En mi caso, voy a usar los datos guardados en Oracle. Me he dado cuenta de que he tenido un problema de espacio por lo que solo se han guardado alrededor de 46000 filas de 51292 que había en el csv. Por simplicidad, se hará el *Data profiling & Eda* con las 46000 filas guardadas en la base de datos.

Is each row an individual and complete shipping order?

```
#Is each row an individual and complete shipping order?
```

```
import oracledb
```

```
connection = oracledb.connect(user="UBD1463", password="cg6nx9CngXtZ", host="diana.lcc.uma.es", sid="ATENEA")
cursor = connection.cursor()
fact_order = cursor.execute("SELECT * FROM FACT_ORDER")

# Obtener los nombres de las columnas
columns = [col[0] for col in cursor.description]

# Recuperar los datos
rows = cursor.fetchall()
```

```
import pandas as pd

# Crear el DataFrame
df_fact_order = pd.DataFrame(rows, columns=columns)
```

Ilustración 1

Se utiliza la librería de *oracledb* para conectarse a la base de datos. Usamos la librería de pandas para crear un *dataframe*.

df_fact_order

	IDFACT_ORDER	PRODUCT_IDPRODUCT	DATE_IDSHIP	DATE_IDORDER	SHIPMENT_IDSHIPMENT	PRIORITY_IDPRIORITY	CUSTOMER_IDCUSTOMER	RETURNS_IDRET
0	248	2592	1216	1210	4	4	11491	
1	249	1639	1216	1210	4	4	11491	
2	250	3546	1571	1571	2	1	6428	
3	251	3409	1820	1815	4	4	12287	
4	252	3574	1687	1685	1	1	16923	
...	...	...	...	...	...	...	...	
45995	43834	269	1857	1852	3	2	13779	
45996	43835	59	1857	1852	3	2	13779	
45997	43836	359	1857	1852	3	2	13779	
45998	43837	1267	1857	1852	3	2	13779	
45999	43838	1785	1857	1852	3	2	13779	

Ilustración 2

ENT_IDSHIPMENT	PRIORITY_IDPRIORITY	CUSTOMER_IDCUSTOMER	RETURNS_IDRETURNS	SALES	QUANTITY	DISCOUNT	PROFIT	SHIPPING_COST	PREPARATION_TIME
4	4	11491	2	105.72	4.0	0.0	43.32	6.73	6.0
4	4	11491	2	6.45	1.0	0.0	1.65	1.37	6.0
2	1	6428	2	182.07	1.0	0.0	40.05	59.24	0.0
4	4	12287	2	85.26	1.0	0.0	0.84	4.06	5.0
1	1	16923	2	553.92	4.0	0.0	22.08	206.56	2.0
...	...	...	...	...	...	...	...	...	...
3	2	13779	2	332.28	2.0	0.0	43.14	60.06	5.0
3	2	13779	2	362.40	1.0	0.0	39.84	42.22	5.0
3	2	13779	2	132.48	1.0	0.0	14.55	18.63	5.0
3	2	13779	2	103.56	2.0	0.0	38.28	8.12	5.0
3	2	13779	2	19.29	1.0	0.0	1.14	3.95	5.0

Ilustración 3

Si visualizamos la tabla vemos atributos como la fecha de envío y la fecha del pedido, coste del envío, tiempo de preparación, precio, … Por lo que se puede intuir que cada fila representa una orden desde que un cliente la pide hasta que es enviado.

How many customers are in the data set?

```
#How many customers are in the data set?
```

```
cursor.execute("SELECT COUNT(*) FROM CUSTOMER")
customers_numbers = cursor.fetchone()[0]
customers_numbers
```

```
17415
```

Ilustración 4

Se hace una consulta en la base de datos para obtener **17415**.

How many orders does each customer place on average?

```
#How many orders does each customer place on average?
```

```
cursor.execute("""
    SELECT AVG(order_count)
    FROM (
        SELECT CUSTOMER_IDCUSTOMER, COUNT(*) AS order_count
        FROM FACT_ORDER
        GROUP BY CUSTOMER_IDCUSTOMER
    )
""")
mean_orders = cursor.fetchone()[0]
mean_orders
```

```
2.8961783038468805
```

Ilustración 5

Al hacer la consulta podemos ver que sobre unos 2.89 pedidos cada cliente de media.

Are there columns with missing values? Which ones? Are they critical to the analysis we are doing?

```
cursor.execute("""
    SELECT
        C.*, P.*,
        D.YEAR AS YEAR_ORDER, D.MONTH AS MONTH_ORDER, D.DAY AS DAY_ORDER,
        D2.YEAR AS YEAR_SHIP, D2.MONTH AS MONTH_SHIP, D2.DAY AS DAY_SHIP,
        PR.*,
        S.*,
        R.*
    FROM FACT_ORDER O
        JOIN  CUSTOMER C ON O.CUSTOMER_IDCUSTOMER = C.IDCUSTOMER
        JOIN PRODUCT P ON O.PRODUCT_IDPRODUCT = P.IDPRODUCT
        JOIN  DATES D ON O.DATE_IDORDER = D.IDDATE
        JOIN DATES D2 ON O.DATE_IDSHIP = D2.IDDATE
        JOIN PRIORITY PR ON O.PRIORITY_IDPRIORITY = PR.IDPRIORITY
        JOIN SHIPMENT S ON O.SHIPMENT_IDSHIPMENT = S.IDSHIPMENT
        JOIN RETURNS R ON O.RETURNS_IDRETURNS = R.IDRETURNS
""")

# Obtener los nombres de las columnas
columns = [col[0] for col in cursor.description]

# Recuperar los datos
rows = cursor.fetchall()

df_complete = pd.DataFrame(rows, columns=columns)
```

Ilustración 6

```
from ydata_profiling import ProfileReport
report = ProfileReport(df_complete, title="Orders profiling", correlations={"auto": {"calculate": True}}, minimal=True)
report
```

Ilustración 7

Podemos hacer un reporte con la librería *ydata_profiling*. Para hacer el reporte obtengo los valores de las dimensiones.

POSTAL_CODE

Text

Missing

Distinct	488
Distinct (%)	5.9%
Missing	37692
Missing (%)	81.9%
Memory size	359.5 KiB

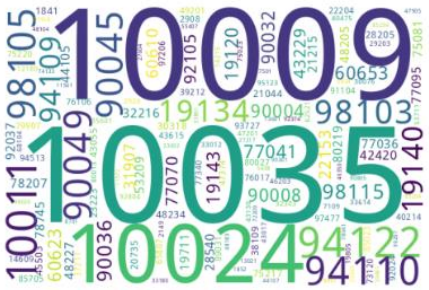

Ilustración 8

Vemos claramente que el 81.2% de códigos postales faltan. En el análisis que estamos haciendo no nos es relevante estos datos, podríamos obviarlo.

Are there columns highly correlated? Which ones? If there are correlations, do they all make sense?

```
# Calcular la matriz de correlación
corr_matrix = df_fact_order.corr()

corr_matrix
```

	IDFACT_ORDER	PRODUCT_IDPRODUCT	DATE_IDSHIP	DATE_IDORDER	SHIPMENT_IDSHIPMENT	PRIORITY_IDPRIORITY	CUSTOMER_IDCUSTOME
IDFACT_ORDER	1.000000	0.012675	0.130995	0.130982	0.009403	0.005982	0.0329
PRODUCT_IDPRODUCT	0.012675	1.000000	-0.004530	-0.004522	0.002939	-0.004896	-0.0051
DATE_IDSHIP	0.130995	-0.004530	1.000000	0.999991	0.008762	0.020916	-0.0138
DATE_IDORDER	0.130982	-0.004522	0.999991	1.000000	0.005746	0.018881	-0.0138
SHIPMENT_IDSHIPMENT	0.009403	0.002939	0.008762	0.005746	1.000000	0.403242	-0.0001
PRIORITY_IDPRIORITY	0.005982	-0.004896	0.020916	0.018881	0.403242	1.000000	-0.0117
CUSTOMER_IDCUSTOMER	0.032946	-0.005119	-0.013858	-0.013859	-0.000135	-0.011700	1.0000
RETURNS_IDRETURNS	0.009249	-0.002713	-0.001102	-0.001113	-0.001540	0.009182	0.0051
SALES	-0.031842	0.019997	-0.001950	-0.001933	0.001207	0.000787	0.0000
QUANTITY	-0.085249	-0.004101	0.002443	0.002420	0.006497	0.005199	0.0024
DISCOUNT	0.116681	-0.031370	-0.010399	-0.010381	-0.013047	0.005034	-0.0014
PROFIT	-0.044356	0.035485	0.004506	0.004499	0.002711	0.002189	-0.0028
SHIPPING_COST	-0.020189	0.015405	-0.002619	-0.002021	-0.146077	-0.188742	0.0016
PREPARATION_TIME	0.003649	-0.002025	0.006070	0.001882	0.720089	0.486162	0.0001

Ilustración 9

Calculamos la matriz de correlación. En primer lugar, de forma trivial se puede decir que una columna tiene una correlación 100% consigo misma. Si consideramos una correlación alta a partir un 70% podemos ver que hay una correlación alta con:

- la fecha de pedido y fecha de envío
- el coste del envío y con el precio
- el id del tipo del pedido con el tiempo de preparación.

Realizamos un filtrado para verlo de forma más cómoda.

```
# Convertir la matriz en un formato de tabla
corr_pairs = corr_matrix.unstack().reset_index()
corr_pairs.columns = ["Variable 1", "Variable 2", "Correlation"]

# Filtrar pares con alta correlación (excluyendo correlaciones de 1 con sí mismo)
high_corr = corr_pairs[(corr_pairs["Correlation"].abs() > 0.7) & (corr_pairs["Variable 1"] != corr_pairs["Variable 2"])]

# Ordenar por correlación en orden descendente
high_corr = high_corr.sort_values(by="Correlation", ascending=False)

high_corr
```

	Variable 1	Variable 2	Correlation
31	DATE_IDSHIP	DATE_IDORDER	0.999991
44	DATE_IDORDER	DATE_IDSHIP	0.999991
124	SALES	SHIPPING_COST	0.764942
176	SHIPPING_COST	SALES	0.764942
69	SHIPMENT_IDSHIPMENT	PREPARATION_TIME	0.720089
186	PREPARATION_TIME	SHIPMENT_IDSHIPMENT	0.720089

Ilustración 10

En primer lugar, tiene sentido que las fechas de pedido y de envío muestren correlación. Ya que lo normal es que la fecha de envío sea poco después de la fecha de pedido para todas las fechas de pedido

Por otra parte, también puede tener sentido que el precio de envío esté algo relacionado con el precio total, ya que muchas empresas ajustan el coste de envío al precio total de compra o productos más caros pueden ser más pesados o frágiles y al ofrecer un envío más seguro puede ser más caro.

La correlación entre el id del *shipment* y el tiempo de preparación también tiene sentido. Un envío *standart* de media debería ser mas lento que un envío en *first class*, por lo que es normal que exista cierta correlación.

After the initial data profiling task, we are going to carry out a brief **Exploratory Data Analysis (EDA) to understand the data at a deeper level.** Perform the necessary operations to answer the following questions:

How much is the average shipping time of packages according to their ship mode?

```
cursor.execute("""
SELECT
    S.SHIPMENT_MODE,
    AVG(O.PREPARATION_TIME) AS  "AVG PREPARING THE SHIPMENT (DAYS)"
FROM FACT_ORDER O JOIN SHIPMENT S ON O.SHIPMENT_IDSHIPMENT = S.IDSHIPMENT
GROUP BY S.SHIPMENT_MODE
""")
# Obtener los nombres de las columnas
columns = [col[0] for col in cursor.description]

# Recuperar los datos
rows = cursor.fetchall()

df_shipment_mode_avgs = pd.DataFrame(rows, columns=columns)
df_shipment_mode_avgs
```

	SHIPMENT_MODE	AVG PREPARING THE SHIPMENT (DAYS)
0	Same Day	0.037344
1	Second Class	3.234119
2	Standard Class	4.994107
3	First Class	2.171254

Ilustración 11

Obtenemos con una consulta las medias en días de tiempo de preparación, a simple vista parecen correctas. Como vemos si es el mismo día el tiempo es menor a 1 día. *First class* tarda alrededor de 2 días, *Second class* tarda alrededor de 3 días y *Standard class* pues tarda casi 5 días de media.

Vamos a comprobar que no exista ningún tiempo de preparación negativo.

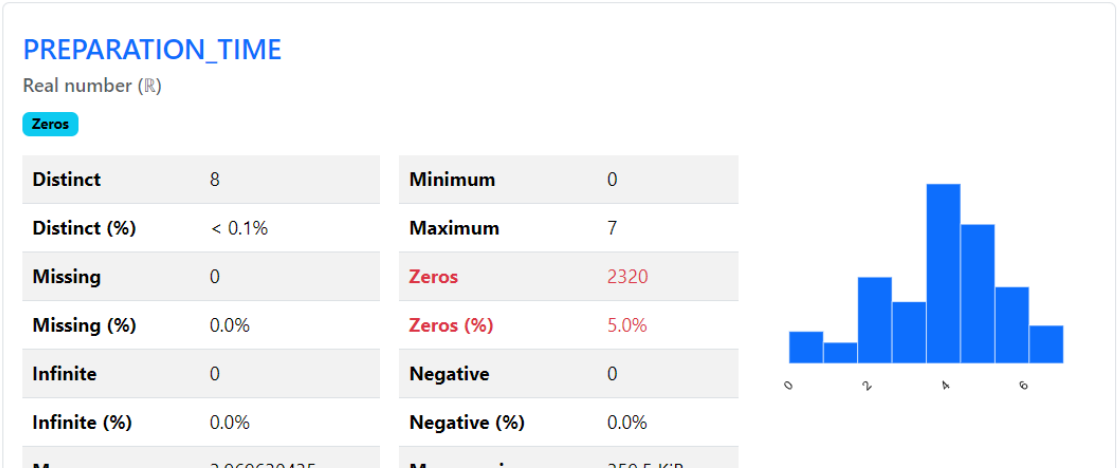

Ilustración 12

Comprobando el reporte general que hemos hecho antes podemos ver que no hay valores negativos, pero si hay muchos valores que son 0. Podríamos retirarlos y volver a calcular la media. Sin embargo, como entendemos que el valor que se guarda es en días, puede tener sentido sobre todo para envíos de tipo *Same Day*.

How much is the average profit that the company obtains per package ship mode? Is there any case where packages incur losses?

```
# How much is the average profit that the company obtains per package ship mode? Is there any case where packages incur in loses?

cursor.execute("""
SELECT
    S.SHIPMENT_MODE,
    AVG(O.PREPARATION_TIME) AS "AVG PREPARING THE SHIPMENT (DAYS)",
    AVG(O.PROFIT) AS "AVG PROFIT PER SHIPMENT"
FROM FACT_ORDER O
JOIN SHIPMENT S ON O.SHIPMENT_IDSHIPMENT = S.IDSHIPMENT
GROUP BY S.SHIPMENT_MODE
ORDER BY "AVG PROFIT PER SHIPMENT" DESC
""")
# Obtener los nombres de las columnas
columns = [col[0] for col in cursor.description]

# Recuperar los datos
rows = cursor.fetchall()

df_shipment_mode_avgs = pd.DataFrame(rows, columns=columns)
df_shipment_mode_avgs
```

Ilustración 13

Se obtienen con una consulta SQL la media para cada modo de envío y lo ordenamos de mayor a menor.

```
df_fact_order_grouped_by_shipment = (
    df_fact_order.groupby(['SHIPMENT_IDSHIPMENT'])['PROFIT']
    .agg(["min","mean","max"])
    .sort_values(by=[('mean')], ascending=True)

)
df_fact_order_grouped_by_shipment
```

SHIPMENT_IDSHIPMENT	min	mean	max
1	-2639.9912	32.830768	6719.9808
3	-2750.2800	34.143650	2799.9840
4	-6599.9780	34.380594	8399.9760
2	-4088.3760	34.552228	2461.3200

Ilustración 14

Como vemos el *profit* de medio de cada modo de envío es más o menos similar. Sin embargo, vamos a ver si existen valores anómalos para el *profit*.

Al ver los valores mínimos y máximos de *profit* podemos ver valores de hasta - 6599.9790 para el *profit* de envíos de clase *standart*. Es posible que estos valores hayan sido apuntados mal, o existan productos muy poco rentables y otros muy muy rentables. Para sacar una aproximación más real a la media podemos probar a retirar valores extremos. Filtramos para obtener los valores entre el percentil 5 y percentil 95:

```
#Retiro valores anómalos
# Calcular los percentiles 5 y 95 para la columna PROFIT
p5 = df_fact_order["PROFIT"].quantile(0.05)
p95 = df_fact_order["PROFIT"].quantile(0.95)

# Filtrar el DataFrame para mantener solo los valores entre los percentiles 5 y 95
df_filtered = df_fact_order[(df_fact_order["PROFIT"] >= p5) & (df_fact_order["PROFIT"] <= p95)]

# Agrupar por SHIPMENT_IDSHIPMENT y calcular min, mean, max después del filtrado
df_filtered_grouped = df_filtered.groupby(['SHIPMENT_IDSHIPMENT'])['PROFIT'].agg(["min", "mean", "max"])
df_filtered_grouped
```

SHIPMENT_IDSHIPMENT	min	mean	max
1	-68.376	24.490043	220.4800
2	-68.394	24.814737	219.5100
3	-68.430	25.553745	219.9904
4	-68.640	25.296196	220.0800

Ilustración 15

Las medias siguen siendo similares, lo cual tiene sentido, pero sale un vale inferior al de antes de media y valores mucho mas razonable en cuanto a *profit* se refiere para los productos.

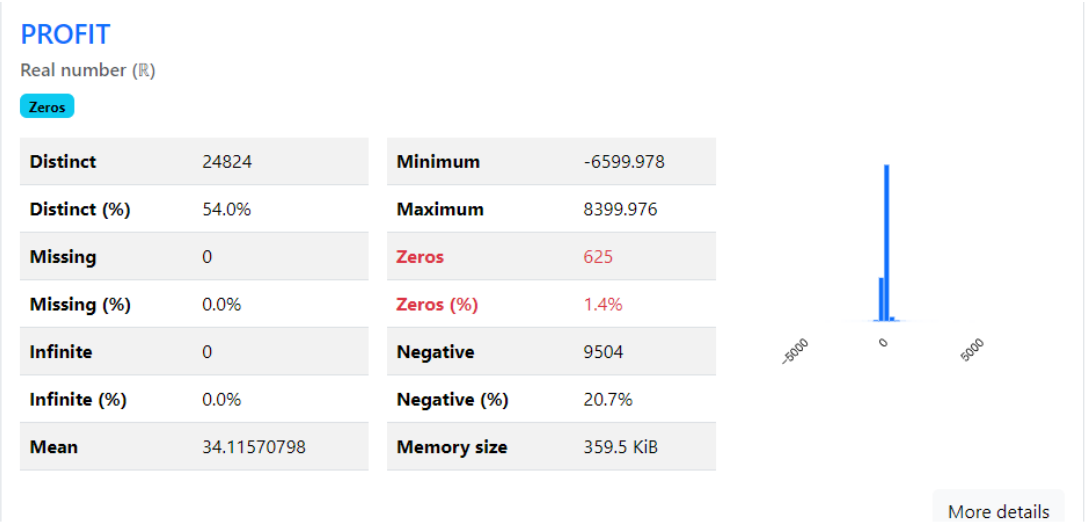

Ilustración 16

Se observa que un número significativo de pedidos generan pérdidas. Según el reporte general, al menos 9,504 pedidos presentan un profit negativo, lo que podría deberse a errores en los datos, aunque no necesariamente implica que sean incorrectos.

Una posible explicación es que el costo de producción o preparación del envío supere el precio de venta final, lo que puede parecer un error, pero no siempre lo es. Muchas empresas adoptan estrategias en las que asumen pérdidas en ciertos productos para maximizar ganancias en otros.

Por ejemplo, una maquinilla de afeitar que se vende a un precio bajo, incluso generando pérdidas. Sin embargo, el verdadero beneficio viene después, con la venta recurrente de cuchillas de recambio, compensando con creces la pérdida inicial.

How much profit do the 10 best clients (profit-wise) generate compared to the rest?

```
cursor.execute("""
 SELECT C.*, FO.TOTAL_PROFIT
FROM CUSTOMER C
JOIN (
    SELECT CUSTOMER_IDCUSTOMER, SUM(PROFIT) AS TOTAL_PROFIT
    FROM FACT_ORDER
    GROUP BY CUSTOMER_IDCUSTOMER
    ORDER BY TOTAL_PROFIT DESC
    FETCH FIRST 10 ROWS ONLY
) FO
ON C.IDCUSTOMER = FO.CUSTOMER_IDCUSTOMER
ORDER BY FO.TOTAL_PROFIT DESC


""")
# Obtener los nombres de las columnas
columns = [col[0] for col in cursor.description]

# Recuperar los datos
rows = cursor.fetchall()

df_customers_bests = pd.DataFrame(rows, columns=columns)
df_customers_bests
```

Ilustración 17

Con esta consulta puedo obtener el resultado de los 10 clientes que más *profit* han generado. El resultado obtenido es:

	IDCUSTOMER	CUSTOMER_NUMBER	NAME	COUNTRY	CUSTOMER_TYPE	POSTAL_CODE	MARKET	REGION	STATE	CITY	TOTAL_PROFIT
0	16286	TC-209801402	Tamara Chand	United States	Corporate	47905	USCA	Central US	Indiana	Lafayette	8745.0635
1	13637	RB-193601404	Raymond Buch	United States	Consumer	98115	USCA	Central US	Washington	Seattle	6807.0879
2	137	AB-101051402	Adrian Barton	United States	Consumer	48205	USCA	Central US	Michigan	Detroit	5362.6135
3	7163	HL-150401406	Hunter Lopez	United States	Consumer	19711	USCA	Central US	Delaware	Newark	5045.8564
4	14744	SC-200951402	Sanjit Chand	United States	Consumer	55407	USCA	Central US	Minnesota	Minneapolis	4668.6935
5	16020	TA-213851406	Tom Ashbrook	United States	Home Office	10024	USCA	Central US	New York	New York City	4599.2073
6	13082	PJ-1883564	Patrick Jones	Italy	Corporate	None	Europe	Southern Europe	Tuscany	Prato	3982.0440
7	2706	CA-1277558	Cynthia Arntzen	India	Consumer	None	Asia Pacific	Southern Asia	Chhattisgarh	Kota	3981.7200
8	2573	BW-106533	Barry Weirich	Democratic Republic of the Congo	Consumer	None	Africa	Central Africa	Kasai-Occidental	Kananga	3261.9300
9	3470	CM-123851408	Christopher Martinez	United States	Consumer	30318	USCA	Central US	Georgia	Atlanta	3197.4580

Ilustración 18

Si quisiéramos compararlos con el resto de los clientes podríamos hacerlo de la siguiente forma:

```python
# Calcular la suma total del profit de los 10 mejores clientes
total_profit_top10 = df_customers_bests["TOTAL_PROFIT"].sum()

# Calcular la suma total del profit de todos los clientes
total_profit_all = df_fact_order["PROFIT"].sum()
# Calcular la suma total del profit del resto de clientes
total_profit_rest = total_profit_all - total_profit_top10
print ("customers total profit:", total_profit_top10)
print ("rest of the customers total profit:", total_profit_rest)
print ("Total profit:", total_profit_all)

# Calcular el porcentaje de profit generado por los 10 mejores clientes
percent_profit_top10 = (total_profit_top10 / total_profit_all) * 100
print("% of profit of 10 best customers:",percent_profit_top10)
# Calcular el porcentaje de clientes que representan los 10 mejores
total_customers = df_fact_order["CUSTOMER_IDCUSTOMER"].nunique()
percent_customers_top10 = (10 / customers_numbers) * 100
print("% of customers the represent the best:",percent_customers_top10)
```

```
customers total profit: 49651.6741
rest of the customers total profit: 1519670.89312
Total profit: 1569322.5672199999
% of profit of 10 best customers: 3.163892187439591
% of customers the represent the best: 0.057421762848119444
```

Ilustración 19

Para hacer esta comparación hacemos lo siguiente:

1. Calculamos la suma del *profit* que generan los 10 mejores clientes
2. Calculamos la suma del resto de clientes
3. De esta forma podemos ver que los mejores clientes representando tan solo un 0.57% de todos los clientes generan un 3.16% de todo el *profit*. Lo cual es bastante

Podemos visualizar un resumen:

```python
# Crear un DataFrame con los resultados
df_summary = pd.DataFrame({
    "Metric": ["Total Profit (Top 10)", "Total Profit (Rest)", "Percent Profit (Top 10)", "Percent Customers (Top 10)"],
    "Value": [total_profit_top10, total_profit_rest, percent_profit_top10, percent_customers_top10]
})
df_summary
```

	Metric	Value
0	Total Profit (Top 10)	4.965167e+04
1	Total Profit (Rest)	1.519671e+06
2	Percent Profit (Top 10)	3.163892e+00
3	Percent Customers (Top 10)	5.742176e-02

Ilustración 20

How much is the total profit per market?

```
cursor.execute("""
 SELECT C.MARKET, SUM(O.PROFIT) AS "TOTAL PROFIT" FROM
FACT_ORDER O JOIN CUSTOMER C ON O.CUSTOMER_IDCUSTOMER = C.IDCUSTOMER
GROUP BY C.MARKET
ORDER BY 2 DESC
""")
# Obtener los nombres de las columnas
columns = [col[0] for col in cursor.description]

# Recuperar los datos
rows = cursor.fetchall()

df_markets_best = pd.DataFrame(rows, columns=columns)
df_markets_best
```

	MARKET	TOTAL PROFIT
0	Europe	428607.49350
1	Asia Pacific	412884.53900
2	LATAM	355248.78872
3	USCA	289408.65400
4	Africa	83173.09200

Ilustración 21

Podemos ver con esta consulta la suma del profit para cada mercado. Si asumiéramos que todos los datos del profit están apuntados correctamente podríamos concluir que Europa es el mercado más rentable. En caso de que cosiderasemos que algunos profit son demasiado altos o bajo y sean probablemente un error de toma de datos podríamos hacer el mismo con estudio obviando valores extremos.

```
#Retiro valores anómalos
# Calcular los percentiles 5 y 95 para la columna PROFIT
p5 = df_fact_order["PROFIT"].quantile(0.05)
p95 = df_fact_order["PROFIT"].quantile(0.95)

# Filtrar el DataFrame para mantener solo los valores entre los percentiles 5 y 95
df_filtered = df_complete[(df_complete["PROFIT"] >= p5) & (df_complete["PROFIT"] <= p95)]

# Agrupar por SHIPMENT_IDSHIPMENT y calcular min, mean, max después del filtrado
df_filtered_grouped = df_filtered.groupby(['MARKET'])['PROFIT'].sum().sort_values( ascending=False)

df_filtered_grouped
```

```
MARKET
Europe          292069.25700
Asia Pacific    276903.06970
LATAM           218402.47848
USCA            179072.79950
Africa           77070.86400
Name: PROFIT, dtype: float64
```

Ilustración 22

En este caso, no sería tan útil como en el cálculo de la media, pero si asumimos que los extremos son valores mal apuntados entonces nos puede servir, aunque sea para dar una idea de valores mas cercanos a los reales, aunque no valores reales. De cualquier manera, el mercado con mayor *profit* sigue siendo Europa.

Focus now on the market that provides the greatest benefit:

How has profit evolved over time in this market?

```
# Focus now on the market that provides the greatest benefit:
df_europe = df_complete[(df_complete["MARKET"] == 'Europe') ]
```

```
# How has profit evolved over time in this market?
df_europe_profit_filtered = df_europe[["PROFIT", "YEAR_ORDER", "MONTH_ORDER", "DAY_ORDER"]]
df_europe_profit_filtered
```

	PROFIT	YEAR_ORDER	MONTH_ORDER	DAY_ORDER
0	43.32	2013	4	24
1	1.65	2013	4	24
2	40.05	2014	4	20
3	0.84	2014	12	20
4	22.08	2014	8	12
...	...	...	...	...
44658	291.60	2015	9	6
44659	1.98	2015	9	6
44660	24.90	2015	9	6
44661	10.80	2015	6	10
44662	132.72	2015	6	23

Ilustración 23

Podríamos visualizar el *profit* a lo largo del tiempo de esta forma. También podríamos crear un campo llamado *order_date*, agrupar por meses y ver el *profit* a lo largo del tiempo. Usaremos *matplotlib* para visualizarlo de forma más cómoda:

```python
# Rename the columns (si no pandas no las pilla por defecto)
df_europe_profit_filtered = df_europe_profit_filtered.rename(columns={
    "YEAR_ORDER": "year",
    "MONTH_ORDER": "month",
    "DAY_ORDER": "day"
})

# Now pandas will recognize these columns automatically
df_europe_profit_filtered["ORDER_DATE"] = pd.to_datetime(
    df_europe_profit_filtered[["year", "month", "day"]]
)
# Agrupar por mes y calcular el profit total por mes
df_profit_trend = df_europe_profit_filtered.groupby(df_europe_profit_filtered["ORDER_DATE"].dt.to_period("M"))["PROFIT"].sum().reset_index()

# Convertir la fecha a string para evitar problemas en el gráfico
df_profit_trend["ORDER_DATE"] = df_profit_trend["ORDER_DATE"].astype(str)


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 24

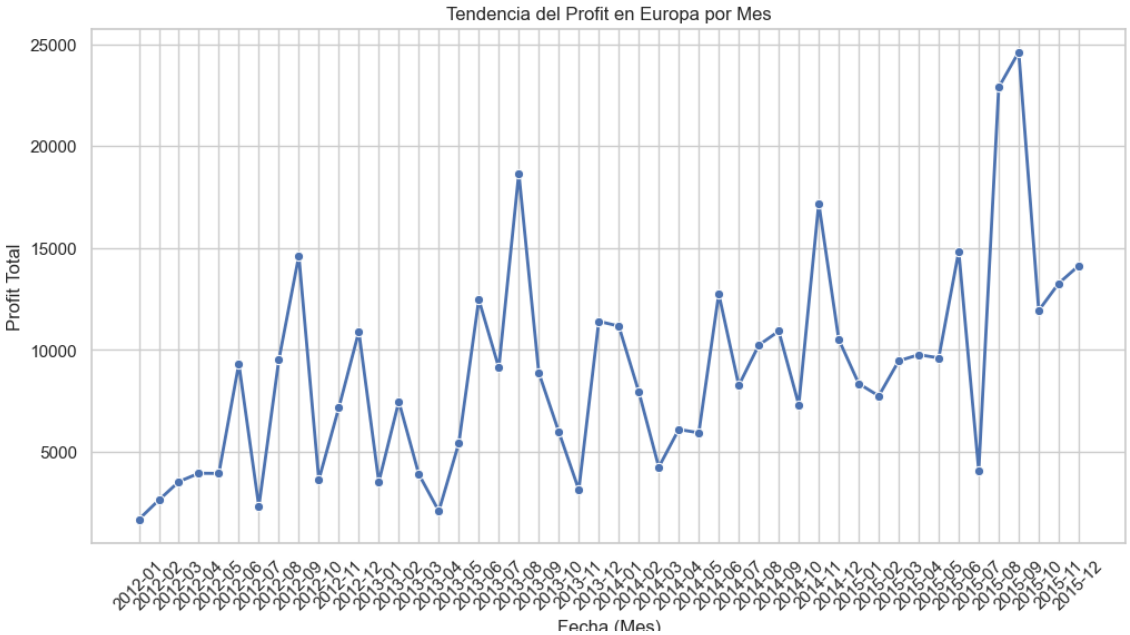

Ilustración 25

Profit de Europa desde enero de 2012 hasta diciembre de 2015. Podemos ver que la tendencia es positiva y parece que hay picos por estación.

What are the best-selling categories (by quantity) in the region of that market that has placed the most orders?

```
cursor.execute("""
SELECT P.CATEGORY, COUNT(*) AS "NUMBER OF ORDERS", SUM(O.PROFIT) AS "TOTAL PROFIT" FROM
FACT_ORDER O JOIN PRODUCT P ON O.PRODUCT_IDPRODUCT = P.IDPRODUCT
GROUP BY P.CATEGORY
ORDER BY 2 DESC
""")
# Obtener los nombres de las columnas
columns = [col[0] for col in cursor.description]

# Recuperar los datos
rows = cursor.fetchall()

df_best_selling_categories = pd.DataFrame(rows, columns=columns)
df_best_selling_categories
```

	CATEGORY	NUMBER OF ORDERS	TOTAL PROFIT
0	Office Supplies	28286	531848.95770
1	Technology	9130	678914.49652
2	Furniture	8584	358559.11300

Ilustración 26

Con esta consulta obtengo que para Europa la categoría con mas ventas es la de material de oficina.

Are the best-selling categories those that generate the highest total profits?

Como podemos ver en el *dataframe*, aunque haya aproximadamente el triple de pedidos para material de oficina los márgenes de *profit* deben de ser mucho menor de lo que se obtiene por producto tecnológico haciéndola más rentable.

Based on all the answers you have obtained, what recommendations would you make to the managers of your company in order to improve results?

- En primer lugar, recomendaría a la empresa a empujar los esfuerzos hacía los aparatos tecnológicos sobre todo en Europa (Ilustración 26) que parece ser la categoría de productos que mas beneficios aporta el mercado más rentable.

- Se puede observar que los meses de junio y septiembre suelen ser malos para las ventas (no siempre, pero la mayoría) por lo que aflojaría todo el tema de la producción, marketing, … (Ilustración 25)
- Los meses más rentables parecen ser siempre uno o un par de mes previa a la caída de los pedidos en los meses de junio y septiembre (Ilustración 25).

- Los 6 primeros clientes que más *profit* generan son US central (Ilustración 18), sin embargo, son de los mercados que menos pedidos realizan (Ilustración 21) es posible que los compradores estén dispuestos a comprar más compulsivamente y gastar más o sean más fieles, por lo que se podría a lo mejor empujar el mercado hacia allí.

- No recomendaría aplicar muchos esfuerzos en la región africana ya que parece la menos rentable (Ilustración 21).