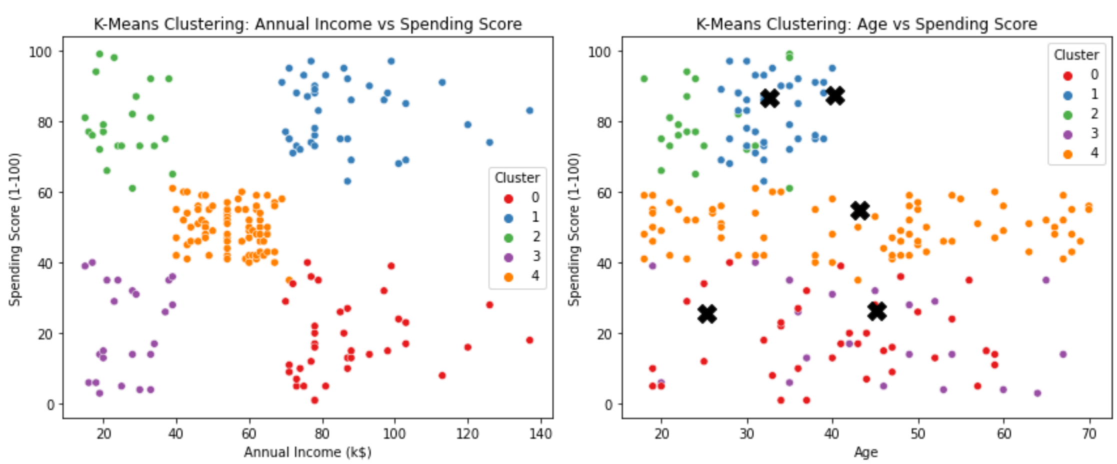


Silhouette Score Method

In [31]:
```
fig, ax = plt.subplots(figsize=(12,5))
ax = sns.lineplot(x=n_clusters, y=s_scores, marker='o', ax=ax)
ax.set_title("Silhouette score method")
ax.set_xlabel("number of clusters")
ax.set_ylabel("Silhouette score")
ax.axvline(6, ls="--", c="red")
plt.grid()
plt.show()
```

Let's try 5 Clusters

In [34]:
```
# Initialize and fit K-Means model
KM_5_clusters = KMeans(n_clusters=5, init='k-means++').fit(X_numerics)

# Create a copy of the DataFrame to store the cluster labels
KM5_clustered = X_numerics.copy()
KM5_clustered['Cluster'] = KM_5_clusters.labels_

# Create a subplot layout with two axes for the scatter plots
fig, axes = plt.subplots(1, 2, figsize=(12, 5))

# First scatter plot: 'Annual Income (k$)' vs 'Spending Score (1-100)' col
scat_1 = sns.scatterplot(x='Annual Income (k$)', y='Spending Score (1-100)
                         data=KM5_clustered, hue='Cluster', ax=axes[0],
                         palette='Set1', legend='full')

# Second scatter plot: 'Age' vs 'Spending Score (1-100)' colored by cluste
sns.scatterplot(x='Age', y='Spending Score (1-100)', data=KM5_clustered,
                hue='Cluster', ax=axes[1], palette='Set1', legend='full')

# Plot cluster centers on the second scatter plot
axes[1].scatter(KM_5_clusters.cluster_centers_[:, 0], KM_5_clusters.cluste
                s=200, c='black', marker='X', label='Cluster Centers')

# Set titles and labels for the plots
axes[0].set_title('K-Means Clustering: Annual Income vs Spending Score')
axes[1].set_title('K-Means Clustering: Age vs Spending Score')
axes[0].set_xlabel('Annual Income (k$)')
axes[1].set_xlabel('Age')
axes[0].set_ylabel('Spending Score (1-100)')
axes[1].set_ylabel('Spending Score (1-100)')

# Show the plots
plt.tight_layout()
plt.show()
```



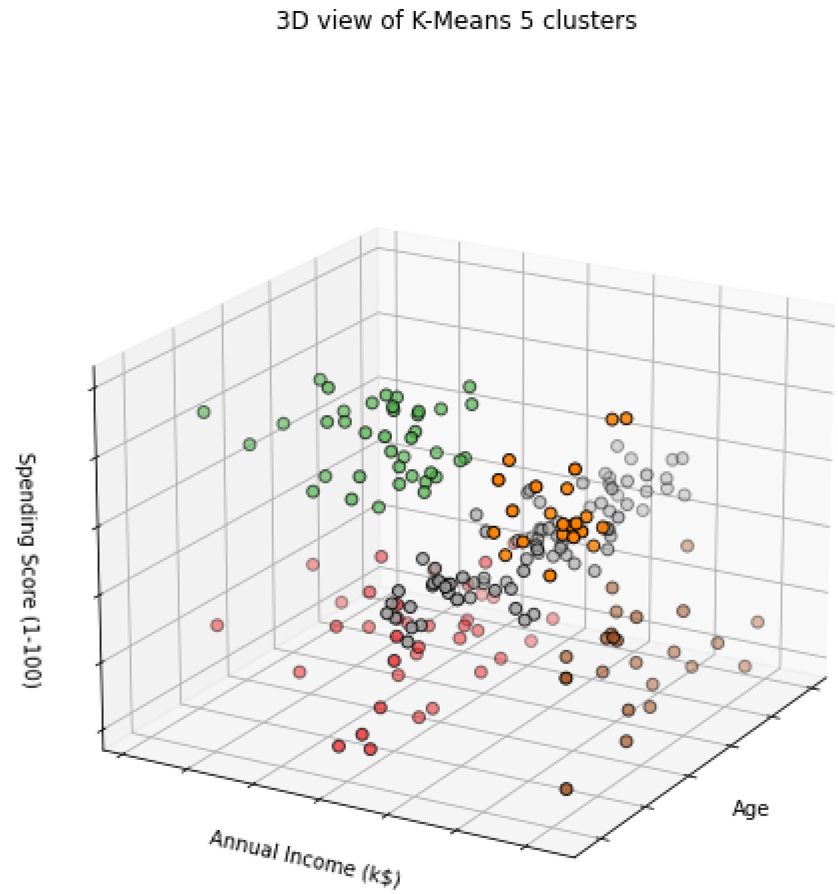
In [35]:

```
# Size of clusters
KM_clust_sizes = KM5_clustered.groupby('Cluster').size().to_frame()
KM_clust_sizes.columns = ["KM_size"]
KM_clust_sizes
```

Out[35]:

	KM_size
Cluster	
0	37
1	39
2	22
3	23
4	79

In [36]:
```
#Create a 3D projection of 5 generated clusters.
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(figsize=(7, 7))
ax = Axes3D(fig, rect=[0, 0, .99, 1], elev=20, azim=210)
ax.scatter(KM5_clustered['Age'],
 KM5_clustered['Annual Income (k$)'],
 KM5_clustered['Spending Score (1-100)'],
 c=KM5_clustered['Cluster'],
 s=35, edgecolor='k', cmap=plt.cm.Set1)
ax.w_xaxis.set_ticklabels([])
ax.w_yaxis.set_ticklabels([])
ax.w_zaxis.set_ticklabels([])
ax.set_xlabel('Age')
ax.set_ylabel('Annual Income (k$)')
ax.set_zlabel('Spending Score (1-100)')
ax.set_title('3D view of K-Means 5 clusters')
ax.dist = 12
plt.show()
```



Let's try 6 Clusters

In [37]:

```
# Initialize and fit K-Means model with 6 clusters
KM_6_clusters = KMeans(n_clusters=6, init='k-means++').fit(X_numerics)

# Create a copy of the DataFrame to store the cluster labels
KM6_clustered = X_numerics.copy()
KM6_clustered['Cluster'] = KM_6_clusters.labels_

# Create a subplot layout with two axes for the scatter plots
fig, axes = plt.subplots(1, 2, figsize=(12, 5))

# First scatter plot: 'Annual Income (k$)' vs 'Spending Score (1-100)' col
sns.scatterplot(x='Annual Income (k$)', y='Spending Score (1-100)',
                data=KM6_clustered, hue='Cluster', ax=axes[0],
                palette='Set1', legend='full')

# Second scatter plot: 'Age' vs 'Spending Score (1-100)' colored by cluste
sns.scatterplot(x='Age', y='Spending Score (1-100)', data=KM6_clustered,
                hue='Cluster', palette='Set1', ax=axes[1], legend='full')

# Plot cluster centers on both scatter plots
axes[0].scatter(KM_6_clusters.cluster_centers_[:, 1], KM_6_clusters.cluste
                s=200, c='black', marker='X', label='Cluster Centers')
axes[1].scatter(KM_6_clusters.cluster_centers_[:, 1], KM_6_clusters.cluste
                s=200, c='black', marker='X', label='Cluster Centers')

# Set titles and labels for the plots
axes[0].set_title('K-Means Clustering: Annual Income vs Spending Score')
axes[1].set_title('K-Means Clustering: Age vs Spending Score')
axes[0].set_xlabel('Annual Income (k$)')
axes[1].set_xlabel('Age')
axes[0].set_ylabel('Spending Score (1-100)')
axes[1].set_ylabel('Spending Score (1-100)')

# Show the plots
plt.tight_layout()
plt.show()
```

In [38]:
```
# Size of Clusters
KM6_clust_sizes = KM6_clustered.groupby('Cluster').size().to_frame()
KM6_clust_sizes.columns = ["KM_size"]
KM6_clust_sizes
```

Out[38]:

Cluster	KM_size
0	45
1	38
2	35
3	21
4	39
5	22

In [42]:

```
import plotly.graph_objs as go
import plotly.io as pio

# ... (assuming KM6_clustered is already defined)

def tracer(db, n, name):
    '''
    This function returns trace object for Plotly
    '''
    return go.Scatter3d(
        x=db[db['Cluster'] == n]['Age'],
        y=db[db['Cluster'] == n]['Spending Score (1-100)'],
        z=db[db['Cluster'] == n]['Annual Income (k$)'],
        mode='markers',
        name=name,
        marker=dict(
            size=5
        )
    )

trace0 = tracer(KM6_clustered, 0, 'Cluster 0')
trace1 = tracer(KM6_clustered, 1, 'Cluster 1')
trace2 = tracer(KM6_clustered, 2, 'Cluster 2')
trace3 = tracer(KM6_clustered, 3, 'Cluster 3')
trace4 = tracer(KM6_clustered, 4, 'Cluster 4')
trace5 = tracer(KM6_clustered, 5, 'Cluster 5')

data = [trace0, trace1, trace2, trace3, trace4, trace5]
layout = go.Layout(
    title='Clusters by K-Means',
    scene=dict(
        xaxis=dict(title='Age'),
        yaxis=dict(title='Spending Score'),
        zaxis=dict(title='Annual Income')
    )
)

fig = go.Figure(data=data, layout=layout)
pio.show(fig)
```

STEP 6: DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

The basis of this algorithm is based on the concept of density. We will use the distance (eps) and the minimum number of points within the distance as parameters. Again, we will use the Euclidean distance for the DBSCAN.

Creating a matrix of investigated combinations and the number of generated clusters.

In [43]:

```
# create a matrix of investigated combinations.
eps_values = np.arange(8,12.75,0.25) # eps values to be investigated
min_samples = np.arange(3,10) # min_samples values to be investigated
DBSCAN_params = list(product(eps_values, min_samples))
```

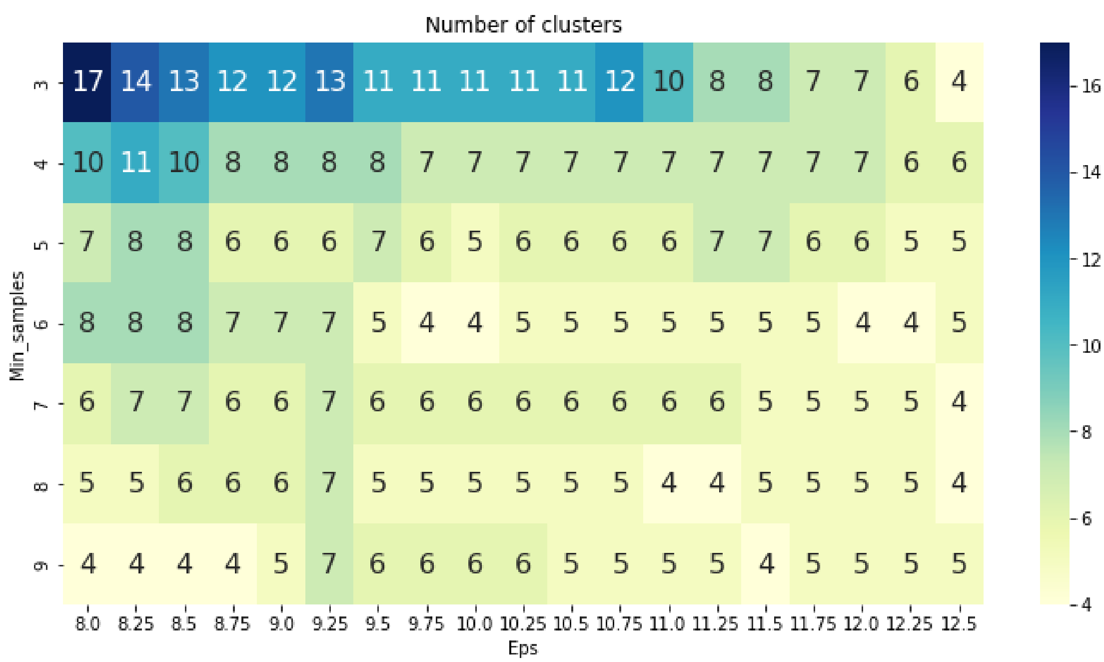
In [44]:

```
# collecting number of generated clusters
no_of_clusters = []
sil_score = []
for p in DBSCAN_params:
 DBS_clustering = DBSCAN(eps=p[0], min_samples=p[1]).fit(X_numerics)
 no_of_clusters.append(len(np.unique(DBS_clustering.labels_)))
 sil_score.append(silhouette_score(X_numerics, DBS_clustering.labels_))
```

Create Heatmaps

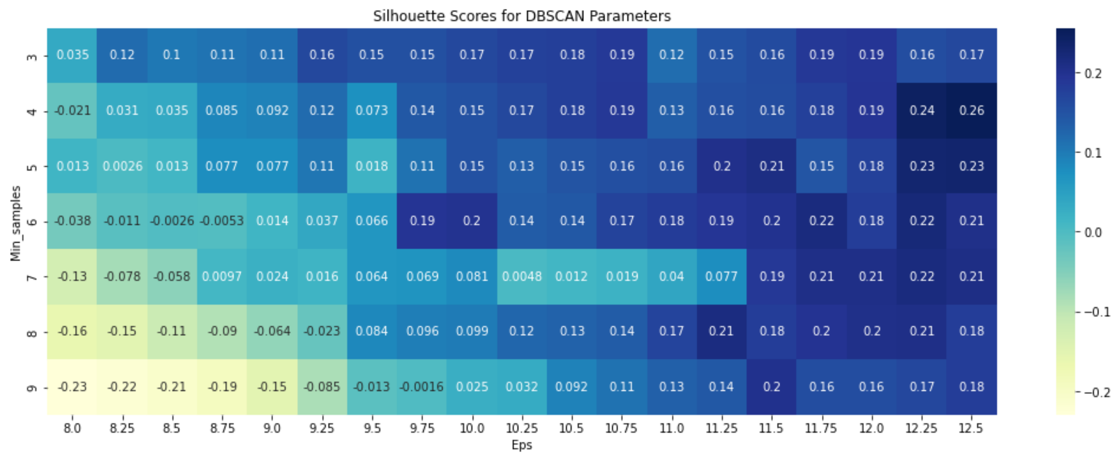
In [46]:

```
tmp = pd.DataFrame.from_records(DBSCAN_params, columns=['Eps', 'Min_sample
tmp['No_of_clusters'] = no_of_clusters

pivot_1 = pd.pivot_table(tmp, values='No_of_clusters', index='Min_samples

fig, ax = plt.subplots(figsize=(12, 6))
sns.heatmap(pivot_1, annot=True, annot_kws={"size": 16}, cmap="YlGnBu", ax
ax.set_title('Number of clusters')
plt.show()
```



In [47]:

```
tmp = pd.DataFrame.from_records(DBSCAN_params, columns=['Eps', 'Min_sample
tmp['Sil_score'] = sil_score

pivot_1 = pd.pivot_table(tmp, values='Sil_score', index='Min_samples', co]

fig, ax = plt.subplots(figsize=(18, 6))
sns.heatmap(pivot_1, annot=True, annot_kws={"size": 10}, cmap="YlGnBu", ax
ax.set_title('Silhouette Scores for DBSCAN Parameters')
plt.show()
```



In [48]:

```
# Based on the graph above let's use the Global maximum settings
DBS_clustering = DBSCAN(eps=12.5, min_samples=4).fit(X_numerics)
DBSCAN_clustered = X_numerics.copy()
DBSCAN_clustered.loc[:,'Cluster'] = DBS_clustering.labels_
```

In [49]:

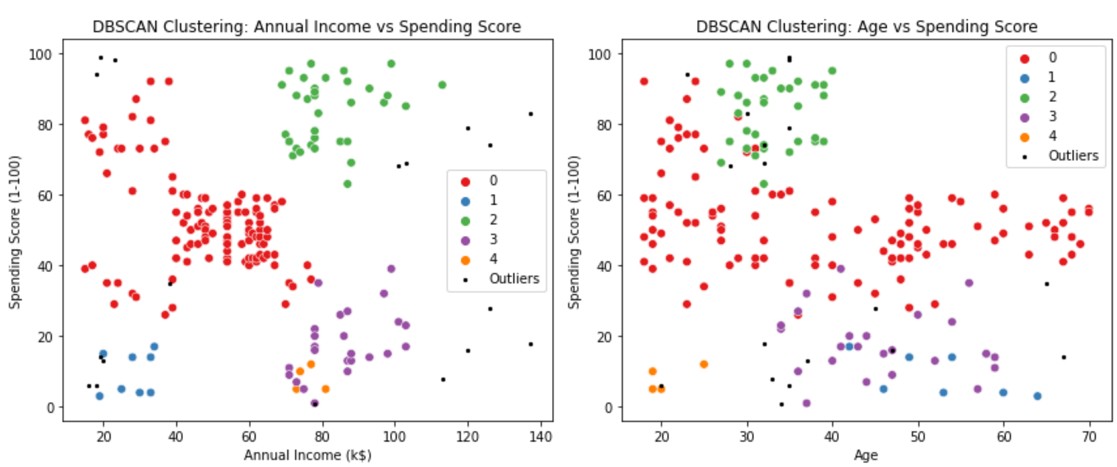
```
DBSCAN_clust_sizes = DBSCAN_clustered.groupby('Cluster').size().to_frame()
DBSCAN_clust_sizes.columns = ["DBSCAN_size"]
DBSCAN_clust_sizes
```

Out[49]:

	DBSCAN_size
Cluster	
-1	18
0	112
1	8
2	34
3	24
4	4

In [50]:

```
# Extract outliers (points with cluster label -1)
outliers = DBSCAN_clustered[DBSCAN_clustered['Cluster'] == -1]

fig, axes = plt.subplots(1, 2, figsize=(12, 5))

# First subplot: 'Annual Income (k$)' vs 'Spending Score (1-100)' colored
sns.scatterplot(x='Annual Income (k$)', y='Spending Score (1-100)',
                data=DBSCAN_clustered[DBSCAN_clustered['Cluster'] != -1],
                hue='Cluster', ax=axes[0], palette='Set1', legend='full',

# Second subplot: 'Age' vs 'Spending Score (1-100)' colored by clusters (e
sns.scatterplot(x='Age', y='Spending Score (1-100)',
                data=DBSCAN_clustered[DBSCAN_clustered['Cluster'] != -1],
                hue='Cluster', palette='Set1', ax=axes[1], legend='full',

# Plot outliers as small black dots on both subplots
axes[0].scatter(outliers['Annual Income (k$)'], outliers['Spending Score (
                s=5, label='Outliers', color='black', marker='o')
axes[1].scatter(outliers['Age'], outliers['Spending Score (1-100)'],
                s=5, label='Outliers', color='black', marker='o')

# Set legends and font size for legends
axes[0].legend()
axes[1].legend()
plt.setp(axes[0].get_legend().get_texts(), fontsize='10')
plt.setp(axes[1].get_legend().get_texts(), fontsize='10')

# Set titles for subplots
axes[0].set_title('DBSCAN Clustering: Annual Income vs Spending Score')
axes[1].set_title('DBSCAN Clustering: Age vs Spending Score')

# Set labels for axes
axes[0].set_xlabel('Annual Income (k$)')
axes[1].set_xlabel('Age')
axes[0].set_ylabel('Spending Score (1-100)')
axes[1].set_ylabel('Spending Score (1-100)')

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

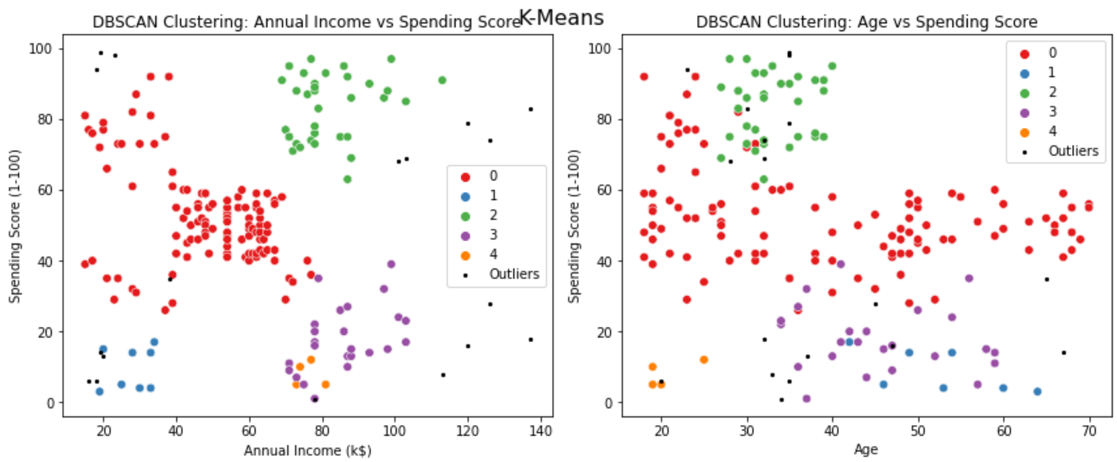


STEP 7: Evaluate

Compare Algorithms

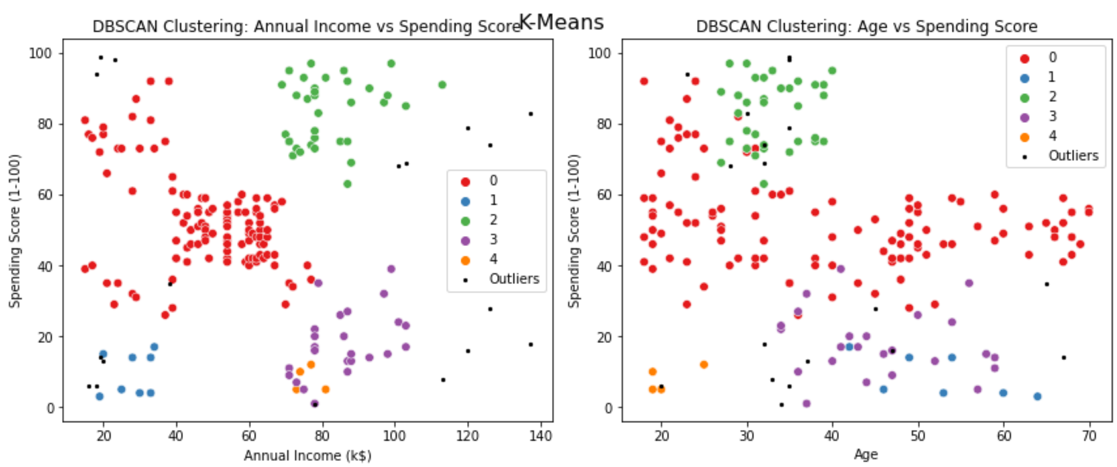
In [52]:
```
# K-Means with K=5
fig.suptitle('K-Means', fontsize=16)
fig
```

Out[52]:



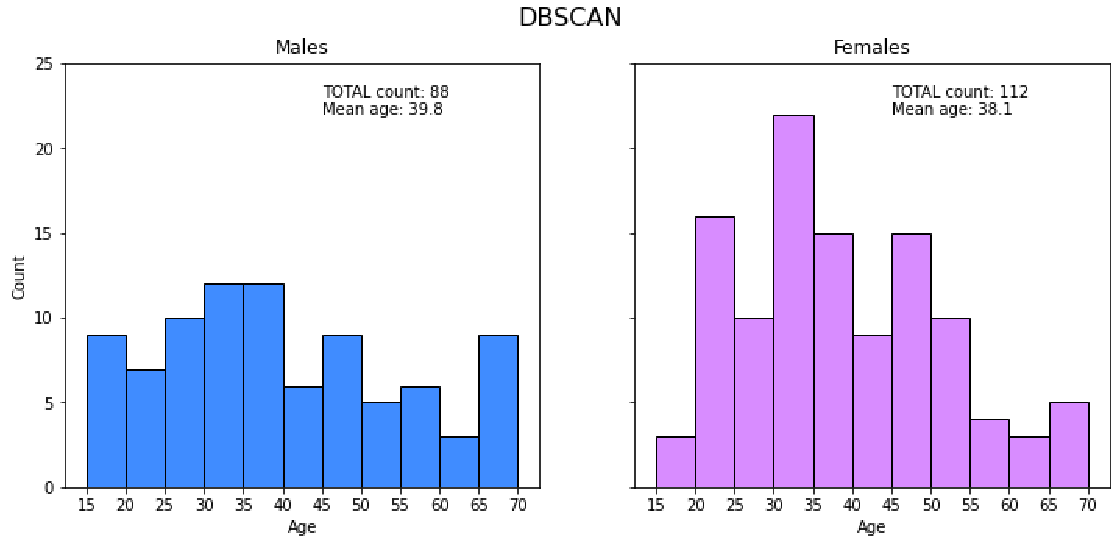
In [54]:
```
# K-Means with K=6
fig.suptitle('K-Means', fontsize=16)
fig
```

Out[54]:



In [56]:
```
# DBSCAN with 5 clusters AND outliers
fig2.suptitle('DBSCAN', fontsize=16)
fig2
```

Out[56]:



In [57]:
```
clusters = pd.concat([KM6_clust_sizes, DBSCAN_clust_sizes],axis=1,
sort=False)
clusters
```

Out[57]:

Cluster	KM_size	DBSCAN_size
0	45.0	112.0
1	38.0	8.0
2	35.0	34.0
3	21.0	24.0
4	39.0	4.0
5	22.0	NaN
-1	NaN	18.0

In [ ]: