Analisis del crecimiento a traves de los ultimos años de una empresa en operaciòn.

autor:
Carlos Cano github: CxrlxsCxnx

Importando librerias

Se realiza la importacion de librerias con las cuales estaremos trabajando durante el analisis de datos.

```
from datetime import datetime
import glob
import json
import os
from pathlib import Path
import re

from IPython.display import HTML, display, Image
import matplotlib
import matplotlib.pyplot as plt
from matplotlib import cm
import missingno as msno
import numpy as np
import pandas as pd
import seaborn as sns
import altair as alt
import calendar


import streamlit as st
```

Lectura de los datos.

```
#echo|True
route = "./Data"
```

```
files = [ Path(i) for i in glob.glob(f'{route}/*.csv') ]
files
```

```
[PosixPath('Data/FACTURAS03.csv'),
 PosixPath('Data/FACTURAS02.csv'),
 PosixPath('Data/FACTURAS01.csv')]
```

```
#echo : True
dfs = []
for csv in files:
    df = pd.read_csv(csv, delimiter= ',')
    dfs.append(df)
```

Ignoramos los indices al concatenar para no recibir indices repetidos

```
data = pd.concat(dfs, ignore_index= True)
```

Creamos una data redefinida con los valores que nos interesan.

```
#echo : True
data = data[['Fecha','Razón Social', 'Total', 'Cancelado', 'Nombre de la Moneda', 'Nombre de

data.Fecha = pd.to_datetime(data.Fecha,format = 'mixed')
```

Eliminamos valores nulos.

Comparamos y eliminamos valores nulos si existen dentro de la data correspondiente, en caso de que no sean de nuestro interes y no afecten los resultados de nuestro analisis, los conservamos.

```
#echo : True
display(data.index.size)
display(data.dropna().index.size)
```

```
3566
```

```
3564
```

```
display(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3566 entries, 0 to 3565
Data columns (total 6 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Fecha                3566 non-null   datetime64[ns]
 1   Razón Social         3566 non-null   object
 2   Total                3566 non-null   float64
 3   Cancelado            3566 non-null   int64
```

```
 4   Nombre de la Moneda  3566 non-null   object
 5   Nombre del agente    3564 non-null   object
dtypes: datetime64[ns](1), float64(1), int64(1), object(3)
memory usage: 167.3+ KB
```

```
None
```

```
{python] msno.matrix(data) msno.bar(data.sample(frac = 0.1))
```

Data Ventas Mxn, Quitando facturas canceladas.º

```
data_mxn = data[(data['Nombre de la Moneda'] == "Peso Mexicano") & (data['Cancelado'] == 0)]
data_mxn.Total = data.Total.astype(float)

grupo_Clientes_Venta = data_mxn.groupby('Razón Social')['Total'].sum()
clientes_venta_sorting = grupo_Clientes_Venta.sort_values(ascending= False).reset_index().he
```

```
/tmp/ipykernel_5192/1176312956.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guid
  data_mxn.Total = data.Total.astype(float)
```

Data Ventas dolares.

```
data_Dls = data[(data['Nombre de la Moneda'] == "Dólar Americano") & (data['Cancelado'] == 0
data_Dls.shape
```

```
(43, 6)
```

¿Quien nos compra más?

Se agrupan los clientes y los totales de compras que han realizado así se puede obtener una vista general de cuantos ingresos se han obtenido de cada uno de ellos y cuales son nuestros clientes más importantes.

```
grupo_Clientes_Venta = data_mxn.groupby('Razón Social')['Total'].sum()
```

```
clientes_venta_sorting = grupo_Clientes_Venta.sort_values(ascending= False).reset_index().he
```

Se crea una grafica con altair sobre el grupo realizado anteriormente.

```
grafica_1 = alt.Chart(clientes_venta_sorting).encode(
    y = alt.X('Razón Social:O', title = "Clientes"),
    x = alt.Y('Total', title = "Ventas Totales")
).properties(
    width = 350

)
grafica_1.mark_bar() + grafica_1.mark_text(align =  'left', dx = 0)
```

Ventas por año.

Agrupamos las ventas por años para poder ver cuanto porcentaje y que tanto se ha vendido en cada año.

```
ventas_anuales = data_mxn.groupby(data_mxn.Fecha.dt.year)['Total'].sum().reset_index()
ventas_anuales
```

	Fecha	Total
0	2020	4729251.56
1	2021	7445951.39
2	2022	7846298.04
3	2023	8563507.47

Crecimiento

Seleccionamos el priner y el ultimo ewlemento de ventas_anuales para obtener el crecimiento total de la empresa durante los 4 años.

```
pElemento = ventas_anuales.Total.iloc[0]
uElemento = ventas_anuales.Total.iloc[-1]

crecimiento_total = ((uElemento - pElemento) / pElemento ) * 100
crecimiento_total
```

```
81.07532156737295
```

Para una mejor visualización obtenemos la grafica, la generamos con altair con una mark_line.

```
grafica_2 = alt.Chart(ventas_anuales).mark_line(point = alt.OverlayMarkDef(filled = False, f
    x = alt.X("Fecha:N").scale(zero = False),
    y = alt.Y("Total:Q").scale(zero = False),
).properties(
    width = 400,
    height = 200
)
display(grafica_2)
```

Tambien nos gustaria obtener el contraste de los meses durante el año y como se ha desarrollado, cual es el mes en que más generamos o si hay una tendencia sobre el mes y el consumo de los clientes.

```
contraste_mensual = data_mxn.groupby(data_mxn.Fecha.dt.to_period('M'))['Total'].sum().reset_
contraste_mensual = contraste_mensual.iloc[:-2]
contraste_mensual['Fecha'] = contraste_mensual['Fecha'].apply(lambda x: x.to_timestamp())
contraste_mensual['Fecha'] = pd.to_datetime(contraste_mensual['Fecha'], format='%Y-%m')
```

```
contraste_mensual['Año'] = contraste_mensual['Fecha'].dt.year
contraste_mensual['Mes'] = contraste_mensual['Fecha'].dt.month

highlight = alt.selection_point(
    on="mouseover", fields=["Año"], nearest=True
)

base = alt.Chart(contraste_mensual).encode(
    x= alt.X('Mes:O', title='Mes'),
    y= alt.Y('Total:Q', title='Total'),
    color= alt.Color('Año:N')
).properties(
    width=600
)
points = base.mark_circle().encode(
    opacity=alt.value(0)
).add_params(
    highlight
).properties(
    width=600
)

lines = base.mark_line( point=alt.OverlayMarkDef(filled=False, fill="white")).encode(
    size=alt.condition(~highlight, alt.value(1), alt.value(3))
)
points + lines
```

Tambien agrupamos por fechas y años para realizar una grafica y datos dinamicos sobre fechas seleccionadas.

```
años_disponibles =  data_mxn['Fecha'].dt.year.unique()
clientesprincipales_y  = data_mxn[['Fecha', 'Razón Social', 'Total']].reset_index(drop = Tru

clientesprincipales_y['Fecha'] = pd.to_datetime(clientesprincipales_y['Fecha'])
clientesprincipales_y['Fecha'] = clientesprincipales_y['Fecha'].dt.year

clientesprincipales_y = clientesprincipales_y.groupby(['Fecha', 'Razón Social'])['Total'].su
clientesprincipales_y = clientesprincipales_y.sort_values(by='Total', ascending=False)

año_seleccionado = 2023
datos_año_seleccionado = clientesprincipales_y[clientesprincipales_y['Fecha'] == año_selecci

total_facturacion = clientesprincipales_y.Total.sum()
total_facturacion
```

```
28585008.459999997
```

Seleccionamos el periodo y creamos grupos para visualizarlos, obtenemos una

tabla ademas de una grafica.

```
año_inicio = 2022
año_fin = 2023


datos_años_seleccionados = clientesprincipales_y[(clientesprincipales_y['Fecha'] >= int(año_

total_facturacion = clientesprincipales_y[(clientesprincipales_y['Fecha'] >= int(año_inicio)
total_facturacion = total_facturacion['Total'].sum()
total_facturacion_del_periodo = clientesprincipales_y.Total.sum()


datos_años_seleccionados['PorcentajeTotal_Facturacion'] = round((datos_años_seleccionados['T
datos_años_seleccionados_table = datos_años_seleccionados.head()
datos_años_seleccionados_table.columns = ['Año', 'Cliente', 'Total', 'Porcentaje Representat
```

Se generan los grupos con los años seleccionados y los grupos que nos permiten visualizar los clientes principales de los años seleccionados, los clientes más frecuentes son con los qu enos moestrara la siguiente grafica y con la que se hara dinamica.

```
total_facturacion = clientesprincipales_y[(clientesprincipales_y['Fecha'] >= int(año_inicio)
total_facturacion = total_facturacion['Total'].sum()
total_facturacion_del_periodo = clientesprincipales_y.Total.sum()


datos_años_seleccionados['PorcentajeTotal_Facturacion'] = round((datos_años_seleccionados['T
datos_años_seleccionados_table = datos_años_seleccionados.head()
datos_años_seleccionados_table.columns = ['Año', 'Cliente', 'Total', 'Porcentaje Representat
datos_años_seleccionados_table = datos_años_seleccionados_table.to_html(index=False)


grupo_clientes_principales = datos_años_seleccionados.groupby('Razón Social')['Total'].sum()
grupo_clientes_principales = grupo_clientes_principales.sort_values(by='Total', ascending=Fa
```

En la grafica se muestra a la dertecha el cliente que más nos ha comprado y se puede modificar cambiando los valores de las variables año_inicio y año_fin. esto también modificara algunos detalles en el link de streamlit.

```
grafica_2 = alt.Chart(grupo_clientes_principales.head(20)).encode(
    y = alt.X('Razón Social:O', title = "Clientes"),
    x = alt.Y('Total', title = "Ventas Totales")
).properties(
    width = 700
)
grafica_2 = grafica_2.mark_bar() + grafica_2.mark_text(align =  'left', dx = 0)
grafica_2
```

En el analisis se usaron las librerias:

altair==5.1.2 ipython==8.16.1 numpy==1.26.1 pandas==2.1.3 streamlit==1.28.1

para cada libreria las versiones son las indicadas.