*Using the data from the previous session OR from the ORACLE database, select and implement the most adequate type of visualization to answer the following questions from the EDA analysis that you had carried out. Remember to justify your choice:*

### • *How much profit do the 10 best clients generate compared to the rest?*

Podemos intentar usar esta tabla para ayudarnos a seleccionar que gráfico usar.

| | VISUALIZATION CONTEXT | Stacked Column Chart | Bubble Chart | Pie Chart |
|---|---|---|---|---|
| **Goal:** | Composition | fit | unfit | fit |
| | Comparison | fit | fit | unfit |
| **Interaction:** | Overview | acceptable | acceptable | fit |
| **User:** | Lay | fit | acceptable | fit |
| **Dimensionality:** | 2-dimensional | unfit | unfit | fit |
| | n-dimensional | fit | fit | unfit |
| **Cardinality:** | Low | fit | acceptable | fit |
| **Independent Type:** | Nominal | fit | unfit | fit |
| **Dependent Type:** | Ratio | fit | fit | fit |

*Ilustración 1*

Al ser una comparación, con más de dos dimensiones, baja cardinalidad podríamos usar un grafo de barras.

```
# Crear el DataFrame
import seaborn as sns

df_plot = pd.DataFrame({
    "Grupo": ["Top 10 Clientes", "Resto de Clientes"],
    "Total Profit": [total_profit_top10,total_profit_rest ]
}
)

# Crear el gráfico de barras
barplot = sns.barplot(data=df_plot, x="Grupo", y="Total Profit", color="steelblue")
barplot.set_title("Comparación del Profit Total: Top 10 Clientes vs Resto")
barplot.set_ylabel("Total Profit (um)")
barplot.set_xlabel("")
barplot.tick_params(axis='x', rotation=45)
barplot
```

*Ilustración 2*

Creamos un *dataframe* con los grupos y los *profits* totales calculados que ya teníamos calculado en de la tarea 4. Posteriormente creamos el gráfico de barras.


*Ilustración 3*

Como vemos el resto de los clientes es un porcentaje mucho mayor que el del resto de clientes, pero para para la gran cantidad de clientes, se podría decir que los 10 primeros aportan bastante *Profit*. En este caso en especial al ser una composición sería más apropiado usar un grafo de barras apiladas (***stacked bar***).

```
#Gráfico Stack bars
# Crear gráfico de barras apiladas
plt.figure(figsize=(3, 6))
plt.bar(["Clientes"], [total_profit_top10], label="Top 10 Clientes", color="red")
plt.bar(["Clientes"], [total_profit_rest], bottom=[total_profit_top10], label="Resto de Clientes", color="steelblue")

# Agregar etiquetas
plt.title("Distribución del Profit Total (Stacked Bar)")
plt.ylabel("Total Profit (um)")

plt.legend()
plt.show()
```

*Ilustración 4*

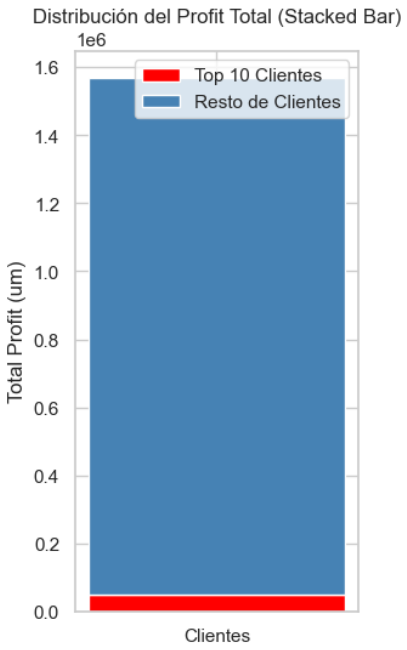



*Ilustración 5*

Este gráfico es útil porque puedes ver el profit total y entender fácilmente la magnitud que representa cada uno.

También podríamos crear un gráfico de tarta (pie):

```
#Crear pies gráfico
import matplotlib.pyplot as plt

plt.figure(figsize=(6, 6))
plt.pie(df_plot["Total Profit"], labels=df_plot["Grupo"], autopct='%1.1f%%', colors=["steelblue", "lightgray"], startangle=140)
plt.title("Distribución del Profit: Top 10 Clientes vs Resto")
plt.show()
```

*Ilustración 6*

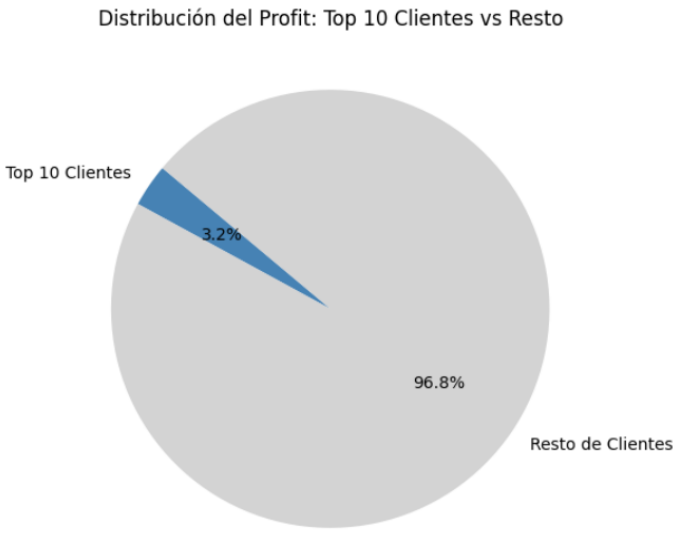



*Ilustración 7*

Aun así, creo que es el gráfico mas apropiado en este caso es el gráfico de barras apiladas o **stack bar**.

### • *How much is the average time and profit of packages according to their category?*

Una visualización muy interesante sería ver si el tiempo de preparación y el *profit* en conjunto. Para ello usamos un gráfico de burbujas para ver si encontramos algún patrón.

```
import matplotlib.pyplot as plt
import seaborn as sns


# Crear el gráfico de burbujas
plt.figure(figsize=(10, 6))
sns.scatterplot(
    data=df_profit_preparation_time,
    x="Preparation_Time",
    y="Profit",
    sizes=(20, 1000),
    hue="Category",

)
```

*Ilustración 8*

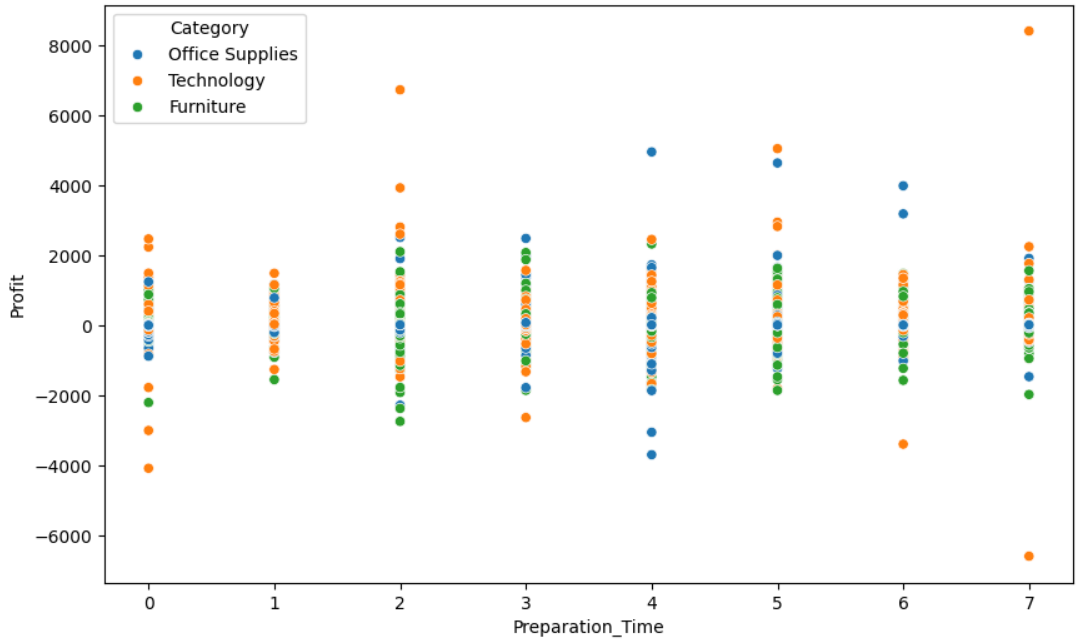



*Ilustración 9 Gráfico de puntos*

Como podemos ver el tiempo de preparación y el *profit* no parece que tengan una correlación. Usamos el color para representar la dimensión de la categoría. Su lectura es un poco difícil debido al gran volumen de datos, podíamos visualizarlo mejor viendo para cada tiempo de preparación la media del *profit* y tiempo de preparación por cada categoría.

```
df_grouped = df_profit_preparation_time.groupby("Category").agg(
    Profit_Avg=("Profit", "mean"),
    Preparation_Time_Avg=("Preparation_Time", "mean"),
    Count=("Category", "count")
).reset_index()


plt.figure(figsize=(10, 6))
sns.scatterplot(
    data=df_grouped,
    x="Preparation_Time_Avg",
    y="Profit_Avg",
    size="Count",
    hue="Category",
    sizes=(20, 1000),
)

plt.title("Tiempo vs Ganancia Promedio por Categoría")
plt.legend( bbox_to_anchor=(1.05, 1), loc="upper left")
```

*Ilustración 10*

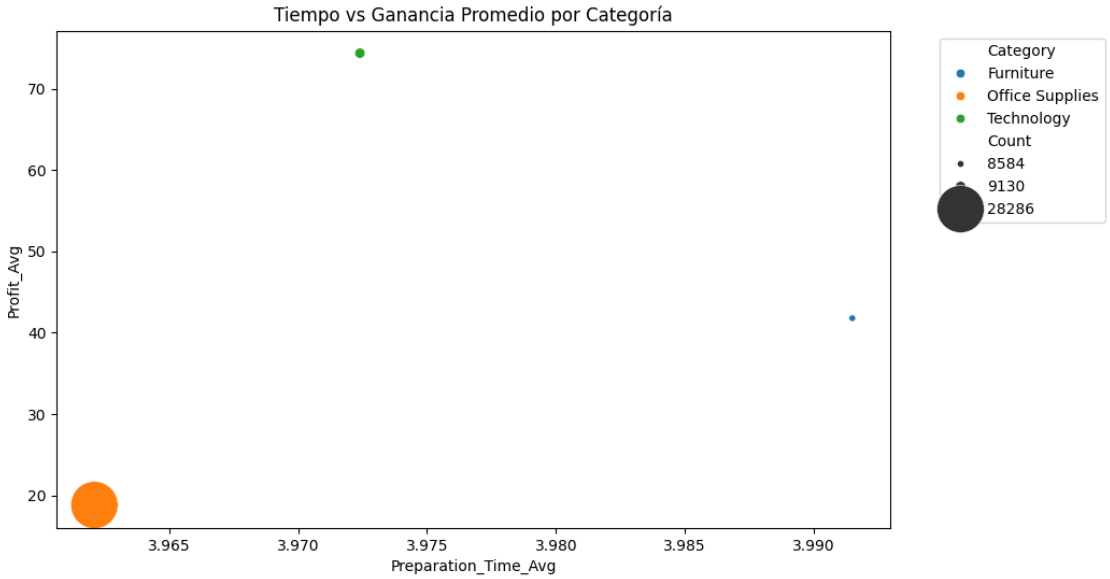



*Ilustración 11*

Con esta visualización podemos ver como los suministros de oficina son los mas se piden pero los que menos margen de *profit* dejan, lo cuál tiene sentido ya que

obtienen mas *profit* por la cantidad de pedidos a un menor precio que por ejemplo los pedidos tecnológicos que son mas caros, se venden menos pero cada uno deja producto deja más *profit*.

Si quisiéramos ver además como se comparan el *profit* y tiempo de preparación entre las categorías entre sí, podríamos usar gráficos de barras

```
cursor.execute("""
    SELECT P.CATEGORY, F.PROFIT, F.PREPARATION_TIME
    FROM FACT_ORDER F
    JOIN PRODUCT P ON F.PRODUCT_IDPRODUCT = P.IDPRODUCT
""")

results = cursor.fetchall()

df_profit_preparation_time = pd.DataFrame(results, columns=["Category", "PROFIT", "Preparation_Time"])
```

*Ilustración 12 Obtenemos los datos necesarios para hacer las vistas.*

```
# Calcular la media por categoría
df_avg = df.groupby("Category").agg({"Profit": "mean", "Preparation_Time": "mean"}).reset_index()
```

*Ilustración 13 Cálculo de media de profit*

```
sns.barplot(x="Category", y="Profit", data=df_avg)
```

```
<Axes: xlabel='Category', ylabel='Profit'>
```

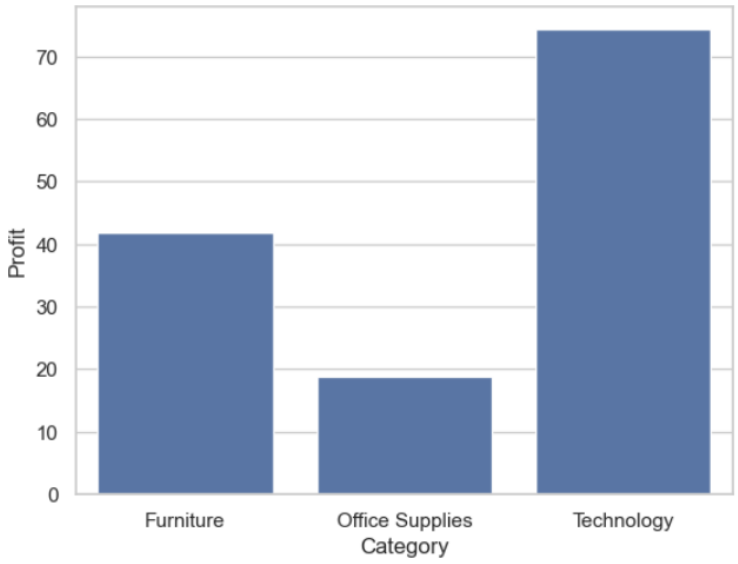



*Ilustración 14 Bar plot que compara el profit medio para cada categoría.*

```
sns.barplot(x="Category", y="Preparation_Time", data=df_avg)
```

```
<Axes: xlabel='Category', ylabel='Preparation_Time'>
```

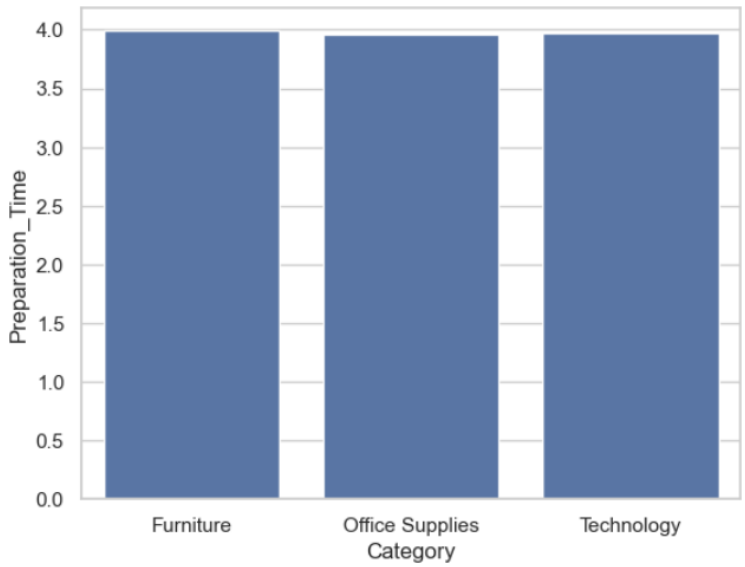



*Ilustración 15 Gráfico que compara el tiempo de preparación medio (días) para cada categoría*

Aunque no lo piden en el enunciado podríamos ponernos en el hipotético caso que quisiéramos estudiar la distribución del *profit* y del tiempo de preparación de cada categoría, podríamos obtener la mediana y los cuartiles y ver con facilidad los valores anómalos usando un gráfico de cajas y bigotes.

```
# Crear el gráfico de cajas para Profit por categoría

sns.boxplot(x="Category", y="Profit", data=df_profit_preparation_time)
```

```
<Axes: xlabel='Category', ylabel='Profit'>
```

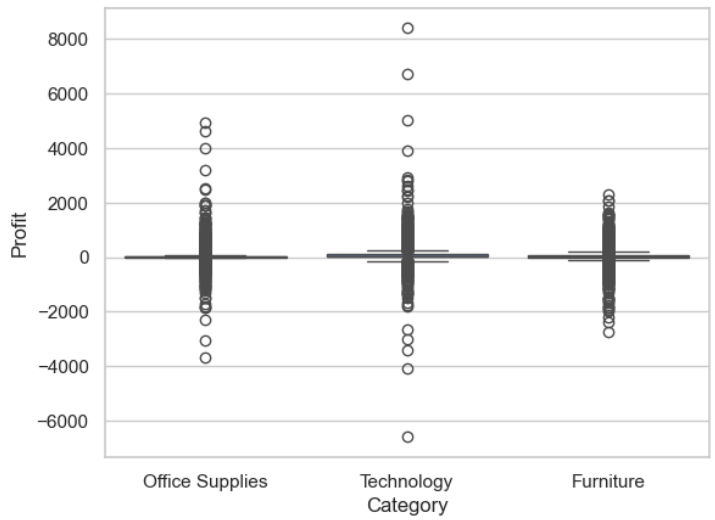



*Ilustración 16 Gráfico de cajas y bigotes del profit respecto a las categorías.*

```
# Crear el gráfico de cajas para Preparation_Time por categoría

sns.boxplot(x="Category", y="Preparation_Time", data=df_profit_preparation_time
```

```
<Axes: xlabel='Category', ylabel='Preparation_Time'>
```

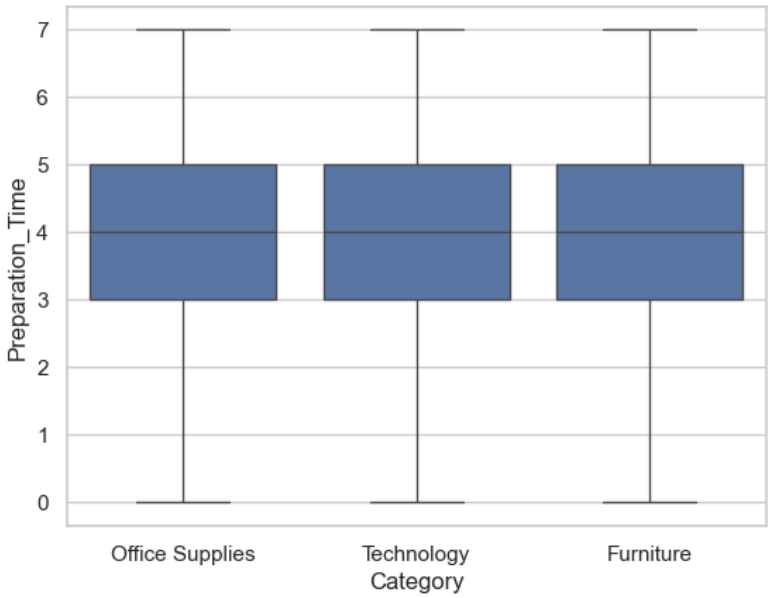



*Ilustración 17 Gráfico de cajas y bigotes del tiempo de preparación en días respecto a las categorías.*

*Focus now on the market that provides the greatest benefit:*

### • *How has profit evolved over time in this market?*

Los gráficos de líneas son ideales para representar la evolución de un valor cardinal a lo largo del tiempo.

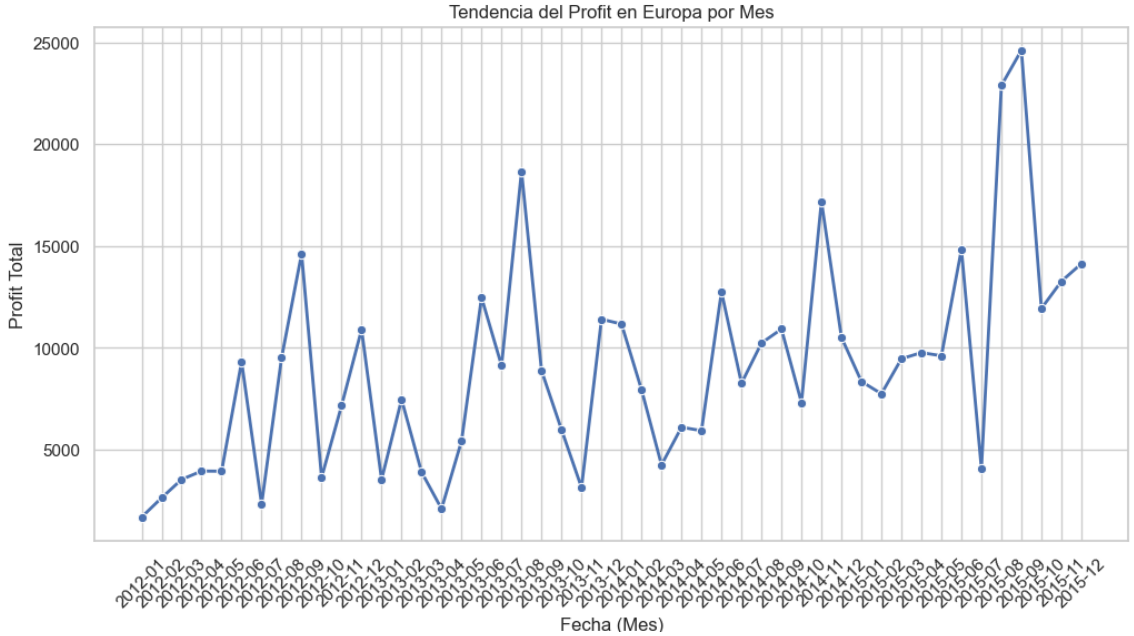
```
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
sns.lineplot(data=df_profit_trend, x="ORDER_DATE", y="PROFIT", marker="o", linewidth=2)

plt.title("Tendencia del Profit en Europa por Mes")
plt.xlabel("Fecha (Mes)")
plt.ylabel("Profit Total")
plt.xticks(rotation=45)  # Rotar etiquetas para mejor legibilidad
plt.grid(True)
```

*Ilustración 18*

El *dataframe* es el *profit* para Europa agrupado por mes que se había obtenido en la tarea anterior. Se combina la librería *seaborn* con *matplotlib* para rotar las etiquetas del eje x y que se visualice mejor el *profit* en cada mes. El resultado es el siguiente:




### • *What is the average quantity ordered in each region in this market according to the order category?*

Esta visualización ha de ser una comparación entre regiones y categorías (**que asumimos que se refiere al modo de envío y no a la categoría del producto**). Como los tipos independientes son región y categoría el *stacked columna chart* es aplicable. La visualización con el gráfico *pie* no es adecuada por comparaciones

con datos n-dimensionales y el gráfico de burbujas es menos apropiado debido a la dimensionalidad y tipos dependientes.

```
cursor.execute("""
     SELECT S.SHIPMENT_MODE,C.REGION, COUNT(*) AS "NUMBER OF ORDERS"
    FROM FACT_ORDER F
    JOIN SHIPMENT S ON F.SHIPMENT_IDSHIPMENT = S.IDSHIPMENT
    JOIN  CUSTOMER C ON F.CUSTOMER_IDCUSTOMER = C.IDCUSTOMER
    WHERE C.MARKET = 'Europe'
    GROUP BY S.SHIPMENT_MODE, C.REGION
""")

results = cursor.fetchall()

df_shipment_market = pd.DataFrame(results, columns=["Shipment_Mode", "Region", "Number_orders"])


# Gráfico de barras agrupadas
sns.barplot(data=df_shipment_market, x="Region", y="Number_orders", hue="Shipment_Mode")
```

*Ilustración 19*

Nos ayudamos de la consulta para obtener el número de pedidos agrupados por modo de envío y región para el mercado de Europa.

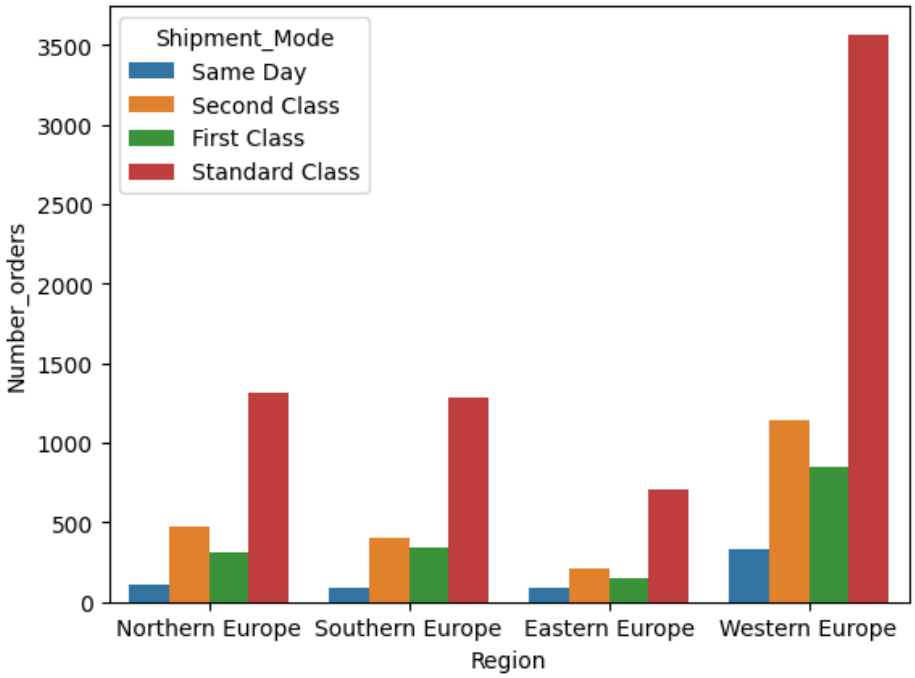



*Ilustración 20*

Como vemos hay una gran diferencia de pedidos en cuando a Westen Europe para los envíos de clase estándar.

*Once you have finished these questions, look at the* ***multidimensional schema*** *of the repository.* ***Propose and answer another analytical question*** *using visualizations* ***that involve at least 2 dimensions and one measure.***

Yo propongo responder a la pregunta analítica sobre visualizar el número de productos devueltos a lo largo del tiempo.

```
cursor.execute("""
    SELECT  D.MONTH, D.YEAR, COUNT(*) AS "NUMBER OF ORDERS RETURNED"
    FROM FACT_ORDER O
    JOIN RETURNS R ON O.RETURNS_IDRETURNS = R.IDRETURNS
    JOIN DATES D ON O.DATE_IDORDER = D.IDDATE
    WHERE Upper(R.RETURNED) = 'RETURNED'
    GROUP BY D.MONTH,D.YEAR
""")

data = cursor.fetchall()
columns = [desc[0] for desc in cursor.description]
df_returns = pd.DataFrame(data, columns=columns)

df_returns = df_returns.rename(columns={"MONTH": "month", "YEAR": "year", "NUMBER OF ORDERS RETURNED": "returns"})

df_returns["ORDER_DATE"] = pd.to_datetime(df_returns[["year", "month"]].astype(str).agg('-'.join, axis=1), format="%Y-%m")

df_returns = df_returns.sort_values("ORDER_DATE")

df_returns["ORDER_DATE"] = df_returns["ORDER_DATE"].dt.strftime("%Y-%m")

plt.figure(figsize=(12, 6))
sns.lineplot(data=df_returns, x="ORDER_DATE", y="returns", marker="o", linewidth=2)


plt.xticks(rotation=45)
plt.grid(True)
```

*Ilustración 21*

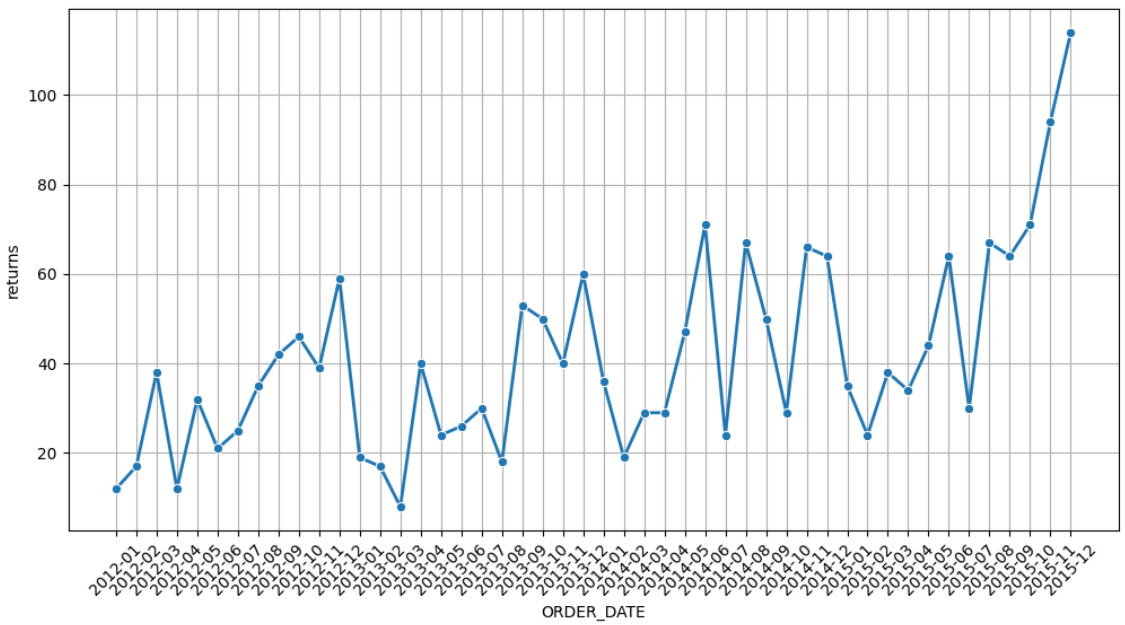



*Ilustración 22*

Parece ser que cada vez el número de devoluciones tiende a crecer, lo cuál tiene sentido porque el número de pedidos también tendía a crecer y probablemente estén mas o menos correlacionados.