In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import normalize, StandardScaler, RobustScaler
from sklearn.decomposition import PCA
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, silhouette_samples

import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
# Configuración de números flotantes a 3 decimales
pd.set_option('display.float_format','{:.3f}'.format)

# Estilo de visualización
sns.set_style("darkgrid", {"grid.color": ".6", "grid.linestyle": ":"})
```

CARGA Y EXPLORACIÓN DE DATOS

In [3]:

```
# Carga de datos
data = pd.read_csv('Superstore.csv', date_parser='Order Date')
data.sample(5)
```

Out[3]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	
5927	5928	CA-2014-114314	10/11/2014	10/15/2014	Standard Class	DB-13555	Dorothy Badders	Corporate	United States	Faye
5033	5034	CA-2016-155166	12/26/2016	1/2/2017	Standard Class	BB-11545	Brenda Bowman	Corporate	United States	Vi
4934	4935	CA-2015-106978	9/28/2015	10/4/2015	Standard Class	ZC-21910	Zuschuss Carroll	Consumer	United States	
9982	9983	US-2016-157728	9/22/2016	9/28/2016	Standard Class	RC-19960	Ryan Crowe	Consumer	United States	

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	
176	177	US-2017-152366	4/21/2017	4/25/2017	Second Class	SJ-20500	Shirley Jackson	Consumer	United States	H

5 rows × 21 columns

In [4]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Row ID         9994 non-null   int64  
 1   Order ID       9994 non-null   object 
 2   Order Date     9994 non-null   object 
 3   Ship Date      9994 non-null   object 
 4   Ship Mode      9994 non-null   object 
 5   Customer ID    9994 non-null   object 
 6   Customer Name  9994 non-null   object 
 7   Segment        9994 non-null   object 
 8   Country        9994 non-null   object 
 9   City           9994 non-null   object 
 10  State          9994 non-null   object 
 11  Postal Code    9994 non-null   int64  
 12  Region         9994 non-null   object 
 13  Product ID     9994 non-null   object 
 14  Category       9994 non-null   object 
 15  Sub-Category   9994 non-null   object 
 16  Product Name   9994 non-null   object 
 17  Sales          9994 non-null   float64
 18  Quantity       9994 non-null   int64  
 19  Discount       9994 non-null   float64
 20  Profit         9994 non-null   float64
dtypes: float64(3), int64(3), object(15)
memory usage: 1.6+ MB
```

In [5]:

```
data.describe().T
```

Out[5]:

	count	mean	std	min	25%	50%	75%	max
Row ID	9994.000	4997.500	2885.164	1.000	2499.250	4997.500	7495.750	9994.000
Postal Code	9994.000	55190.379	32063.693	1040.000	23223.000	56430.500	90008.000	99301.000
Sales	9994.000	229.858	623.245	0.444	17.280	54.490	209.940	22638.480
Quantity	9994.000	3.790	2.225	1.000	2.000	3.000	5.000	14.000
Discount	9994.000	0.156	0.206	0.000	0.000	0.200	0.200	0.800
Profit	9994.000	28.657	234.260	-6599.978	1.729	8.666	29.364	8399.976

In [6]:
```
data.columns
```

Out[6]:
```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Country', 'City', 'State',
       'Postal Code', 'Region', 'Product ID', 'Category', 'Sub-Category',
       'Product Name', 'Sales', 'Quantity', 'Discount', 'Profit'],
      dtype='object')
```

TRANSFORMACIONES DE DATOS

In [7]:
```
# Datos de los corporativos
corporate_data = data[data.Segment=='Corporate']

# Transformación de tipo de dato
corporate_data['Order Date'] = pd.to_datetime(corporate_data['Order Date'])

# Seleccinar los datos del último año - 2017
corporate_data = corporate_data[corporate_data['Order Date'] >= '2017']

# Calculo de variable 'Discount'
corporate_data['Discount'] = corporate_data['Discount']  * corporate_data['Sales
```

In [8]:
```
# Datos agrupados
corporate_data = corporate_data.groupby(['Customer ID','Order ID','Order Date'])
corporate_data.drop(['Row ID','Postal Code'], axis=1, inplace=True)
corporate_data = corporate_data.reset_index([0,1,2])

corporate_data.sample(3)
```

Out[8]:

	Customer ID	Order ID	Order Date	Sales	Quantity	Discount	Profit
482	VG-21805	CA-2017-166499	2017-03-19	8.940	3	0.000	2.414
337	LW-16990	CA-2017-162978	2017-05-04	502.474	9	96.973	40.432
64	BO-11350	CA-2017-144883	2017-08-15	50.400	8	0.000	23.184

In [9]:
```
# CREATING RFM FEATURES
# ==============================================================================

snapshot_date = corporate_data['Order Date'].max() + dt.timedelta(days=1)

# Aggregate data on a customer level
corporateData = corporate_data.groupby('Customer ID').agg({'Order Date': lambda
                                          'Order ID': 'count',
                                          'Sales': 'sum'})

# Rename columns
corporateData.rename(columns={'Order Date': 'Recency',
                        'Order ID': 'Frequency',
                        'Sales': 'MonetaryValue'}, inplace=True)

corporateData.sample(5)
```

Out[9]:

Recency	Frequency	MonetaryValue

Customer ID	Recency	Frequency	MonetaryValue
Customer ID			
RS-19765	37	5	2128.808
MH-17785	99	1	9.248
GD-14590	6	2	1103.356
GM-14695	118	1	1322.352
KH-16690	3	2	845.800

VARIABLES SIN ESCALADO

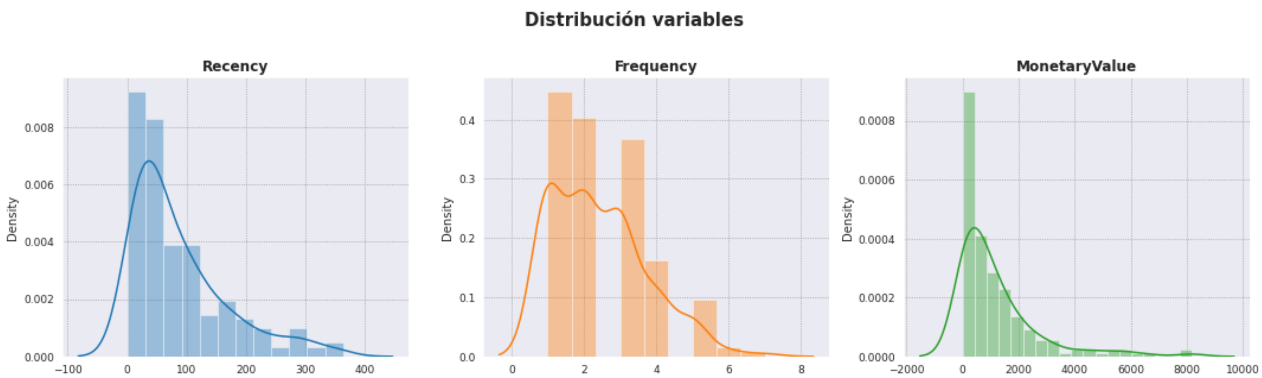
In [10]:

```
# DISTRIBUCIONES DE VARIABLES SIN ESCALAR
# ==============================================================================

fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(15, 4))
axes = axes.flat

for i, feature in enumerate(corporateData.columns):
    sns.distplot(
        x       = corporateData[feature],
        color   = (list(plt.rcParams['axes.prop_cycle'])*2)[i]["color"],
        ax      = axes[i]
    )
    axes[i].set_title(feature, fontsize = 12, fontweight = "bold")
    axes[i].tick_params(labelsize = 9)
    axes[i].set_xlabel("")


fig.tight_layout()
fig.suptitle('Distribución variables', y=1.1,fontsize = 15, fontweight = "bold")
```



In [11]:

```
corporateData.describe().T
```

Out[11]:

	count	mean	std	min	25%	50%	75%	max
Recency	204.000	87.098	80.515	1.000	29.000	58.000	119.250	363.000
Frequency	204.000	2.417	1.290	1.000	1.000	2.000	3.000	7.000
MonetaryValue	204.000	1185.529	1468.036	1.188	217.842	697.842	1518.147	8167.420

PREPROCESAMINETO DE VARIABLES

In [12]:

```
# Transformación logarítmica
corporateData_log = np.log(corporateData)

# x - X.mean / X.std
scaler = StandardScaler()

#
scaler.fit(corporateData_log)

# Escalado de variables
scaled_features = scaler.transform(corporateData_log)

# Variables escaladas - DataFrame
scaled_features = pd.DataFrame(scaled_features, index=corporateData.index, colum
scaled_features.sample(5)
```

Out[12]:

Customer ID	Recency	Frequency	MonetaryValue
NH-18610	1.106	0.656	0.938
BF-11020	0.311	1.175	0.439
KB-16585	1.273	-0.076	0.089
JM-15655	-1.500	-0.076	-0.155
CV-12805	-1.847	2.185	0.613

VARIABLES ESCALADAS

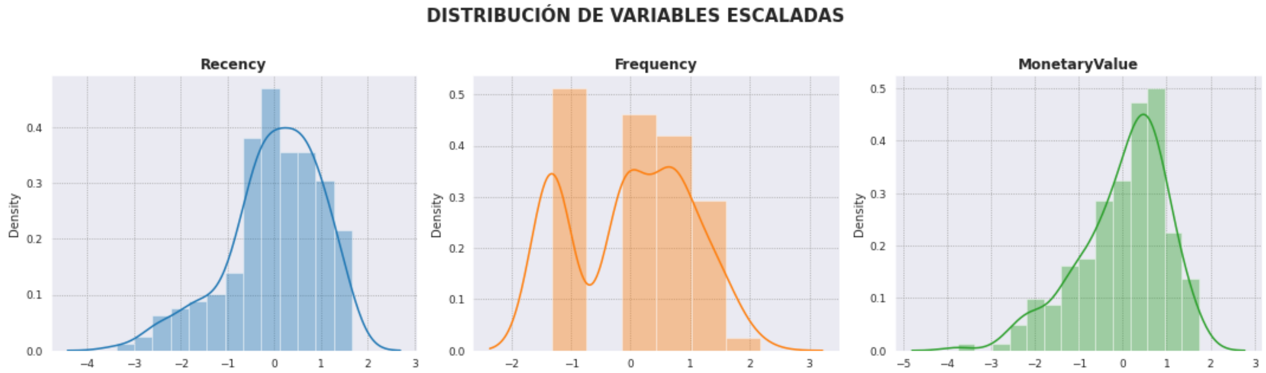
In [13]:

```
# DISTRIBUCIONES DE VARIABLES ESCALADAS
# ==============================================================================

fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(15, 4))
axes = axes.flat


for i, feature in enumerate(scaled_features.columns):
    sns.distplot(
        x       = scaled_features[feature],
        color   = (list(plt.rcParams['axes.prop_cycle'])*2)[i]["color"],
        ax      = axes[i]
    )
    axes[i].set_title(feature, fontsize = 12, fontweight = "bold")
    axes[i].tick_params(labelsize = 9)
    axes[i].set_xlabel("")


fig.tight_layout()
plt.subplots_adjust(top = 0.9)
fig.suptitle('DISTRIBUCIÓN DE VARIABLES ESCALADAS', y=1.1, fontsize = 15, fontwe
```



In [14]:
```
scaled_features.describe().T
```

Out[14]:

	count	mean	std	min	25%	50%	75%	max
Recency	204.000	0.000	1.002	-3.378	-0.500	0.092	0.708	1.659
Frequency	204.000	-0.000	1.002	-1.327	-1.327	-0.076	0.656	2.185
MonetaryValue	204.000	-0.000	1.002	-3.757	-0.515	0.210	0.693	1.740

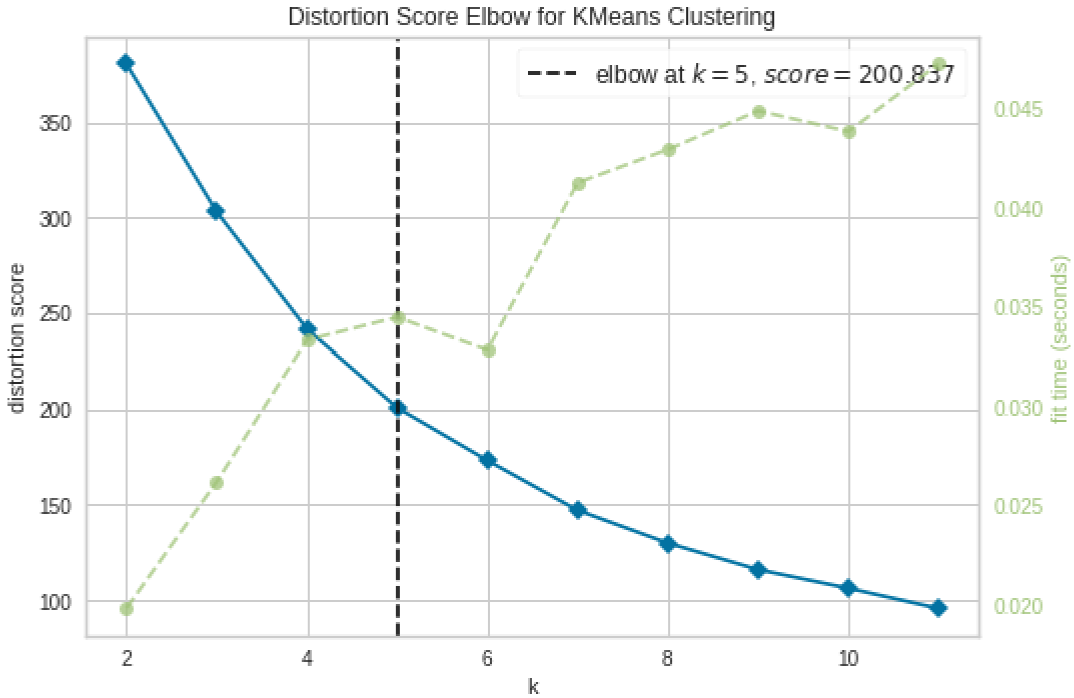
KMeans

MÉTODO DE CODO

In [15]:
```
# MÉTODO DE CODO
# ==============================================================================
from yellowbrick.cluster import SilhouetteVisualizer
from yellowbrick.cluster import KElbowVisualizer


model = KMeans()
visualizer = KElbowVisualizer(model, k=(2,12))

visualizer.fit(scaled_features)        # Fit the data to the visualizer
visualizer.show()        # Finalize and render the figure
plt.show()
```



ENTRENAMIENTO DEL MODELO - KMEANS

In [16]:

```
# Modelo - KMeans
kmeans_model = KMeans(n_clusters=5)

# Modelo entrenado
kmeans_model.fit_transform(scaled_features)

corporateData['ClusterLabels_k5'] = kmeans_model.labels_

kmeansData = corporateData.groupby('ClusterLabels_k5').agg({'Recency':'mean',
                                                            'Frequency':'mean',
                                                            'MonetaryValue':['mean','count']})

kmeansData
```

Out[16]:

	Recency	Frequency	MonetaryValue	
	mean	mean	mean	count
ClusterLabels_k5				
0	59.193	3.719	2425.641	57
1	97.536	2.357	585.789	56
2	125.303	1.273	47.393	33
3	6.500	3.042	1306.377	24
4	136.500	1.000	1113.678	34

Hierarchical Clustering

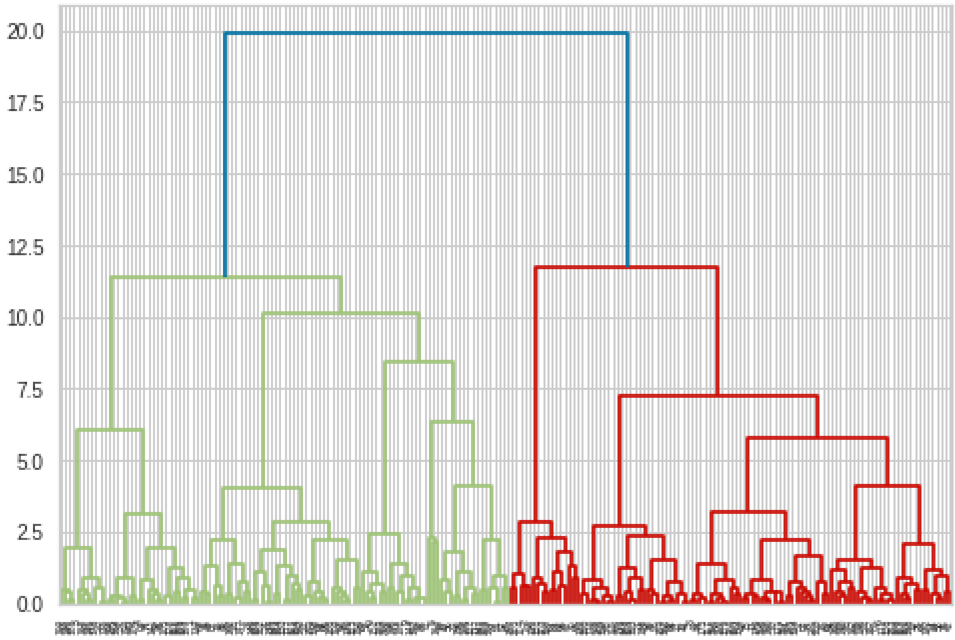
In [17]:

```
# Modelo de segmentación jerárquica
hierarchy_model = linkage(scaled_features, method='ward')
```

In [18]:

```
# Dendrograma
drendogram = dendrogram(hierarchy_model)

# Visualización del dendrograma
plt.show()
```



In [19]:

```
# Punto de corte
cluster_labels = fcluster(hierarchy_model, 9, criterion='distance')

hierarchicalData = corporateData.copy()

# Asignación de las etiquetas de clusters a los datos
hierarchicalData['ClusterLabels_H'] = cluster_labels

hierarchicalData = hierarchicalData.groupby('ClusterLabels_H').agg({'Recency':'m
                                                    'Frequency':'mean',
                                                    'MonetaryValue':['mean','cour



hierarchicalData
```

Out[19]:

	Recency	Frequency	MonetaryValue	
	mean	mean	mean	count
ClusterLabels_H				
1	193.125	1.000	157.530	32
2	67.649	2.405	183.645	37
3	57.765	1.147	1172.732	34
4	5.250	3.875	1893.507	16
5	82.788	3.188	1880.506	85