Importación de librerías

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

from scipy import stats
from sklearn.metrics import mean_squared_error, r2_score, classification_report, confusion_ma

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge, Lasso, LogisticRegression
from sklearn.preprocessing import StandardScaler
```

Entendimiento contextual del proyecto a desarrollar

La empresa: RETAIL PLUS

RetailPlus, una cadena de tiendas desenvolviendo dentro del sector minorista peruano. Nació como un modesto distribuidor de abarrotes, la empresa ha sabido adaptarse a las cambiantes dinámicas del mercado, diversificando su oferta y expandiendo su huella geográfica. Hoy, su modelo de negocio se caracteriza por una doble vertiente: la venta mayorista a un portafolio diversificado de clientes institucionales y la venta minorista a través de sus propias sucursales y un creciente canal online.

Principales características de la empresa

- Diversidad en su cartera de clientes, clasificados de la siguiente manera:
  - Tradicional o empresa: pequeñas bodegas y restaurantes.
  - Alimento.
  - Química.
  - Textil.
  - Laboratorio.
  - Distribuidor.

  Cada uno de ellos en un SECTOR CLIENTE y RUBRO DE CLIENTE definido.

- La gestión del inventario y la logística: desde productos perecederos hasta bienes duraderos, con centros de despacho estratégicamente ubicados ("PE20", "PE21", "PE22", entre otros). La relación entre el centro de despacho y la ubicación geográfica del cliente podría revelar patrones de demanda regional o costos logísticos asociados.

- Diversas condiciones de pago. Identificar si ciertas condiciones de pago están correlacionadas con mayores volúmenes de compra u otra correlación existente.

El problema:

- Excesos de Inventario y Quiebres de Stock.
- Decisiones de Precios No Óptimas: No existe una comprensión profunda de la elasticidad de la demanda por tipo de cliente o producto.

- Gestión de Crédito y Cobranza: La extensión de plazos de pago puede generar problemas de liquidez.

Las variables se describen de la siguiente manera:

- FECHA_FACTURA: Fecha en que se emitió la factura para la transacción.
- COD_CLIENTE: Identificador único del cliente que realizó la compra.
- SECTOR_CLIENTE: Sector económico al que pertenece el cliente.
- RUBRO_CLIENTE: Rubro específico del negocio del cliente, más detallado que el sector.
- COD_MATERIAL: Identificador único del material o producto vendido.
- MARCA: Marca del producto vendido.
- REGION: Región geográfica donde se encuentra el cliente o se realizó la entrega.
- PROVINCIA: Provincia donde se encuentra el cliente o se realizó la entrega.
- DISTRITO: Distrito donde se encuentra el cliente o se realizó la entrega.
- CIUDAD: Ciudad donde se encuentra el cliente o se realizó la entrega.
- CENTRO_DESPACHO: Código del centro de despacho desde donde se envió el producto.
- CONDICION_PAGO: Término de pago acordado para la transacción.
- FACTURA: Número de factura único para la transacción.
- UNIDAD_MEDIDA: Unidad de medida utilizada para la cantidad del producto (ej. litros, kilogramos, unidades).
- Cantidad: Cantidad de unidades del producto vendido en esta transacción.
- Monto USD: Monto total de la transacción en dólares estadounidenses

Lectura de archivo

In [3]:
```
df = pd.read_csv("/content/dataset_module2.txt", encoding='latin1', sep=',')
```

EDA

Información general del data set

- Se realizará en análisis general de los datos: descriptivos, identificación de datos faltantes, así como ouliers.

In [4]:
```
#INFORMACION GENERAL DEL TXT.
df.shape
```

Out[4]:
```
(16749, 17)
```

In [5]:
```
# Eliminamos las variables identificadores
df = df.drop(['COD_CLIENTE', 'COD_MATERIAL', 'FACTURA'], axis=1)
df.head()
```

Out[5]:

	FECHA_FACTURA	SECTOR_CLIENTE	RUBRO_CLIENTE	COD_MARCA	REGION	PROVINCIA	DISTRITO
0	2020-01-02	EMPRESA	ALIMENTO	A	Lima	LIMA	LA VICTORIA
1	2020-01-02	EMPRESA	ALIMENTO	B	Lima	LIMA	LA VICTORIA
2	2020-01-02	EMPRESA	QUIMICA	B	Lima	LIMA	SAN ISIDRO
3	2020-01-02	EMPRESA	ALIMENTO	B	Lima	LIMA	ATE
4	2020-01-02	EMPRESA	TEXTIL	B	Lima	LIMA	ATE

In [6]:
```
df.describe(include="all")
```

Out[6]:

	FECHA_FACTURA	SECTOR_CLIENTE	RUBRO_CLIENTE	COD_MARCA	REGION	PROVINCIA	DIST
count	16749	16682	16682	16748	16748	16748	
unique	448	6	24	4	21	52	
top	2020-02-29	TRADICIONAL	DISTRIBUIDOR	C	Lima	LIMA	
freq	461	9059	8456	9288	10185	9335	
mean	NaN	NaN	NaN	NaN	NaN	NaN	
std	NaN	NaN	NaN	NaN	NaN	NaN	
min	NaN	NaN	NaN	NaN	NaN	NaN	
25%	NaN	NaN	NaN	NaN	NaN	NaN	
50%	NaN	NaN	NaN	NaN	NaN	NaN	
75%	NaN	NaN	NaN	NaN	NaN	NaN	
max	NaN	NaN	NaN	NaN	NaN	NaN	

Se observará la variable Unidad de Medida, ya que se tiene que considerar una misma unidad de mediad para todo los datos, en todo caso se hará un conversión

In [7]:
```
df.groupby('UNIDAD_MEDIDA').count()
```

Out[7]:

UNIDAD_MEDIDA	FECHA_FACTURA	SECTOR_CLIENTE	RUBRO_CLIENTE	COD_MARCA	REGION	PROVII
TO	16748	16682	16682	16748	16748	1

Se detectó una inconsistencia en la unidad de medida ('C/U'), por lo que se eliminará este registro para no sesgar los resultados.

In [8]:
```
df = df[df['UNIDAD_MEDIDA'] != 'C/U']
df.groupby('UNIDAD_MEDIDA').count()
```

Out[8]:

	FECHA_FACTURA	SECTOR_CLIENTE	RUBRO_CLIENTE	COD_MARCA	REGION	PROVII
UNIDAD_MEDIDA						
TO	16748	16682	16682	16748	16748	1

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16749 entries, 0 to 16748
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   FECHA_FACTURA    16749 non-null  object 
 1   SECTOR_CLIENTE   16682 non-null  object 
 2   RUBRO_CLIENTE    16682 non-null  object 
 3   COD_MARCA        16748 non-null  object 
 4   REGION           16748 non-null  object 
 5   PROVINCIA        16748 non-null  object 
 6   DISTRITO         16748 non-null  object 
 7   CIUDAD           16682 non-null  object 
 8   CENTRO_DESPACHO  16632 non-null  object 
 9   CONDICION_PAGO   16748 non-null  object 
 10  UNIDAD_MEDIDA    16748 non-null  object 
 11  Cantidad         16748 non-null  float64
 12  Flete            16748 non-null  float64
 13  Monto USD        16748 non-null  float64
dtypes: float64(3), object(11)
memory usage: 1.8+ MB
```

-Se observa que existen variables con un porcentaje de valores faltantes: SECTOR_CLIENTE, RUBRO_CLIENTE, CIUDAD y CENTRO_DESPACHO. Analizamos el porcentaje da valores nulos por cada una de ellas

In [10]:
```
df.isnull().sum().sum()
```

Out[10]:
```
np.int64(327)
```

Hay presencia de un total de 530 datos null. A continuación se observará el porcentaje de representación por cada variable.

Identificación de NULOS

In [11]:
```
#Vemos el porcentaje de valores faltantes por cada variable

# Missing data:

total = df.isnull().sum().sort_values(ascending = False)
percent = (df.isnull().sum() / df.isnull().count()).sort_values(ascending = False)
missing_data = pd.concat([total, percent], axis = 1, keys = ['Total', 'Percent'])


missing_data = missing_data[missing_data['Total'] > 0]
missing_data
```

Out[11]:

	Total	Percent
CENTRO_DESPACHO	117	0.006985
SECTOR_CLIENTE	67	0.004000
CIUDAD	67	0.004000
RUBRO_CLIENTE	67	0.004000
REGION	1	0.000060
COD_MARCA	1	0.000060
UNIDAD_MEDIDA	1	0.000060
PROVINCIA	1	0.000060
DISTRITO	1	0.000060
CONDICION_PAGO	1	0.000060
Flete	1	0.000060
Cantidad	1	0.000060
Monto USD	1	0.000060

- Las variables SECTOR_CLIENTE,RUBRO_CLIENTE, CIUDAD y CENTRO_DESPACHO tienen menos del 1% de valores faltantes.

Tratamiento valores null

- Dado que los valores nulos representan menos del 1% de los datos, se completarán con la moda correspondiente a la cada variable.

In [12]:
```
moda_ciudad = df['CIUDAD'].mode()[0]
moda_rubro = df['RUBRO_CLIENTE'].mode()[0]
moda_sector_cliente = df['SECTOR_CLIENTE'].mode()[0]
moda_centro = df['CENTRO_DESPACHO'].mode()[0]

df['CIUDAD']         = df['CIUDAD'].fillna(moda_ciudad)
df['RUBRO_CLIENTE'] = df['RUBRO_CLIENTE'].fillna(moda_rubro)
df['SECTOR_CLIENTE'] = df['SECTOR_CLIENTE'].fillna(moda_sector_cliente)
df['CENTRO_DESPACHO'] = df['CENTRO_DESPACHO'].fillna(moda_centro)
```

In [13]:
```
#df = df.dropna(subset=['CIUDAD', 'RUBRO_CLIENTE', 'SECTOR_CLIENTE', 'CENTRO_DESPACHO'])
```

In [14]:
```
#Vemos el porcentaje de valores faltantes por cada variable

# Missing data:

total = df.isnull().sum().sort_values(ascending = False)
percent = (df.isnull().sum() / df.isnull().count()).sort_values(ascending = False)
missing_data = pd.concat([total, percent], axis = 1, keys = ['Total', 'Percent'])


missing_data = missing_data[missing_data['Total'] > 0]
missing_data
```

Out[14]:

	Total	Percent
REGION	1	0.00006
COD_MARCA	1	0.00006
PROVINCIA	1	0.00006
Cantidad	1	0.00006
CONDICION_PAGO	1	0.00006
DISTRITO	1	0.00006
Flete	1	0.00006
Monto USD	1	0.00006
UNIDAD_MEDIDA	1	0.00006

Dataset completo

In [15]:
```
df.describe()
```

Out[15]:

	Cantidad	Flete	Monto USD
count	16748.000000	16748.000000	16748.000000
mean	15.491938	278.490604	2230.710048
std	13.320534	495.899254	2675.423198
min	-31.200000	0.000000	-7812.990000
25%	2.500000	0.000000	298.550000
50%	12.000000	49.100000	1444.125000
75%	30.000000	330.865000	4932.465000
max	156.000000	2861.100000	27152.880000

- Se observa una alta dispersión en las variables Flete y Monto USD, ya que la desviación estandar es mayor que la media.
- Posibles valores outliers para las varibales Cantidad, Flete y Monto USD, ya qye se observa que los valores máximos distan mucho del tercer cuartil.
- Se observa que las variables Cantidad y monto USD presentan valores negativos, por lo que se realizará el filtrado correspondiente.

In [16]:
```
#Transformamos variable FECHA_FACTURA de tipo de dato OBJECT a DATETIME
df['FECHA_FACTURA'] = pd.to_datetime(df['FECHA_FACTURA'])
```

ANALISIS DE VARIABLE OBJETIVO

In [17]:
```
df['Monto USD'].describe()
```

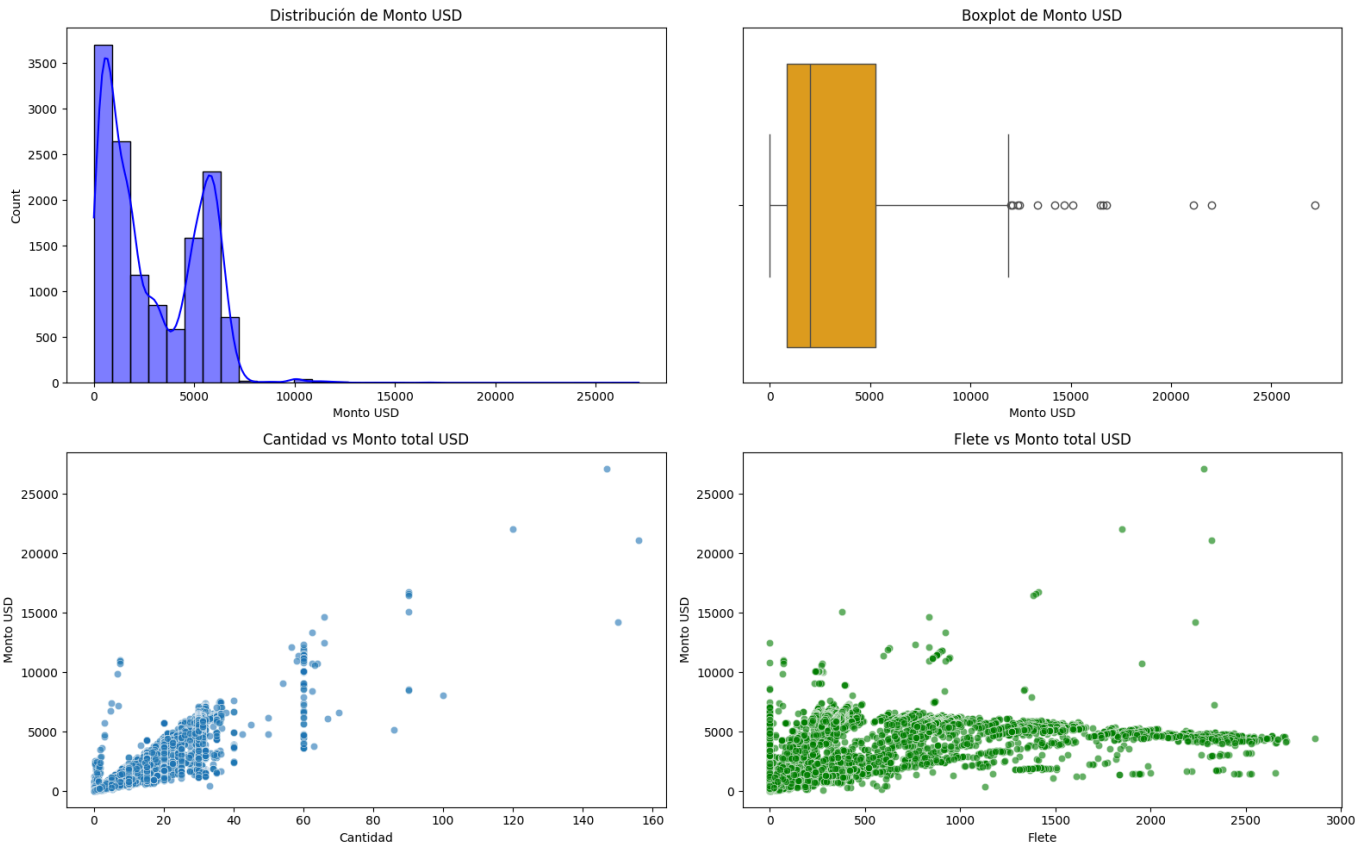
Out[17]:

	Monto USD
count	16748.000000
mean	2230.710048
std	2675.423198
min	-7812.990000
25%	298.550000
50%	1444.125000
75%	4932.465000
max	27152.880000

dtype: float64

In [18]:

```
df = df[df['Cantidad'] > 0]
df = df[df['Monto USD'] > 0]
# Visualización para explorar patrones y detectar outliers
plt.figure(figsize=(16, 10))

# Histograma de Monto total USD
plt.subplot(2, 2, 1)
sns.histplot(df["Monto USD"], bins=30, kde=True, color="blue")
plt.title("Distribución de Monto USD")

# Boxplot de Monto total USD
plt.subplot(2, 2, 2)
sns.boxplot(x=df["Monto USD"], color="orange")
plt.title("Boxplot de Monto USD")

# Relación Cantidad vs Monto total USD
plt.subplot(2, 2, 3)
sns.scatterplot(x="Cantidad", y="Monto USD", data=df, alpha=0.6)
plt.title("Cantidad vs Monto total USD")

# Relación Flete vs Monto total USD
plt.subplot(2, 2, 4)
sns.scatterplot(x="Flete", y="Monto USD", data=df, alpha=0.6, color="green")
plt.title("Flete vs Monto total USD")

plt.tight_layout()
plt.show()
```



Apartir de los graficos de observa lo siguiente:

MONTO_USD:

- La mayoria de montos estan entre 0 y 10 mil dolares, que es donde más se repiten ese monto de venta.
- Hay unos cuantos valores muy altos, pero son pocos, se puede ver por la cola larga a la derecha.
- Cantidad vs Monto Total: Se observa la existencia de una relación lineal positiva entre las variables Cantidad y Monto Total
- En Flete vs Monto Total: Existe un relación lineal debil, se verá a más detalle en la Matriz de correlación.
- Se observa la prescencia de datos atípicos.

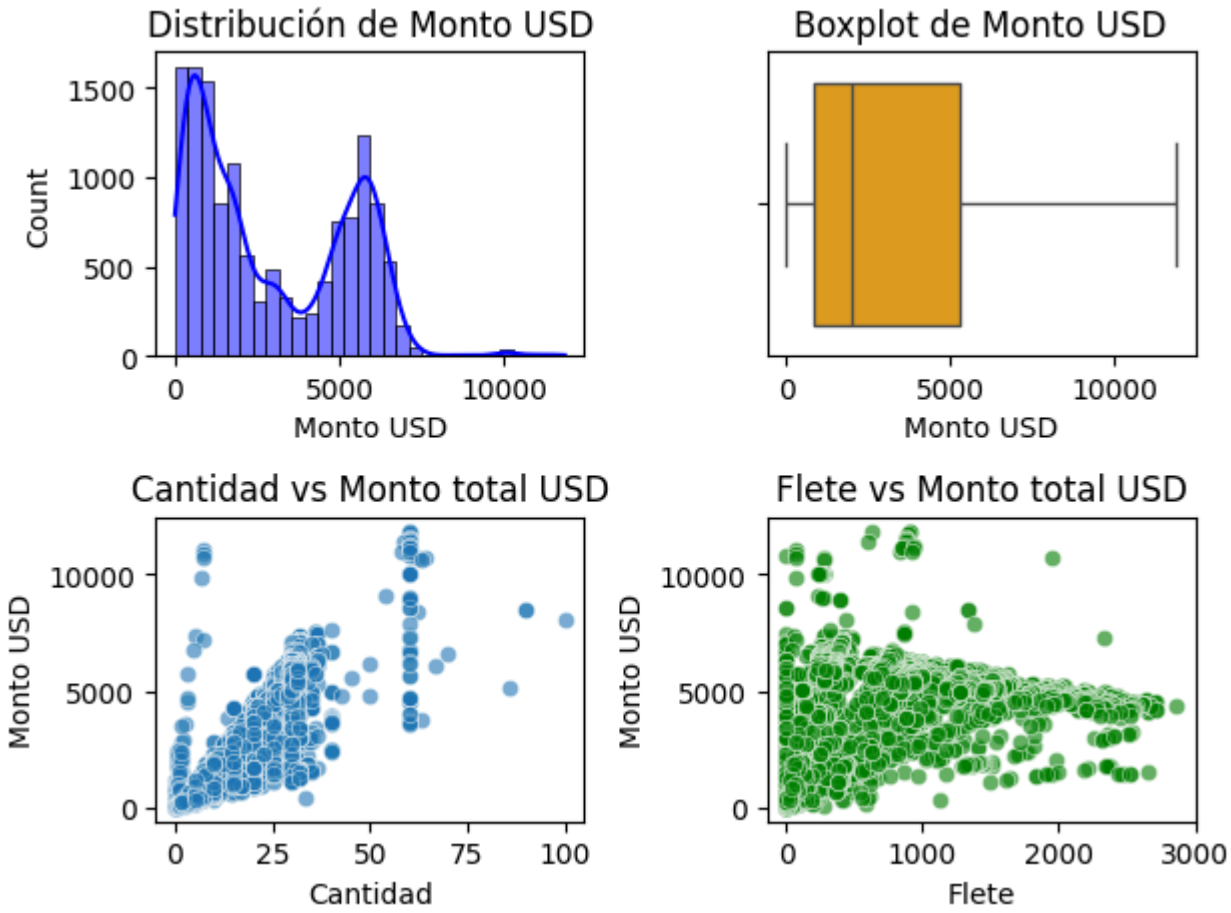
TRATAMIENTO DE OUTLIERS

In [19]:
```
Q1 = df["Monto USD"].quantile(0.25)#calcular los cuartiles
Q3 = df["Monto USD"].quantile(0.75)
IQR = Q3 - Q1
```

In [20]:
```
limite_inferior = Q1 - 1.5 * IQR#definimos los limites
limite_superior = Q3 + 1.5 * IQR

df = df[(df["Monto USD"] >= limite_inferior) & (df["Monto USD"] <= limite_superior)]# filtra
```

In [21]:
```
# Histograma de Monto total USD
plt.subplot(2, 2, 1)
sns.histplot(df["Monto USD"], bins=30, kde=True, color="blue")
plt.title("Distribución de Monto USD")

# Boxplot de Monto total USD
plt.subplot(2, 2, 2)
sns.boxplot(x=df["Monto USD"], color="orange")
```

```
plt.title("Boxplot de Monto USD")

# Relación Cantidad vs Monto total USD
plt.subplot(2, 2, 3)
sns.scatterplot(x="Cantidad", y="Monto USD", data=df, alpha=0.6)
plt.title("Cantidad vs Monto total USD")

# Relación Flete vs Monto total USD
plt.subplot(2, 2, 4)
sns.scatterplot(x="Flete", y="Monto USD", data=df, alpha=0.6, color="green")
plt.title("Flete vs Monto total USD")

plt.tight_layout()
plt.show()
```



OBSERVACIONES: Se muestra que los datos estan ajustado a la distribución general despues del tratamiento de outliers.

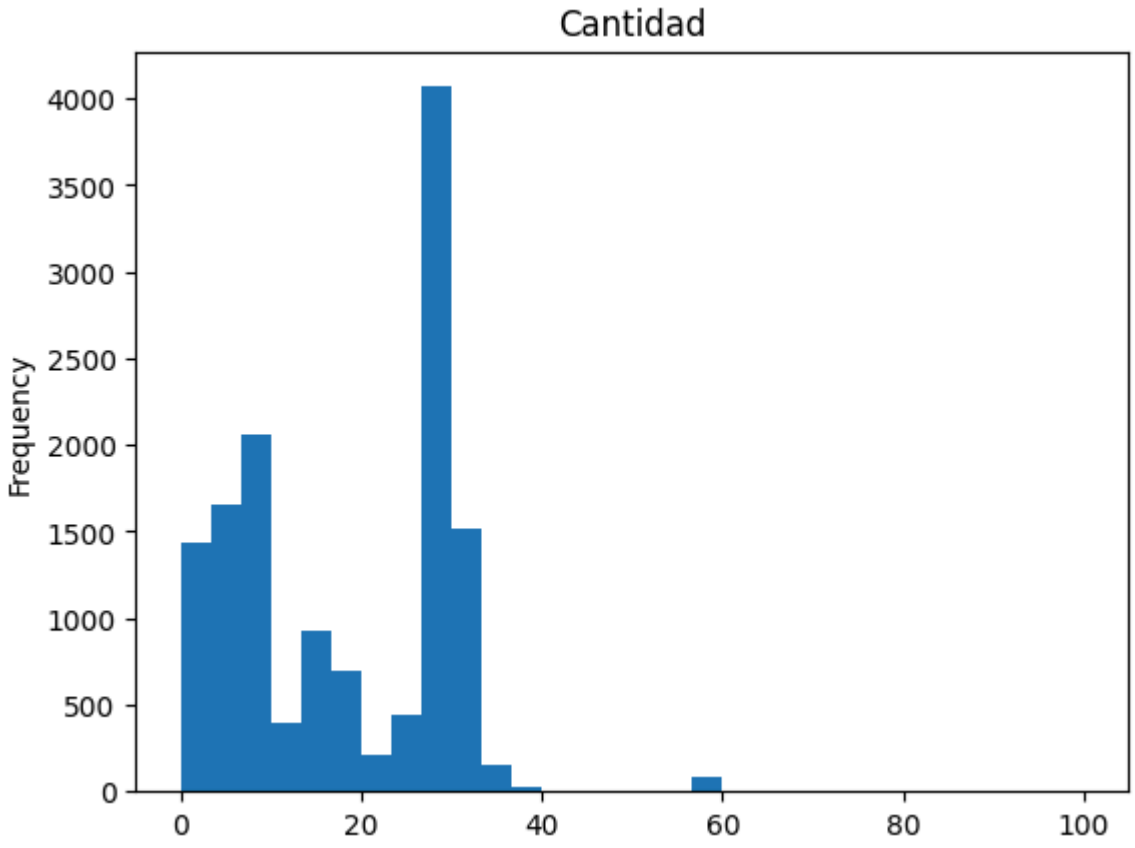
Manejo de Outliers

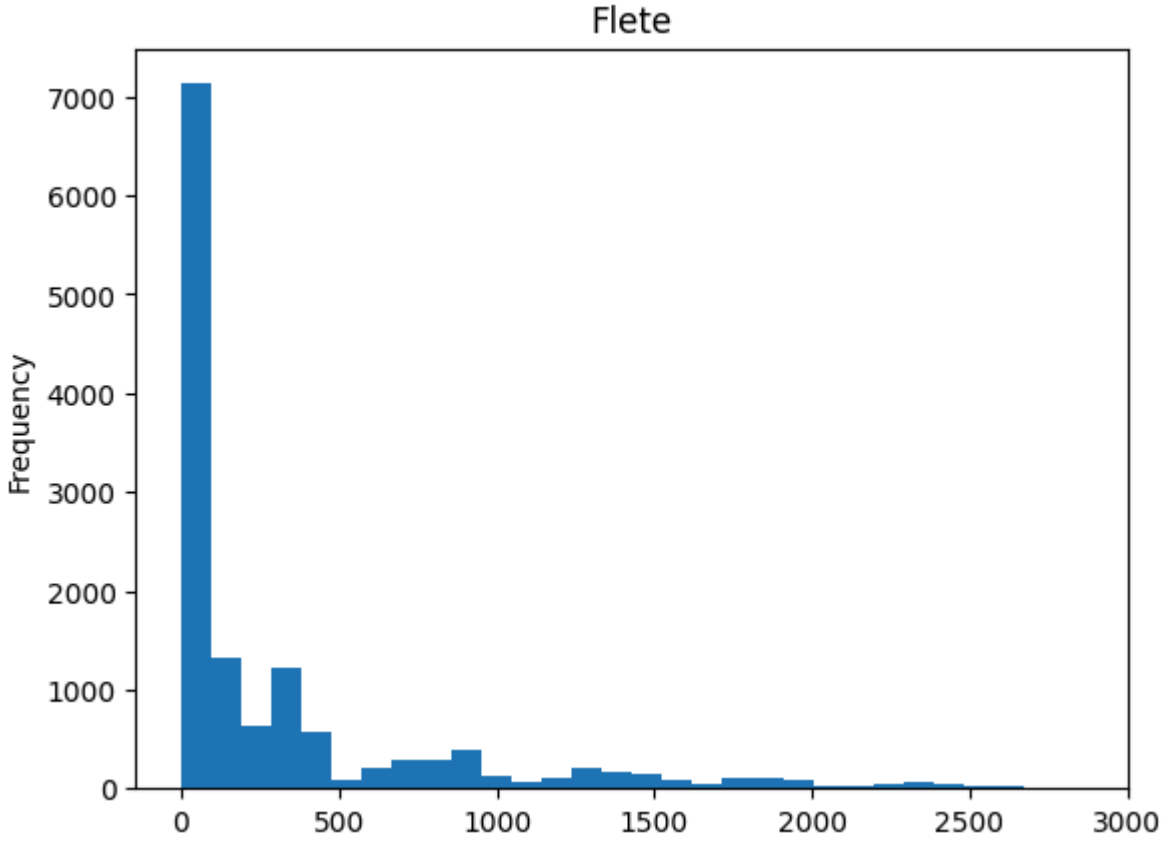
In [22]:
```
#DETECTAR VARIABLES CATEGORICAS Y NUMERICAS
lista_cualitativas = df.select_dtypes(include='object').columns
lista_cuantitativas = df.select_dtypes(include=['int64', 'float64']).columns

print(f"Variables categoricas: {lista_cualitativas}")
print(f"Variables numericas: {lista_cuantitativas}")
```

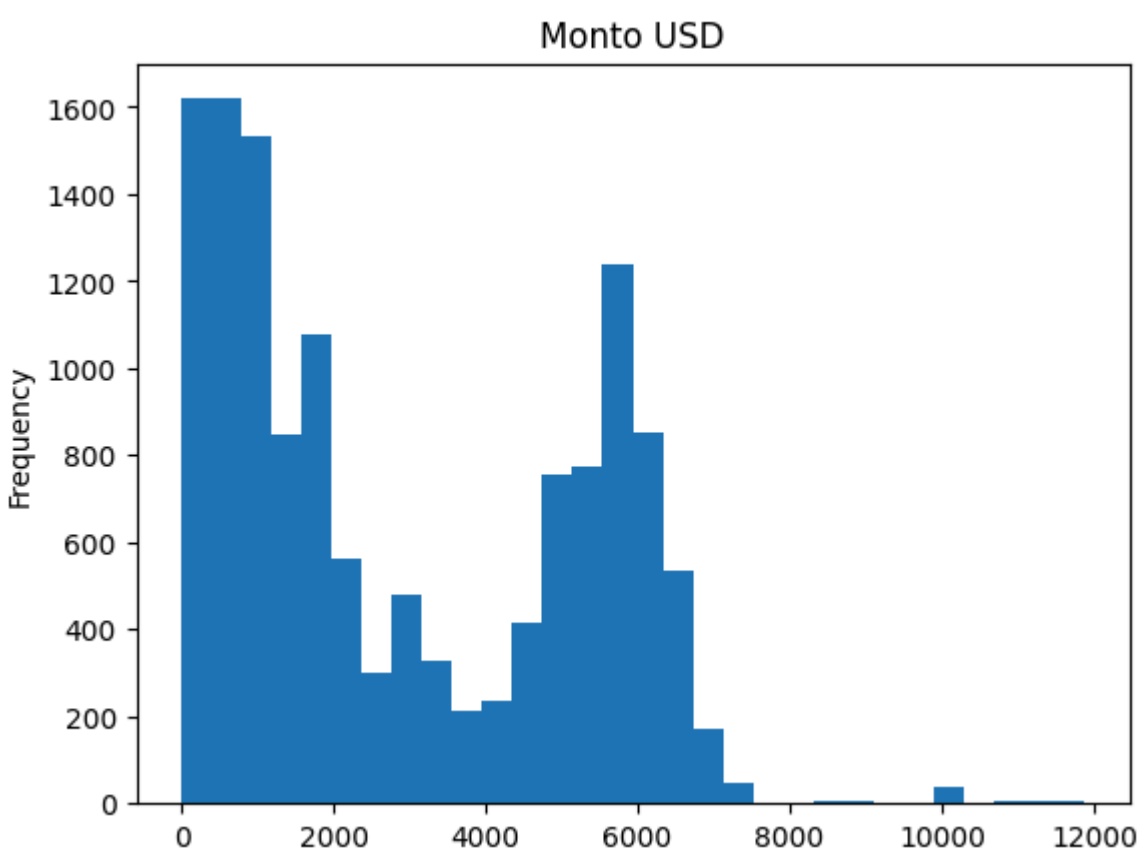
```
Variables categoricas: Index(['SECTOR_CLIENTE', 'RUBRO_CLIENTE', 'COD_MARCA', 'REGION', 'PROVI
NCIA',
       'DISTRITO', 'CIUDAD', 'CENTRO_DESPACHO', 'CONDICION_PAGO',
       'UNIDAD_MEDIDA'],
      dtype='object')
Variables numericas: Index(['Cantidad', 'Flete', 'Monto USD'], dtype='object')
```

Revisando las variables explicativas cuantitativas:

In [23]:
```
# realizamos histogramas para revisar la distribución de cada variable cuantitativa

for var in lista_cuantitativas:
    df[var].plot(kind='hist', title=var, bins=30)
    plt.show()
```

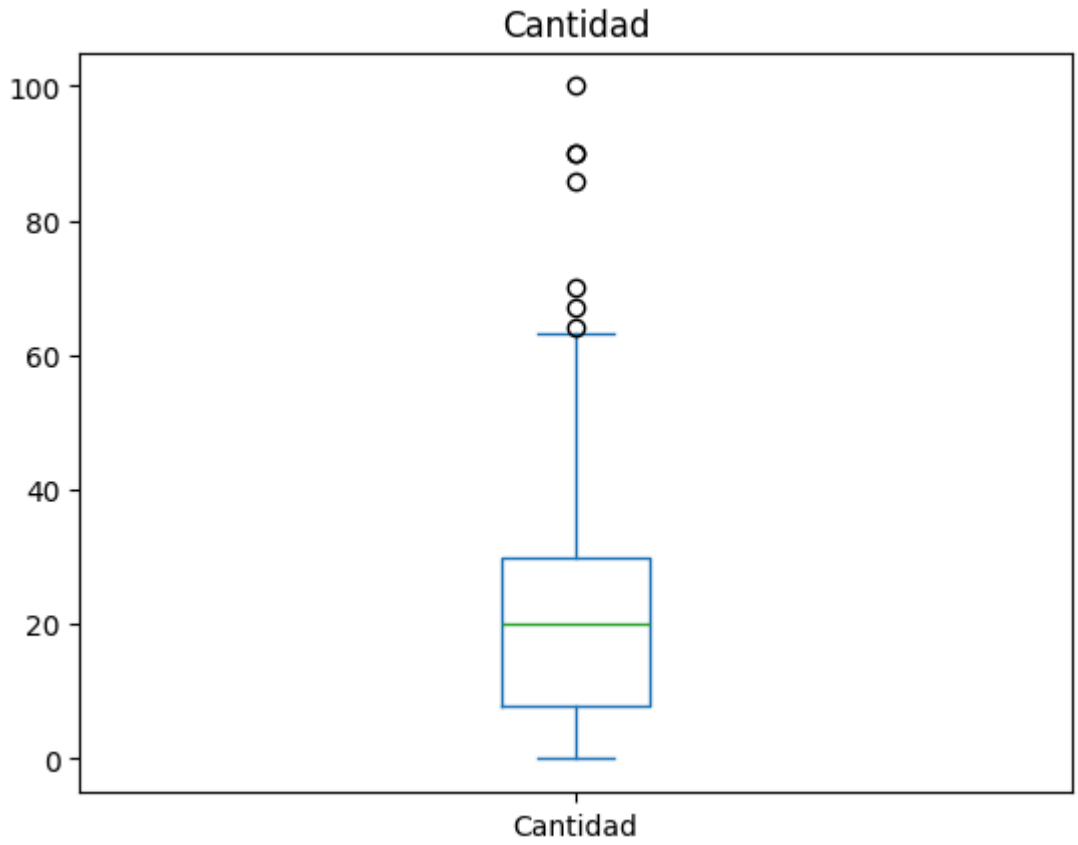


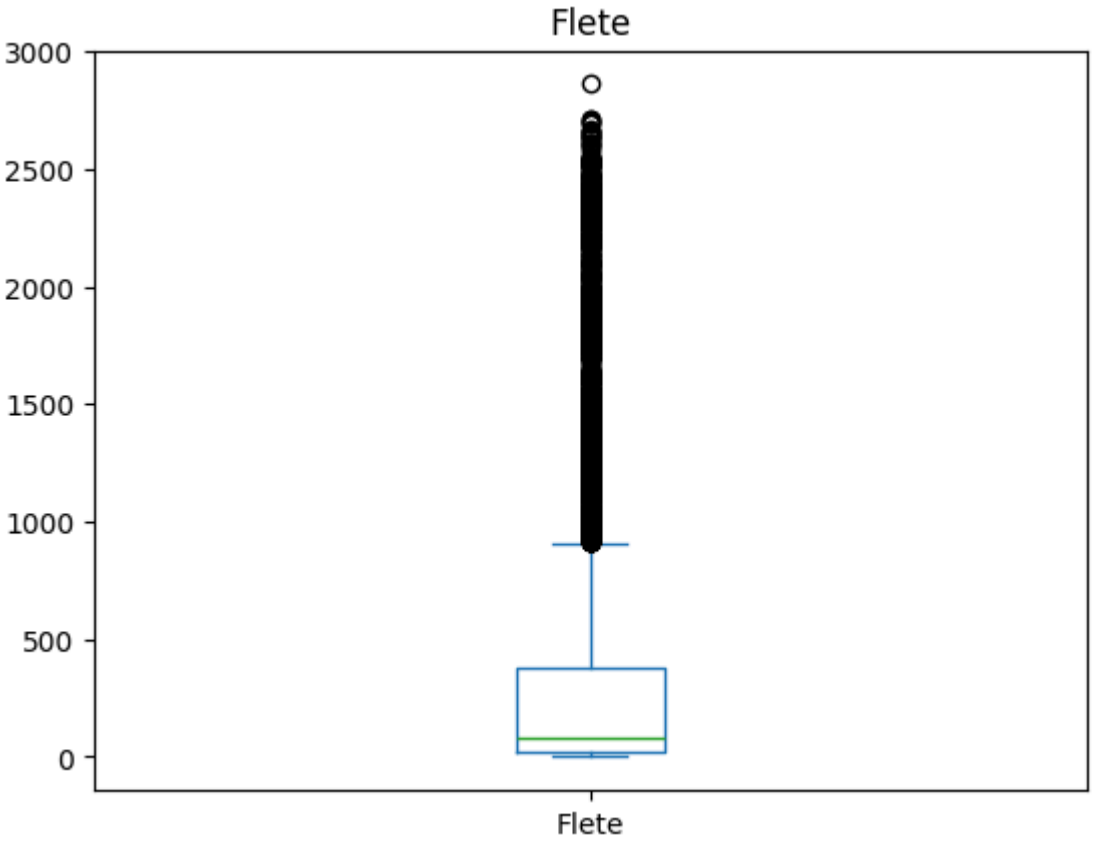


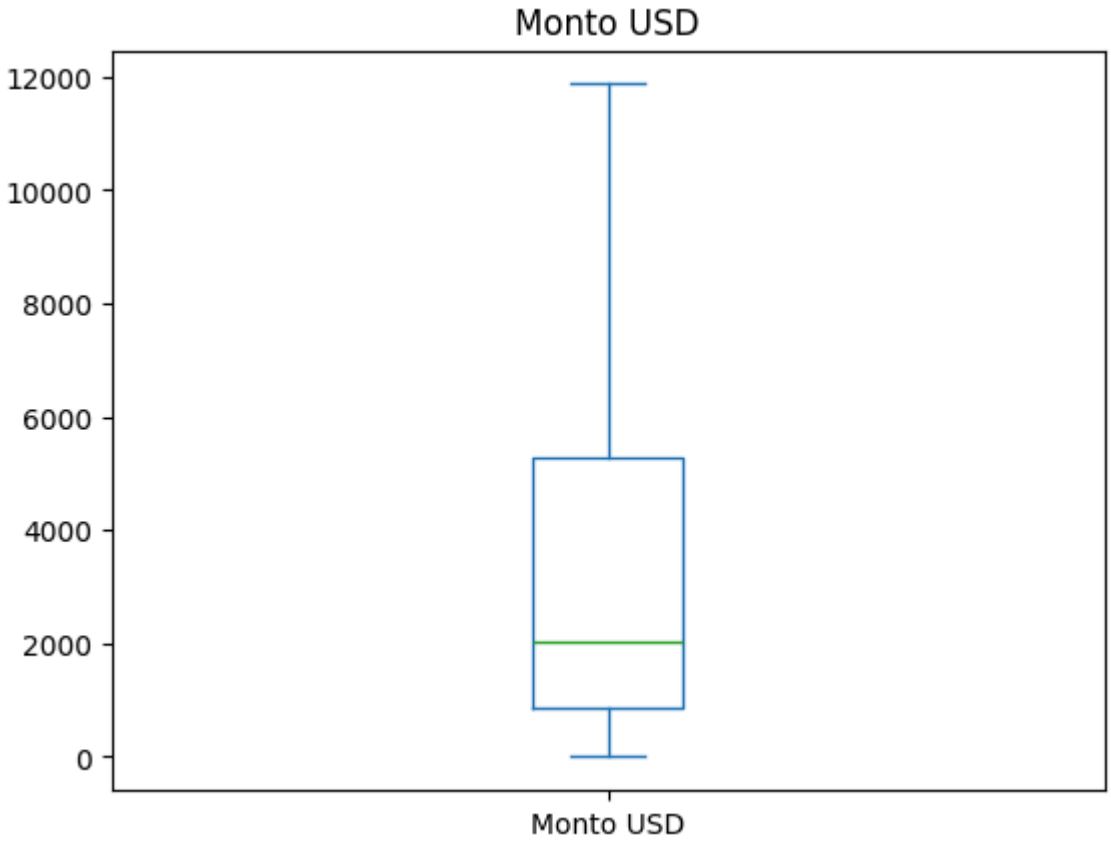


- Para la variable cantidad, los valores con mayores frecuencia se encuentran entre menos a 50.
- Las variables Flete y Monto USD tienen una distribución con cola hacia la derecha.

In [24]:
```
# realizamos diagramas de cajas para revisar la distribución de cada variable cuantitativa
for var in lista_cuantitativas:
    df[var].plot(kind='box', title=var)
    plt.show()
```





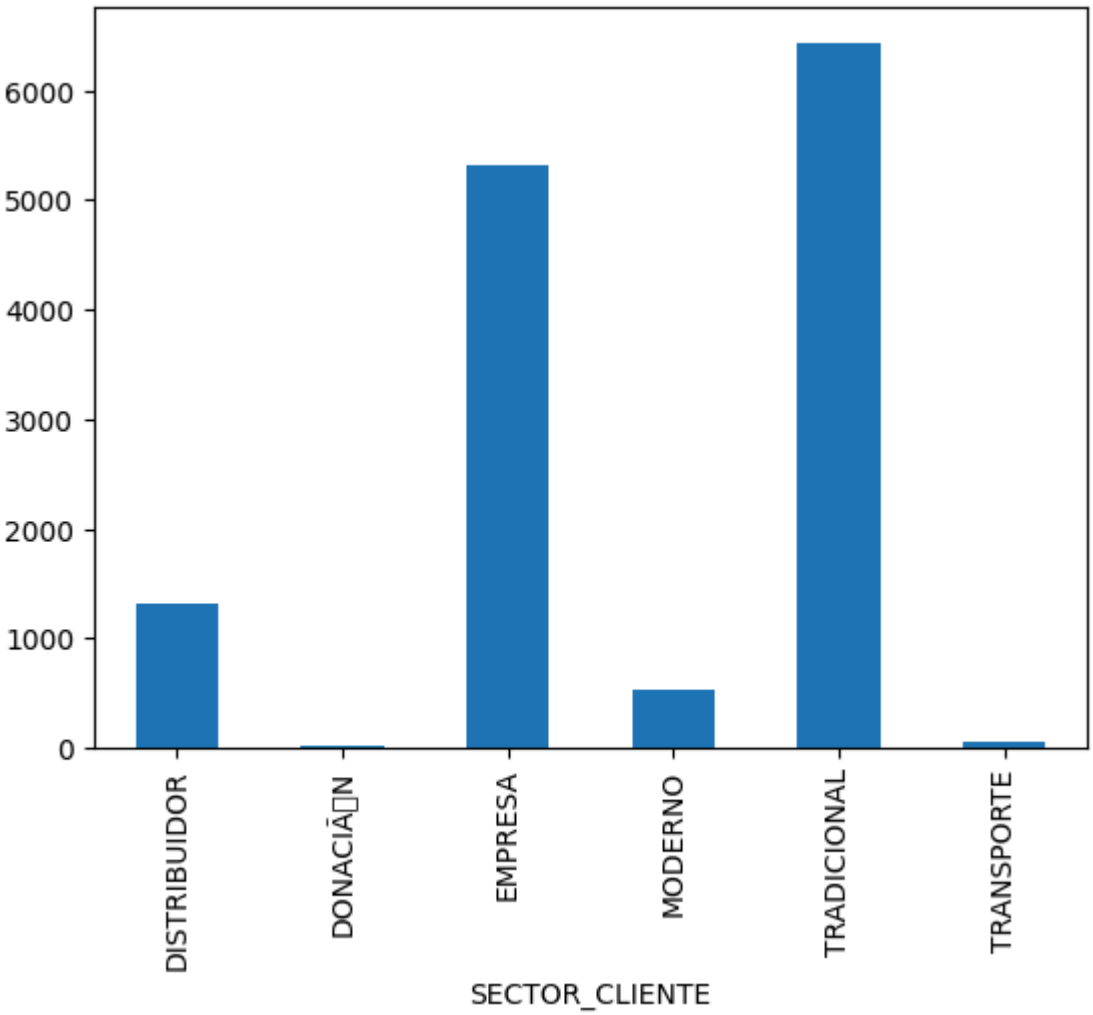


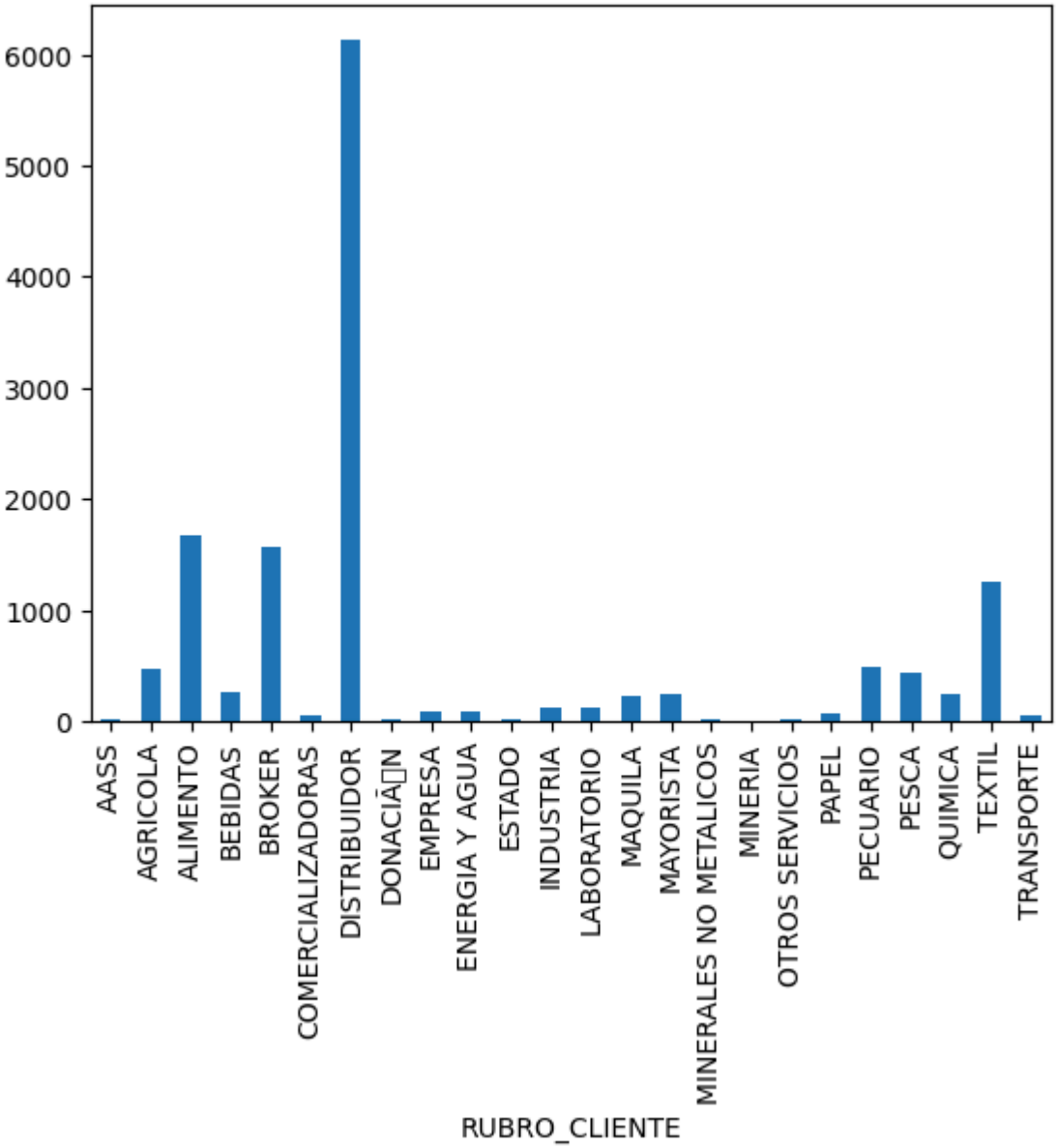
- La variable cantidad presenta valores atipicos aproximadamente entre 60 y 200. Los datos se concentran en valores menores a 50, pues la media se aproxima a cero. -Para la variable Flete, se observa outliers en un rango de 1000 a 3000. -Para la variable Monto USD, Se observa muchos valores atípicos mayores a 10000. Los valores para Monto USD se encuentran concentrados menores a 5000.

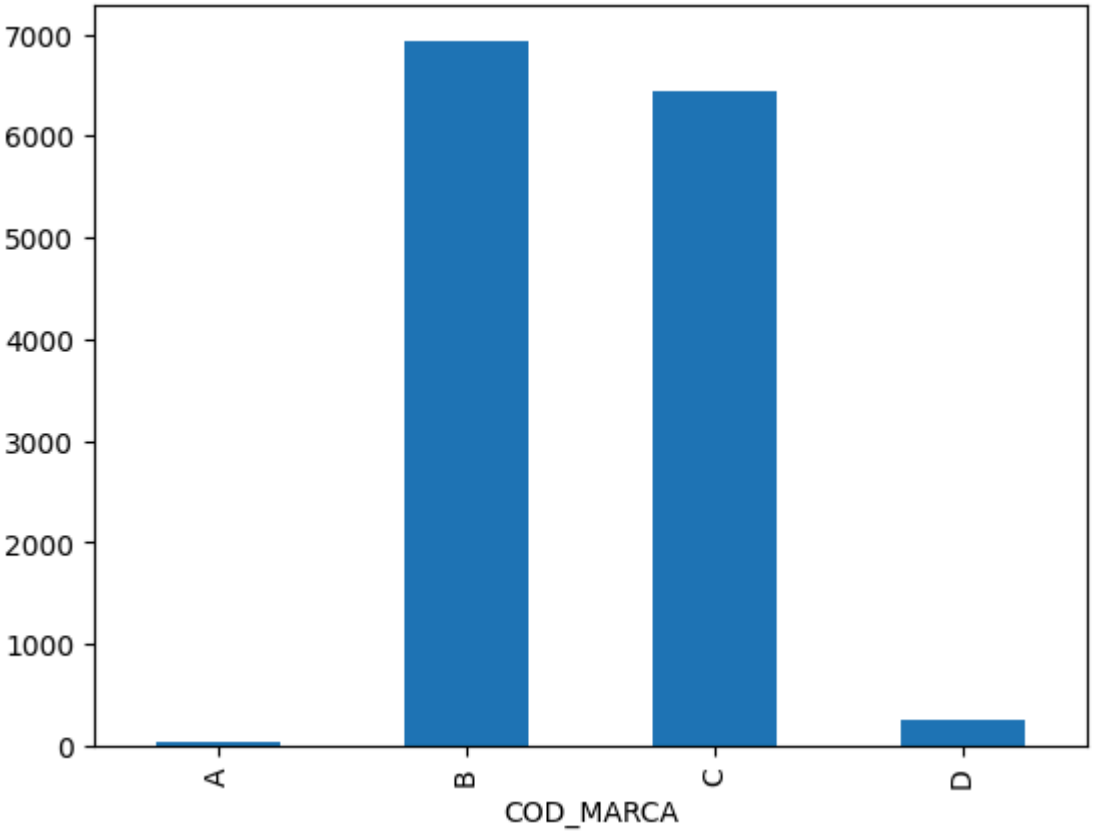
Revisando las variables explicactivas cualitativas:

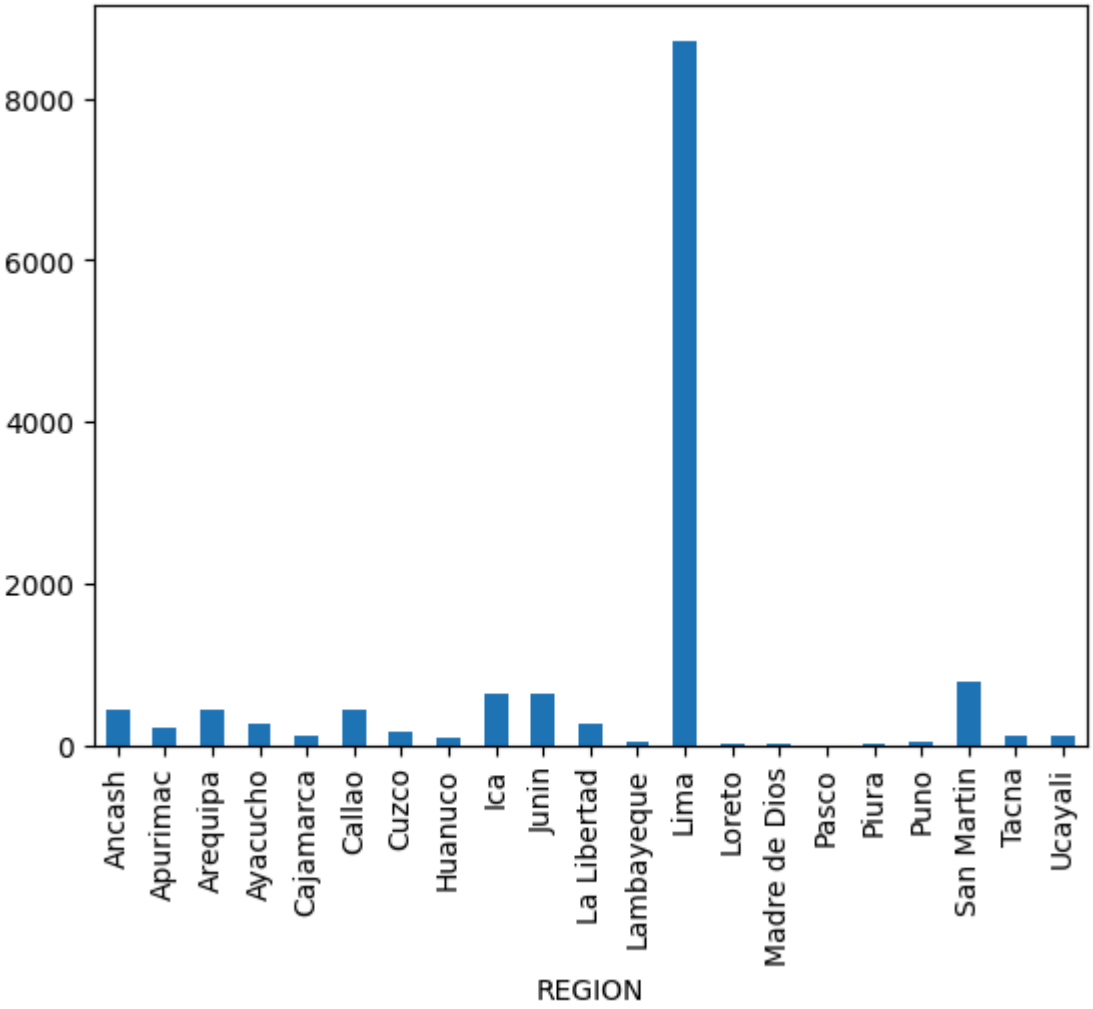
In [25]:
```
# realizamos diagramas de barras para revisar la distribución de frecuencias de cada variable
for var in lista_cualitativas:
```

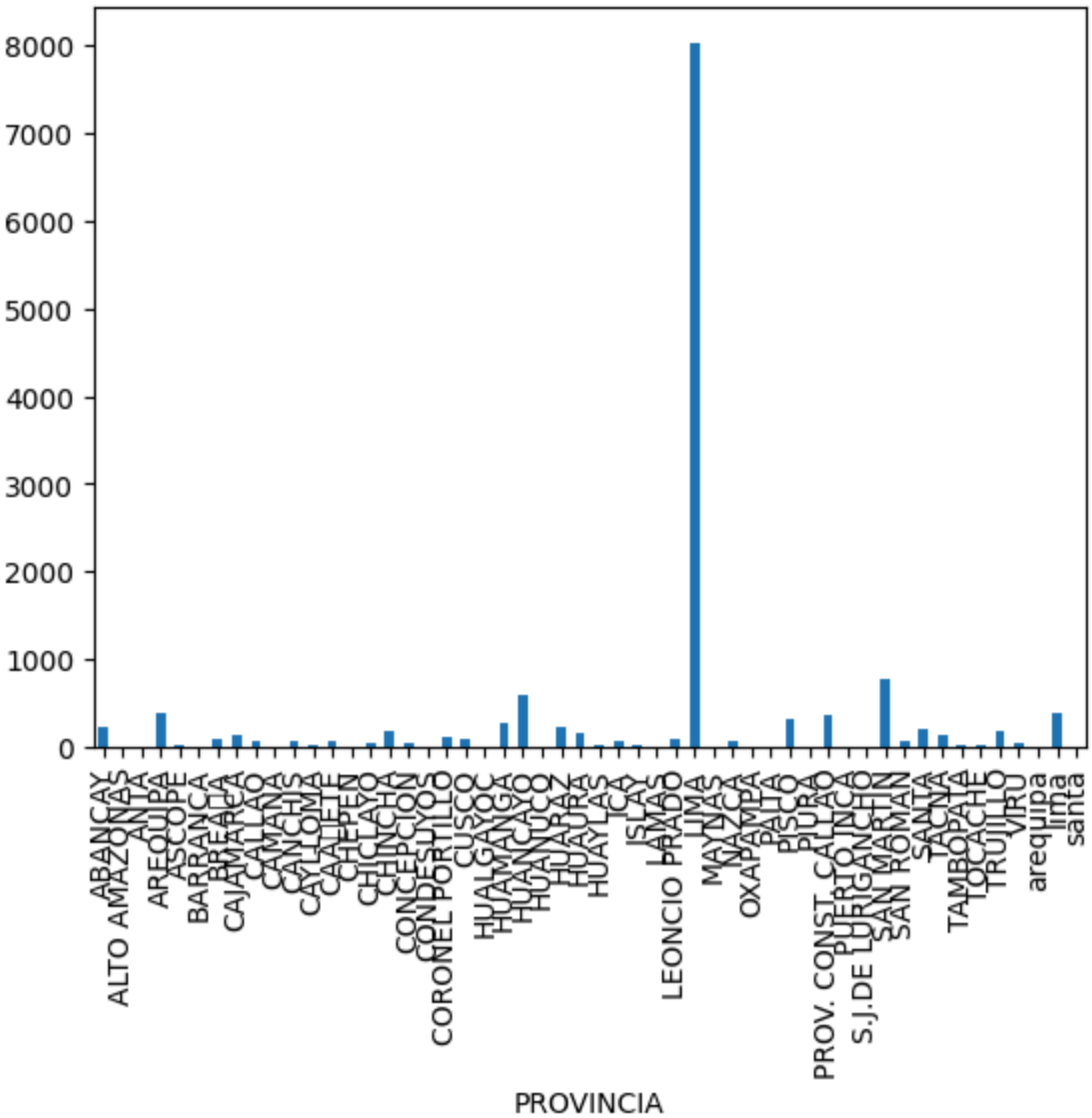
```
    df.groupby(var)['Monto USD'].count().plot.bar()
    plt.show()
```

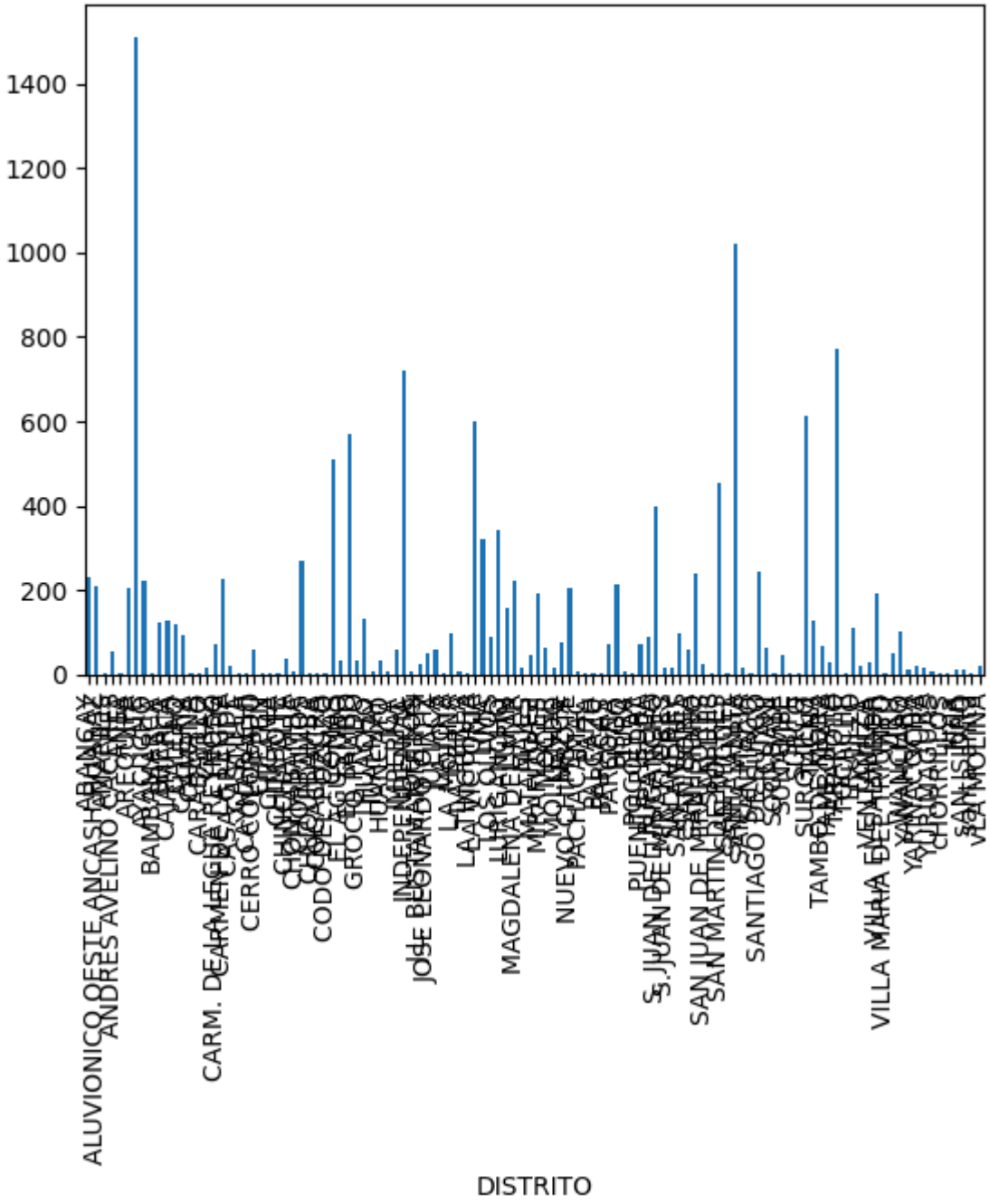


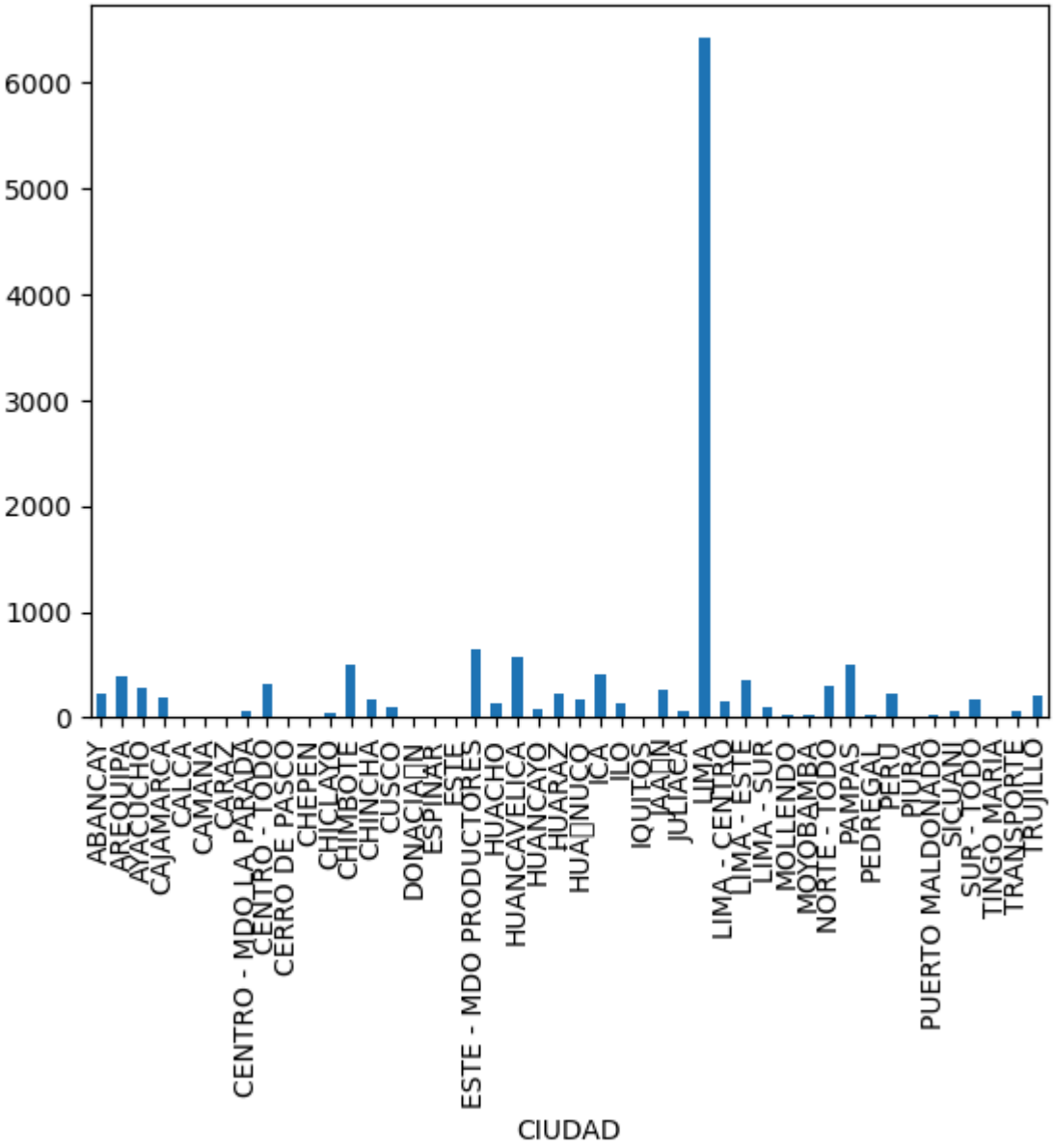

6000
5000
4000
3000
2000
1000
0
AASS
AGRICOLA
ALIMENTO
BEBIDAS
BROKER
COMERCIALIZADORAS
DISTRIBUIDOR
DONACIÃ□N
EMPRESA
ENERGIA Y AGUA
ESTADO
INDUSTRIA
LABORATORIO
MAQUILA
MAYORISTA
MINERALES NO METALICOS
MINERIA
OTROS SERVICIOS
PAPEL
PECUARIO
PESCA
QUIMICA
TEXTIL
TRANSPORTE
RUBRO_CLIENTE

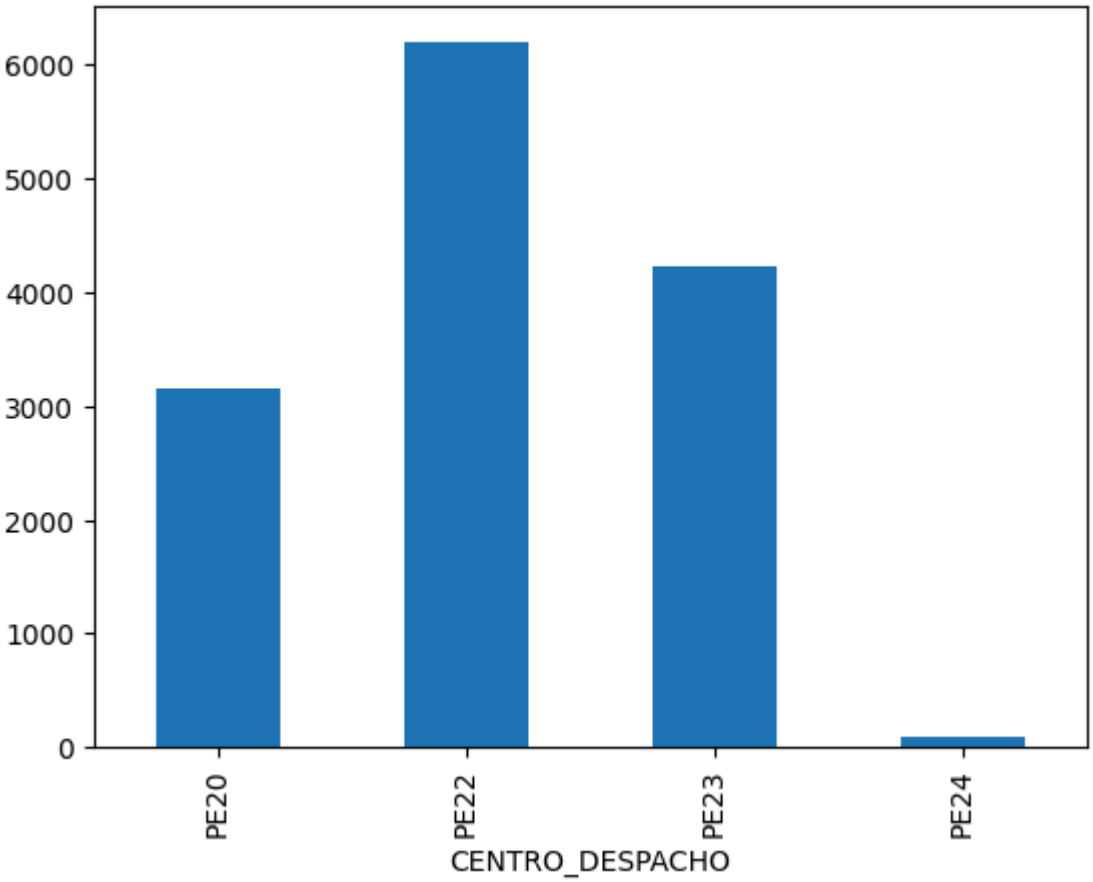

7000
6000
5000
4000
3000
2000
1000
0
A
B
C
D
COD_MARCA

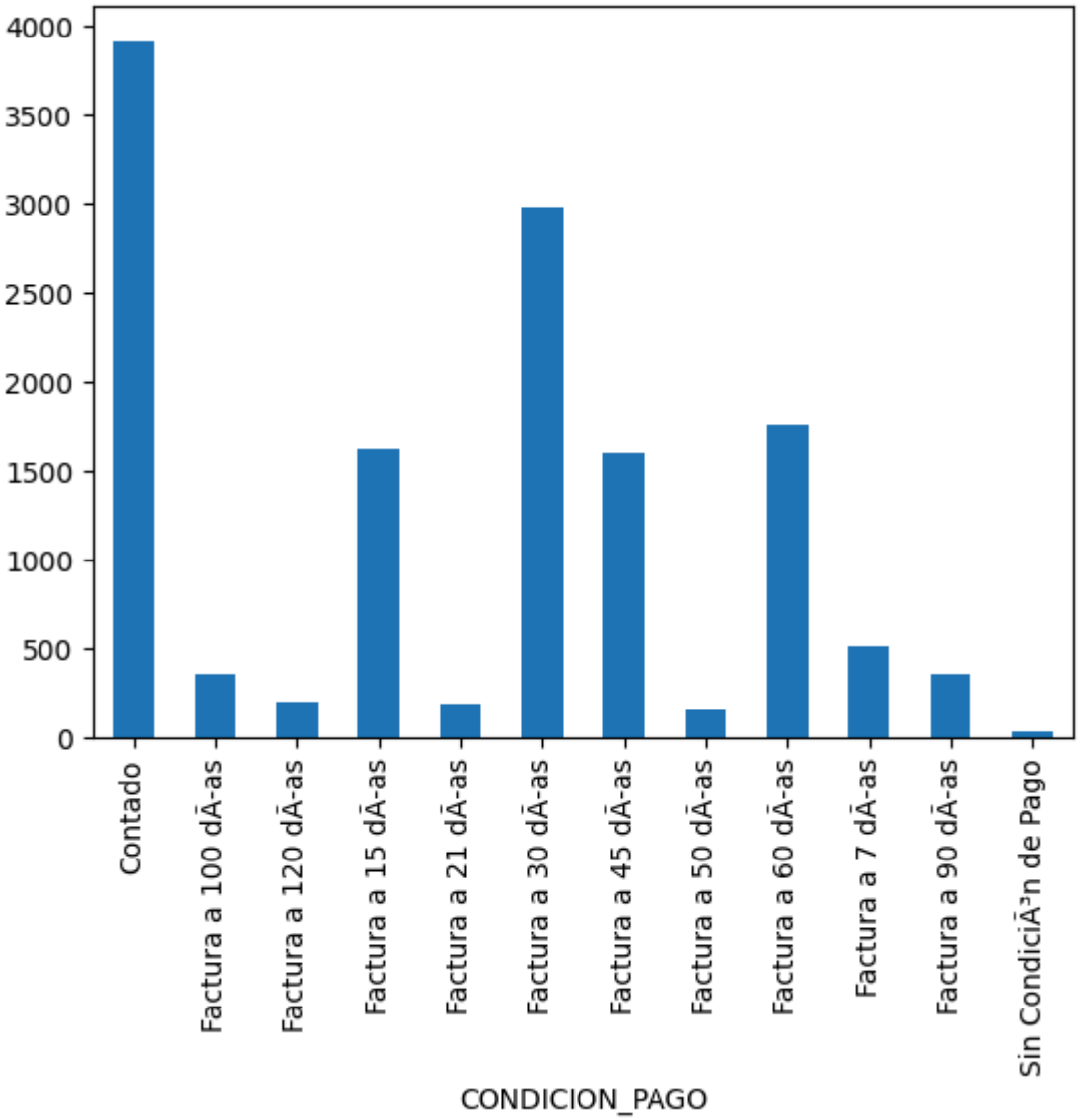

8000
6000
4000
2000
0
Ancash
Apurimac
Arequipa
Ayacucho
Cajamarca
Callao
Cuzco
Huanuco
Ica
Junin
La Libertad
Lambayeque
Lima
Loreto
Madre de Dios
Pasco
Piura
Puno
San Martin
Tacna
Ucayali
REGION

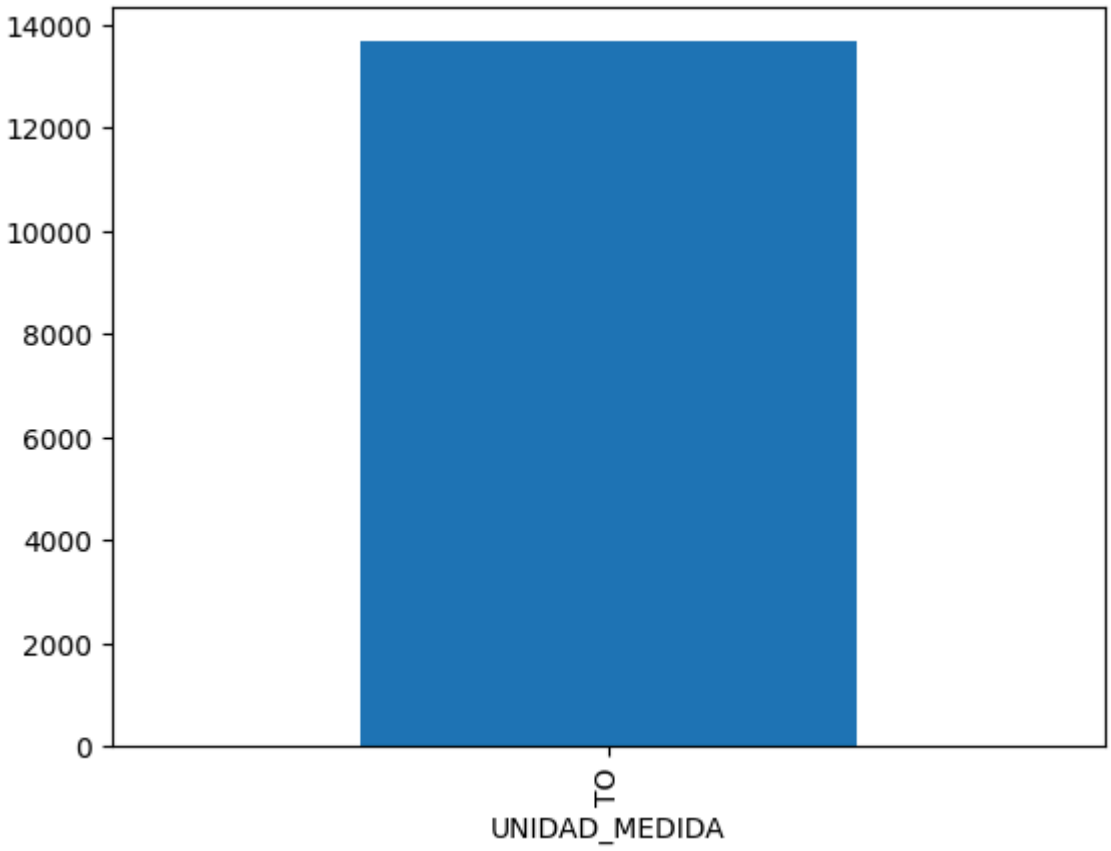

8000
7000
6000
5000
4000
3000
2000
1000
0
PROVINCIA


1400
1200
1000
800
600
400
200
0
DISTRITO


6000
5000
4000
3000
2000
1000
0
ABANCAY
AREQUIPA
AYACUCHO
CAJAMARCA
CALCA
CAMANA
CARAZ
CENTRO - MDO LA PARADA
CENTRO - TODO
CERRO DE PASCO
CHEPEN
CHICLAYO
CHIMBOTE
CHINCHA
CUSCO
DONACIAN
ESPINAR
ESTE
ESTE - MDO PRODUCTORES
HUACHO
HUANCAVELICA
HUANCAYO
HUARAZ
HUANUCO
ICA
ILO
IQUITOS
JAEN
JULIACA
LIMA
LIMA - CENTRO
LIMA - ESTE
LIMA - SUR
MOLLENDO
MOYOBAMBA
NORTE - TODO
PAMPAS
PEDREGAL
PERU
PIURA
PUERTO MALDONADO
SICUANI
SUR - TODO
TINGO MARIA
TRANSPORTE
TRUJILLO
CIUDAD


6000
5000
4000
3000
2000
1000
0
PE20
PE22
PE23
PE24
CENTRO_DESPACHO



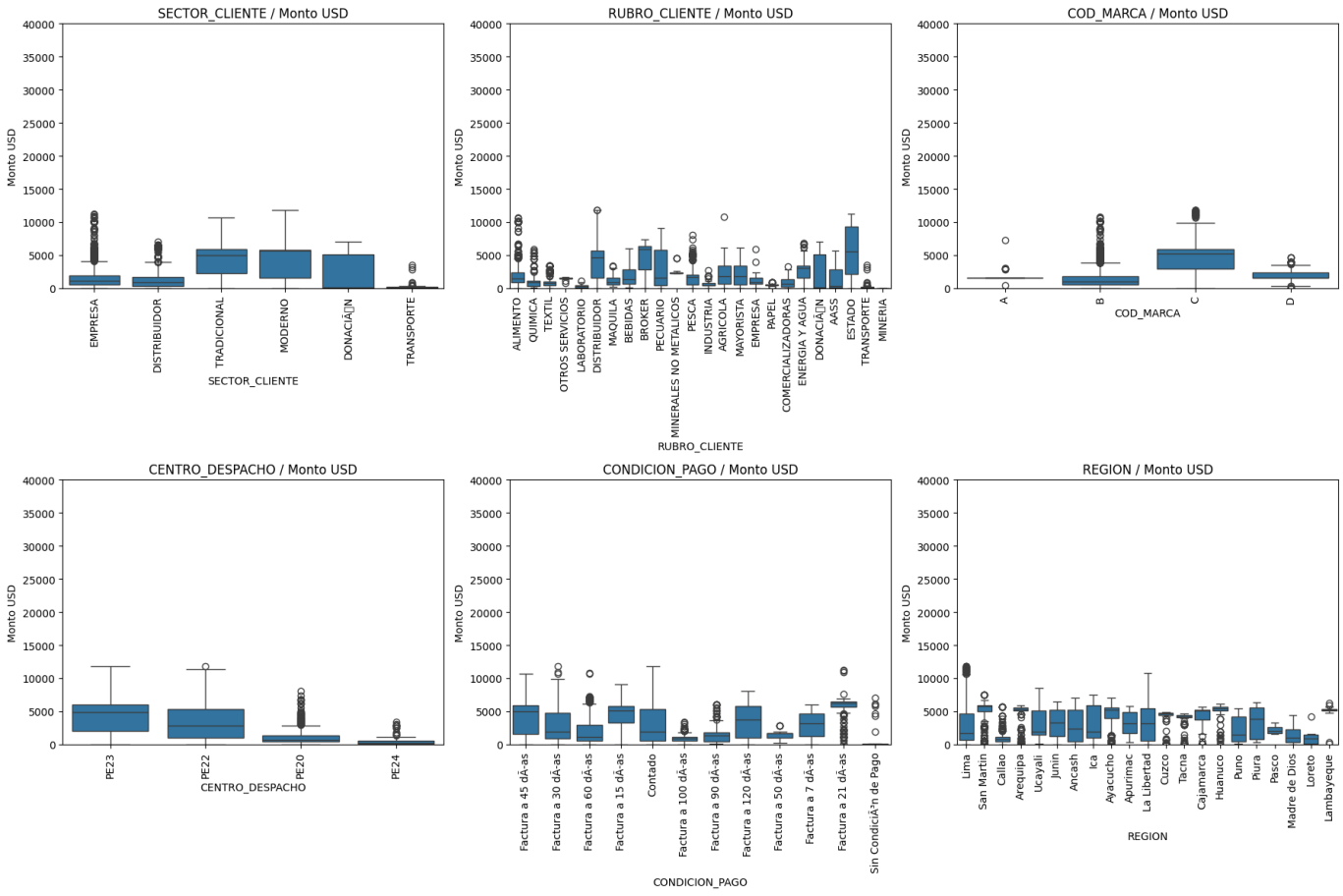


- En el gráfico de la variable SECTOR_CLIENTE, se observa que el MONTO USD se encuentra concentrado en EMPRESA y TRADICIONAL.
- RUBRO_CLIENTE: El gráfico muestra que el rubro DISTRIBUIDOS tiene mayor monto de ventas.

- MARCA: Las marcas B y D tienen valores mayores de MONTO USD.
- CIUDAD: Los valores mayores para MONTO USD se encuentra concentrado en Lima.
- CENTRO DE DESPACHO: los centros de despacho con mayor monto de ventas son PE22 y PE23.
- CONDICION DE PAGO: los valores mayores para MONTO USD se concentran en la condición de pago: al contado y factura a 30 días y a 45 días.

Relación de variables numéricas continuas con numericas discretas

In [26]:
```python
def plot_boxplot_subplots(df):
    variables = [
        'SECTOR_CLIENTE',
        'RUBRO_CLIENTE',
        'COD_MARCA',
        'CENTRO_DESPACHO',
        'CONDICION_PAGO',
        'REGION',
        'PROVINCIA',
        'DISTRITO',
        'CIUDAD',
    ]
    y_col = 'Monto USD'
    fig, axes = plt.subplots(2, 3, figsize=(18, 12))
    axes = axes.flatten()

    for ax, var in zip(axes, variables):
        sns.boxplot(x=var, y=y_col, data=df, ax=ax)
        ax.set_ylim(0, 40000)
        ax.set_title(f'{var} / {y_col}')
        ax.tick_params(axis='x', rotation=90)

    for ax in axes[len(variables):]:
        ax.axis('off')

    plt.tight_layout()
    plt.show()

# Uso:
plot_boxplot_subplots(df)
```



Se considerará las siguientes Variables:

- SECTOR_CLIENTES
- COD_MARCA

In [27]:
```
df_ohe = df.copy()
lista_cualitativas_new = list(df_ohe.select_dtypes(include='object').columns)
df_ohe = pd.get_dummies(df_ohe, columns=lista_cualitativas_new, dtype=float)
lista_cuantitativas_new = list(df_ohe.select_dtypes(include=['int64', 'float64']).columns)
```

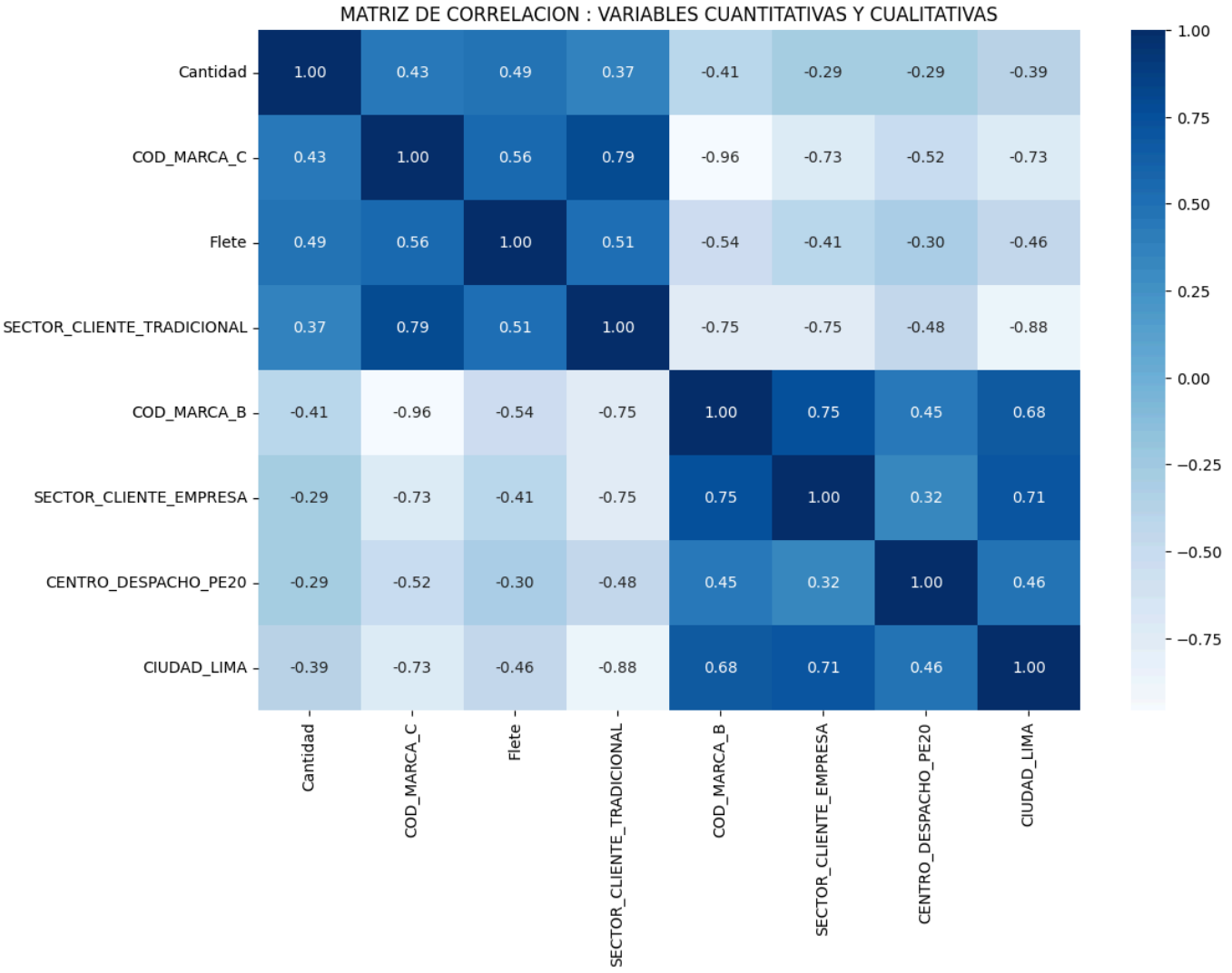
In [28]:
```
import seaborn as sns
import matplotlib.pyplot as plt

df_numeric = df_ohe.select_dtypes(include=['number'])
corrmat = df_numeric.corr()
corrmat[['Monto USD']].sort_values(by = 'Monto USD',ascending = False).head(10).style.backgro
```

Out[28]:

	Monto USD
Monto USD	1.000000
Cantidad	0.870147
COD_MARCA_C	0.627233
SECTOR_CLIENTE_TRADICIONAL	0.496034
Flete	0.487090
CENTRO_DESPACHO_PE23	0.344875
RUBRO_CLIENTE_DISTRIBUIDOR	0.302316
RUBRO_CLIENTE_BROKER	0.267366
DISTRITO_TARAPOTO	0.266666
PROVINCIA_SAN MARTIN	0.266666

In [29]:

```
corrmat = df_ohe[['Cantidad', 'COD_MARCA_C','Flete','SECTOR_CLIENTE_TRADICIONAL','COD_MARCA_B
corrmat

plt.figure(figsize=(12,8))
sns.heatmap(corrmat, annot=True, fmt=".2f", cmap='Blues')
plt.title("MATRIZ DE CORRELACION : VARIABLES CUANTITATIVAS Y CUALITATIVAS")
plt.show()
```



Variables a considerar:

- Cantidad
- Flete

- SECTOR_CLIENTE_TRADICIONAL
- SECTOR_CLIENTE_EMPRESA
- CENTRO_DESPACHO_PE20

Selección de variables

In [30]:
```
df_ohe = df_ohe[['Cantidad', 'CENTRO_DESPACHO_PE20','SECTOR_CLIENTE_EMPRESA','Flete', 'Monto
X = df_ohe.drop(['Monto USD'], axis=1)
df_ohe.head()
y = df_ohe['Monto USD']
```

In [31]:
```
X.shape, y.shape
```

Out[31]:
```
((13671, 5), (13671,))
```

Construción del modelo (Regresión o Clasificación)

In [32]:
```
#DIVISION EN TRAIN Y TEST
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10936 entries, 14704 to 9123
Data columns (total 5 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Cantidad                    10936 non-null  float64
 1   CENTRO_DESPACHO_PE20        10936 non-null  float64
 2   SECTOR_CLIENTE_EMPRESA      10936 non-null  float64
 3   Flete                       10936 non-null  float64
 4   SECTOR_CLIENTE_TRADICIONAL  10936 non-null  float64
dtypes: float64(5)
memory usage: 512.6 KB
```

In [33]:
```
from sklearn.linear_model import LinearRegression
modelo = LinearRegression()
modelo.fit(X_train, y_train)
```

Out[33]:
```
LinearRegression
LinearRegression()
```

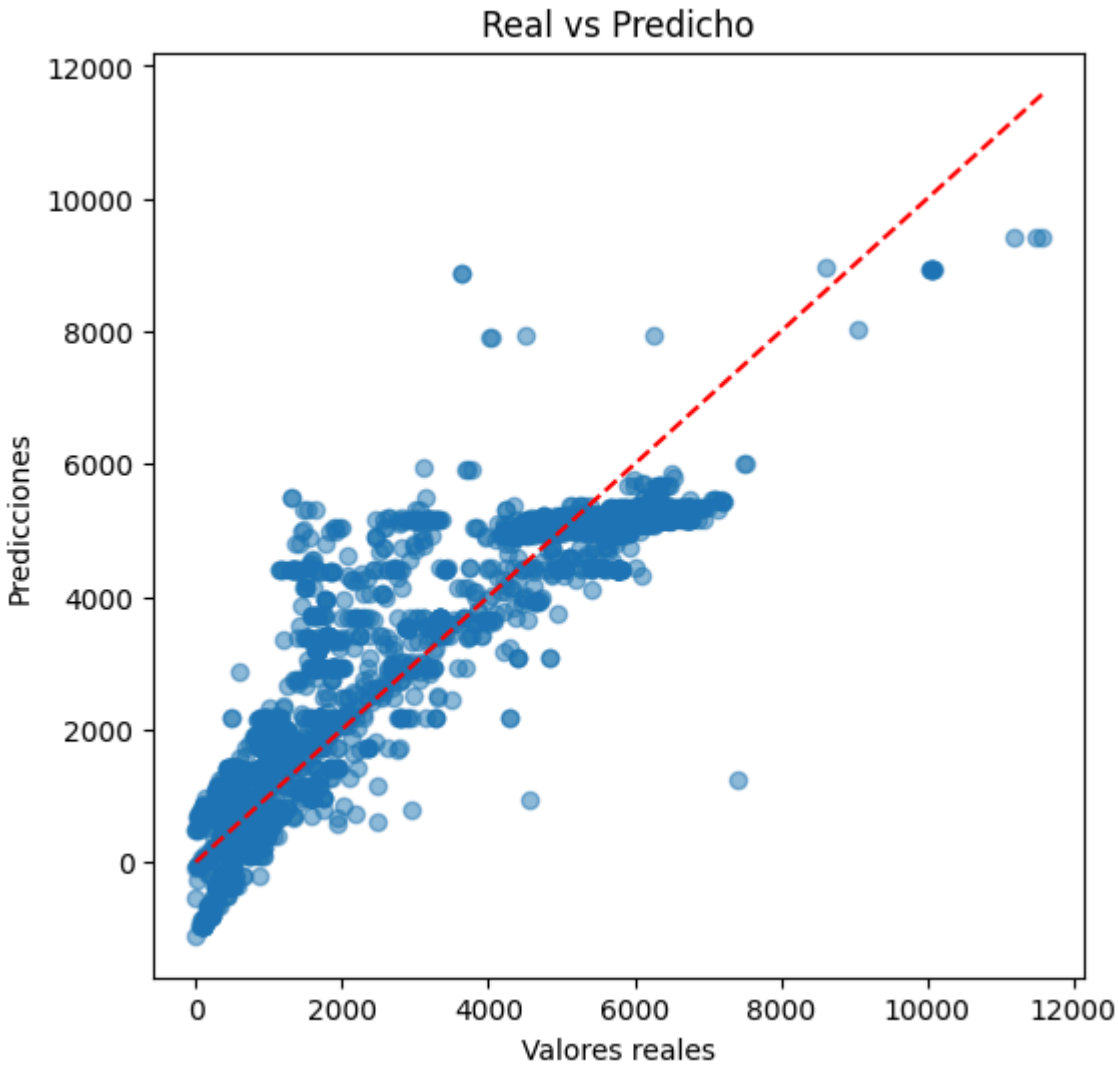
In [34]:
```
y_pred = modelo.predict(X_test)
```

Evaluación del modelo (Regresión o Clasificación)

In [35]:
```
from sklearn.metrics import mean_squared_error, r2_score

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Error cuadrático medio (MSE): {mse:.2f}')
print(f'R²: {r2:.2f}')
```

```
Error cuadrático medio (MSE): 920982.03
R²: 0.82
```

In [36]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(6,6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '--r')
plt.xlabel('Valores reales')
plt.ylabel('Predicciones')
plt.title('Real vs Predicho')
plt.show()
```



OBSERVACIONES:

- La mayoría de puntos reflejan estar alrededor de la línea roja, lo cual indica que el modelo generaliza bien en los datos de prueba.
- Se observa la presencia de puntos alejados, sobre todo en montos altos, lo que puede deberse a valores atípicos o casos especiales.
- El modelo tiene un comportamiento aceptable, pero se puede mejorar para ajustar mejor los extremos.

Mejora del modelo (Regresión o Clasificación)

In [37]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.ensemble import RandomForestRegressor
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

modelo_rforest = RandomForestRegressor().fit(X_train_scaled, y_train)
y_pred_rforest = modelo_rforest.predict(X_test_scaled)

modelo_lin = LinearRegression().fit(X_train_scaled, y_train)
y_pred_lin = modelo_lin.predict(X_test_scaled)

modelo_ridge = Ridge(alpha=1.0).fit(X_train_scaled, y_train)
y_pred_ridge = modelo_ridge.predict(X_test_scaled)

modelo_lasso = Lasso(alpha=0.1, max_iter=5000).fit(X_train_scaled, y_train)
y_pred_lasso = modelo_lasso.predict(X_test_scaled)

print(f"RandomForestRegressor: MSE: {mean_squared_error(y_test, y_pred_rforest):.2f}, R2: {r2_
print(f"Regresión lineal: MSE: {mean_squared_error(y_test, y_pred_lin):.2f}, R2: {r2_score(y_
print(f"Ridge:            MSE: {mean_squared_error(y_test, y_pred_ridge):.2f}, R2: {r2_score(
print(f"Lasso:            MSE: {mean_squared_error(y_test, y_pred_lasso):.2f}, R2: {r2_score(
```

```
RandomForestRegressor: MSE: 313031.05, R2: 0.94
Regresión lineal: MSE: 920982.03, R2: 0.82
Ridge:            MSE: 920981.07, R2: 0.82
Lasso:            MSE: 920981.62, R2: 0.82
```

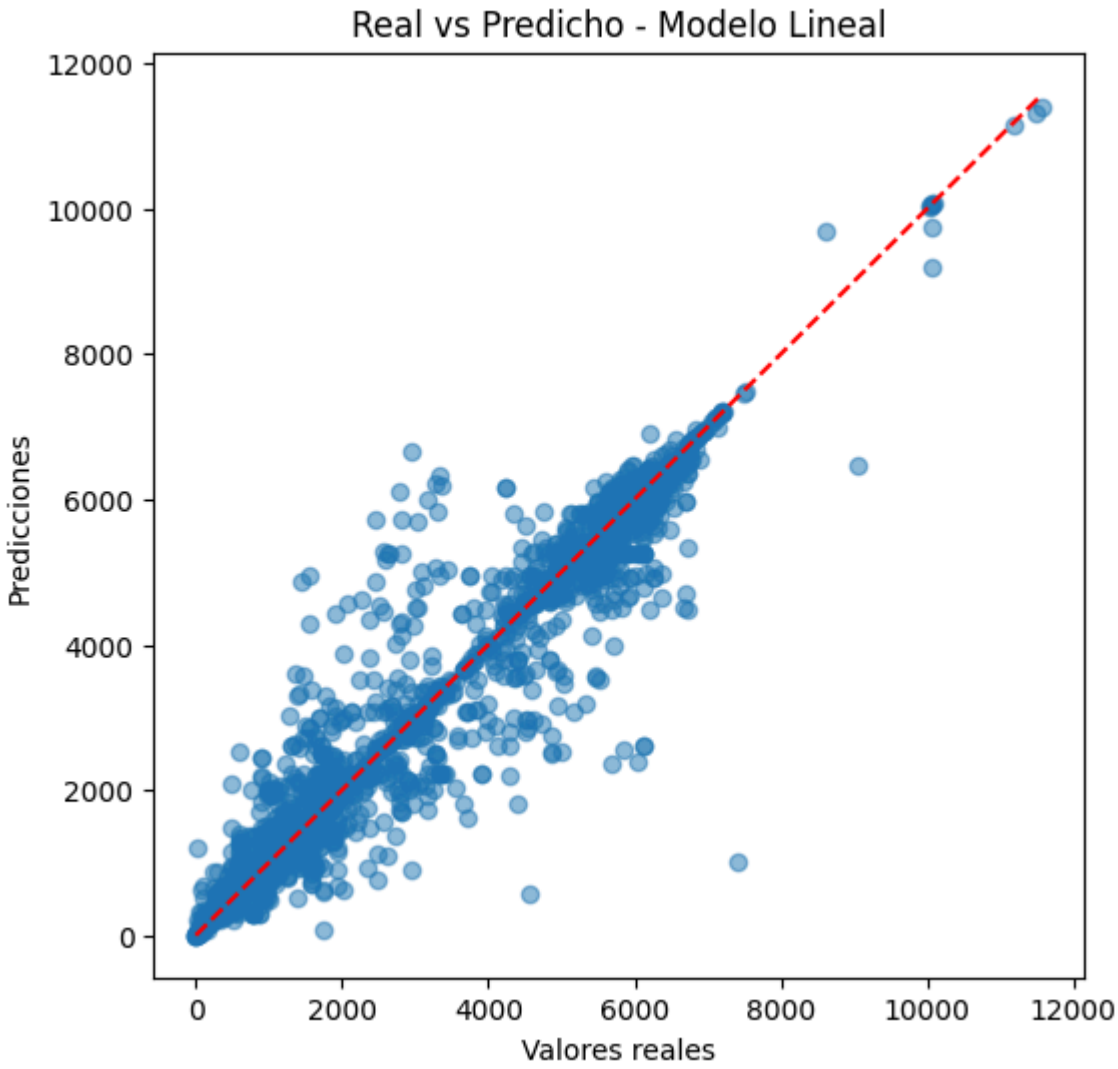
OBSERVACIONES:

El modelo Random Forest, muestra mayor valor de determinación R^2: 0.91 (explica el 91% de la variabilidad del Monto Toral). La diferencia en los MSE es de apenas unas pocas unidades sobre un millón (menos de un 0.001 %), excepto a comparación del Modelo Random Forest.

Evaluación del modelo (Regresión o Clasificación)

In [38]:

```
plt.figure(figsize=(6,6))
plt.scatter(y_test, y_pred_rforest, alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '--r')
plt.xlabel('Valores reales')
plt.ylabel('Predicciones')
plt.title('Real vs Predicho - Modelo Lineal')
plt.show()
```



OBSERVACIONES:

- Al volver a con el nuevo modelo (Random Forest), los resultados mejoraron significativamente, al mostrarse el compartiento de los datos pegados a la linea respecto al modelo, sin embargo los predicción están un poco alejadas a la realidad.
- Esto confirma que el modelo Random Forest es el más adecuado para explicar la relación entre las variables y el Monto USD.

Comentario o Recomendaciones (De acuerdo al modelo elaborado)

- Los factores más importantes que impactan en el Monto Total USD son el **Sector Cliente y Ciudad**.
- Con la variable fecha se podría predecir en un futuro los picos más altos de ventas para cada temporada.
- Podría analizarle de forma estratificada, ya que la mayor concentración de las ventas se registraron en la Región de Lima.