Análisis de Datos y Clasificación ABC en una Cadena de Suministros

Realizaremos un análisis exhaustivo de datos ficticios correspondientes a una cadena de suministros. Este proceso incluirá la visualización de datos para identificar patrones y tendencias, así como la clasificación ABC tanto de productos como de clientes. La visualización facilitará la comprensión de la distribución y el rendimiento, mientras que la clasificación ABC ayudará a categorizar y gestionar los elementos clave en la cadena de suministros.

In [1]:
```
import pandas as pd
from datetime import datetime
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

In [2]:
```
#Lectura de los datos "sales.csv"
df = pd.read_csv('sales.csv')
df.head()
```

Out[2]:

	ORDER_NO	DATE	LINE	CUSTOMER_NO	ITEM	NS_ORDER	NS_SHIP
0	539758	Tuesday, January 3, 2017	1	1268538.0	111501	70.0	70.0
1	539761	Tuesday, January 3, 2017	1	1254786.0	1029071	10.0	10.0
2	539761	Tuesday, January 3, 2017	2	1254786.0	1031541	10.0	10.0
3	539761	Tuesday, January 3, 2017	3	1254786.0	1040827	5.0	5.0
4	539761	Tuesday, January 3, 2017	4	1254786.0	10108111	10.0	10.0

- ORDER_NO: Identificador de la orden
- DATE: Fecha de la orden
- LINE: Línea de embarque
- CUSTOMER_NO: Identificador del cliente
- ITEM: Identificador del SKU
- NS_ORDER: Cantidad de ítems ordenados
- NS_SHIPED: Cantidad de ítems despachados

In [3]:
```
#Convertir la fecha de formato string a formato date
def string_to_date(fecha_str):
    return datetime.strptime(fecha_str[:-5], '%A, %B %d, %Y')

df['DATE'] = df['DATE'].apply(string_to_date)
```

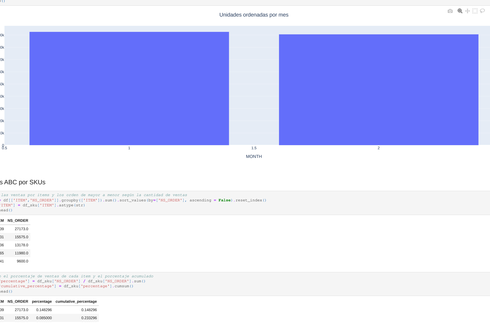
Volumen diario de ordenes

In [4]:
```
#Volumen diario
df_daily_volume = df[['DATE','NS_ORDER']].groupby(['DATE']).sum().reset_index()
df_daily_volume.head()
```

Out[4]:

	DATE	NS_ORDER
0	2017-01-03	4630.0
1	2017-01-04	2890.0
2	2017-01-05	1213.0
3	2017-01-06	3070.0
4	2017-01-07	449.0

In [5]:
```
#Realizo un grafico de lineas con plotly
fig = px.line(df_daily_volume, x='DATE', y='NS_ORDER')
fig.update_layout(
    title_text = "Volumen de unidades ordenadas por dia",
    title_x = 0.5
)
fig.show()
```



Volumen mensual de unidades ordenadas

In [6]:
```
#Volumen de volumen mensual de unidades ordenadas
df_monthly_volume = df[['DATE','NS_ORDER']].groupby(['DATE']).sum().reset_index()
df_monthly_volume['DATE'] = pd.to_datetime(df_monthly_volume['DATE'])
df_monthly_volume['MONTH'] = df_monthly_volume['DATE'].dt.month
df_monthly_summary = df_monthly_volume.groupby('MONTH')['NS_ORDER'].sum().reset_index()
```

In [7]:
```
# Crear gráfico de barras
fig = px.bar(df_monthly_summary, x='MONTH', y='NS_ORDER', )
fig.update_layout(
    title_text='Unidades ordenadas por mes',
    title_x = 0.5
)
fig.show()
```

Análisis ABC por SKUs

In [8]:
```
#Agrupo las ventas por items y los ordeno de mayor a menor segùn la cantidad de ventas
df_abc = df[['ITEM','NS_ORDER']].groupby(['ITEM']).sum().sort_values(by='NS_ORDER', ascending = False).reset_index()
df_abc['ITEM'] = df_abc['ITEM'].astype(str)
df_abc.head()
```

Out[8]:

	ITEM	NS_ORDER
0	10000739	271713
1	111501	15575.0
2	1041106	15178.0
3	1040795	11980.0
4	110441	9600.0

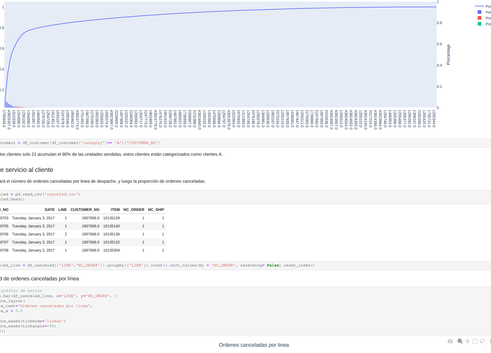
In [9]:
```
#Calculo el porcentaje de ventas de cada item y el porcentaje acumulado
df_abc['percentage'] = df_abc['NS_ORDER'] / df_abc['NS_ORDER'].sum()
df_abc['cumulative_percentage'] = df_abc['percentage'].cumsum()
df_abc.head()
```

Out[9]:

	ITEM	NS_ORDER	percentage	cumulative_percentage
0	10000739	271713	0.146296	0.146296
1	111501	15575.0	0.008500	0.231096
2	1041106	15178.0	0.071918	0.302515
3	1040795	11980.0	0.065381	0.370996
4	110441	9600.0	0.052392	0.423907

In [10]:
```
#Categorizo los ABC
def categorize_abc(percentage):
    if cum_percentage <= 0.35:
        category = 'A'
    elif cum_percentage > 0.5 and cum_percentage < 0.95:
        category = 'B'
    else:
        category = 'C'
    return category

df_abc['category'] = df_abc['cumulative_percentage'].apply(abc)
```

In [11]:
```
fig = make_subplots(specs=[[{"secondary_y": True}]])

# Agregar gráfico de líneas
fig.add_trace(
    go.Scatter(x=df_abc['ITEM'], y=df_abc['cumulative_percentage'], mode='lines', name='Porcentaje acumulado'),
    secondary_y=False,
)

# Agregar gráfico de barras con colores distintos por categoría
colors = px.colors.qualitative.Plotly

categories = df_abc['category'].unique()
for i, category in enumerate(categories):
    df_category = df_abc[df_abc['category'] == category]
    fig.add_trace(
        go.Bar(x=df_category['ITEM'],
               y=df_category['percentage'],
               name='Porcentaje {category}',
               marker_color=colors[i % len(colors)]  # Ciclo los colores si hay más categorías que colores
        ),
        secondary_y=True
    )

# Configurar el rango del eje Porcentaje
fig.update_layout(
    yaxis=dict(
        range=[0, 1],
        title_text="Porcentaje",
        secondary_y=True
    )
)

fig.update_layout(
    title_text="Clasificación ABC por SKU",
    title_x=0.5
)

fig.show()
```



Los SKU categoría "A" explican el 80% de las unidades vendidas, los SKU categoría "B" explican el 15% de las unidades vendidas y los SKU categoría "C" explican el 5% de unidades vendidas.

Análisis ABC por Cliente

In [12]:
```
#Agrupo las ventas por cliente y los ordeno de mayor a menor segùn la cantidad de ventas
df_customer = df[['CUSTOMER_NO','NS_ORDER']].groupby(['CUSTOMER_NO']).sum().sort_values(by='NS_ORDER', ascending = False).reset_index()
df_customer['CUSTOMER_NO'] = df_customer['CUSTOMER_NO'].astype(str)
df_customer.head()
```

Out[12]:

	CUSTOMER_NO	NS_ORDER
0	1790849.0	14588.0
1	1254223.0	11629.0
2	1255648.0	11567.0
3	1740842.0	10687.0
4	1254786.0	9584.0

In [13]:
```
#Calculo el porcentaje de ventas de cada cliente y el porcentaje acumulado
df_customer['percentage'] = df_customer['NS_ORDER'] / df_customer['NS_ORDER'].sum()
df_customer['cumulative_percentage'] = df_customer['percentage'].cumsum()
df_customer.shape
```

Out[13]:
```
(7784, 4)
```

In [14]:
```
#Categorizo a los clientes
df_customer['category'] = df_abc['cumulative_percentage'].apply(abc)
```

In [15]:
```
fig = make_subplots(specs=[[{"secondary_y": True}]])

# Agregar gráfico de líneas
fig.add_trace(
    go.Scatter(x=df_customer['CUSTOMER_NO'], y=df_customer['cumulative_percentage'], mode='lines', name='Porcentaje acumulado'),
    secondary_y=False,
)

# Agregar gráfico de barras con colores distintos por categoría
colors = px.colors.qualitative.Plotly

categories = df_customer['category'].unique()
for i, category in enumerate(categories):
    df_category = df_customer[df_customer['category'] == category]
    fig.add_trace(
        go.Bar(x=df_category['CUSTOMER_NO'],
               y=df_category['percentage'],
               name='Porcentaje {category}',
               marker_color=colors[i % len(colors)]
        ),
        secondary_y=True
    )

# Configurar el rango del eje Porcentaje
fig.update_layout(
    yaxis=dict(
        range=[0, 1],
        title_text="Porcentaje",
        secondary_y=True
    )
)

fig.update_layout(
    title_text="Clasificación ABC por Cliente",
    title_x=0.5
)

fig.show()
```

In [16]:
```
df_A_customers = df_customer[df_customer['category']== 'A']['CUSTOMER_NO']
```

De todos los clientes solo 21 acumulan el 80% de las unidades vendidas, estos clientes están categorizados como clientes A.

Nivel de servicio al cliente

Se analizará el número de ordenes canceladas por linea de despacho, y luego la proporción de ordenes canceladas.

In [17]:
```
df_canceled = pd.read_csv('canceled.csv')
df_canceled.head()
```

Out[17]:

	ORDER_NO	DATE	LINE	CUSTOMER_NO	ITEM	NC_ORDER	NC_SHIP
0	539703	Tuesday, January 3, 2017	1	1857566.0	10139199	1	1
1	539705	Tuesday, January 3, 2017	1	1857566.0	10135140	1	1
2	539706	Tuesday, January 3, 2017	2	1857566.0	10139199	1	1
3	539707	Tuesday, January 3, 2017	1	1857566.0	10139152	1	1
4	539708	Tuesday, January 3, 2017	1	1857566.0	10139359	1	1

In [18]:
```
df_canceled_line = df_canceled['LINE','NC_ORDER'].groupby(['LINE']).count().sort_values(by = 'NC_ORDER', ascending= False).reset_index()
```

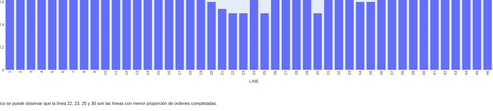
Cantidad de ordenes canceladas por linea

In [19]:
```
# Crear gráfico de barras
fig = px.bar(df_canceled_line, x='LINE', y='NC_ORDER', )
fig.update_layout(
    title_text='Ordenes canceladas por linea',
    title_x = 0.5
)
fig.update_xaxes(tickmode='linear')
fig.update_xaxes(tickangle=-90)
fig.show()
```

Proporción de ordenes canceladas por linea

In [20]:
```
df_accepted_line = df[['LINE','NS_ORDER']].groupby(['LINE']).count().sort_values(by = 'NS_ORDER', ascending= False).reset_index()
df_percentage_accepted = df_accepted_line
df_percentage_accepted['percentage_accepted'] = df_accepted_line['NS_ORDER']/(df_accepted_line['NS_ORDER'] + df_canceled_line['NC_ORDER'])
df_percentage_accepted['percentage_accepted'] = df_percentage_accepted['percentage_accepted'].fillna(1)
df_percentage_accepted.sort_values(by = 'percentage_accepted', ascending= False, inplace= True)
```

In [21]:
```
# Crear gráfico de barras
fig = px.bar(df_percentage_accepted, x='LINE', y='percentage_accepted', )
fig.update_layout(
    title_text='Proporción de ordenes aceptadas por linea',
    title_x = 0.5
)
fig.update_xaxes(tickmode='linear')
fig.update_xaxes(tickangle=-90)
fig.show()
```



En el gráfico se puede observar que la línea 22, 23, 25 y 30 son las líneas con menor proporción de ordenes completadas.