



MATERIAL DE LA CLASE

ANÁLISIS DE DATOS

CLASE 3

Bienvenidos a la inmersión de la Semana de Python en la Práctica

¡Es un placer tenerte aquí con nosotros en esta inmersión!

Este folleto pretende presentar de forma sencilla y directa todo el contenido que se vio en la clase en vivo.

Aquí en Daxus Latam creemos firmemente en el aprendizaje basado en proyectos y por eso, cada día de inmersión trae un nuevo proyecto a desarrollar.

Recuerde siempre en nuestra metodología ORA:

OBSERVAR: Durante la clase en vivo, simplemente siga lo que se está haciendo y tome nota de las explicaciones.

REPETIR: luego, rehaz el proyecto con el material en mano y si tienes alguna dificultad, repasa la clase. Todas las clases se graban y están disponibles en YouTube durante la semana de inmersión.

APLICAR: Ahora que has observado y repetido, es hora de aplicar lo aprendido. Crear nuevos proyectos con los conocimientos adquiridos.

¿Qué aprenderemos?

- › Usar la biblioteca de Análisis de Datos más utilizada en el mundo: **Pandas**
- › Cargar datos de un archivo **Excel**
- › Realizar un análisis exploratorio de los datos
- › Generar estadísticas
- › Generar gráficos interactivos con la biblioteca **Plotly**
- › Enviar correos electrónicos de forma automática

Proyecto de la clase

Has sido contratado(a) como **Analista de Datos** por una cadena de **tiendas de bebidas** y deberás **crear análisis sobre el negocio** utilizando una **base de datos en un archivo Excel**, que fue extraída del sistema de ventas de la cadena.

Bibliotecas que utilizaremos

Instalando las bibliotecas Pandas, OpenpyXL y Plotly Express

TERMINAL

```
!pip install pandas
```

```
!pip install openpyxl
```

```
!pip install plotly nbformat
```

Cargando datos del archivo Excel

ENTRADA

```
1 import pandas as pd  
2  
3 data = pd.read_excel("ventas.xlsx")
```

Análisis Exploratorio

Verificando las primeras y últimas líneas

ENTRADA

```
1 data.head()
```

SALIDA

	region	fecha	tienda	Ciudad	País	tamaño	local_consumo	precio	forma_pago
0	PED1994	2022-01-01	Tienda 4	Guadalajara	México	300ml	Consumo en tienda	5	Efectivo
1	PED2246	2022-01-01	Tienda 6	Santiago de Chile	Chile	500ml	Consumo en tienda	11	Débito
2	PED3876	2022-01-01	Tienda 3	Bogotá	Colombia	300ml	Delivery	7	Crédito
3	PED4352	2022-01-01	Tienda 1	Buenos Aires	Argentina	1000ml	Consumo en tienda	7	Débito
4	PED8633	2022-01-01	Tienda 5	Ciudad de México	México	200ml	Delivery	9	Crédito
5	PED10668	2022-01-01	Tienda 4	Guadalajara	México	500ml	Delivery	7	Crédito
6	PED11326	2022-01-01	Tienda 4	Guadalajara	México	300ml	Delivery	11	Efectivo
7	PED11327	2022-01-01	Tienda 4	Guadalajara	México	200ml	Delivery	5	Nequi
8	PED11506	2022-01-01	Tienda 4	Guadalajara	México	300ml	Consumo en tienda	5	Débito
9	PED16937	2022-01-01	Tienda 4	Guadalajara	México	700ml	Consumo en tienda	13	Efectivo

Verificando las últimas líneas

ENTRADA

```
1 data.tail()
```

SALIDA

	region	fecha	tienda	Ciudad	País	tamaño	local_consumo	precio	forma_pago
69995	PED67084	2024-12-31	Tienda 6	Santiago de Chile	Chile	500ml	Consumo en tienda	11	Crédito
69996	PED67857	2024-12-31	Tienda 3	Bogotá	Colombia	200ml	Consumo en tienda	7	Nequi
69997	PED69171	2024-12-31	Tienda 4	Guadalajara	México	500ml	Consumo en tienda	5	Efectivo
69998	PED69229	2024-12-31	Tienda 4	Guadalajara	México	300ml	Consumo en tienda	9	Nequi
69999	PED69356	2024-12-31	Tienda 1	Buenos Aires	Argentina	300ml	Delivery	9	Nequi

Cantidad de filas y columnas

ENTRADA

```
1 data.shape
```

SALIDA

```
(70000, 9)
```

Información sobre las columnas

ENTRADA

```
1 data.info()
```

SALIDA

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70000 entries, 0 to 69999
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   region           70000 non-null   object 
 1   fecha             70000 non-null   datetime64[ns]
 2   tienda            70000 non-null   object 
 3   Ciudad            70000 non-null   object 
 4   País              70000 non-null   object 
 5   tamaño            70000 non-null   object 
 6   local_consumo    70000 non-null   object 
 7   precio            70000 non-null   int64  
 8   forma_pago        70000 non-null   object 
dtypes: datetime64[ns](1), int64(1), object(7)
memory usage: 4.8+ MB
```

Generando las estadísticas

Ahora, vamos generar algunas estadísticas importantes sobre la columna **precio**.

ENTRADA

```
1 data.describe()
```

SALIDA

	precio
count	70000.000000
mean	8.355200
std	2.653061
min	5.000000
25%	7.000000
50%	7.000000
75%	11.000000
max	13.000000

Obteniendo los valores únicos de una columna

ENTRADA

```
1 data["tienda"].unique()
```

SALIDA

```
array(['Tienda 4', 'Tienda 6', 'Tienda 3', 'Tienda 1', 'Tienda 5',
       'Tienda 2'], dtype=object)
```

Conteo de valores

ENTRADA

```
1 data["tienda"].value_counts()
```

SALIDA

```
tienda
Tienda 4    13483
Tienda 6    13075
Tienda 1    12344
Tienda 5    12177
Tienda 3    10603
Tienda 2    8318
Name: count, dtype: int64
```

Agrupando datos

El método **groupby()** realiza el agrupamiento de datos por una determinada columna. Siempre que utilizamos este método, necesitamos definir la **función de agregación** que se aplicará a los datos. Ejemplo: **sum()** para suma y **mean()** para promedio.

ENTRADA

```
1 data.groupby("tienda")["precio"].sum()
```

SALIDA

```
tienda
Tienda 1    103162
Tienda 2    69592
Tienda 3    88357
Tienda 4    112379
Tienda 5    102189
Tienda 6    109185
Name: precio, dtype: int64
```

Gráficos interactivos

ENTRADA

```

1  grafico = px.histogram(data, x="tienda",
2                            y="precio",
3                            text_auto=True,
4                            title="Facturación",
5                            color="forma_pago")
6
7  grafico.show()

```

SALIDA



Listas y el comando for

ENTRADA

```

1  lista_nombres = ["Nico", "Zai", "Goku"]
2
3  for nombre in lista_nombres:
4      print(nombre)

```

SALIDA

Nico
Zai
Goku



SEMANA DE python)

EN LA PRÁCTICA

Daxus
LATAM