SPRINT S8.01 – PYTHON: VISUALIZACIONES EN PYTHON

ANEXO: RESUMEN DE ACTUALIZACIONES

A continuación se presentan a grandes rasgos las principales modificaciones a los ejercicios, siguiendo las correcciones propuestas. Para leer el informe completo actualizado, por favor **dirigirse al documento principal**, denominado: **SprintS08_01_JosephT.ipynb**

NIVEL 1

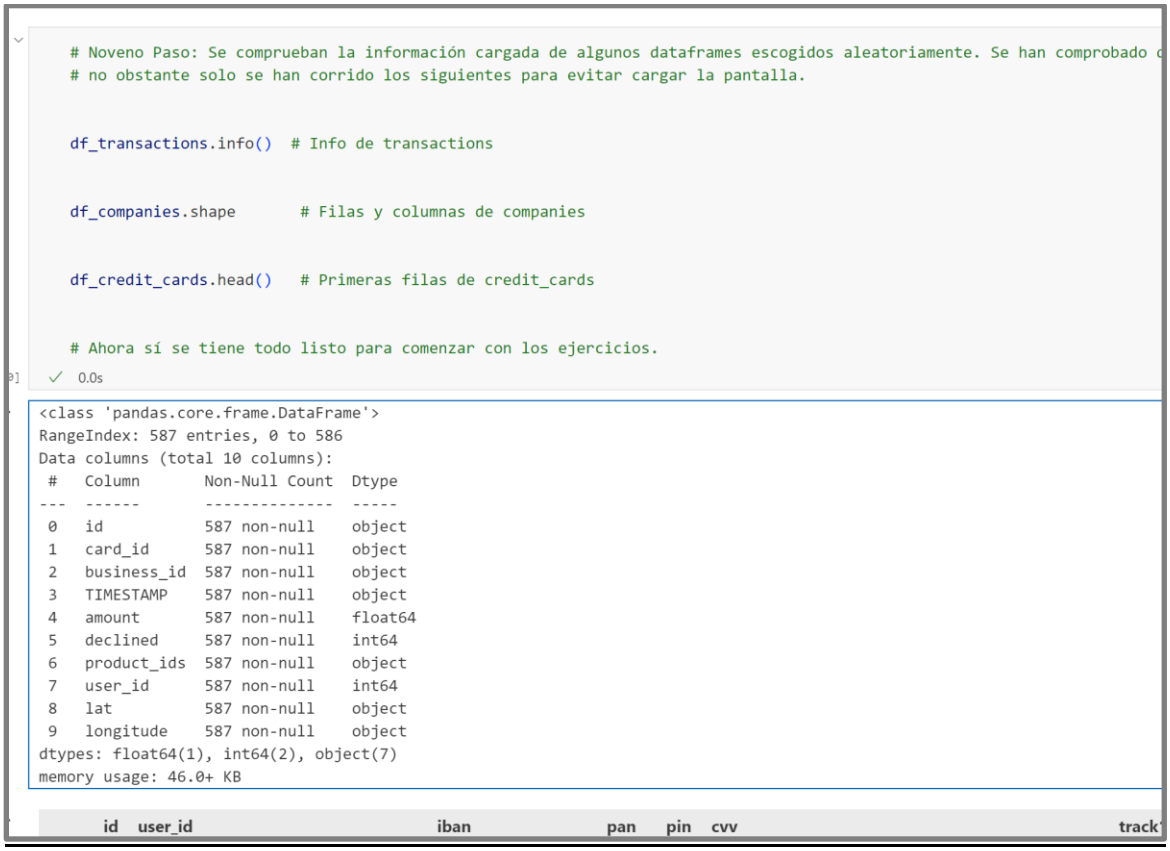
Cambios generales:

- Se ha reducido volumen dentro de los scripts de cada ejercicio, evitando hacer comentarios sobre funciones o partes del código muy sencillas y fáciles de entender.

- Se han realizado modificaciones a los ejercicios, los cuales se detallan a lo largo del presente documento.

- A la hora de verificar la carga correcta de los datos se ha decidido por inspeccionar los DataFrame uno a uno, creando diferentes celdas de código, mejorando asi la lectura.

i) Antes



```
# Noveno Paso: Se comprueban la información cargada de algunos dataframes escogidos aleatoriamente. Se han comprobado c
# no obstante solo se han corrido los siguientes para evitar cargar la pantalla.

df_transactions.info()  # Info de transactions

df_companies.shape       # Filas y columnas de companies

df_credit_cards.head()   # Primeras filas de credit_cards

# Ahora sí se tiene todo listo para comenzar con los ejercicios.
```

0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 587 entries, 0 to 586
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   id           587 non-null    object
 1   card_id      587 non-null    object
 2   business_id  587 non-null    object
 3   TIMESTAMP    587 non-null    object
 4   amount       587 non-null    float64
 5   declined     587 non-null    int64
 6   product_ids  587 non-null    object
 7   user_id      587 non-null    int64
 8   lat          587 non-null    object
 9   longitude    587 non-null    object
dtypes: float64(1), int64(2), object(7)
memory usage: 46.0+ KB
```

id	user_id	iban	pan	pin	cvv	track

i) Ahora:

```
# Resumen de la estructura de 'transactions':

df_transactions.info()
```

✓ 0.3s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 587 entries, 0 to 586
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   id           587 non-null    object 
 1   card_id      587 non-null    object 
 2   business_id  587 non-null    object 
 3   TIMESTAMP    587 non-null    object 
 4   amount       587 non-null    float64
 5   declined     587 non-null    int64  
 6   product_ids  587 non-null    object 
 7   user_id      587 non-null    int64  
 8   lat          587 non-null    object 
 9   longitude    587 non-null    object 
dtypes: float64(1), int64(2), object(7)
memory usage: 46.0+ KB
```

```
# Primeras filas de 'credit_cards':

df_credit_cards.head()
```

✓ 0.0s

	id	user_id	iban	
0	CcU-2938	275	TR301950312213576817638661	5424465566813
1	CcU-2945	274	DO28547637485374752165686 89	5142423821948

Ejercicio 1.1

Una variable numérica.

Modificaciones:

ii) Antes:

iii) Ahora:

```
# E 1.1:

df_histogram = df_transactions[df_transactions['declined'] == 0]
amount = df_histogram['amount']

plt.figure(figsize=(8, 5))
sns.histplot(data=amount)

plt.xlabel('Monto de transacción', fontsize=14)
plt.ylabel('Número de transacciones', fontsize=14)
mean_value = amount.mean()
plt.axvline(mean_value, color='red', linestyle='--', linewidth=1)
plt.text(mean_value + 100, 20, f'Promedio: {mean_value:.2f}', color='red')
plt.title('E 1.1 Distribución de frecuencias del monto de las transacciones', fontsize=15)

# Se muestra el gráfico

plt.show()
```

✓ 0.1s

Resumen:

- El script tiene menos volumen pues se han minimizado los comentarios y su lectura es más clara.
- La gráfica resultante es la misma.
- Actualización de la interpretación.

Ejercicio 1.2

Dos variales numéricas.

Modificaciones:

i) Antes:

```
# Ejercicio 1.2: Dos variales numéricas --> Se utilizan los campos 'price' y 'weight' de la tabla 'products'.

# Se extraen las columnas 'price' y 'weight' del DataFrame

price = df_products['price']
weight = df_products['weight']

# Se crea un gráfico de dispersión

plt.scatter(x=price, y=weight, color='brown', s=100)   # Puntos color marrón y tamaño 100.

plt.xlabel('Precio', fontsize=14)    # Etiqueta del eje X
plt.ylabel('Peso', fontsize=14)      # Etiqueta del eje Y
plt.title("E 1.2 Relación entre el precio y el peso del producto", fontsize=15)   # Título del gráfico
plt.grid(True, linestyle='--', alpha=0.7)           # Cuadrícula punteada con transparencia

# Se muestra el gráfico
plt.show()
```

ii) Ahora:

```
# E 1.2:

price = df_products['price']
weight = df_products['weight']

plt.scatter(x=price, y=weight, color='brown', s=100)

plt.xlabel('Precio', fontsize=14)
plt.ylabel('Peso', fontsize=14)
plt.title("E 1.2 Relación entre el precio y el peso del producto", fontsize=15)
plt.grid(True, linestyle='--', alpha=0.7)

# Se muestra el gráfico
plt.show()
```

✓ 0.2s

Resumen:

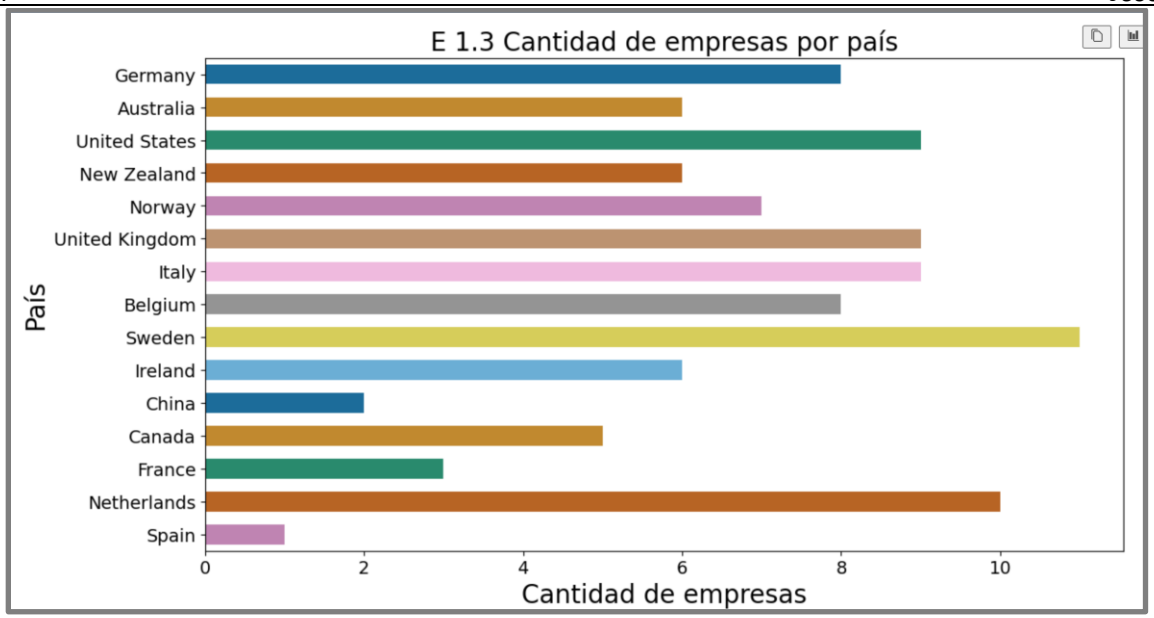
- Al igual que el ejercicio anterior se ha evitado hacer comentarios poco relevantes dentro del script.
- La gráfica resultante es la misma.
- Actualización de la interpretación.

Ejercicio 1.3

Una variable categórica.

i) Antes:

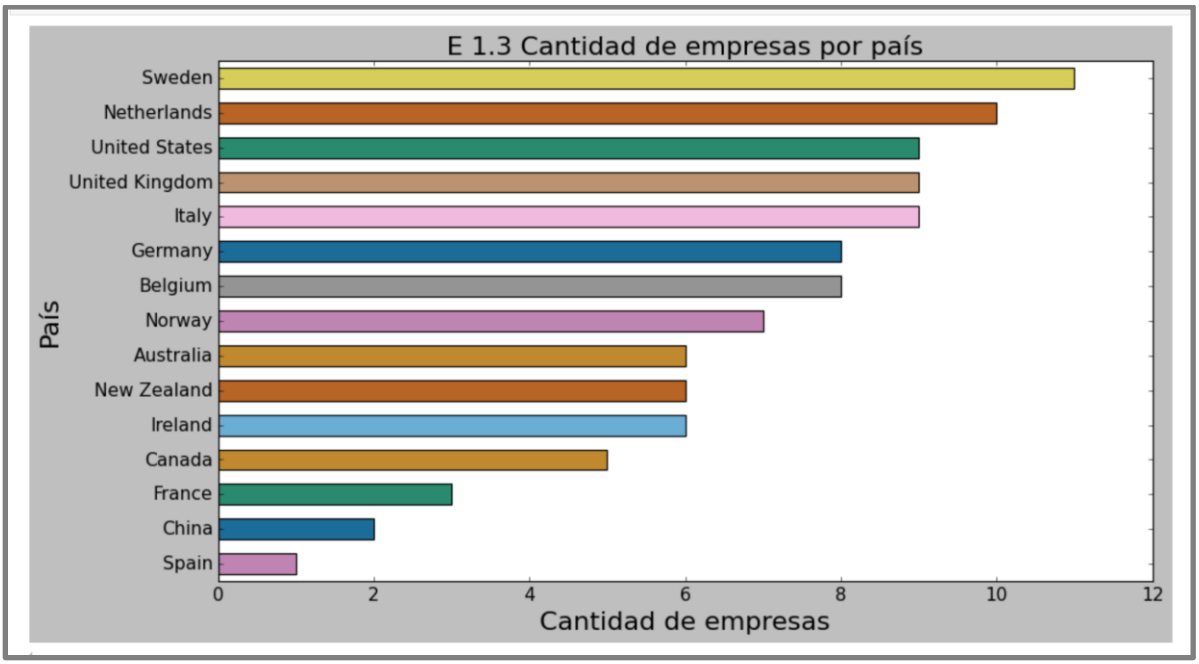
```
# Ejercicio 1.3: Una variable categórica --> Se utiliza el campo 'country' de la tabla 'companies'.


# Se crea un gráfico de barras para contar las empresas por país

plt.figure(figsize=(13, 7))    # Tamaño de la gráfica
sns.countplot(y='country', data=df_companies, hue='country', palette='colorblind', width=0.6)

# Se añaden las etiquetas para el título y los ejes.
plt.xlabel('Cantidad de empresas', fontsize=20)
plt.ylabel('País', fontsize=20)
plt.title('E 1.3 Cantidad de empresas por país', fontsize=20)

# Ajuste del tamaño de las etiquetas de los ejes
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)

# Se muestra el gráfico:
plt.show()
```



i) Ahora:

```
# E 1.3:

orden_country = df_companies['country'].value_counts().index

plt.figure(figsize=(13, 7))
sns.countplot(y='country', data=df_companies, hue='country',order = orden_country, palette='colorblind', width=0.6)

plt.xlabel('Cantidad de empresas', fontsize=20)
plt.ylabel('País', fontsize=20)
plt.title('E 1.3 Cantidad de empresas por país', fontsize=20)

plt.xticks(fontsize=14)
plt.yticks(fontsize=14)

# Se muestra el gráfico:
plt.show()
```

✓ 0.6s



Resumen:

- Reducción de los comentarios dentro del script.
- Se ha ordenado la lista de países por cantidad de empresas de manera descendente.
- Se visualiza más orden en la gráfica y facilita su interpretación.

Ejercicio 1.4

Una variable categórica y una numérica.

Modificaciones:

i) Antes:

```
# Ejercicio 1.4: Una variable categórica y una numérica --> Se utilizan los campos 'country' y 'amount' de las tablas 'companies'
#  y 'transactions', respectivamente.

# Combina las tablas transacciones y empresas, añadiendo 'country'
companies_transactions = df_transactions.merge(df_companies[['company_id', 'country']], left_on='business_id', right_on='company_id')

ventas_pais = companies_transactions.groupby('country')['amount'].sum()  # Se agrupa por país y suma las ventas
ventas_pais = ventas_pais.sort_values(ascending=False)          #  Se ordena las ventas de mayor a menor

# Se crea el gráfico

plt.figure(figsize=(15,9))
ventas_pais.plot(kind='barh')     # Gráfica de barras horizontales

# Título y etiquetas de los ejes

plt.xlabel('ventas totales', fontsize=17)
plt.ylabel('país', fontsize=17)
plt.title('E 1.4 ventas totales por país', fontsize=20)
plt.xticks(rotation=70)  # Se rota las etiquetas del eje X

# Se muestra el gráfico

plt.show()
```

ii) Ahora:

```
# E 1.4:

companies_transactions = df_transactions.merge(df_companies[['company_id', 'country']], left_on='business_id', right_on='company_id')

ventas_pais = companies_transactions.groupby('country')['amount'].sum()
ventas_pais = ventas_pais.sort_values(ascending=False)

plt.figure(figsize=(15,9))
ventas_pais.plot(kind='barh')

plt.xlabel('ventas totales', fontsize=17)
plt.ylabel('país', fontsize=17)
plt.title('E 1.4 ventas totales por país', fontsize=20)
plt.xticks(rotation=70)

# Se muestra el gráfico

plt.show()
```

✓ 0.4s

Resumen:

- Reducción de los comentarios dentro del script.
- La gráfica resultante es la misma.

Ejercicio 1.5

Dos variables categóricas.

Modificaciones:

i) Antes:

```
# Ejercicio 1.5: Dos variables categóricas --> Se utilizan los campos 'declined' y 'país' de las tablas 'transactions'
#  y 'companies', respectivamente.

# Se reutilizará el merge creado anteriormente denominado companies_transactions (del E 1.4)

# Se agrupa por país y se cuentan las transacciones aceptadas y rechazadas
data_agrupada = companies_transactions.groupby('country')['declined'].value_counts().unstack()

data_agrupada.columns = ['Aceptadas', 'Rechazadas']         # Se renombran columnas

# Se crea el gráfico de barras apiladas

data_agrupada.plot(kind='bar', stacked=True, fontsize=8, colormap ='coolwarm' )

# Título y etiquetas de los ejes X e Y.
plt.title('E 1.5 Transacciones aceptadas y rechazadas por países', fontsize=18)
plt.xlabel('País')
plt.ylabel('Cantidad de transacciones')

plt.xticks(rotation=60)    # Se rotan las etiquetas del eje X

# Se muestra el gráfico

plt.show()
```

ii) Ahora:

```
# E 1.5:

data_agrupada = companies_transactions.groupby('country')['declined'].value_counts().unstack()

data_agrupada.columns = ['Aceptadas', 'Rechazadas']

data_agrupada.plot(kind='bar', stacked=True, fontsize=8, colormap ='coolwarm' )

plt.title('E 1.5 Transacciones aceptadas y rechazadas por países', fontsize=18)
plt.xlabel('País')
plt.ylabel('Cantidad de transacciones')
plt.xticks(rotation=60)

# Se muestra el gráfico

plt.show()
```

✓ 0.3s

Resumen:

- Reducción de los comentarios dentro del script.
- La gráfica resultante es la misma.

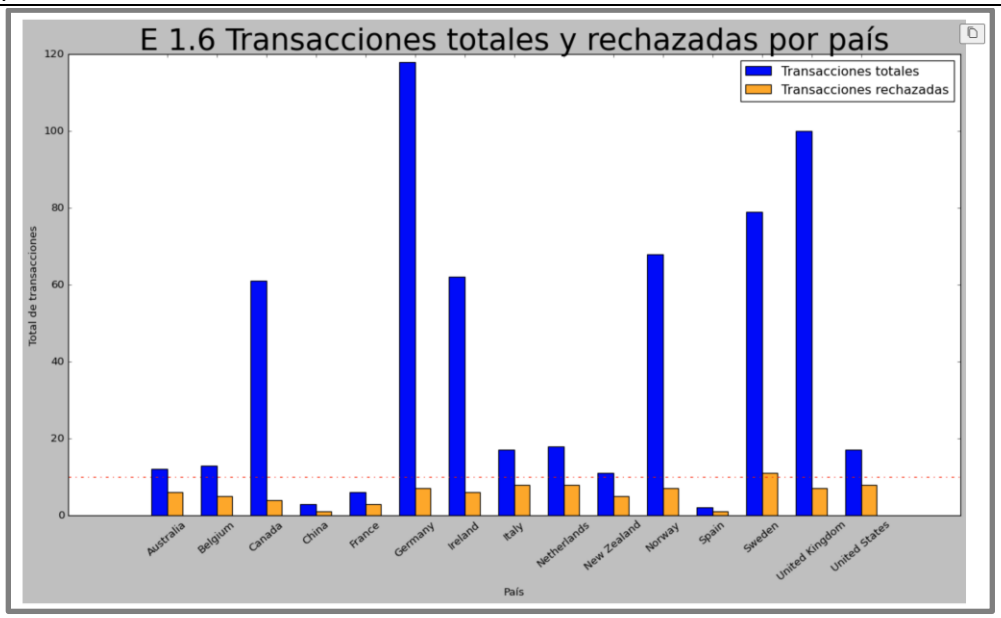
Ejercicio 1.6

Tres variables.

Modificaciones:

i) Antes:

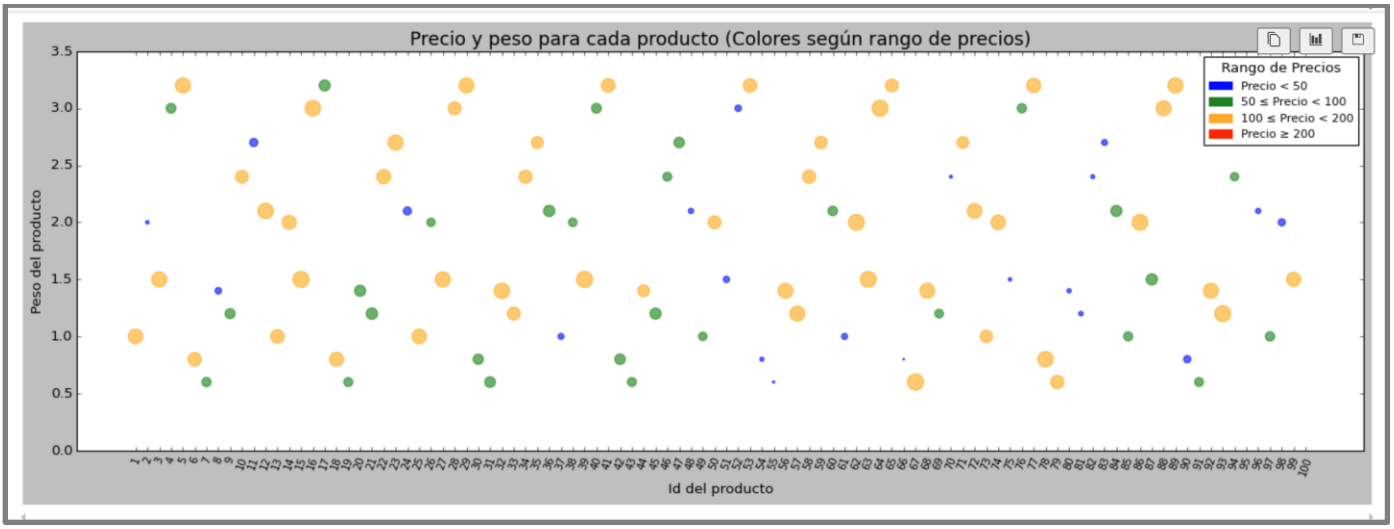
```
# Se reutilizará el merge creado anteriormente denominado companies_transactions (del E 1.4)

# Se agrupa las transacciones totales por país

ventas_pais = companies_transactions.groupby('country')['id'].count()

# Se agrupa las transacciones rechazadas por país

rechazos_pais = companies_transactions.groupby('country')['declined'].sum()

labels = ventas_pais.index      # Etiquetas (nombres de países)

# Valores de transacciones totales y rechazadas
transacciones_totales = ventas_pais.values
transacciones_rechazadas = rechazos_pais.values

bar_width = 0.32         # Ancho de las barras
x = np.arange(len(labels))  # Posiciones de los países en el eje X

# Se crea la gráfica
plt.figure(figsize=(18, 9))

# Barras de transacciones totales
plt.bar(x - bar_width/2, transacciones_totales, width=bar_width, label='Transacciones totales', color='blue')
# Barras de transacciones rechazadas
plt.bar(x + bar_width/2, transacciones_rechazadas, width=bar_width, label='Transacciones rechazadas', color='orange')

plt.xlabel('País')
plt.ylabel('Total de transacciones')
plt.title('E 1.6 Transacciones totales y rechazadas por país', fontsize=34)
plt.axhline(y=10, color='red', linestyle='-.')  # Línea horizontal de referencia con valor de 10
plt.xticks(x, labels, rotation=40)      # Etiquetas del eje X con nombres de los países
plt.legend()               # Leyenda

# Se muestra el gráfico
plt.show()
```



ii) Ahora:

```
# E 1.6:

colores = np.where(df_products['price'] < 50, 'blue', np.where(df_products['price'] < 100, 'green', np.where(df_products['price'] < 200, 'orange', 'red')) )

plt.figure(figsize=(20, 6))
scatter = plt.scatter(df_products['id'], df_products['weight'], color=colores, alpha=0.6, s=df_products['price'])

plt.xlabel('Id del producto', fontsize=12)
plt.ylabel('Peso del producto', fontsize=12)
plt.title('Precio y peso para cada producto (Colores según rango de precios)', fontsize=16)
plt.xticks(rotation=70, fontsize=9)

# Se crea una leyenda manual para explicar los colores
import matplotlib.patches as mpatches
leyenda_azul = mpatches.Patch(color='blue', label='Precio < 50')
leyenda_verde = mpatches.Patch(color='green', label='50 ≤ Precio < 100')
leyenda_naranja = mpatches.Patch(color='orange', label='100 ≤ Precio < 200')
leyenda_rojo = mpatches.Patch(color='red', label='Precio ≥ 200')

plt.legend(handles=[leyenda_azul, leyenda_verde, leyenda_naranja, leyenda_rojo], title='Rango de Precios', fontsize=10)

# Mostrar el gráfico
plt.show()
```

✓ 0.6s



Resumen:

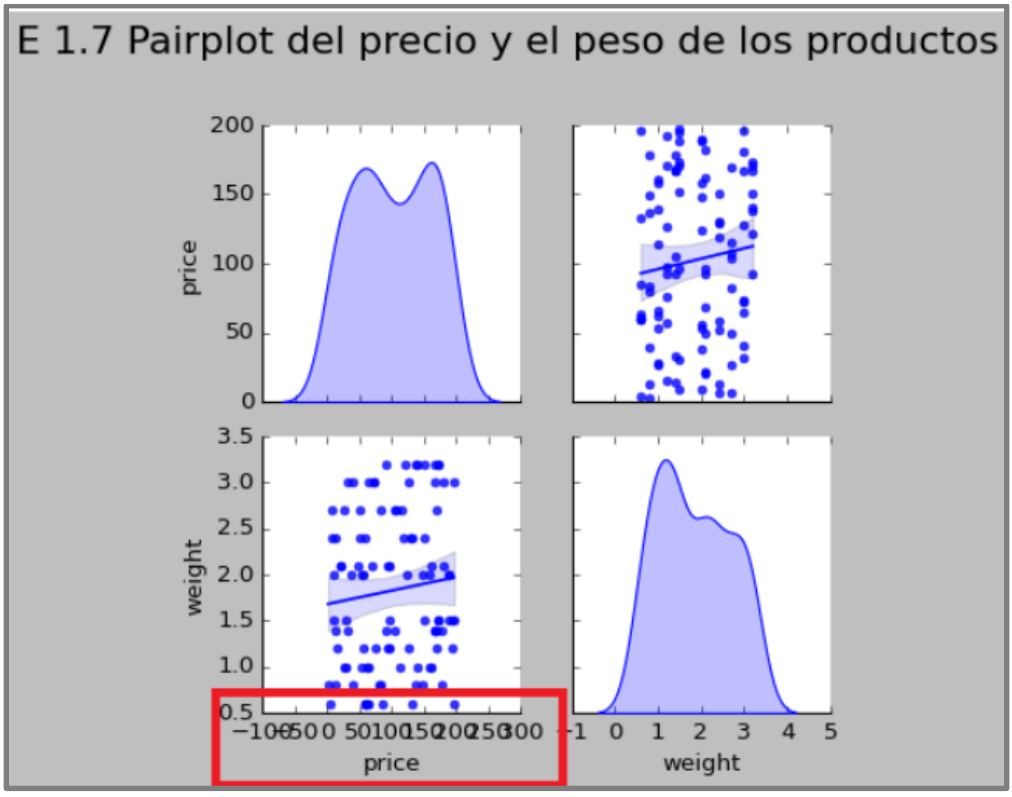
- Se ha substituído la gráfica de barras por una gráfica de dispersión, considerando 3 variables: precio, peso e id de los productos.
- Se ha actualizado la interpretación.

Ejercicio 1.7

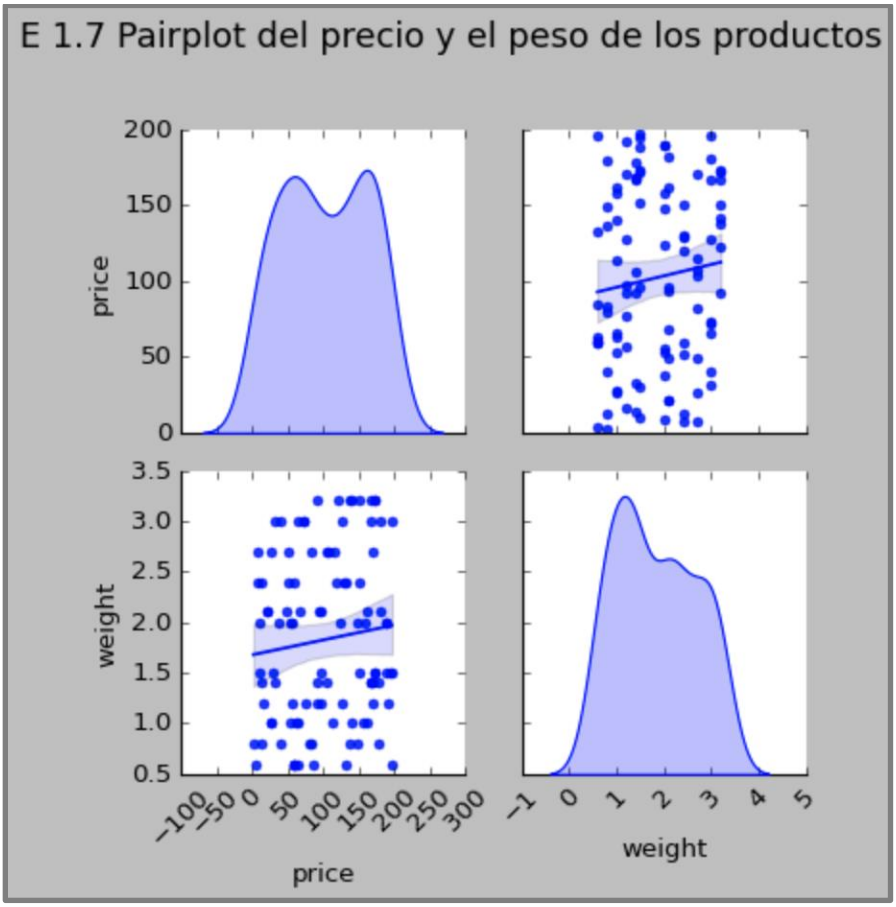
Pairplot.

Modificaciones:

i) Antes:



ii) Ahora:



Resumen:

- Reducción de los comentarios dentro del script.
- Se ha corregido el sobrelapado de las etiquetas del eje x en la gráfica que muestra precio y peso.
- Se ha actualiza la interpretación.