sales_data_sample

February 4, 2020

1 Getting Insights from Data

Demostración del dataset

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


path = "https://docs.google.com/spreadsheets/d/e/
↪2PACX-1vRORx2TSYJz4kdDYn2Mev8sDrWkc6eT6PwRoRjK4fZdyedsSTWs1c80A8ZbsSkFNPqe_BYn8kaQUHal/
↪pub?output=csv"

data = pd.read_csv(path)

data.head()
```

[1]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	\
0	10107	30	95.70	2	2871.00	
1	10121	34	81.35	5	2765.90	
2	10134	41	94.74	2	3884.34	
3	10145	45	83.26	6	3746.70	
4	10159	49	100.00	14	5205.27	

	ORDERDATE	ORDERMONTH	STATUS	QTR_ID	MONTH_ID	…	\
0	2003-02-24	2003-02	Shipped	1	2	…	
1	2003-05-07	2003-05	Shipped	2	5	…	
2	2003-07-01	2003-07	Shipped	3	7	…	
3	2003-08-25	2003-08	Shipped	3	8	…	
4	2003-10-10	2003-10	Shipped	4	10	…	

	ADDRESSLINE1	ADDRESSLINE2	CITY	STATE	POSTALCODE	\
0	897 Long Airport Avenue	NaN	NYC	NY	10022	
1	59 rue de l'Abbaye	NaN	Reims	NaN	51100	
2	27 rue du Colonel Pierre Avia	NaN	Paris	NaN	75508	

```
3             78934 Hillside Dr.          NaN       Pasadena    CA      90003
4                7734 Strong St.          NaN  San Francisco    CA        NaN

  COUNTRY TERRITORY CONTACTLASTNAME CONTACTFIRSTNAME DEALSIZE
0     USA       NaN              Yu             Kwai    Small
1  France      EMEA         Henriot             Paul    Small
2  France      EMEA        Da Cunha           Daniel   Medium
3     USA       NaN           Young            Julie   Medium
4     USA       NaN           Brown            Julie   Medium

[5 rows x 26 columns]
```

[2]:
```
data_day = data.groupby(("ORDERDATE")).agg(
                                    {'SALES': 'sum', "ORDERDATE": "max" })
```

1.1 Ventas por dias

[3]:
```
# Figura
fig = go.Figure()
fig.add_trace(go.Scatter(x = data_day["ORDERDATE"], y = data_day["SALES"],␣
 ↪name="Sales"))


fig.update_layout(title_text='Ventas por dias',
                  xaxis_rangeslider_visible=True,
                  scene = dict( aspectratio = dict( x = 1.7, y = 0.5, z = 1)␣
 ↪), margin=dict(l=0, r=0, b=20, t=0),
                  template="plotly_dark"
                 )
fig.show()
```

1.2 Ventas por meses

[4]:
```
data_month = data.groupby(("ORDERMONTH")).agg(
                                    {'SALES': 'sum', "ORDERMONTH": "max" })


data_month.insert(loc=0, column='index', value=np.arange(len(data_month)))
dic1 = pd.merge(data_month, data_month["SALES"], left_on = data_month["index"],␣
 ↪right_on = data_month["index"]+1, how='outer')
dic1 = dic1.dropna(axis = 0, thresh = 3)
dic1["%"] = round(((dic1["SALES_x"]/dic1["SALES_y"])-1)*100, 2)
```

[5]:
```
# Figura
fig = make_subplots(specs=[[{"secondary_y": True}]])
```

```
fig.add_trace(go.Scatter(x = dic1["ORDERMONTH"], y = dic1["SALES_x"],␣
 ↪name="Sales"),secondary_y=False)
fig.add_trace(go.Bar(x = dic1["ORDERMONTH"], y = dic1["%"],␣
 ↪name="%"),secondary_y=True),

fig.update_layout(title_text='Ventas por dias',

                  scene = dict( aspectratio = dict( x = 1.7, y = 0.5, z = 1)␣
 ↪), margin=dict(l=10, r=10, b=20, t=0),
                  template="plotly_dark"
                 )
fig.show()
```

1.3 Where go we sell more, and what do we sell in those places?

[6]:
```
data_contry = data.groupby(("COUNTRY")).agg(
                                  {'SALES': 'sum'})
data_contry.sort_values(by=['SALES'], ascending=False).head(15)
```

[6]:

COUNTRY	SALES
USA	3627982.83
Spain	1215686.92
France	1110916.52
Australia	630623.10
UK	478880.46
Italy	374674.31
Finland	329581.91
Norway	307463.70
Singapore	288488.41
Denmark	245637.15
Canada	224078.56
Germany	220472.09
Sweden	210014.21
Austria	202062.53
Japan	188167.81

[7]:
```
data_contry1 = data.groupby(("COUNTRY", "PRODUCTLINE")).agg(
                                  {'SALES': 'sum'})
data_contry1.sort_values(by=['SALES', "COUNTRY"], ascending=False).head(30)
```

```
/home/edgar/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
FutureWarning:

Interpreting tuple 'by' as a list of keys, rather than a single key. Use
'by=[…]' instead of 'by=(…)'. In the future, a tuple will always mean a
single key.
```

[7]:

COUNTRY	PRODUCTLINE	SALES
USA	Classic Cars	1344638.22
	Vintage Cars	757755.90
	Motorcycles	520371.70
Spain	Classic Cars	476165.15
USA	Trucks and Buses	397842.42
France	Classic Cars	388951.20
USA	Planes	328432.89
Spain	Vintage Cars	229514.51
France	Motorcycles	226390.31
USA	Ships	209688.14
Australia	Classic Cars	193085.54
	Vintage Cars	189555.32
Spain	Trucks and Buses	177556.78
France	Vintage Cars	176609.81
UK	Classic Cars	159377.70
Denmark	Classic Cars	157182.48
Finland	Classic Cars	153552.24
Germany	Classic Cars	148315.00
Norway	Classic Cars	134787.37
Singapore	Classic Cars	132890.44
Italy	Classic Cars	128576.65
Spain	Ships	124459.97
UK	Vintage Cars	123798.74
Switzerland	Classic Cars	117713.56
France	Trucks and Buses	116982.22
Italy	Vintage Cars	110450.74
France	Planes	108155.51
Austria	Classic Cars	101459.47
Italy	Planes	98185.65
Spain	Planes	89985.51

1.4 How many customers do we have?

[8]:
```
data_contry2 = data.groupby(("CUSTOMERNAME")).nunique()
data_contry2 = data_contry2.shape
print("El total de clientes es: " + str(data_contry2[0]) )
```

```
El total de clientes es: 92
```

1.5 Is there any product line that has decreased sales dramatically during the last year?

[51]:
```
fecha1 = ["2005-01-01", "2005-12-31"]
data1 = np.logical_and( data["ORDERDATE"] >= fecha1[0], data["ORDERDATE"] <=␣
 ↪fecha1[1] )
data1 = data[data1]

data_product = data1.groupby(("ORDERMONTH", "PRODUCTLINE")).agg(
                                    {'SALES': 'sum', "ORDERMONTH": "max",␣
 ↪"PRODUCTLINE": "max" })
dic2 = pd.merge(data_product, data_product["SALES"], left_on = data_product.
 ↪index, right_on = data_product.index, how='outer')
dic2
```

```
/home/edgar/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5:
FutureWarning:

Interpreting tuple 'by' as a list of keys, rather than a single key. Use
'by=[…]' instead of 'by=(…)'. In the future, a tuple will always mean a
single key.
```

[51]:

	key_0	SALES_x	ORDERMONTH	PRODUCTLINE	\
0	(2005-01, Classic Cars)	139087.23	2005-01	Classic Cars	
1	(2005-01, Motorcycles)	39913.36	2005-01	Motorcycles	
2	(2005-01, Planes)	11262.27	2005-01	Planes	
3	(2005-01, Ships)	31807.85	2005-01	Ships	
4	(2005-01, Trains)	6510.17	2005-01	Trains	
5	(2005-01, Trucks and Buses)	36577.20	2005-01	Trucks and Buses	
6	(2005-01, Vintage Cars)	74385.34	2005-01	Vintage Cars	
7	(2005-02, Classic Cars)	146148.57	2005-02	Classic Cars	
8	(2005-02, Motorcycles)	47951.42	2005-02	Motorcycles	
9	(2005-02, Planes)	31040.41	2005-02	Planes	
10	(2005-02, Ships)	10586.06	2005-02	Ships	
11	(2005-02, Trains)	7209.84	2005-02	Trains	
12	(2005-02, Trucks and Buses)	37174.16	2005-02	Trucks and Buses	
13	(2005-02, Vintage Cars)	78075.72	2005-02	Vintage Cars	
14	(2005-03, Classic Cars)	91389.86	2005-03	Classic Cars	
15	(2005-03, Motorcycles)	47830.83	2005-03	Motorcycles	
16	(2005-03, Planes)	79735.05	2005-03	Planes	
17	(2005-03, Ships)	52765.15	2005-03	Ships	
18	(2005-03, Trains)	12939.45	2005-03	Trains	
19	(2005-03, Trucks and Buses)	14579.38	2005-03	Trucks and Buses	
20	(2005-03, Vintage Cars)	75023.04	2005-03	Vintage Cars	
21	(2005-04, Classic Cars)	111562.51	2005-04	Classic Cars	
22	(2005-04, Motorcycles)	59862.22	2005-04	Motorcycles	

23	(2005-04, Planes)	43763.59	2005-04	Planes
24	(2005-04, Ships)	6284.00	2005-04	Ships
25	(2005-04, Vintage Cars)	40160.97	2005-04	Vintage Cars
26	(2005-05, Classic Cars)	184385.11	2005-05	Classic Cars
27	(2005-05, Motorcycles)	39389.70	2005-05	Motorcycles
28	(2005-05, Planes)	34272.85	2005-05	Planes
29	(2005-05, Ships)	26735.01	2005-05	Ships
30	(2005-05, Trains)	10257.87	2005-05	Trains
31	(2005-05, Trucks and Buses)	89726.28	2005-05	Trucks and Buses
32	(2005-05, Vintage Cars)	73094.24	2005-05	Vintage Cars

	SALES_y
0	139087.23
1	39913.36
2	11262.27
3	31807.85
4	6510.17
5	36577.20
6	74385.34
7	146148.57
8	47951.42
9	31040.41
10	10586.06
11	7209.84
12	37174.16
13	78075.72
14	91389.86
15	47830.83
16	79735.05
17	52765.15
18	12939.45
19	14579.38
20	75023.04
21	111562.51
22	59862.22
23	43763.59
24	6284.00
25	40160.97
26	184385.11
27	39389.70
28	34272.85
29	26735.01
30	10257.87
31	89726.28
32	73094.24

[93]:
```
fig = px.line(dic2, x="ORDERMONTH", y="SALES_x", color='PRODUCTLINE')
fig.update_layout(scene = dict(aspectratio = dict( x = 2, y = 1.1),
                               yaxis = dict( nticks = 7 ),


                               xaxis_title='Month',
                               yaxis_title=''
                              ),template="plotly_dark",
                  margin = dict(l=0, r=0, b=30, t=0)
                 )
fig.show()
```

[95]:
```
from IPython.display import HTML

HTML('''<script>
code_show=true;
function code_toggle() {
 if (code_show){
 $('div.input').hide();
 } else {
 $('div.input').show();
 }
 code_show = !code_show
}
$( document ).ready(code_toggle);
</script>
<form action="javascript:code_toggle()"><input type="submit" value="Click here␣
 ↪to toggle on/off the raw code."></form>''')
```

[95]:
```
<IPython.core.display.HTML object>
```