In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [3]:
```
import sweetviz # importing sweetviz for auto EDA
from AutoClean import AutoClean # Importing AutoClean library for automated data cleaning
```

In [5]:
```
from sklearn.preprocessing import MinMaxScaler # Importing MinMaxscaler for featre scaling
from sklearn.pipeline import make_pipeline # Importing Make_pipeline for creatin a pipeline of preprocessing st
```

In [9]:
```
from scipy.cluster.hierarchy import linkage,dendrogram # Importing functions for hierarchical clustering
from sklearn.cluster import AgglomerativeClustering # Importing AgglomerativeCluster for hierarchical clusterin
```

In [11]:
```
from sklearn import metrics # Importing metric models from sklearn for evaluating cluster
from clusteval import clusteval # Importing clusteval for cluster evaluation
```

In [13]:
```
from sqlalchemy import create_engine,text # importing Create_engine and text for database interaction
```

In [15]:
```
airline = pd.read_csv(r"C:\Users\DELL\Downloads\Data Set\Data Set (5)\AirTraffic_Passenger_Statistics.csv")
```

In [17]:
```
# credentials to connect the database
user = 'root'
pw = "Venkat#123"
db = 'Airline'
```

In [19]:
```
engine = create_engine(f"mysql+pymysql://{user}:{pw}@Localhost/{db}")
```

In [21]:
```
airline.to_sql('airline_tbl', con = engine, if_exists = 'replace', chunksize =1000, index = False)
```

Out[21]: 15007

In [23]:
```
sql ='select * from airline_tbl;' # to read the data from Mysql database
```

In [25]:
```
df = pd.read_sql_query(text(sql), engine.connect())
```

In [27]:
```
# data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15007 entries, 0 to 15006
Data columns (total 9 columns):
 #   Column                      Non-Null Count  Dtype 
---  ------                      --------------  ----- 
 0   Activity Period             15007 non-null  int64 
 1   Operating Airline           15007 non-null  object
 2   Operating Airline IATA Code  14953 non-null  object
 3   GEO Region                  15007 non-null  object
 4   Terminal                    15007 non-null  object
 5   Boarding Area               15007 non-null  object
 6   Passenger Count             15007 non-null  int64 
 7   Year                        15007 non-null  int64 
 8   Month                       15007 non-null  object
dtypes: int64(3), object(6)
memory usage: 1.0+ MB
```

In [29]:
```
# EDA
# Generating descriptive statistics of the DataFrame df, including count mean,std, min, max,etc...
df.describe()
```

Out[29]:

	Activity Period	Passenger Count	Year
count	15007.000000	15007.000000	15007.000000
mean	201045.073366	29240.521090	2010.385220
std	313.336196	58319.509284	3.137589
min	200507.000000	1.000000	2005.000000
25%	200803.000000	5373.500000	2008.000000
50%	201011.000000	9210.000000	2010.000000
75%	201308.000000	21158.500000	2013.000000
max	201603.000000	659837.000000	2016.000000

In [31]:
```
# Data preprocessing
# Auto EDA
my_report = sweetviz.analyze([df,'df'])
```

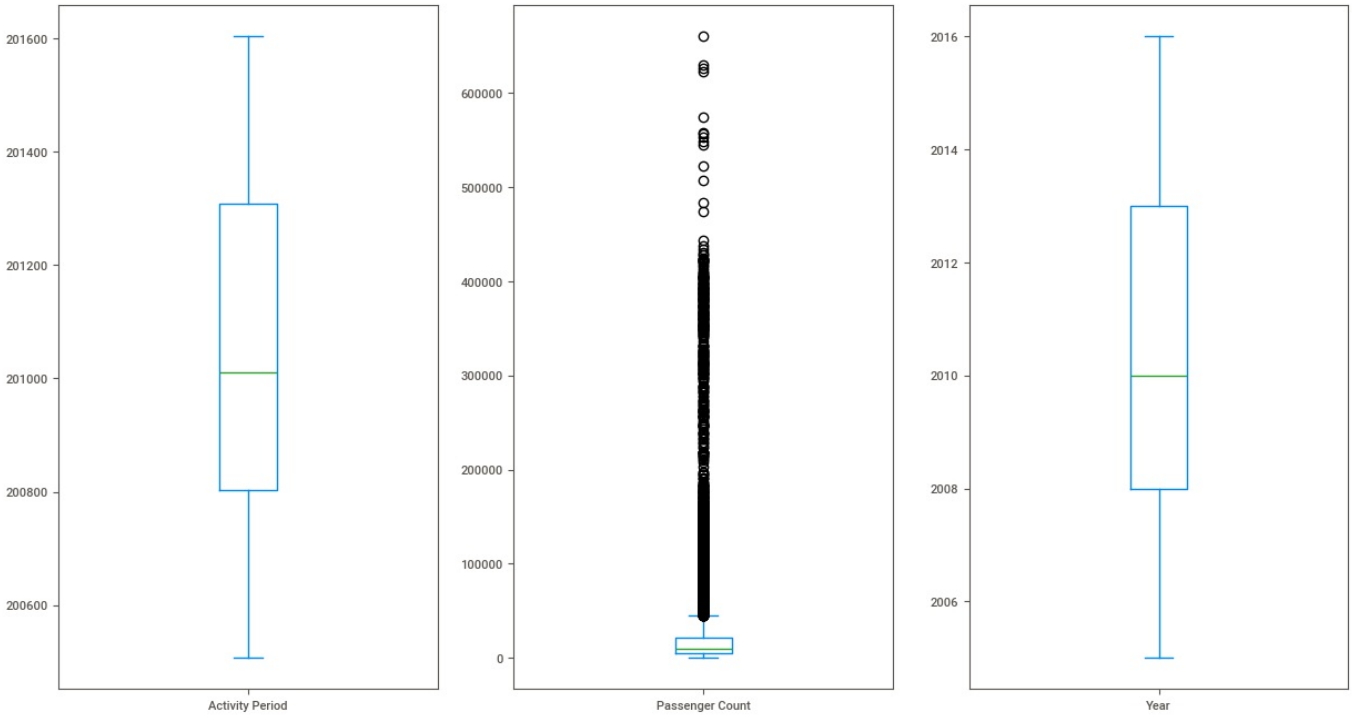
```
                                                |          | [  0%]   00:00 -> (? left)
```

In [32]:
```
my_report.show_html('Report.html')
```

```
Report Report.html was generated! NOTEBOOK/COLAB USERS: the web browser MAY not pop up, regardless, the report I
S saved in your notebook/colab files.
```

In [33]:
```
# generating the boxplot for outliers
df.plot(kind = "box", subplots = True, sharey = False, figsize = (15,8))
```

Out[33]:
```
Activity Period       Axes(0.125,0.11;0.227941x0.77)
Passenger Count    Axes(0.398529,0.11;0.227941x0.77)
Year               Axes(0.672059,0.11;0.227941x0.77)
dtype: object
```



In [34]:
```
clean_pipeline = AutoClean(
    df.iloc[:, :],        # Selecting all rows and columns for cleaning
    mode='manual',         # Setting the cleaning mode to 'manual'
    missing_num='auto',    # Specifying automatic handling of missing numerical values
    outliers='winz',       # Specifying Winsorization method for outlier handling
    encode_categ='auto'  # Specifying automatic encoding of categorical variables
)
```

```
AutoClean process completed in 4.428676 seconds
Logfile saved to: C:\Users\DELL\autoclean.log
```

In [39]:
```
df_clean = clean_pipeline.output
```

In [41]:
```
df_clean.head()
```

Out[41]:

	Activity Period	Operating Airline	Operating Airline IATA Code	GEO Region	Terminal	Boarding Area	Passenger Count	Year	Month	Terminal_International	...	Month_lab	Re
0	200507	ATA Airlines	TZ	US	Terminal 1	B	27271	2005	July	False	...	5	
1	200507	ATA Airlines	TZ	US	Terminal 1	B	29131	2005	July	False	...	5	
2	200507	ATA Airlines	TZ	US	Terminal 1	B	5415	2005	July	False	...	5	
3	200507	Air Canada	AC	Canada	Terminal 1	B	35156	2005	July	False	...	5	
4	200507	Air Canada	AC	Canada	Terminal 1	B	34090	2005	July	False	...	5	

5 rows × 32 columns

In [43]:
```
# droping the categ columns as created the dummy variables
df_clean.drop(['Operating Airline','Operating Airline IATA Code','GEO Region','Terminal','Boarding Area','Month
```

In [45]:
```
df_clean.head()
```

Out[45]:

	Activity Period	Passenger Count	Year	Terminal_International	Terminal_Other	Terminal_Terminal 1	Terminal_Terminal 2	Terminal_Terminal 3	Boa A
0	200507	27271	2005	False	False	True	False	False	
1	200507	29131	2005	False	False	True	False	False	
2	200507	5415	2005	False	False	True	False	False	
3	200507	35156	2005	False	False	True	False	False	
4	200507	34090	2005	False	False	True	False	False	

5 rows × 26 columns

In [47]:
```
# normalization and minmaxscaler to address the scale diff
df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15007 entries, 0 to 15006
Data columns (total 26 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Activity Period                 15007 non-null  Int64
 1   Passenger Count                 15007 non-null  Int64
 2   Year                            15007 non-null  Int64
 3   Terminal_International          15007 non-null  bool
 4   Terminal_Other                  15007 non-null  bool
 5   Terminal_Terminal 1             15007 non-null  bool
 6   Terminal_Terminal 2             15007 non-null  bool
 7   Terminal_Terminal 3             15007 non-null  bool
 8   Boarding Area_A                 15007 non-null  bool
 9   Boarding Area_B                 15007 non-null  bool
 10  Boarding Area_C                 15007 non-null  bool
 11  Boarding Area_D                 15007 non-null  bool
 12  Boarding Area_E                 15007 non-null  bool
 13  Boarding Area_F                 15007 non-null  bool
 14  Boarding Area_G                 15007 non-null  bool
 15  Boarding Area_Other             15007 non-null  bool
 16  Month_lab                       15007 non-null  Int64
 17  GEO Region_Asia                 15007 non-null  bool
 18  GEO Region_Australia / Oceania  15007 non-null  bool
 19  GEO Region_Canada               15007 non-null  bool
 20  GEO Region_Central America      15007 non-null  bool
 21  GEO Region_Europe               15007 non-null  bool
 22  GEO Region_Mexico               15007 non-null  bool
 23  GEO Region_Middle East          15007 non-null  bool
 24  GEO Region_South America        15007 non-null  bool
 25  GEO Region_US                   15007 non-null  bool
dtypes: Int64(4), bool(22)
memory usage: 850.1 KB
```

In [49]:
```
cols = list(df_clean.columns) # creating the list of columns names from cleaned dataFrame
cols
```

Out[49]:
```
['Activity Period',
 'Passenger Count',
 'Year',
 'Terminal_International',
 'Terminal_Other',
 'Terminal_Terminal 1',
 'Terminal_Terminal 2',
 'Terminal_Terminal 3',
 'Boarding Area_A',
 'Boarding Area_B',
 'Boarding Area_C',
 'Boarding Area_D',
 'Boarding Area_E',
 'Boarding Area_F',
 'Boarding Area_G',
 'Boarding Area_Other',
 'Month_lab',
 'GEO Region_Asia',
 'GEO Region_Australia / Oceania',
 'GEO Region_Canada',
 'GEO Region_Central America',
 'GEO Region_Europe',
 'GEO Region_Mexico',
 'GEO Region_Middle East',
 'GEO Region_South America',
 'GEO Region_US']
```

In [51]:
```
# Creating a pipelin using Make_pipeline to apply MinmaxScaler for feature scaling
```

```
pipe1 = make_pipeline(MinMaxScaler())
```

In [53]:
```
# Train the data preprocessing pipeline on data
# Applying the pipeline pipe1 to transformed the cleaned DataFrame
df_pipelined = pd.DataFrame(pipe1.fit_transform(df_clean), columns = cols, index = df_clean.index)
```

In [55]:
```
df_pipelined.head() #Displaying first fewrows of the dataframe
```

Out[55]:

	Activity Period	Passenger Count	Year	Terminal_International	Terminal_Other	Terminal_Terminal 1	Terminal_Terminal 2	Terminal_Terminal 3	Boa A
0	0.0	0.608230	0.0	0.0	0.0	1.0	0.0	0.0	
1	0.0	0.649716	0.0	0.0	0.0	1.0	0.0	0.0	
2	0.0	0.120754	0.0	0.0	0.0	1.0	0.0	0.0	
3	0.0	0.784097	0.0	0.0	0.0	1.0	0.0	0.0	
4	0.0	0.760321	0.0	0.0	0.0	1.0	0.0	0.0	

5 rows × 26 columns

In [57]:
```
df_pipelined.describe()
```

Out[57]:

	Activity Period	Passenger Count	Year	Terminal_International	Terminal_Other	Terminal_Terminal 1	Terminal_Terminal 2	Te
count	15007.000000	15007.000000	15007.000000	15007.000000	15007.000000	15007.000000	15007.000000	
mean	0.490943	0.355247	0.489565	0.612847	0.001799	0.215966	0.021590	
std	0.285891	0.336684	0.285235	0.487115	0.042380	0.411504	0.145345	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.270073	0.119828	0.272727	0.000000	0.000000	0.000000	0.000000	
50%	0.459854	0.205398	0.454545	1.000000	0.000000	0.000000	0.000000	
75%	0.730839	0.471897	0.727273	1.000000	0.000000	0.000000	0.000000	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

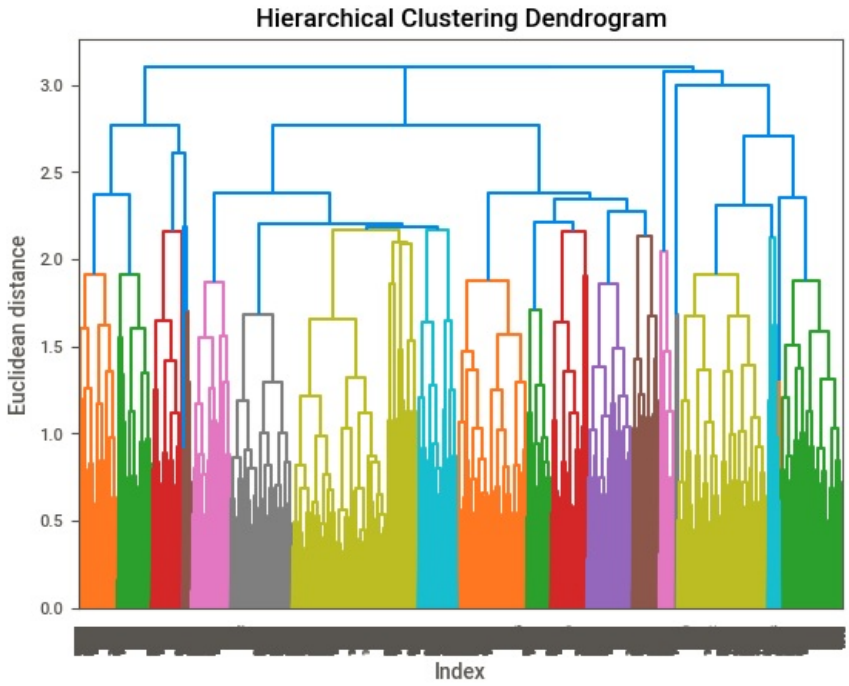
8 rows × 26 columns

In [59]:
```
# ModelBulding
plt.figure(1, figsize = (16,8)) # creating new figure with specified size for the dendrogram plot
```

Out[59]:
```
<Figure size 1600x800 with 0 Axes>
```

```
<Figure size 1600x800 with 0 Axes>
```

In [61]:
```
tree_plot = dendrogram(linkage(df_pipelined, method = 'complete')) # generating a dendrogram plot using hierarc
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel("Index") # setting the label x axis
plt.ylabel("Euclidean distance") # setting the label y axis
plt.show()
```



In [65]:
```
#Applying the agglomerative clustering and grouping data into
hc1 = AgglomerativeClustering(n_clusters = 7, metric = 'euclidean', linkage = 'complete')
```

In [67]:
```
# fitting agglomerative clustering model to the data and predicting the clster labels for each sample
y_hc1 = hc1.fit_predict(df_pipelined)
```

In [69]:
```
# displaying the cluster labels assiged by the agglomerativeclustering
y_hc1
```

Out[69]:
```
array([0, 0, 0, ..., 3, 6, 6], dtype=int64)
```

In [71]:
```
# Accessing the cluster labels directly from the Agglomerativeclustering
hc1.labels_
```

Out[71]:
```
array([0, 0, 0, ..., 3, 6, 6], dtype=int64)
```

In [73]:
```
# Converting the cluster labels into pandas series for further analysis
cluster_labels = pd.Series(hc1.labels_)
```

In [75]:
```
# combine the labels obtained with the data
# Concatenating the cluster labels with cleaned DataFrame (df_clean) along the column axis
df_clust = pd.concat([cluster_labels,df_clean],axis = 1)
```

In [77]:
```
df_clust.head()
```

Out[77]:

	0	Activity Period	Passenger Count	Year	Terminal_International	Terminal_Other	Terminal_Terminal 1	Terminal_Terminal 2	Terminal_Terminal 3
0	0	200507	27271	2005	False	False	True	False	False
1	0	200507	29131	2005	False	False	True	False	False
2	0	200507	5415	2005	False	False	True	False	False
3	0	200507	35156	2005	False	False	True	False	False
4	0	200507	34090	2005	False	False	True	False	False

5 rows × 27 columns

In [79]:
```
# Displaying the column names of the DataFrame df_clust
df_clust.columns
```

Out[79]:
```
Index([                            0,               'Activity Period',
                   'Passenger Count',                          'Year',
           'Terminal_International',                'Terminal_Other',
              'Terminal_Terminal 1',            'Terminal_Terminal 2',
              'Terminal_Terminal 3',                'Boarding Area_A',
                  'Boarding Area_B',                'Boarding Area_C',
                  'Boarding Area_D',                'Boarding Area_E',
                  'Boarding Area_F',                'Boarding Area_G',
             'Boarding Area_Other',                     'Month_lab',
                  'GEO Region_Asia', 'GEO Region_Australia / Oceania',
                'GEO Region_Canada',     'GEO Region_Central America',
                'GEO Region_Europe',               'GEO Region_Mexico',
           'GEO Region_Middle East',       'GEO Region_South America',
                     'GEO Region_US'],
      dtype='object')
```

In [84]:
```
# renaming the first column name ( contaiing cluster labels) to cluster for better clarit
df_clust = df_clust.rename(columns={0: 'cluster'})
```

In [86]:
```
df_clust.head()
```

Out[86]:

	cluster	Activity Period	Passenger Count	Year	Terminal_International	Terminal_Other	Terminal_Terminal 1	Terminal_Terminal 2	Terminal_Termi
0	0	200507	27271	2005	False	False	True	False	Fa
1	0	200507	29131	2005	False	False	True	False	Fa
2	0	200507	5415	2005	False	False	True	False	Fa
3	0	200507	35156	2005	False	False	True	False	Fa
4	0	200507	34090	2005	False	False	True	False	Fa

5 rows × 27 columns

In [90]:
```
# cluster Evaluation
metrics.silhouette_score(df_pipelined,cluster_labels)
```

Out[90]:
```
0.34299621489496235
```

In [92]:
```
ce = clusteval(evaluate = 'silhouette')
```

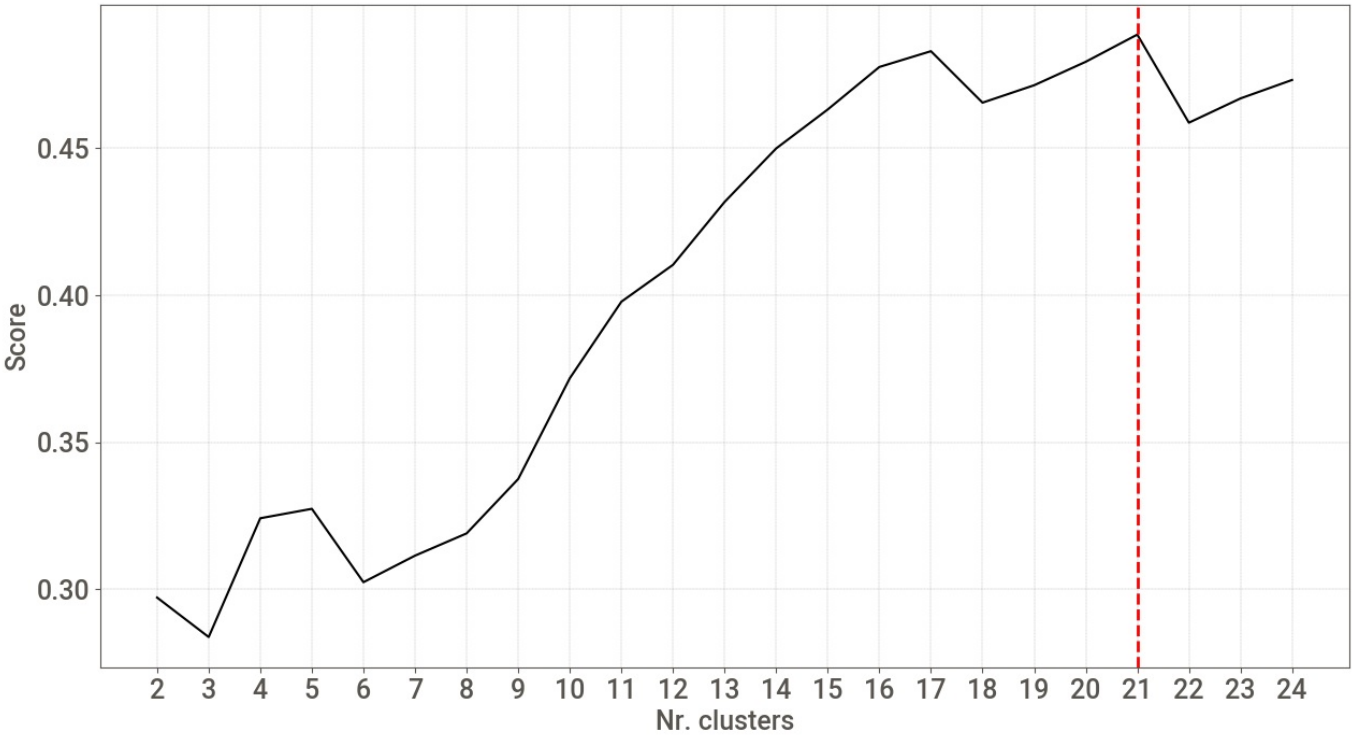
In [94]:
```
# converting the datafram of preprocessed and scaled data ( df_pipelined) into array
df_array = np.array(df_pipelined)
```

In [96]:
```
# fitting the clusteval instance to the data array to compute silhouette score for diff numbers of clusters
ce.fit(df_array)
```

```
[clusteval] >INFO> Saving data in memory.
[clusteval] >INFO> Fit with method=[agglomerative], metric=[euclidean], linkage=[ward]
[clusteval] >INFO> Evaluate using silhouette.
[clusteval] >INFO: 100%|██████████| 23/23 [01:59<00:00,  5.19s/it]
[clusteval] >INFO> Compute dendrogram threshold.
[clusteval] >INFO> Optimal number clusters detected: [21].
[clusteval] >INFO> Fin.
```

Out[96]:
```
{'evaluate': 'silhouette',
 'score':     cluster_threshold  clusters     score
 0                    2         2  0.297200
 1                    3         3  0.283777
 2                    4         4  0.324158
 3                    5         5  0.327366
 4                    6         6  0.302418
 5                    7         7  0.311451
 6                    8         8  0.319013
 7                    9         9  0.337509
 8                   10        10  0.371765
 9                   11        11  0.397781
 10                  12        12  0.410311
 11                  13        13  0.431681
 12                  14        14  0.449874
 13                  15        15  0.463122
 14                  16        16  0.477564
 15                  17        17  0.482929
 16                  18        18  0.465434
 17                  19        19  0.471349
 18                  20        20  0.479326
 19                  21        21  0.488575
 20                  22        22  0.458619
 21                  23        23  0.466873
 22                  24        24  0.473149,
'labx': array([ 7,  7,  8, ...,  2, 16, 16]),
'fig': {'silscores': array([0.29719981, 0.28377652, 0.32415788, 0.32736638, 0.30241848,
        0.31145055, 0.31901291, 0.33750919, 0.37176487, 0.39778062,
        0.41031082, 0.43168112, 0.44987373, 0.46312152, 0.47756441,
        0.48292891, 0.46543358, 0.47134928, 0.4793258 , 0.48857547,
        0.4586187 , 0.46687346, 0.47314867]),
 'sillclust': array([ 2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
        19, 20, 21, 22, 23, 24]),
 'clustcutt': array([ 2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
        19, 20, 21, 22, 23, 24])},
'max_d': 21.52129683570493,
'max_d_lower': 21.19014760189956,
'max_d_upper': 21.852446069510297}
```

In [98]:
```
ce.plot()
```



Out[98]:
```
(<Figure size 1500x800 with 1 Axes>,
 <Axes: xlabel='Nr. clusters', ylabel='Score'>)
```

In [126…
```
hc2_clust = AgglomerativeClustering(n_clusters = 4, metric = 'euclidean', linkage= 'ward')
```

In [128…
```
y_hc_2clust = hc2_clust.fit_predict(df_pipelined)
```

In [129…
```
hc2_clust.labels_
```

Out[129…
```
array([0, 0, 0, ..., 3, 1, 1], dtype=int64)
```

In [130…
```
cluster_label2 = pd.Series(hc2_clust.labels_)
```

In [131…
```
# concatenating the clusterlabels with clean DataFrame
df_2clust =pd.concat([cluster_label2,df_clean],axis=1)
```

In [132…
```
# renaming the cluster column name
df_2clust = df_2clust.rename(columns={0: 'cluster'})
```

In [133…
```
df_2clust.head()
```

Out[133…

	cluster	Activity Period	Passenger Count	Year	Terminal_International	Terminal_Other	Terminal_Terminal 1	Terminal_Terminal 2	Terminal_Termi
0	0	200507	27271	2005	False	False	True	False	Fa
1	0	200507	29131	2005	False	False	True	False	Fa
2	0	200507	5415	2005	False	False	True	False	Fa
3	0	200507	35156	2005	False	False	True	False	Fa
4	0	200507	34090	2005	False	False	True	False	Fa

5 rows × 27 columns

In [134…
```
metrics.silhouette_score(df_pipelined,cluster_label2)
```

Out[134… 0.32415788357766323

In [ ]: