



ADVANCE
Consortium



ATENEA 2025 BOOTCAMPS



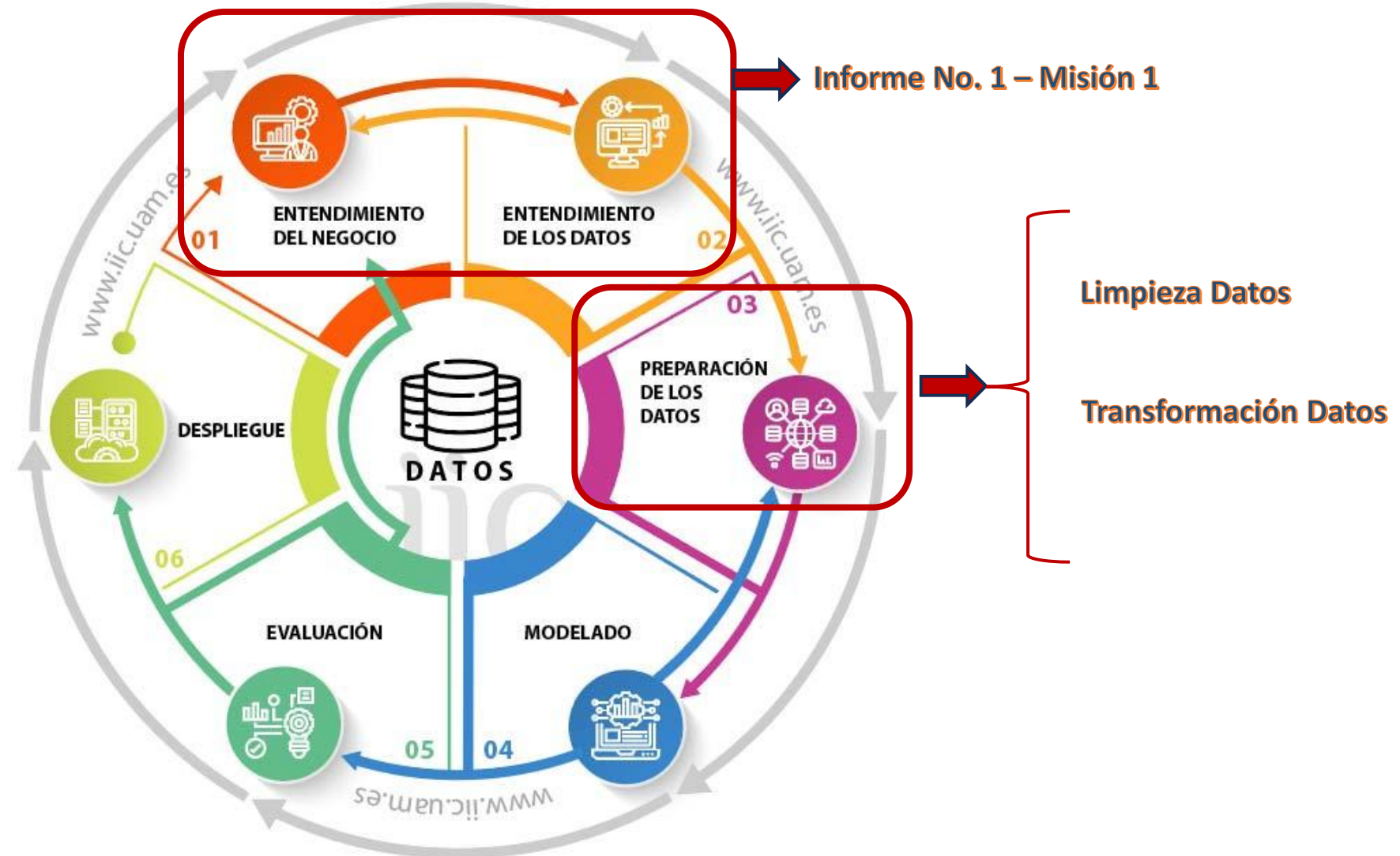
Identificación Material

Bootcamp	Soluciones de Futuro con IA y Datos- Intermedio
Módulo	Desarrollo de soluciones basadas en datos
Eje Temático	Limpieza y Transformación de Datos

Limpieza de Datos

Transformación Datos

Fases en la Metodología CRISP-DM



MAPA DE CONTENIDOS

LIMPIEZA DE DATOS

01



IDENTIFICACIÓN Y
RECOPILACIÓN DE DATOS

02



HALLZAGOS Y LIMPIEZA

03



LIMPIEZA CON PYTHON



Universidad Tecnológica de Bolívar

Identificación y recopilación – Fuentes de Datos

Dimensiones en la Calidad de los Datos



Compleitud: el grado en el que todos los atributos del dato están presentes.

Precisión / Exactitud: si los datos no son precisos, estos no pueden ser utilizados. En este sentido, para detectar si estos son precisos, se compara el dato con una fuente de referencia

Integridad: Se centra en el hecho de saber si toda la información relevante de un registro está presente de forma que se pueda utilizar.

Conformidad: los datos deben estar en un formato estándar y legible

Consistencia: al hacer el cruce de información con los registros, se debe evitar la información contradictoria, es decir, los datos serán siempre los mismos.

Unicidad: es importante saber si se tiene la misma información en formatos iguales o similares dentro de la fuente de información

Jerarquía Calidad del Dato



La identificación y recopilación de fuentes de datos relevantes es un paso crucial en cualquier proyecto de analítica de datos. La calidad y pertinencia de los datos determinarán en gran medida la efectividad de los análisis y las conclusiones que se puedan extraer. Aquí hay algunas consideraciones clave en este proceso:

- 1. Definición de Objetivos:** Antes de comenzar a buscar fuentes de datos, es esencial tener claridad sobre los objetivos del proyecto de analítica de datos. **¿Qué preguntas se están buscando responder? ¿Cuáles son los problemas o áreas de mejora específicos que se abordarán?**
- 2. Identificación de Fuentes de Datos Potenciales:** Enumera las posibles fuentes de datos que podrían ser relevantes para tu proyecto. Esto puede incluir bases de datos internas, datos de clientes, datos de redes sociales, datos gubernamentales, entre otros.
- 3. Evaluación de Calidad de Datos:** Antes de seleccionar una fuente de datos, evalúa la calidad de los datos disponibles. Esto implica revisar la integridad, precisión y actualidad de los datos. Si la calidad es deficiente, podría afectar la validez de los análisis.

4. **Consolidación de Datos Internos y Externos:** Examina las fuentes de datos internas de la organización, como bases de datos de clientes, registros de transacciones y datos operativos. También considera la integración de datos externos que puedan enriquecer el análisis.
5. **Exploración de Datos No Estructurados:** No te limites a datos estructurados. Explora datos no estructurados, como comentarios en redes sociales, registros de chat, correos electrónicos y otros tipos de información que pueden aportar perspectivas valiosas.

Datos estructurados



Los datos agregados están contenidos en una sola dimensión.



Se puede almacenar en MS Access, Oracle, SQL Server y otros sistemas de bases de datos tradicionales similares.



Se puede almacenar en diferentes columnas y filas.



Un ejemplo de datos estructurados son las transacciones de aplicaciones en línea.



Se puede definir fácilmente dentro del modelo de datos.



Viene con un tamaño y contenido fijos.

Datos no estructurados



Los datos se dividen en diferentes tablas de dimensiones.



No se puede almacenar en un sistema de base de datos tradicional.



No se puede almacenar en filas y columnas.



Ejemplos de datos no estructurados son Tweets, búsquedas de Google, Me gusta de Facebook, etc.



No se puede definir según el modelo de datos.

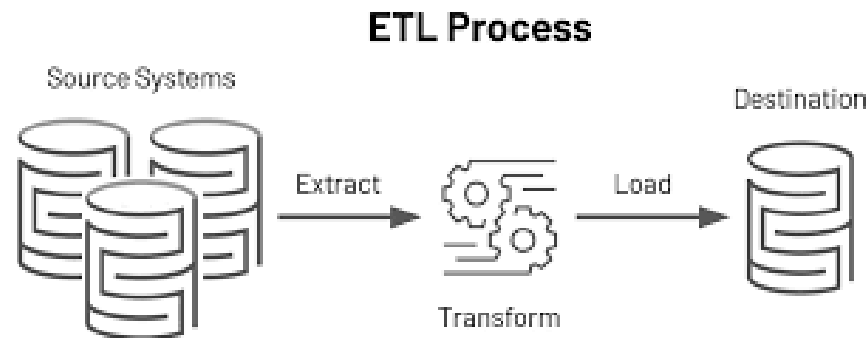



Viene en varios tamaños y contenidos.

6. Revisión de Fuentes Gubernamentales y de Terceros: Examina si existen fuentes de datos gubernamentales relevantes para tu industria. Además, considera datos proporcionados por terceros, como empresas de investigación de mercado, que puedan complementar tus fuentes internas.

7. Consideración de Ética y Privacidad: Asegúrate de cumplir con las normativas de ética y privacidad al recopilar datos. La transparencia y el consentimiento son fundamentales, y es esencial proteger la privacidad de los individuos.

8. **Extracción y Transformación de Datos (ETL):** Desarrolla procesos de extracción, transformación y carga de datos (ETL) para preparar y limpiar los datos. Esto implica la conversión de datos en un formato utilizable y la eliminación de cualquier ruido o inconsistencia.



9. Documentación de Metadatos: Documenta los metadatos de las fuentes de datos. Esto incluye detalles sobre el origen, la frecuencia de actualización, la estructura de los datos y cualquier transformación realizada. Facilita la comprensión futura de los datos. 

10. Prueba y Validación de Datos: Antes de comenzar análisis exhaustivos, realiza pruebas y validaciones preliminares para asegurarte de que los datos recopilados sean coherentes y cumplan con las expectativas.

11. Planificación para la Escalabilidad: Considera la escalabilidad a medida que seleccionas fuentes de datos. ¿Podrán manejar un volumen creciente de datos a medida que el proyecto se expanda?



Limpieza de Datos Python

Preparación de los datos

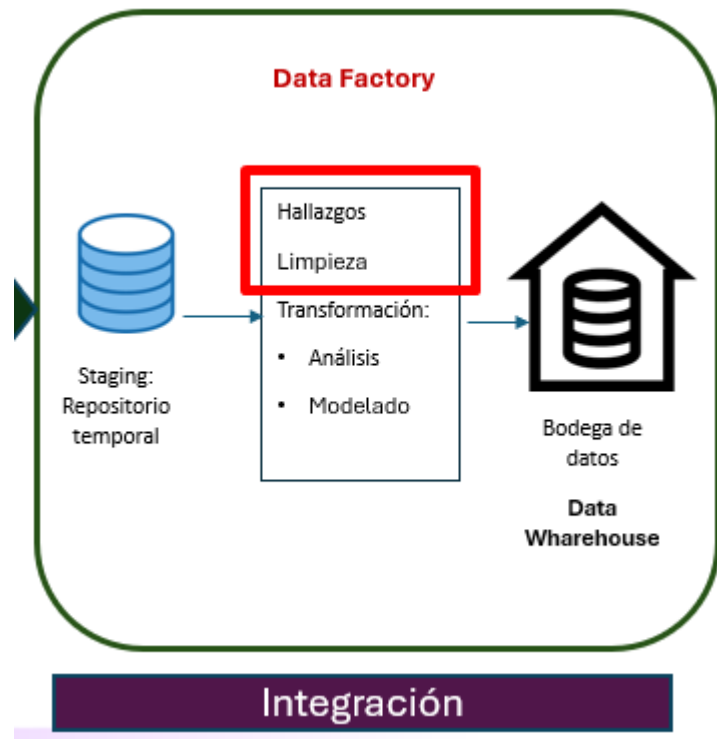
Organizar los datos de modo que se pueda ver lo que ellos cuentan. Algunas preguntas: “¿Cómo deben ser tratados los valores nulos? ¿Los atributos están en los formatos correctos? ¿Será necesario hacer alguna fusión con otros datos? ¿Qué variables serán utilizadas en el modelado?”

Datos con Basura -
Ortografía

Manejo de Valores nulos

Valores duplicados

Manejo de Outliers



Remediación o Imputación

Transformación de Columnas

En esta fase de limpieza y preparación de datos en un proyecto, el manejo de valores faltantes y la eliminación de duplicados son aspectos críticos, con las siguientes estrategias y prácticas se podría abordar estos dos aspectos

- **Documentación:**

- Documenta claramente las decisiones tomadas en cuanto al manejo de valores faltantes y duplicados. Esto facilita la comprensión y la reproducción del proceso.

- **Impacto en el Análisis:**

- Antes y después de realizar acciones de limpieza, evalúa cómo estas afectarán el análisis final. Es crucial comprender cómo las decisiones de limpieza pueden influir en los resultados.

- **Consistencia:**

- Mantén la consistencia en la aplicación de las estrategias de manejo de valores faltantes y duplicados en todo el conjunto de datos para evitar sesgos o inconsistencias.

La limpieza de datos es crucial para garantizar la calidad de los análisis. Algunas técnicas avanzadas son:

Manejo de Valores nulos:

```
# Eliminar filas con valores nulos
df_sin_nulos = df.dropna()

# Rellenar valores nulos con un valor específico
df_con_relleno = df.fillna(valor_relleno)
```

Detección y eliminación de duplicados

```
# Detectar duplicados basados en todas las columnas
df_sin_duplicados = df.drop_duplicates()

# Detectar duplicados basados en una columna específica
df_sin_duplicados_columna = df.drop_duplicates(subset='columna')
```


LIMPIEZA DE DATOS

Con base en la fuente de datos: Mercado_casa.xlsx: Hoja: Datos

Año	Mes	Viveres	Verduras	Frutas	Carnes	Lácteos
2024	Ene	200.000	60.000	40.000		100.000
2024	Feb	210.000	65.000	40.000	270.000	105.000
2024	Mar	210.000	70.000	45.000	275.000	105.000
2024	Abr	225.000	80.000	42.000		110.000
2024	May	240.000	85.000	50.000	280.000	115.000
2024	Ago	250.000	90.000	70.000	285.000	135.000
2024	Jun	240.000	85.000	45.000	285.000	130.000
2024	Jul	245.000	90.000	65.000	285.000	130.000
2024	Ago	250.000	90.000	70.000	285.000	135.000
2024	Ago	250.000	90.000	70.000	285.000	135.000
2024	Sep		90.000	65.000	285.000	130.000

Con base en la fuente de datos: Mercado_casa.xlsx:

Manejo de Valores nulos:

```
import numpy as np
import pandas as pd

#Leer archivo excel e imprimir las primeras 5 filas
df = pd.read_excel('C:\\Otros\\UTB\\TalentoTech\\AnálisisDatos\\Material\\Semana_4\\Ejercicios\\Limpieza\\Mercado_casa.xlsx')
print("\nMercado Casa")
print(df)
```

Mercado Casa							
	Año	Mes	Víveres	Verduras	Frutas	Carnes	Lácteos
0	2024.0	Ene	200000.0	60000.0	40000.0	NaN	100000.0
1	2024.0	Feb	210000.0	65000.0	40000.0	270000.0	105000.0
2	2024.0	Mar	210000.0	70000.0	45000.0	275000.0	105000.0
3	2024.0	Abr	225000.0	80000.0	42000.0	NaN	110000.0
4	2024.0	May	240000.0	85000.0	50000.0	280000.0	115000.0
5	2024.0	Ago	250000.0	90000.0	70000.0	285000.0	135000.0
6	2024.0	Jun	240000.0	85000.0	45000.0	285000.0	130000.0
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8	2024.0	Jul	245000.0	90000.0	65000.0	285000.0	130000.0
9	2024.0	Ago	250000.0	90000.0	70000.0	285000.0	135000.0
10	2024.0	Ago	250000.0	90000.0	70000.0	285000.0	135000.0
11	2024.0	Sep	NaN	90000.0	65000.0	285000.0	130000.0

LIMPIEZA DE DATOS

Con base en la fuente de datos: Mercado_casa.xlsx:

Manejo de Valores nulos:

```
#Limpieza valores nulos (eliminarlos) (Todos los datos de la fila debe ser nulos)
df_sin_nulos1=df.dropna(how="all")
print("\nMercado Casa sin nulos (Toda la fila tenga nulos)")
print(df_sin_nulos1)
```

	Año	Mes	Víveres	Verduras	Frutas	Carnes	Lácteos
0	2024.0	Ene	200000.0	60000.0	40000.0	NaN	100000.0
1	2024.0	Feb	210000.0	65000.0	40000.0	270000.0	105000.0
2	2024.0	Mar	210000.0	70000.0	45000.0	275000.0	105000.0
3	2024.0	Abr	225000.0	80000.0	42000.0	NaN	110000.0
4	2024.0	May	240000.0	85000.0	50000.0	280000.0	115000.0
5	2024.0	Ago	250000.0	90000.0	70000.0	285000.0	135000.0
6	2024.0	Jun	240000.0	85000.0	45000.0	285000.0	130000.0
8	2024.0	Jul	245000.0	90000.0	65000.0	285000.0	130000.0
9	2024.0	Ago	250000.0	90000.0	70000.0	285000.0	135000.0
10	2024.0	Ago	250000.0	90000.0	70000.0	285000.0	135000.0
11	2024.0	Sep	NaN	90000.0	65000.0	285000.0	130000.0

LIMPIEZA DE DATOS

Con base en la fuente de datos: Mercado_casa.xlsx:

Manejo de Valores nulos:

```
#Limpieza valores nulos (eliminarlos) (Todos los datos donde alguna celda sea nulo))  
df_sin_nulos2=df.dropna()  
print("\nMercado Casa sin nulos (Alguna celda que se nula)")  
print(df_sin_nulos2)
```

```
Mercado Casa sin nulos (Alguna celda que se nula)  
   Año  Mes  Víveres  Verduras  Frutas  Carnes  Lácteos  
1  2024.0  Feb  210000.0   65000.0  40000.0  270000.0  105000.0  
2  2024.0  Mar  210000.0   70000.0  45000.0  275000.0  105000.0  
4  2024.0  May  240000.0   85000.0  50000.0  280000.0  115000.0  
5  2024.0  Ago  250000.0   90000.0  70000.0  285000.0  135000.0  
6  2024.0  Jun  240000.0   85000.0  45000.0  285000.0  130000.0  
8  2024.0  Jul  245000.0   90000.0  65000.0  285000.0  130000.0  
9  2024.0  Ago  250000.0   90000.0  70000.0  285000.0  135000.0  
10 2024.0  Ago  250000.0   90000.0  70000.0  285000.0  135000.0
```

LIMPIEZA DE DATOS

Con base en la fuente de datos: Mercado_casa.xlsx:

Manejo de Valores nulos:

```
#Limpieza valores nulos (Poner algun valor) (Pone un valor en donde alguna celda sea nulo))
df_sin_nulos3=df.dropna(how="all")
df_sin_nulos4=df_sin_nulos3.fillna(-1)
print("\nMercado Casa (Elimina las filas totalmente nulas y las celdas nulas retantanes las llena -1)")
print(df_sin_nulos4)
```

```
Mercado Casa (Elimina las filas totalmente nulas y las celdas nulas retantanes las llena -1)
   Año  Mes  Víveres  Verduras  Frutas  Carnes  Lácteos
0  2024.0  Ene  200000.0  60000.0  40000.0    -1.0  100000.0
1  2024.0  Feb  210000.0  65000.0  40000.0  270000.0  105000.0
2  2024.0  Mar  210000.0  70000.0  45000.0  275000.0  105000.0
3  2024.0  Abr  225000.0  80000.0  42000.0    -1.0  110000.0
4  2024.0  May  240000.0  85000.0  50000.0  280000.0  115000.0
5  2024.0  Ago  250000.0  90000.0  70000.0  285000.0  135000.0
6  2024.0  Jun  240000.0  85000.0  45000.0  285000.0  130000.0
8  2024.0  Jul  245000.0  90000.0  65000.0  285000.0  130000.0
9  2024.0  Ago  250000.0  90000.0  70000.0  285000.0  135000.0
10 2024.0  Ago  250000.0  90000.0  70000.0  285000.0  135000.0
11 2024.0  Sep    -1.0  90000.0  65000.0  285000.0  130000.0
```


LIMPIEZA DE DATOS

Con base en la fuente de datos: Mercado_casa.xlsx:

Datos duplicados

```
import numpy as np
import pandas as pd

#Leer archivo excel e imprimir las primeras 5 filas
df = pd.read_excel('C:\\Otros\\UTB\\TalentoTech\\AnalisisDatos\\Material\\Semana_4\\Ejercicios\\Limpieza\\Mercado_casa.xlsx')
print("\nMercado Casa")
print(df)
```

Mercado Casa							
	Año	Mes	Víveres	Verduras	Frutas	Carnes	Lácteos
0	2024.0	Ene	200000.0	60000.0	40000.0	NaN	100000.0
1	2024.0	Feb	210000.0	65000.0	40000.0	270000.0	105000.0
2	2024.0	Mar	210000.0	70000.0	45000.0	275000.0	105000.0
3	2024.0	Abr	225000.0	80000.0	42000.0	NaN	110000.0
4	2024.0	May	240000.0	85000.0	50000.0	280000.0	115000.0
5	2024.0	Ago	250000.0	90000.0	70000.0	285000.0	135000.0
6	2024.0	Jun	240000.0	85000.0	45000.0	285000.0	130000.0
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8	2024.0	Jul	245000.0	90000.0	65000.0	285000.0	130000.0
9	2024.0	Ago	250000.0	90000.0	70000.0	285000.0	135000.0
10	2024.0	Ago	250000.0	90000.0	70000.0	285000.0	135000.0
11	2024.0	Sep	NaN	90000.0	65000.0	285000.0	130000.0

LIMPIEZA DE DATOS

Con base en la fuente de datos: Mercado_casa.xlsx:

Datos duplicados

```
#Limpieza valores duplicados (eliminarlos) (Todos los datos de la fila que están duplicados)  
df_sin_duplicados1=df.drop_duplicates()  
print("\nMercado Casa duplicados toda la fila")  
print(df_sin_duplicados1)
```

	Año	Mes	Víveres	Verduras	Frutas	Carnes	Lácteos
0	2024.0	Ene	200000.0	60000.0	40000.0	NaN	100000.0
1	2024.0	Feb	210000.0	65000.0	40000.0	270000.0	105000.0
2	2024.0	Mar	210000.0	70000.0	45000.0	275000.0	105000.0
3	2024.0	Abr	225000.0	80000.0	42000.0	NaN	110000.0
4	2024.0	May	240000.0	85000.0	50000.0	280000.0	115000.0
5	2024.0	Ago	250000.0	90000.0	70000.0	285000.0	135000.0
6	2024.0	Jun	240000.0	85000.0	45000.0	285000.0	130000.0
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8	2024.0	Jul	245000.0	90000.0	65000.0	285000.0	130000.0
11	2024.0	Sep	NaN	90000.0	65000.0	285000.0	130000.0

LIMPIEZA DE DATOS

Con base en la fuente de datos: Mercado_casa.xlsx:

Datos duplicados

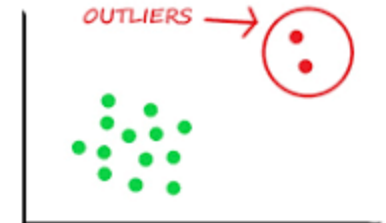
```
#Limpieza valores duplicados (eliminarlos) (Por columnas)  
df_sin_duplicados2=df.drop_duplicates(subset="Carnes")  
print("\nMercado Casa duplicados por columna (carnes)")  
print(df_sin_duplicados2)
```

	Año	Mes	Víveres	Verduras	Frutas	Carnes	Lácteos
0	2024.0	Ene	200000.0	60000.0	40000.0	NaN	100000.0
1	2024.0	Feb	210000.0	65000.0	40000.0	270000.0	105000.0
2	2024.0	Mar	210000.0	70000.0	45000.0	275000.0	105000.0
4	2024.0	May	240000.0	85000.0	50000.0	280000.0	115000.0
5	2024.0	Ago	250000.0	90000.0	70000.0	285000.0	135000.0

La limpieza de datos es crucial para garantizar la calidad de los análisis. Algunas técnicas avanzadas son:

Manejo de Outliers

```
# Identificar y manejar outliers  
limite_superior = df['columna'].quantile(0.95)  
df_sin_outliers = df[df['columna'] < limite_superior]
```



LIMPIEZA DE DATOS

Con base en la fuente de datos: Mercado_casa.xlsx: Hoja: Datos1

Año	Mes	Víveres	Verduras	Frutas	Carnes	Lácteos
2024	Ene	200.000	60.000	40.000	1.650.000	100.000
2024	Feb	210.000	65.000	40.000	270.000	105.000
2024	Mar	210.000	70.000	45.000	275.000	105.000
2024	Abr	225.000	80.000	42.000	20.000	110.000
2024	May	240.000	85.000	50.000	280.000	115.000
2024	Ago	250.000	90.000	70.000	285.000	135.000
2024	Jun	240.000	85.000	45.000	285.000	130.000
2024	Jul	245.000	90.000	65.000	285.000	130.000
2024	Ago	250.000	90.000	70.000	285.000	135.000

LIMPIEZA DE DATOS

Con base en la fuente de datos: Mercado_casa.xlsx:

Quitar Outliers

```
import numpy as np
import pandas as pd

#Leer archivo excel e imprimir las primeras 5 filas
df = pd.read_excel('C:\\Otro\\UTB\\TalentoTech\\AnálisisDatos\\Material\\Semana_4\\Ejercicios\\Limpieza\\Mercado_casa.xlsx')
print("\nMercado Casa")
print(df)
```

	Mercado Casa						
	Año	Mes	Víveres	Verduras	Frutas	Carnes	Lácteos
0	2024	Ene	200000	60000	40000	1650000	100000
1	2024	Feb	210000	65000	40000	270000	105000
2	2024	Mar	210000	70000	45000	275000	105000
3	2024	Abr	225000	80000	42000	20000	110000
4	2024	May	240000	85000	50000	280000	115000
5	2024	Ago	250000	90000	70000	285000	135000
6	2024	Jun	240000	85000	45000	285000	130000
7	2024	Jul	245000	90000	65000	285000	130000
8	2024	Ago	250000	90000	70000	285000	135000

LIMPIEZA DE DATOS

Con base en la fuente de datos: Mercado_casa.xlsx:

Quitar Outliers

```
#Limpieza valores Outliers (eliminarlos) (Tomando el cuartil superior)
limiteSuperior= df["Carnes"].quantile(0.95)
df_sin_outliers=df[df["Carnes"]<limiteSuperior]
print("\nMercado Casa sin Outliers Por el límite superior")
print(df_sin_outliers)
```

	Año	Mes	Víveres	Verduras	Frutas	Carnes	Lácteos
1	2024	Feb	210000	65000	40000	270000	105000
2	2024	Mar	210000	70000	45000	275000	105000
3	2024	Abr	225000	80000	42000	20000	110000
4	2024	May	240000	85000	50000	280000	115000
5	2024	Ago	250000	90000	70000	285000	135000
6	2024	Jun	240000	85000	45000	285000	130000
7	2024	Jul	245000	90000	65000	285000	130000
8	2024	Ago	250000	90000	70000	285000	135000

LIMPIEZA DE DATOS

Con base en la fuente de datos: Mercado_casa.xlsx:

Quitar Outliers

```
#Limpieza valores Outliers (eliminarlos) (Tomando el cuartil inferior)  
limiteInferior= df["Carnes"].quantile(0.05)  
df_sin_outliers=df_sin_outliers[df_sin_outliers["Carnes"]>limiteInferior]  
print("\nMercado Casa sin Outliers por el límite inferior")  
print(df_sin_outliers)
```

	Año	Mes	Víveres	Verduras	Frutas	Carnes	Lácteos
1	2024	Feb	210000	65000	40000	270000	105000
2	2024	Mar	210000	70000	45000	275000	105000
4	2024	May	240000	85000	50000	280000	115000
5	2024	Ago	250000	90000	70000	285000	135000
6	2024	Jun	240000	85000	45000	285000	130000
7	2024	Jul	245000	90000	65000	285000	130000
8	2024	Ago	250000	90000	70000	285000	135000

La limpieza de datos es crucial para garantizar la calidad de los análisis. Algunas técnicas avanzadas son:

Datos con Basura - Ortografía



Función	Descripción
<code>str.isalnum()</code>	Retorna True si la cadena es alfanumérica, caso contrario retorna False
<code>str.isalpha()</code>	Retorna True si la cadena es alfabética, caso contrario retorna False
<code>str.isdigit()</code>	Retorna True si la cadena es numérica, caso contrario retorna False
<code>str.isdecimal()</code>	Retorna True si la cadena es decimal, caso contrario retorna False
<code>str.islower()</code>	Retorna True si la cadena contiene solo minúsculas, caso contrario retorna False
<code>str.isupper()</code>	Retorna True si la cadena contiene solo mayúsculas, caso contrario retorna False

Función	Descripción
<code>lower()</code>	Devuelve un string con todos sus caracteres en minúsculas.
<code>upper()</code>	Devuelve un string con todos sus caracteres en mayúsculas.
<code>replace(x,y)</code>	Devuelve un string con cada ocurrencia de x reemplazada por y.
<code>count(x)</code>	Cuenta el número de ocurrencias de x en un string.
<code>index(x)</code>	Devuelve la ubicación de la primera ocurrencia de x.
<code>isalpha()</code>	Devuelve True si cada carácter del string es una letra.
<code>a,b=split(x)</code>	Divide la cadena en N variables de acuerdo al carácter x
<code>a,b=rsplit(x)</code>	Divide la cadena en N variables de acuerdo al carácter x (reverse)

LIMPIEZA DE DATOS

Con base en la fuente de datos: LimpiezaOrtografia.xlsx: Hoja: Datos

Id	Fecha	Nombre	Ciudad	Estrato	ValorFactura
1	2/01/2024	Sergio Arturo Medina Castillo	Bogota	4	120.000
2	3/01/2024	LUISA LANE	Bogotá	3	85.000
3	4/02/2024	Pedro LOPEZ	Medellín	5	180.000
4	10/02/2024	Carlos Contreras	Medellín	2	50.000
5	5/03/2024	Catalina Torres	Cartagena	3	100.000
6	6/03/2024	VALENTINA yepez	Cartagena	2	60.000
7	10/04/2024	Hugo García	Cali	5	200.000
8	11/04/2024	Rosa Urrutia	Cali	4	100.000
9	4/05/2024	Gustavo Jerez	Bogota	3	90.000
10	5/05/2024	Gabriela Soto	Bucaramanga	2	55.000
11	10/05/2024	Hugo Gómez	Medellin	5	79.000

Leer Fuente de Datos

Datos con Basura - Ortografía

```
# Limpieza de Datos
# Datos con B́asura o errores ortogŕaficos

import numpy as nd
import pandas as pd

#Leer archivo excel e imprimir las primeras 5 filas
df_original = pd.read_excel('C:\\Otros\\UTB\\TalentoTech\\AnálisisDatos\\Material\\Semana_4\\Ejercicios\\Limpieza\\LimpiezaO
print("Fuente de Datos")
print(df_original)
```

	Id	Fecha	Nombre	Ciudad	Estrato	\
0	1	2024-01-02	Sergio Arturo Medina Castillo	Bogota	4	
1	2	2024-01-03	LUISA LANE	Bogotá	3	
2	3	2024-02-04	Pedro LOPEZ	Medellín	5	
3	4	2024-02-10	Carlos Contreras	Medellín	2	
4	5	2024-03-05	Catalina Torres	Cartagena	3	
5	6	2024-03-06	VALENTINA yepez	Cartagena	2	
6	7	2024-04-10	Hugo García	Cali	5	
7	8	2024-04-11	Rosa Urrutia	Cali	4	
8	9	2024-05-04	Gustavo Jerez	Bogota	3	
9	10	2024-05-05	Gabriela Soto	Bucaramanga	2	
10	11	2024-05-10	Hugo Gómez	Medellin	5	
		ValorFactura				
0		120000				
1		85000				
2		180000				
3		50000				
4		100000				
5		60000				
6		200000				
7		100000				
8		90000				
9		55000				
10		79000				

Convertir a Mayúsculas (upper) - Nombre, Ciudad

Datos con Basura - Ortografía

```
# Funciones de Cadenas de Caracteres
# Convertir a Mayúsculas (upper) o minúsculas (uplow)
# Nombres y ciudades en Mayúsculas
df=df_original
df["Nombre"]=df["Nombre"].str.upper()
df["Ciudad"]=df["Ciudad"].str.upper()
print("Convertir a Mayúsculas")
print(df)
```

	Id	Fecha	Nombre	Ciudad	Estrato	\
0	1	2024-01-02	SERGIO ARTURO MEDINA CASTILLO	BOGOTA	4	
1	2	2024-01-03	LUISA LANE	BOGOTÁ	3	
2	3	2024-02-04	PEDRO LOPEZ	MEDELLÍN	5	
3	4	2024-02-10	CARLOS CONTRERAS	MEDELLÍN	2	
4	5	2024-03-05	CATALINA TORRES	CARTAGENA	3	
5	6	2024-03-06	VALENTINA YEPEZ	CARTAGENA	2	
6	7	2024-04-10	HUGO GARCÍA	CALI	5	
7	8	2024-04-11	ROSA URRUTIA	CALI	4	
8	9	2024-05-04	GUSTAVO JEREZ	BOGOTA	3	
9	10	2024-05-05	GABRIELA SOTO	BUCARAMANGA	2	
10	11	2024-05-10	HUGO GÓMEZ	MEDELLIN	5	
		ValorFactura				
0		120000				
1		85000				
2		180000				
3		50000				
4		100000				
5		60000				
6		200000				
7		100000				
8		90000				
9		55000				
10		79000				

Agrupar datos categóricos – Calidad de Dato

Ejemplo: Ciudad

Datos con Basura - Ortografía

```
# Agrupar datos categóricos - Limpieza de Datos  
df1=df_original.groupby("Ciudad").ValorFactura.sum().reset_index()  
print(df1)
```

	Ciudad	ValorFactura
0	BOGOTA	210000
1	BOGOTÁ	85000
2	BUCARAMANGA	55000
3	CALI	300000
4	CARTAGENA	160000
5	MEDELLIN	79000
6	MEDELLÍN	230000

Errores Ortográficos o Basura (replace) – Ciudad: BOGOTA

Datos con Basura - Ortografía

```
df.replace({"Ciudad":{"BOGOTA":"BOGOTÁ","MEDELLIN":"MEDELLÍN"}}, inplace=True)  
print(df)
```

	Id	Fecha	Nombre	Ciudad	Estrato
0	1	2024-01-02	SERGIO ARTURO MEDINA CASTILLO	BOGOTA	4
1	2	2024-01-03	LUISA LANE	BOGOTÁ	3
2	3	2024-02-04	PEDRO LOPEZ	MEDELLÍN	5
3	4	2024-02-10	CARLOS CONTRERAS	MEDELLÍN	2
4	5	2024-03-05	CATALINA TORRES	CARTAGENA	3
5	6	2024-03-06	VALENTINA YEPEZ	CARTAGENA	2
6	7	2024-04-10	HUGO GARCÍA	CALI	5
7	8	2024-04-11	ROSA URRUTIA	CALI	4
8	9	2024-05-04	GUSTAVO JEREZ	BOGOTA	3
9	10	2024-05-05	GABRIELA SOTO	BUCARAMANGA	2
10	11	2024-05-10	HUGO GÓMEZ	MEDELLIN	5
ValorFactura					
0		120000			
1		85000			
2		180000			
3		50000			
4		100000			
5		60000			
6		200000			
7		100000			
8		90000			
9		55000			
10		79000			



Transformación de Datos

LIMPIEZA – TRANSFORMACIÓN DE DATOS

La limpieza de datos es crucial para garantizar la calidad de los análisis. Algunas técnicas avanzadas son:

Operación de Merge

```
# Combinar dos DataFrames por una columna común  
df_combinado = pd.merge(df1, df2, on='columna_comun', how='inner')
```

Operaciones de Pivot y Melt

```
# Pivotar datos  
df_pivotado = df.pivot(index='fila', columns='columna', values='valor')  
  
# Derretir datos  
df_derretido = pd.melt(df_pivotado, id_vars='fila', value_vars=['col1', 'col2'])
```

Operaciones de Ventana(Window Functions)

```
# Calcular promedio móvil  
df['promedio_movil'] = df['columna'].rolling(window=3).mean()
```

Operación:

Operación de Merge

Cruzar información por una columna común

DataFrame 1:

	id	nombre	estrato
0	10	Juan	1
1	20	Pedro	3
2	30	Luis	4

DataFrame 2:

	estrato	descripcion
0	1	bajo-bajo
1	2	bajo
2	3	medio
3	4	medio-alto
4	5	alto



DataFrame Merge:

	id	nombre	estrato	descripcion
0	10	Juan	1	bajo-bajo
1	20	Pedro	3	medio
2	30	Luis	4	medio-alto

LIMPIEZA DE DATOS - TRANSFORMACIÓN

Operación:

Operación de Merge

```
import pandas as pd

df1=pd.DataFrame({"id":[10,20,30],"nombre":["Juan","Pedro","Luis"],"estrato":[1,3,4]})
df2=pd.DataFrame({"estrato":[1,2,3,4,5],"descripcion":["bajo-bajo","bajo","medio","medio-alto","alto"]})

print("DataFrame 1: ")
print(df1)
print("DataFrame 2: ")
print(df2)
```

```
DataFrame 1:
   id nombre  estrato
0  10   Juan        1
1  20  Pedro        3
2  30   Luis        4
DataFrame 2:
   estrato descripcion
0         1   bajo-bajo
1         2         bajo
2         3         medio
3         4   medio-alto
4         5         alto
```

Operación:

Operación de Merge

```
df_merge=pd.merge(df1,df2,on="estrato",how="inner")  
print("DataFrame Merge: ")  
print(df_merge)
```

```
DataFrame Merge:  
   id nombre  estrato descripcion  
0  10   Juan        1    bajo-bajo  
1  20  Pedro        3         medio  
2  30   Luis        4    medio-alto
```


LIMPIEZA DE DATOS - TRANSFORMACIÓN

Operación: Operaciones de Pivot

Convertir Filas en Columnas

Información original:

	Fecha	Producto	Venta
0	2023-01	Samsung	100
1	2023-01	Apple	150
2	2023-02	Samsung	200
3	2023-02	Apple	120



Información pivoteada:

Producto	Apple	Samsung
Fecha		
2023-01	150	100
2023-02	120	200

Operación: Operaciones de Pivot

```
import pandas as pd

products_df = pd.DataFrame({
    'Fecha': ['2023-01', '2023-01', '2023-02', '2023-02'],
    'Producto': ['Samsung', 'Apple', 'Samsung', 'Apple'],
    'Venta': [100, 150, 200, 120]
})

print("Información original:")
print(products_df)
```

Información original:

	Fecha	Producto	Venta
0	2023-01	Samsung	100
1	2023-01	Apple	150
2	2023-02	Samsung	200
3	2023-02	Apple	120

Operación: Operaciones de Pivot

```
pivot_df = products_df.pivot(index='Fecha', columns='Producto', values='Venta')  
print("\nInformación pivoteada:")  
print(pivot_df)
```

```
Información pivoteada:  
Producto  Apple  Samsung  
Fecha  
2023-01      150      100  
2023-02      120      200
```

LIMPIEZA DE DATOS - TRANSFORMACIÓN

Operación: Operaciones Melt

Convertir Columnas en filas

Información original:

	Fecha	Samsung	Apple
0	2023-01	400	300
1	2023-02	600	550



Resultado del Melt

	Fecha	variable	value
0	2023-01	Samsung	400
1	2023-02	Samsung	600
2	2023-01	Apple	300
3	2023-02	Apple	550

Operación: Operaciones Melt

```
import pandas as pd

products_df = pd.DataFrame({
    'Fecha': ['2023-01', '2023-02'],
    'Samsung': [400, 600],
    'Apple': [300, 550]
})

print("Información original:")
print(products_df)
```

Información original:			
	Fecha	Samsung	Apple
0	2023-01	400	300
1	2023-02	600	550

Operación: Operaciones Melt

```
derretido_df=pd.melt(products_df,id_vars="Fecha",value_vars=["Samsung","Apple"])  
print("\nResultado del Melt")  
print(derretido_df)
```

	Fecha	variable	value
0	2023-01	Samsung	400
1	2023-02	Samsung	600
2	2023-01	Apple	300
3	2023-02	Apple	550

❖ **Uso de condiciones para filtrar datos:**

- Filtrar basado en una condición
- Filtrar con varias condiciones(AND)
- Filtrar con varias condiciones (OR)

❖ **Operadores Lógicos en la selección de datos**

- Operador 'isin()' el cual se utiliza para filtrar filas donde los valores de una columna específica están presentes en una lista dada. Es útil cuando deseas seleccionar filas que contienen valores específicos en una columna.
- Operador 'between()': su función es filtrar filas donde los valores de una columna están dentro de un rango especificado. Esta técnica es útil cuando se desea seleccionar filas basadas en valores numéricos dentro de un rango específico.

Programa con Filtros:

Datos de prueba

```
import pandas as pd

df= pd.DataFrame({
    'A': [1,2,3],
    'B': [4,5,6],
    'C': [-1,0,1]
})
print("\nInformación original:")
print(df)
```

Información original:

	A	B	C
0	1	4	-1
1	2	5	0
2	3	6	1

Programa con Filtros:

Filtrado por condición: Elementos de la columna C
que sean mayores que cero

```
#1. Filtrar por condiciones
#1.1. Filtrado basado en una condición
df_filtrado_simple=df[df["C"]>0]
print("\nFiltrado basado en una condición:")
print(df_filtrado_simple)
```

Filtrado basado en una condición:

	A	B	C
2	3	6	1

Programa con Filtros:

Filtrado por varias condiciones – Operador AND:
Elementos de la columna C que sean mayores o iguales que cero y la columna B que sea mayor o igual a 6

```
#1.2. Filtrado basado en dos condiciones (AND)
df_filtrado_and=df[(df["C"]>=0) & (df["B"]<=6)]
print("\nFiltrado condicionado con AND:")
print(df_filtrado_and)
```

Filtrado condicionado con AND:

	A	B	C
1	2	5	0
2	3	6	1

Programa con Filtros:

Filtrado por varias condiciones – Operador OR:
Elementos de la columna C que sean mayores o iguales que cero o la columna A que sea mayor que 1

```
#1.3. Filtrado basado en dos condiciones (OR)
df_filtrado_or=df[(df["C"]>=0) | (df["A"]>1)]
print("\nFiltrado condicionado con OR:")
print(df_filtrado_or)
```

Filtrado condicionado con OR:

	A	B	C
1	2	5	0
2	3	6	1

Programa con Filtros:

Filtrado con método `isin`: Selecciona valores del DataFrame que se encuentren en una lista dada

```
#2. Operadores lógicos en la selección de datos
#2.1. Operador isin()
valores_seleccionar=[1,3,5]
df_filtrado_isin=df[df["C"].isin(valores_seleccionar)]
print("\nFiltrado con isin():")
print(df_filtrado_isin)
```

```
Filtrado con isin():
   A  B  C
2  3  6  1
```

Fuente de Datos:

Columna	Descripción
ID Cliente	Identificación del cliente (Código)
Zona	Zona de la venta en el mundo. Se manejan 5 zonas. Algo parecido a los continentes
País	País del cliente donde se realiza la venta
Tipo de producto	Son los productos que se ofrecen para las ventas
Canal de venta	Si es en línea o fuera de línea
Prioridad	Prioridad asignada a la venta, puede ser alta, media, baja, crítica
Fecha pedido	Fecha de realización del pedido (DD/MM/AAAA)
ID Pedido	Identificación o código asignado al pedido
Fecha envío	Fecha de envío del pedido (DD/MM/AAAA)
Unidades	Unidades del producto vendidas en el pedido
Precio Unitario	Precio unitario del producto
Coste unitario	Costo de producción del producto en el pedido
Importe venta total	Valor total de la venta
Importe Coste total	Valor total de producción de los productos del pedido

Programa con Filtros:

Filtrado con método `between`: Selecciona valores del DataFrame que se encuentren en el rango de valores especificado

```
#2.1. Operador between()  
df_filtrado_between=df[df["A"].between(2,5)]  
print("\nFiltrado con between():")  
print(df_filtrado_between)
```

Filtrado con `between()`:

	A	B	C
1	2	5	0
2	3	6	1

Podemos realizar agrupación de datos usando el método 'groupby' de la siguiente manera:



Posterior a dicha agrupación usando 'groupby' es común realizar operaciones agregadas, es decir, aplicar funciones que resumen o agregan información dentro de cada grupo. Algunas funciones agregadas comunes incluyen calcular la suma, el promedio, el mínimo, el máximo, la desviación estándar, etc.

Esto lo podríamos realizar así:

```
ventas = pd.DataFrame({  
    "Producto": ["A", "B", "C", "B", "A", "A"],  
    "Ventas": [6, 2, 1, 4, 5, 2]  
})  
ventas
```

```
ventas.groupby(by = "Producto").mean()
```

También se pueden realizar cálculos básicos como suma, promedio y conteo así:

```
suma_por_grupo = grupos['columna_1'].sum()
```

Suma por Grupo

```
promedio_por_grupo = grupos['columna_2'].mean()
```

Promedio por Grupo

```
conteo_por_grupo = grupos.size()
```

Conteo de Elementos por grupo

Programa con Agrupación:

Datos de prueba

```
import pandas as pd

df= pd.DataFrame({
    'producto': ["A","B","C","A","B","A","C"],
    'venta': [10,15,20,20,25,20,30]
})
print("\nInformación original:")
print(df)
```

	producto	venta
0	A	10
1	B	15
2	C	20
3	A	20
4	B	25
5	A	20
6	C	30

Programa con Agrupación:

Agrupar por la columna producto y generar la media o promedio

```
#Agrupar por producto  
grupos=df.groupby(by="producto").mean()  
print("\nMedia de una Agrupación:")  
print(grupos)
```

```
Media de una Agrupación:  
      venta  
producto  
A      16.666667  
B      20.000000  
C      25.000000
```

La limpieza de datos es crucial para garantizar la calidad de los análisis. Algunas técnicas avanzadas son:

Transformación de Columnas

```
# Aplicar una función a una columna
df['columna'] = df['columna'].apply(funcion)

# Crear una nueva columna basada en valores existentes
df['nueva_columna'] = df['columna'].map(lambda x: x * 2)
```

Con base en la fuente de datos: Mercado_casa.xlsx:

Transformación

Fuente de Datos

```
import numpy as nd
import pandas as pd

ventas=pd.DataFrame(
    {"A":[1,3,9],
     "B":[4,8,2],
     "C":[7,3,5]},
    index=["Ene","Feb","Mar"])
print("\nVentas")
print(ventas)
```

Ventas			
	A	B	C
Ene	1	4	7
Feb	3	8	3
Mar	9	2	5

LIMPIEZA DE DATOS - TRANSFORMACIÓN

Con base en la fuente de datos: Mercado_casa.xlsx:

Transformación

Calcular la suma de las columnas

```
#Función para hallar la diferencia del mayor y menos por columna  
def rango(s):  
    return max(s)-min(s)  
salida=ventas.apply(rango)  
print("\nSalida Diferencia mayor y menor por columnas")  
print(salida)
```

```
Salida Diferencia mayor y menor por columnas  
A      8  
B      6  
C      4  
dtype: int64
```

LIMPIEZA DE DATOS - TRANSFORMACIÓN

Con base en la fuente de datos: Mercado_casa.xlsx:

Transformación

Calcular la suma de las filas

```
#Función para hallar la diferencia del mayor y menos por fila  
salida1=ventas.apply(rango,axis=1)  
print("\nSalida Diferencia mayor y menor por filas")  
print(salida1)
```

```
Salida Diferencia mayor y menor por filas  
Ene      6  
Feb      5  
Mar      7  
dtype: int64
```

LIMPIEZA DE DATOS - TRANSFORMACIÓN

Con base en la fuente de datos: Mercado_casa.xlsx:

Transformación

Multiplicar por dos los elementos de la Columna A

```
#Transformación Columna (A se elevara al cuadrado)  
ventas["duplicada"] = ventas["A"].apply(lambda x: x*2)  
print("\nSalida columna nueva duplicandolos valores de la columna A")  
print(ventas)
```

```
Salida columna nueva duplicandolos valores de la columna A  
   A  B  C  duplicada  
Ene  1  4  7         2  
Feb  3  8  3         6  
Mar  9  2  5        18
```




ADVANCE
Consortium

GRACIAS

