

```
In [25]: import pandas as pd
df = pd.read_csv('transfermarkt_value_history.csv')
df.head()
```

```
Out[25]:
```

	player_id	valuation_date	value_eur	value_display	club	age_at_date	year	month
0	108390	01/06/2016	35000000	€35.00m	Chelsea FC	24	2016	5
1	108390	01/07/2015	35000000	€35.00m	Chelsea FC	23	2015	6
2	108390	01/08/2012	15000000	€15.00m	Atlético de Madrid	20	2012	7
3	108390	01/08/2016	35000000	€35.00m	Chelsea FC	24	2016	7
4	108390	02/01/2018	50000000	€50.00m	Chelsea FC	25	2018	1

```
In [26]: len(df)
```

```
Out[26]: 9764
```

```
In [27]: # Check for missing values
df.isnull().sum()
```

```
Out[27]: player_id      0
valuation_date    0
value_eur         0
value_display     0
club              0
age_at_date       0
year              0
month             0
dtype: int64
```

```
In [28]: # لمعرفة القيم الفريدة مثلا كم نادي موجود في البيانات
df['club'].nunique()
```

```
Out[28]: 555
```

```
In [29]: import numpy as np
# فارغاً عمداً لنطبق كود الحذف player_id سنقوم بجعل 40% من عمود
df.loc[:len(df)//2.5, 'player_id'] = np.nan

# تأكد أن العمود أصبح فيه قيم مفقودة الآن
print(df.isnull().sum())
```

```

player_id      3906
valuation_date    0
value_eur        0
value_display    0
club            0
age_at_date      0
year            0
month           0
dtype: int64

```

In [30]: `df_cleaned.head()`

Out[30]:

	valuation_date	value_eur	value_display	club	age_at_date	year	month
0	01/06/2016	35000000	€35.00m	Chelsea FC	24	2016	5
1	01/07/2015	35000000	€35.00m	Chelsea FC	23	2015	6
2	01/08/2012	15000000	€15.00m	Atlético de Madrid	20	2012	7
3	01/08/2016	35000000	€35.00m	Chelsea FC	24	2016	7
4	02/01/2018	50000000	€50.00m	Chelsea FC	25	2018	1

In [31]:

```

# 1. حساب نسبة القيم المفقودة لكل عمود
missing_percentage = df.isnull().mean() * 100

# 2. تحديد الحد المسموح به (%30)
threshold = 30

# 3. player_id سيتعرف الكود على اختيار الأعمدة التي تجاوزت النسبة
columns_to_drop = missing_percentage[missing_percentage > threshold].index

# 4. تنفيذ عملية الحذف
df_cleaned = df.drop(columns=columns_to_drop)

# 5. طباعة النتائج لتأكيد الحذف أمام المعلمة
print(f"Columns dropped: {list(columns_to_drop)}")
print(f"Remaining columns: {df_cleaned.columns.tolist()}")

```

Columns dropped: ['player_id']

Remaining columns: ['valuation_date', 'value_eur', 'value_display', 'club', 'age_at_date', 'year', 'month']

In [32]:

```

# حساب الربع الاول و الربع و الثالث باستخدام البيانات المنظفة
Q1 = df_cleaned['value_eur'].quantile(0.25)
Q3 = df_cleaned['value_eur'].quantile(0.75)

# حساب المدى الربيعي (IQR)
IQR = Q3 - Q1

# تحديد الحدود
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

```

```
print(f"الحد الأدنى : {lower_bound}")
print(f"الحد الأعلى : {upper_bound}")
```

الحد الأدنى : -37500000.0

الحد الأعلى : 70500000.0

```
In [33]: # (لاني حذفت كولم df بدلاً من df_cleaned استخدام)
outliners_only = df_cleaned[(df_cleaned['value_eur'] < lower_bound) | (df_cleaned['value_eur'] > upper_bound)]

# عرض الجدول
outliners_only
```

```
Out[33]:
```

	valuation_date	value_eur	value_display	club	age_at_date	year	month
7	05/01/2021	75000000	€75.00m	Real Madrid	28	2021	1
13	08/10/2020	75000000	€75.00m	Real Madrid	28	2020	10
50	05/01/2021	90000000	€90.00m	Atlético de Madrid	27	2021	1
56	08/04/2020	80000000	€80.00m	Atlético de Madrid	27	2020	4
57	08/10/2020	90000000	€90.00m	Atlético de Madrid	27	2020	10
...	...	...	...	...	...	...	...
9435	21/03/2024	75000000	€75.00m	FC Barcelona	16	2024	3
9438	27/12/2024	180000000	€180.00m	FC Barcelona	17	2024	12
9517	09/06/2025	80000000	€80.00m	FC Barcelona	18	2025	6
9519	12/12/2025	80000000	€80.00m	FC Barcelona	18	2025	12
9637	09/12/2025	80000000	€80.00m	Chelsea FC	18	2025	12

489 rows × 7 columns

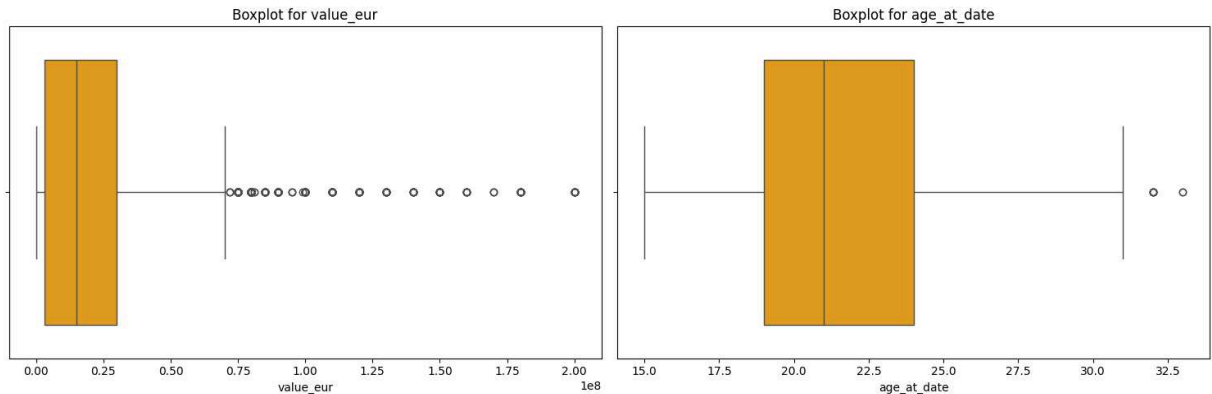
```
In [34]: import matplotlib.pyplot as plt
import seaborn as sns

# 1. df_cleaned (تأكد من استخدام) إعداد شكل الرسم
plt.figure(figsize=(15, 5))

# 2. تحديد الأعمدة المهمة التي نريد رؤية القيم الشاذة فيها
cols_to_plot = ['value_eur', 'age_at_date']

# 3. لرسم كل عمود في مكان مخصص (Loop) دوران
for i, col in enumerate(cols_to_plot):
    plt.subplot(1, 2, i+1) # إنشاء أماكن بجانب بعضها
    sns.boxplot(x=df_cleaned[col], color='orange') # رسم الصندوق باللون البرتقالي
```

```
plt.title(f"Boxplot for {col}") # وضع عنوان لكل رسمه
# 4. تحسين المسافات
plt.tight_layout()
plt.show()
```



```
In [35]: import numpy as np

# 1. أخذ نسخة من البيانات للعمل عليها
data = df_cleaned.copy()

# 2. تعريف دالة لاستبدال القيم الشاذة بـ NaN
def replace_outliers_with_nan(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # NaN استبدال أي قيمة خارج الحدود بـ
    return column.where((column >= lower_bound) & (column <= upper_bound), np.nan)

# 3. تطبيق الدالة على الأعمدة الرقمية فقط (العمر والسعر)
cols_to_fix = ['value_eur', 'age_at_date']

for col in cols_to_fix:
    data[col] = replace_outliers_with_nan(data[col])

# 4. الآن (القيم الشاذة المكتشفة) NaN حساب عدد القيم التي أصبحت
null_counts = data.isnull().sum()
print("في كل عمود NaN عدد القيم الشاذة التي تم تحويلها إلى:")
print(null_counts[cols_to_fix])
```

```
: في كل عمود NaN عدد القيم الشاذة التي تم تحويلها إلى
value_eur      489
age_at_date      8
dtype: int64
```

```
In [36]: # 1. حساب الوسيط لكل عمود (الوسيط هو الرقم الذي يقع في المنتصف ولا يتأثر بالقيم الشاذة)
median_value = data['value_eur'].median()
median_age = data['age_at_date'].median()

# 2. تعبئة الفراغات (التي كانت قيمًا شاذة) بالوسيط
data['value_eur'] = data['value_eur'].fillna(median_value)
```

```
data['age_at_date'] = data['age_at_date'].fillna(median_age)
```

```
# 3. التأكد من أن جميع الفراغات تم تعبئتها بنجاح
print("التحقق من القيم المفقودة بعد التعبئة بالوسيط:")
print(data[['value_eur', 'age_at_date']].isnull().sum())
```

التحقق من القيم المفقودة بعد التعبئة بالوسيط:

```
value_eur      0
age_at_date    0
dtype: int64
```

```
In [37]: # 1. تعبئة الفراغات التي نتجت عن القيم الشاذة باستخدام الوسيط (Median)
# اخترنا الوسيط لأنه لا يتأثر بالقيم المتطرفة
data['value_eur'] = data['value_eur'].fillna(data['value_eur'].median())
data['age_at_date'] = data['age_at_date'].fillna(data['age_at_date'].median())

# 2. الآن نحسب عدد القيم المفقودة مرة أخرى
print("عدد القيم المفقودة بعد المعالجة النهائية:")
print(data[['value_eur', 'age_at_date']].isnull().sum())
```

عدد القيم المفقودة بعد المعالجة النهائية:

```
value_eur      0
age_at_date    0
dtype: int64
```

```
In [38]: # 1. أخذ نسخة من البيانات
cleaned_data = df_cleaned.copy()

# 2. حساب الحدود لعمود السعر
Q1_v = cleaned_data['value_eur'].quantile(0.25)
Q3_v = cleaned_data['value_eur'].quantile(0.75)
IQR_v = Q3_v - Q1_v
lower_v = Q1_v - 1.5 * IQR_v
upper_v = Q3_v + 1.5 * IQR_v

# 3. حساب الحدود لعمود العمر
Q1_a = cleaned_data['age_at_date'].quantile(0.25)
Q3_a = cleaned_data['age_at_date'].quantile(0.75)
IQR_a = Q3_a - Q1_a
lower_a = Q1_a - 1.5 * IQR_a
upper_a = Q3_a + 1.5 * IQR_a

# 4. حذف الصفوف التي تخرج عن هذه الحدود في العمودين
data_no_outliers = cleaned_data[
    (cleaned_data['value_eur'] >= lower_v) & (cleaned_data['value_eur'] <= upper_v)
    & (cleaned_data['age_at_date'] >= lower_a) & (cleaned_data['age_at_date'] <= upper_a)
]

# 5. طباعة النتائج لرؤية الفرق في عدد الصفوف
print(f"عدد البيانات الأصلي: {len(cleaned_data)}")
print(f"عدد البيانات بعد حذف القيم الشاذة: {len(data_no_outliers)}")
print("\nعدد القيم المفقودة الآن:")
print(data_no_outliers[['value_eur', 'age_at_date']].isnull().sum())
```

عدد البيانات الأصلي: 9764
عدد البيانات بعد حذف القيم الشاذة: 9267

عدد القيم المفقودة الآن:

value_eur 0
age_at_date 0
dtype: int64

```
In [39]: #Basic Vertical Bar Chart
import matplotlib.pyplot as plt

# 1. تجهيز البيانات: حساب متوسط قيمة اللاعبين لكل نادي واختيار أعلى 5
top_clubs = df.groupby('club')['value_eur'].mean().sort_values(ascending=False).head(5)

# 2. رسم Bar Chart (نفس الطريقة في صورتك)
plt.figure(figsize=(10, 6))
plt.bar(top_clubs.index, top_clubs.values, color='lightcoral') # نفس اللون في مثالك

# 3. ليكن الرسم مفهوم (Labels) إضافة التسميات
plt.title('متوسط أسعار اللاعبين في أعلى 5 أندية')
plt.xlabel('النادي')
plt.ylabel('متوسط السعر (بالأورو)')

# 4. عرض الرسم
plt.show()
```



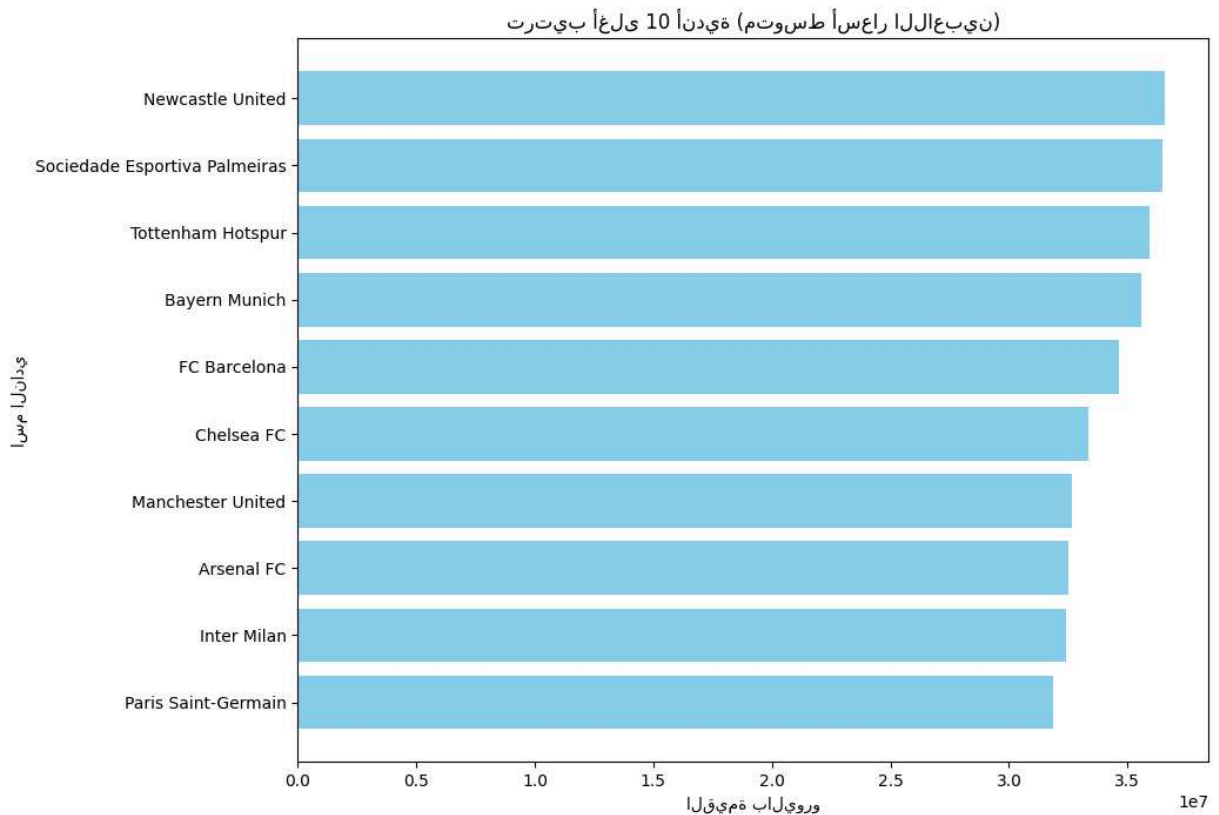
```
In [40]: #. Horizontal Bar Chart
import matplotlib.pyplot as plt

# 1. تجهيز البيانات (أعلى 10 أندية)
top_10_clubs = data.groupby('club')['value_eur'].mean().sort_values(ascending=True)
```

```
# 2. رسم الأعمدة الأفقية باستخدام barh
plt.figure(figsize=(10, 8))
plt.barh(top_10_clubs.index, top_10_clubs.values, color='skyblue')

# 3. إضافة العناوين
plt.title('ترتيب أعلى 10 أندية (متوسط أسعار اللاعبين)')
plt.xlabel('القيمة باليورو')
plt.ylabel('اسم النادي')

# 4. عرض الرسم
plt.show()
```



```
In [41]: # Grouped Bar Chart (Comparing Two Categories)
import numpy as np
import matplotlib.pyplot as plt

# 1. تجهيز البيانات: حساب متوسط العمر لأشهر 5 أندية
# هنا لأنها النسخة المنظمة من القيم الشاذة 'data' استخدمنا
top_clubs = data['club'].value_counts().head(5).index
avg_ages = data[data['club'].isin(top_clubs)].groupby('club')['age_at_date'].mean()

# قيمة مرجعية للمقارنة (عمر 18 سنة) لـ 5 أندية
min_age_ref = [18] * 5

# 2. إعداد مواقع الأعمدة
x = np.arange(len(top_clubs))
width = 0.35 # عرض العمود لضمان عدم التداخل

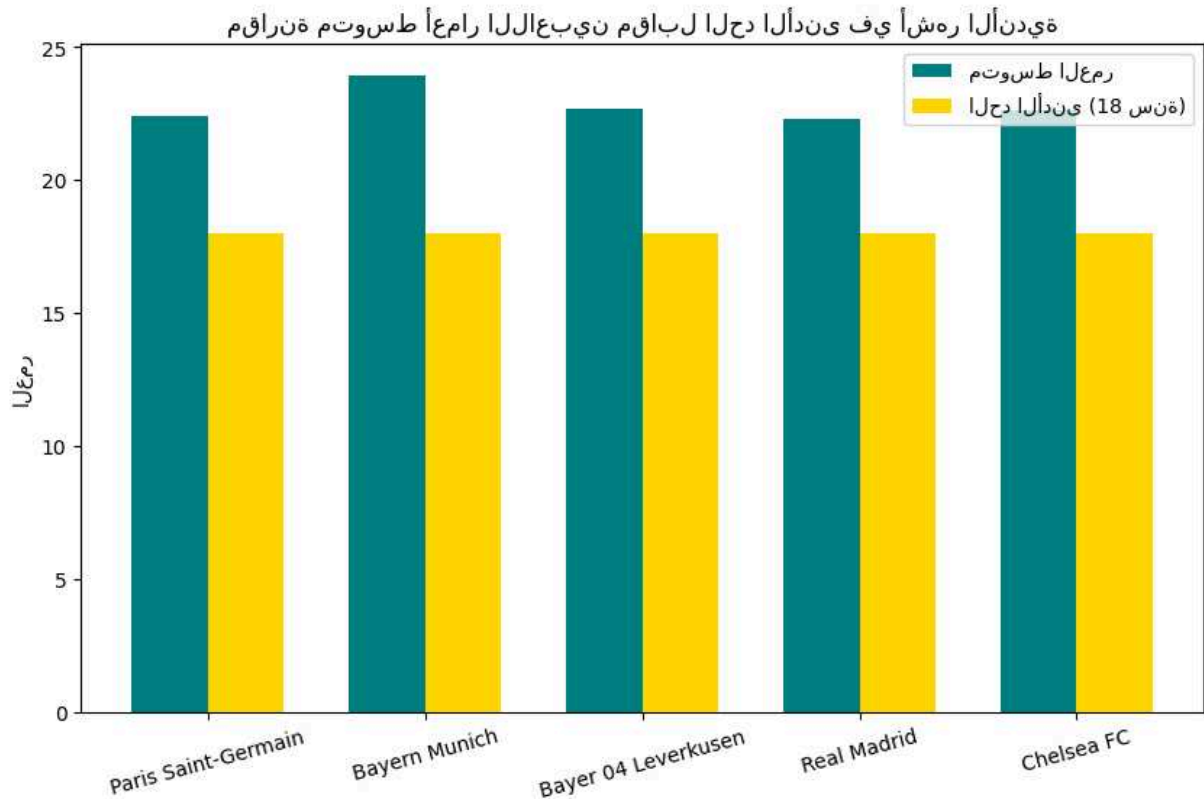
# 3. رسم الأعمدة المتجمعة
plt.figure(figsize=(10, 6))
```

```
# رسم عمود متوسط العمر
plt.bar(x - width/2, avg_ages, width, label='متوسط العمر', color='teal')

# رسم عمود الحد الأدنى المرجعي
plt.bar(x + width/2, min_age_ref, width, label='الحد الأدنى (18 سنة)', color='gold')

# 4. تحسين شكل الرسم
plt.xticks(x, top_clubs, rotation=15) # تدوير الأسماء لتظهر بوضوح
plt.title('مقارنة متوسط أعمار اللاعبين مقابل الحد الأدنى في أشهر الأندية')
plt.ylabel('العمر')
plt.legend()

plt.show()
```



```
In [42]: #Stacked Bar Chart
import matplotlib.pyplot as plt

# 1. تجهيز البيانات (أشهر 5 أندية)
top_clubs = df['club'].value_counts().head(5).index

# حساب عدد اللاعبين الشباب والكبار في كل نادي
young_players = []
senior_players = []

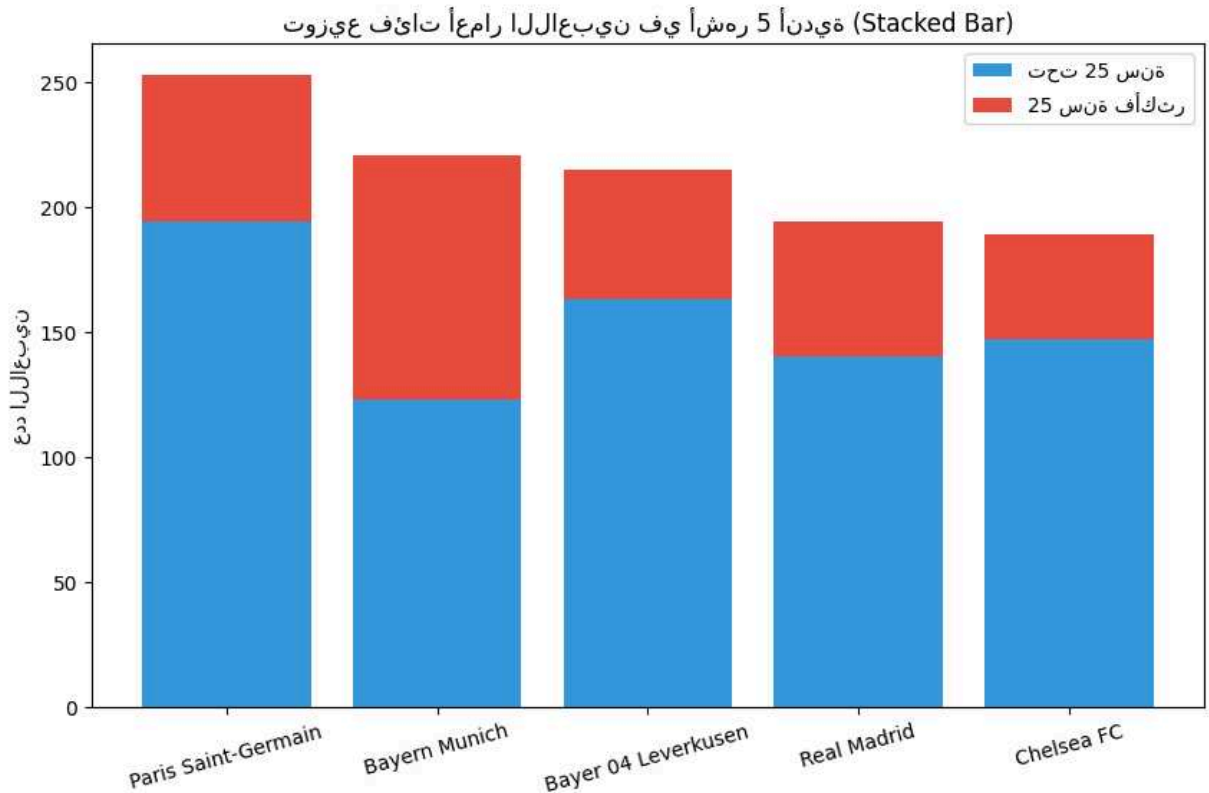
for club in top_clubs:
    club_data = df[df['club'] == club]
    young_count = len(club_data[club_data['age_at_date'] < 25])
    senior_count = len(club_data[club_data['age_at_date'] >= 25])
    young_players.append(young_count)
    senior_players.append(senior_count)
```

```
# 2. رسم ال Stacked Bar Chart
plt.figure(figsize=(10, 6))

# رسم الفئة الأولى (الأساس)
plt.bar(top_clubs, young_players, label='تحت 25 سنة', color='#3498db')

# رسم الفئة الثانية (استخدام) (فوق الأولى باستخدام)
plt.bar(top_clubs, senior_players, bottom=young_players, label='سنة فأكثر 25', col

# 3. اللمسات النهائية
plt.title('توزيع فئات أعمار اللاعبين في أشهر 5 أندية (Stacked Bar)')
plt.ylabel('عدد اللاعبين')
plt.legend()
plt.xticks(rotation=15)
plt.show()
```



```
In [43]: print(df.columns)
```

```
Index(['player_id', 'valuation_date', 'value_eur', 'value_display', 'club',
      'age_at_date', 'year', 'month'],
      dtype='object')
```

```
In [44]: #Pie Chart 1. Basic Pie Chart
import matplotlib.pyplot as plt
```

```
# تجهيز البيانات: حساب عدد اللاعبين في كل نادي واختيار أعلى 5
club_counts = df['club'].value_counts()
top_5_clubs = club_counts.head(5)
others_clubs = club_counts.iloc[5:].sum()

# دمجهم في قائمة واحدة للرسم
labels = list(top_5_clubs.index) + ['أندية أخرى']
```

```

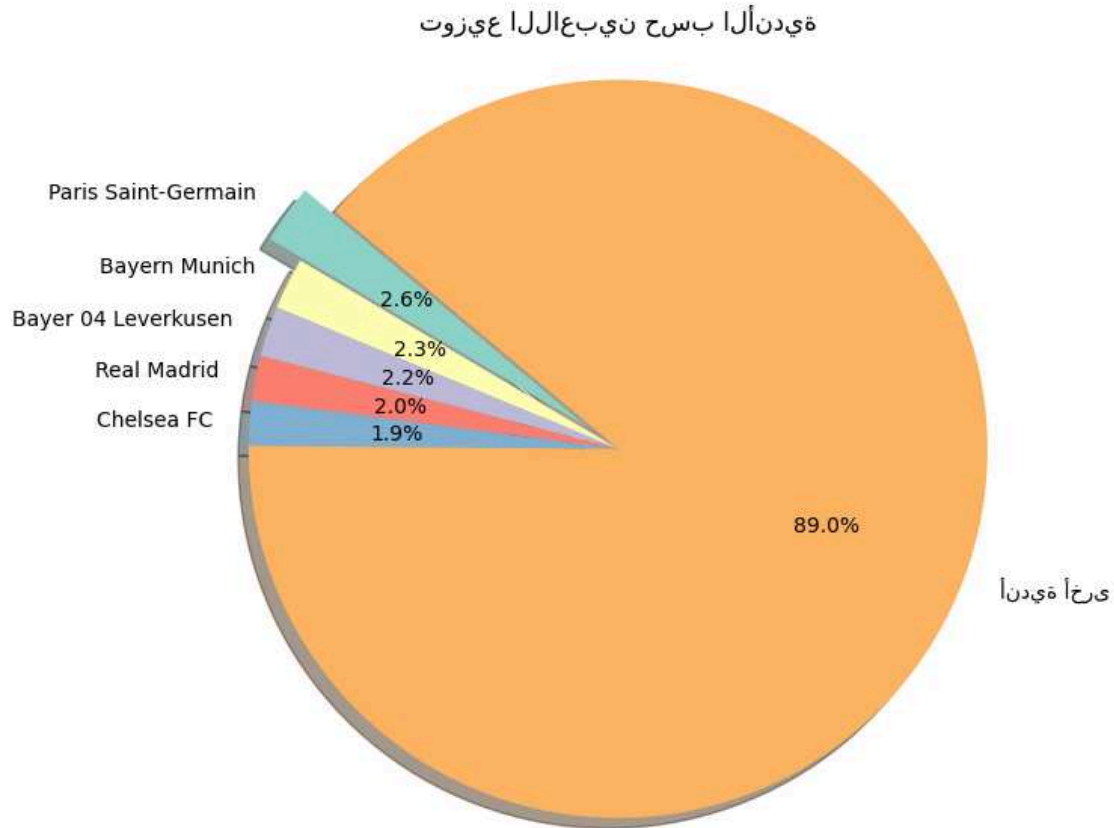
sizes = list(top_5_clubs.values) + [others_clubs]

# 2. رسم ال Pie Chart
plt.figure(figsize=(10, 7))
# تجعل أول شريحة تبرز قليلاً لجمالية الرسم
explode = (0.1, 0, 0, 0, 0, 0)

plt.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=140, colors=plt.cm.Set3.colors)

plt.title('توزيع اللاعبين حسب الأندية')
plt.axis('equal') # لجعل الدائرة متساوية الأبعاد
plt.show()

```



```

In [45]: #Pie Chart with Percentages
import matplotlib.pyplot as plt

# 1. تجهيز البيانات: حساب تكرار الأندية واختيار أعلى 5
club_data = df['club'].value_counts()
top_5 = club_data.head(5)
others = club_data.iloc[5:].sum()

# دمجهم في قائمة واحدة
labels = list(top_5.index) + ['Others']
sizes = list(top_5.values) + [others]

# 2. رسم الدائرة
plt.figure(figsize=(9, 9))

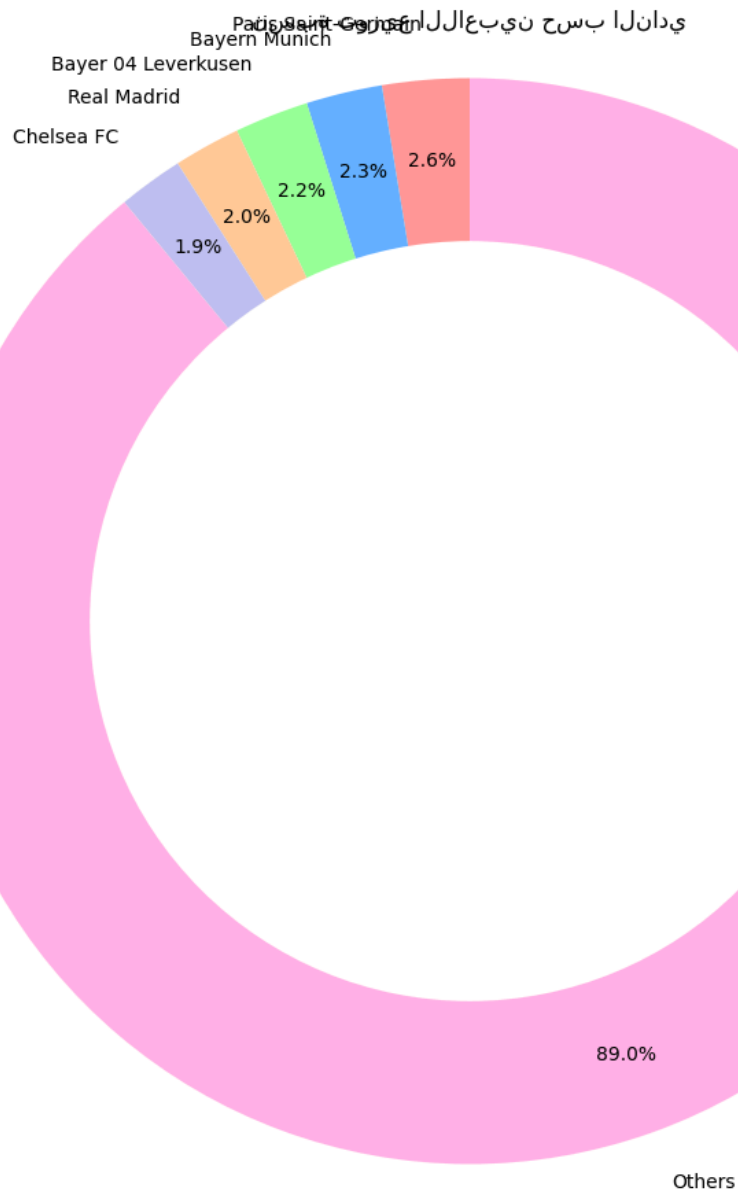
```

```
# colors: اختيار ألوان هادئة ومتناسقة
colors = ['#ff9999', '#66b3ff', '#99ff99', '#ffcc99', '#c2c2ff', '#ffb3e6']

# autopct: هذا هو الجزء المسؤول عن إظهار النسبة المئوية
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90, colors=colors, pctd

# اختياري لكن شكلها أجمل) Donut Chart إضافة دائرة بيضاء في الوسط لجعلها تبدو كأنها
centre_circle = plt.Circle((0,0), 0.70, fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

plt.title('نسبة توزيع اللاعبين حسب النادي')
plt.axis('equal')
plt.tight_layout()
plt.show()
```



```
In [46]: #. Exploded (Highlighted) Pie Chart
import matplotlib.pyplot as plt
```

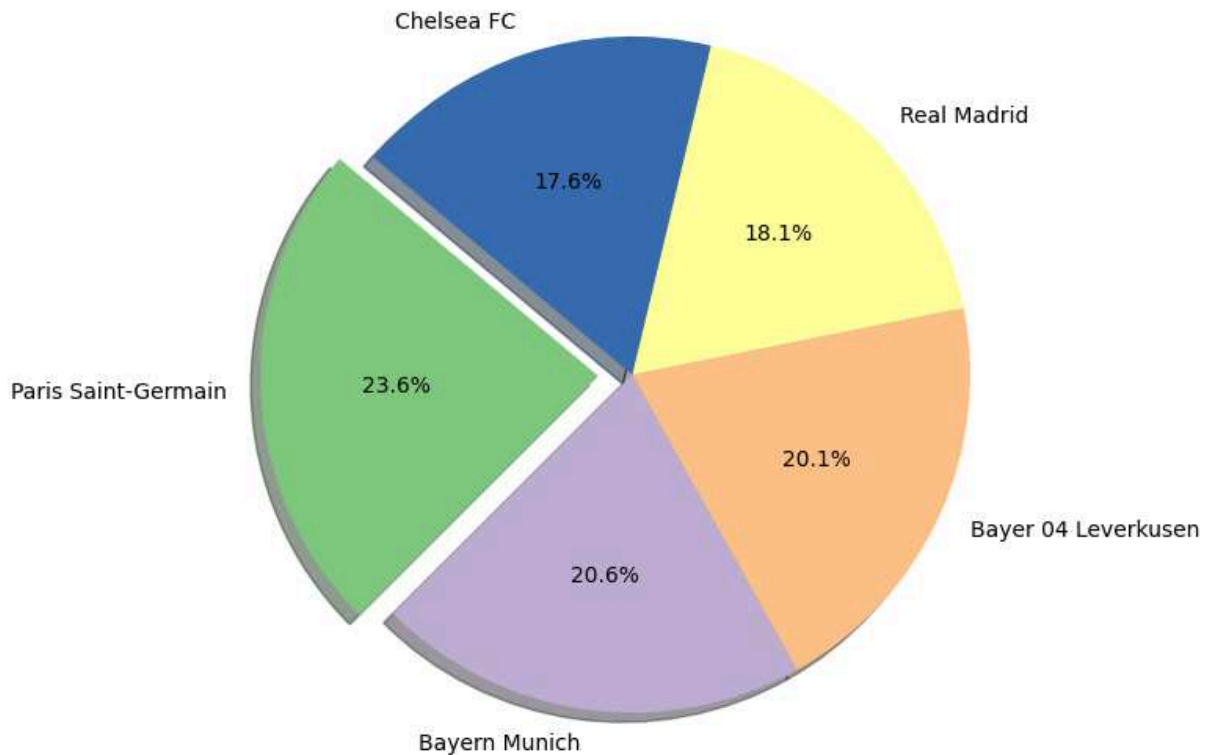
```
# 1. تجهيز البيانات (أشهر 5 أندية)
club_data = df['club'].value_counts().head(5)
labels = club_data.index
sizes = club_data.values

# 2. (Explode) "تحديد أي جزء سيتم فصله"
# القيمة 0.1 تعني فصل الجزء الأول قليلاً عن المركز، والباقي (0) يبقون في مكانهم
explode = (0.1, 0, 0, 0, 0)

# 3. رسم الدائرة
plt.figure(figsize=(10, 7))
plt.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=140, colors=plt.cm.Accent.colors)

# 4. إضافة عنوان
plt.title('إبراز النادي الأكثر تكراراً (Exploded Pie Chart)')
plt.show()
```

(Exploded Pie Chart) إبراز النادي الأكثر تكراراً



```
In [47]: #Ring Chart
import matplotlib.pyplot as plt

# 1. تجهيز البيانات (حساب تكرار السنوات)
year_counts = df['year'].value_counts()
labels = year_counts.index
sizes = year_counts.values

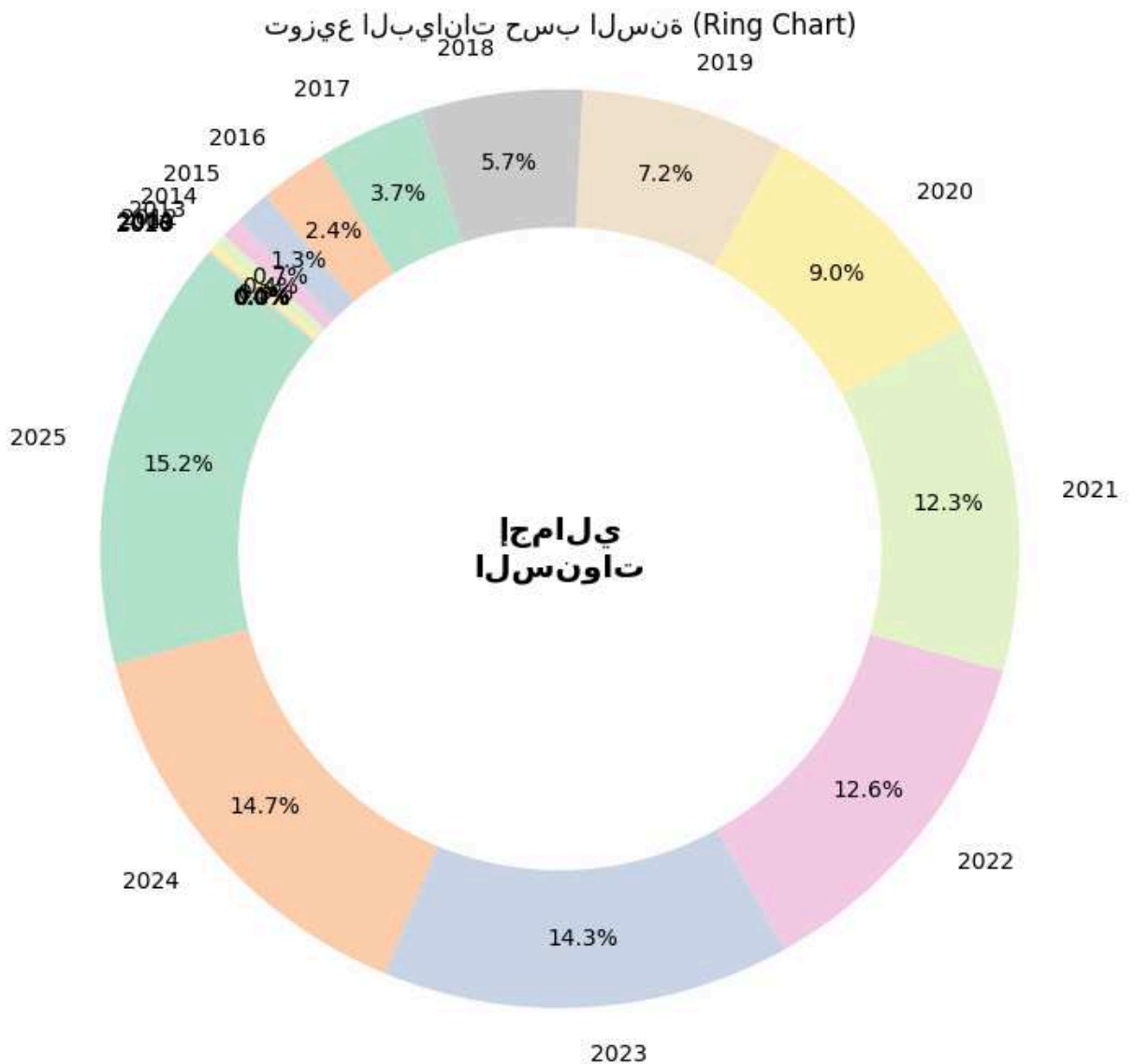
# 2. رسم الدائرة الأساسية
plt.figure(figsize=(8, 8))
```

```
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140, colors=plt.cm.Pastel3.colors)

# 3. تحويل الدائرة إلى حلقة (السفر في هذه الخطوة)
# رسم دائرة بيضاء في المركز
centre_circle = plt.Circle((0,0), 0.70, fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

# 4. إضافة نص في منتصف الحلقة
plt.text(0, 0, 'السنوات \n إجمالي', ha='center', va='center', fontsize=14, fontweight='bold')

plt.title('Ring Chart توزيع البيانات حسب السنة')
plt.axis('equal')
plt.show()
```



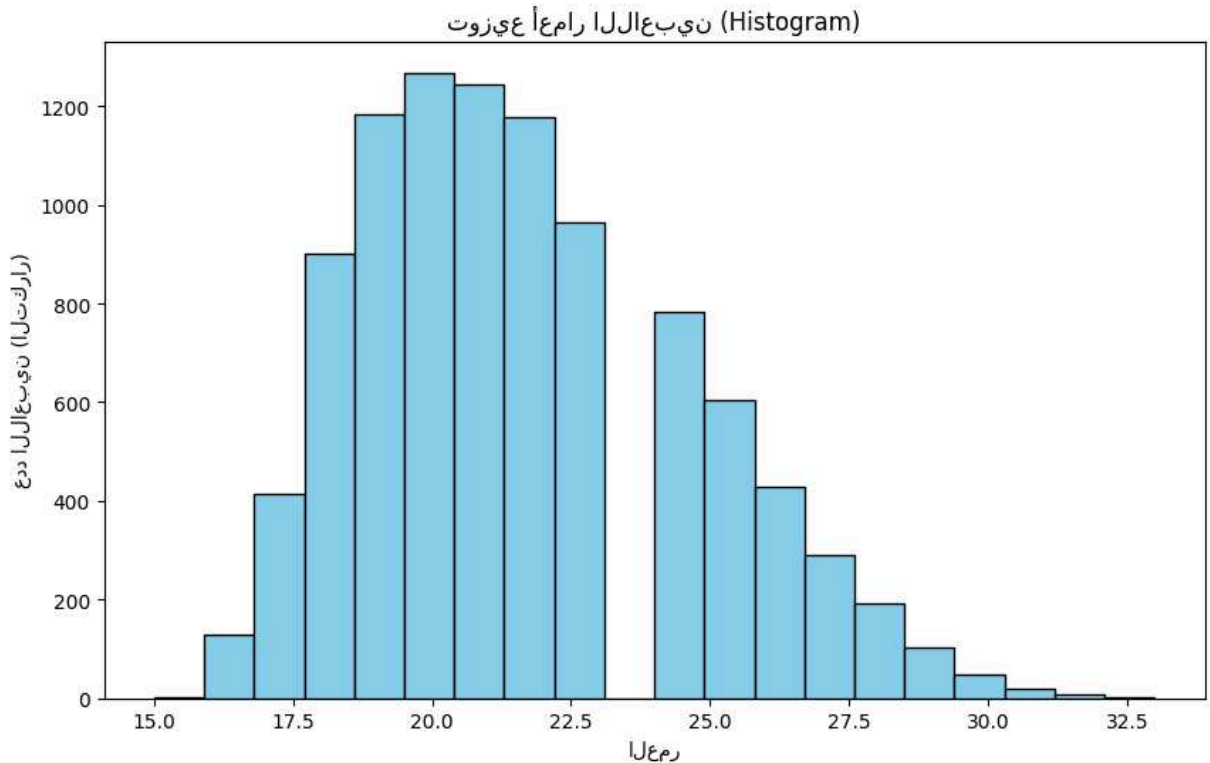
```
In [48]: #Histogram
import matplotlib.pyplot as plt

# 1. رسم ال Histogram
plt.figure(figsize=(10, 6))
# تعني تقسيم الأعمار إلى 20 مجموعة (أعمدة)
```

```
plt.hist(df['age_at_date'], bins=20, color='skyblue', edgecolor='black')

# 2. إضافة التسميات
plt.title('توزيع أعمار اللاعبين (Histogram)')
plt.xlabel('العمر')
plt.ylabel('عدد اللاعبين (التكرار)')

# 3. عرض الرسم
plt.show()
```

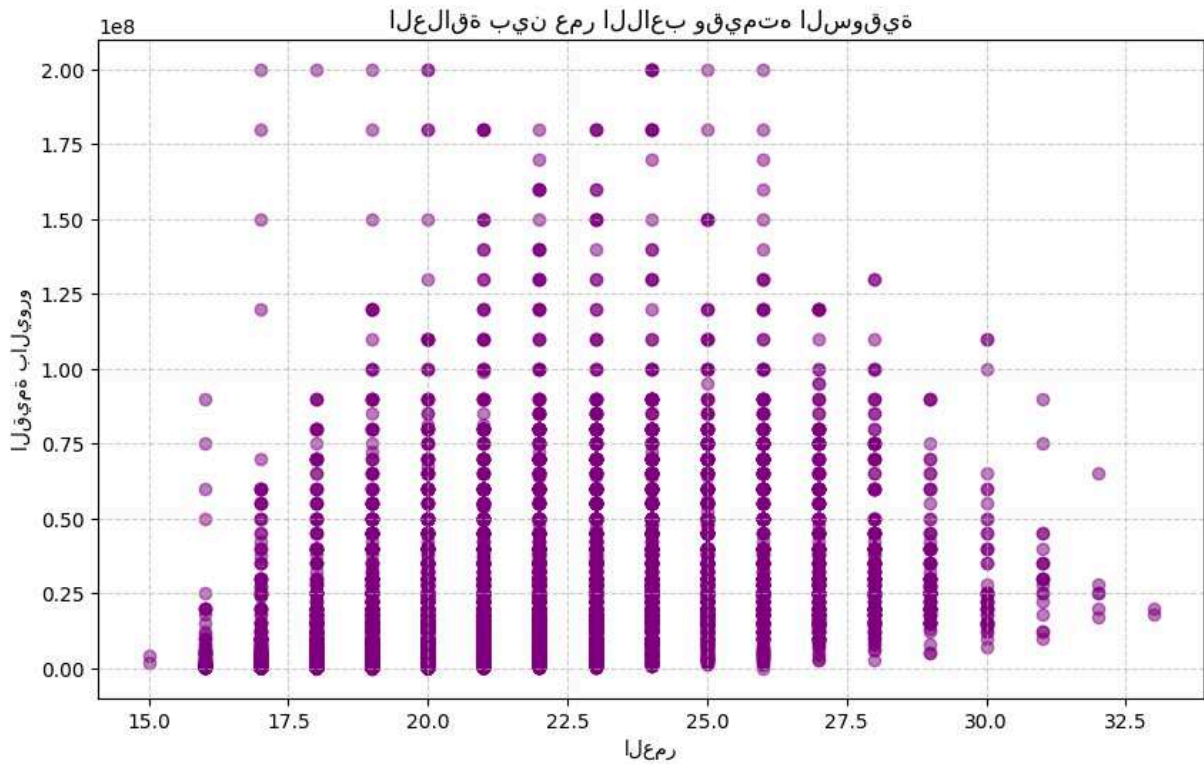


```
In [49]: #Boxplot
import matplotlib.pyplot as plt

# 1. رسم Scatter Plot
plt.figure(figsize=(10, 6))
plt.scatter(df['age_at_date'], df['value_eur'], alpha=0.5, color='purple')

# 2. إضافة التسميات
plt.title('العلاقة بين عمر اللاعب وقيمته السوقية')
plt.xlabel('العمر')
plt.ylabel('القيمة باليورو')

# 3. تحسين المظهر
plt.grid(True, linestyle='--', alpha=0.6) # إضافة شبكة خلفية لتسهيل القراءة
plt.show()
```



```
In [50]: #Scatter plot
import matplotlib.pyplot as plt

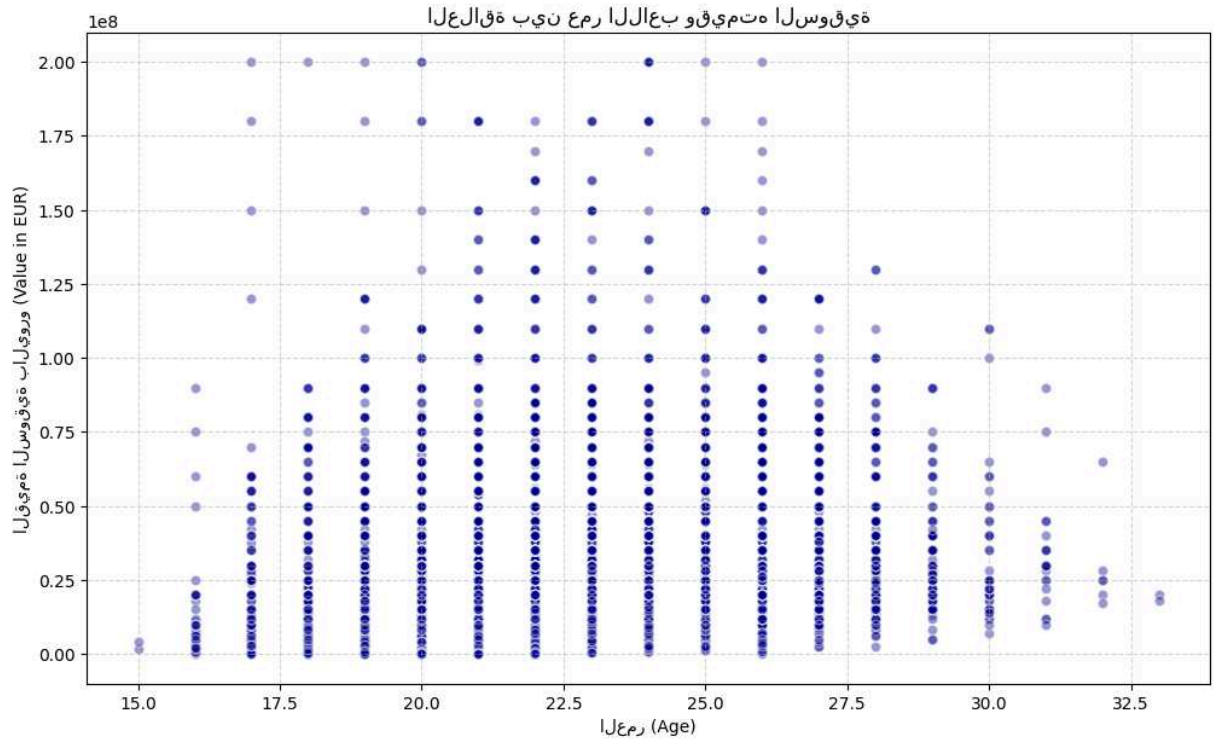
# 1. تحديد حجم الرسم
plt.figure(figsize=(12, 7))

# 2. رسم النقاط
# جعل النقاط شفافة قليلاً لتري التجمعات المزدحمة بوضوح
plt.scatter(df['age_at_date'], df['value_eur'], alpha=0.4, color='darkblue', edgeco

# 3. إضافة التسميات والعناوين
plt.title('العلاقة بين عمر اللاعب وقيمته السوقية')
plt.xlabel('العمر (Age)')
plt.ylabel('القيمة السوقية باليورو (Value in EUR)')

# 4. إضافة شبكة خلفية لتسهيل قراءة القيم
plt.grid(True, linestyle='--', alpha=0.5)

# 5. عرض الرسم
plt.show()
```



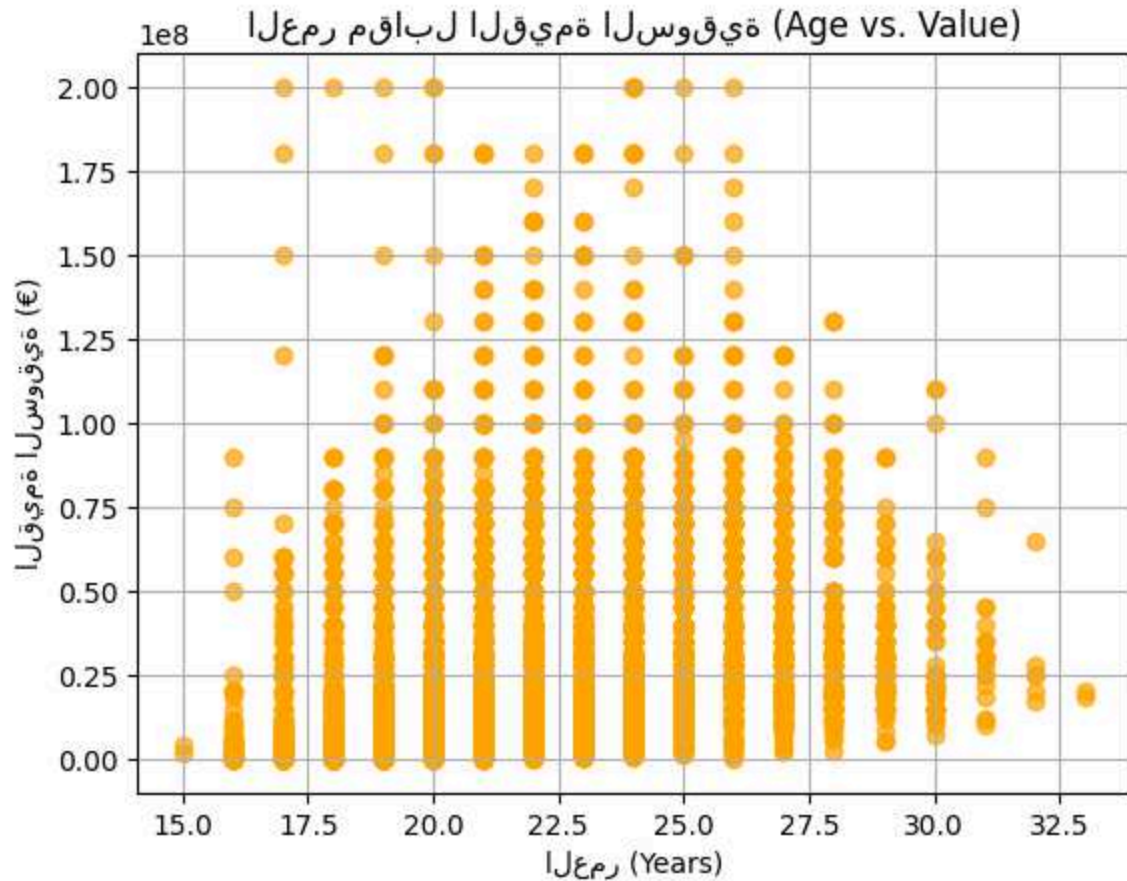
```
In [51]: import matplotlib.pyplot as plt

# 1. تحديد البيانات من ملفك
# سنستخدم العمر على المحور الأفقي والقيمة على المحور الرأسي
plt.scatter(df['age_at_date'], df['value_eur'], color='orange', alpha=0.7)

# 2. إضافة العناوين (بنفس أسلوب مثالك)
plt.title("العمر مقابل القيمة السوقية (Age vs. Value)")
plt.xlabel("العمر (Years)")
plt.ylabel("القيمة السوقية (€)")

# 3. إضافة الشبكة كما في المثال
plt.grid(True)

# 4. عرض الرسم
plt.show()
```



```
In [52]: #Line graph
import matplotlib.pyplot as plt

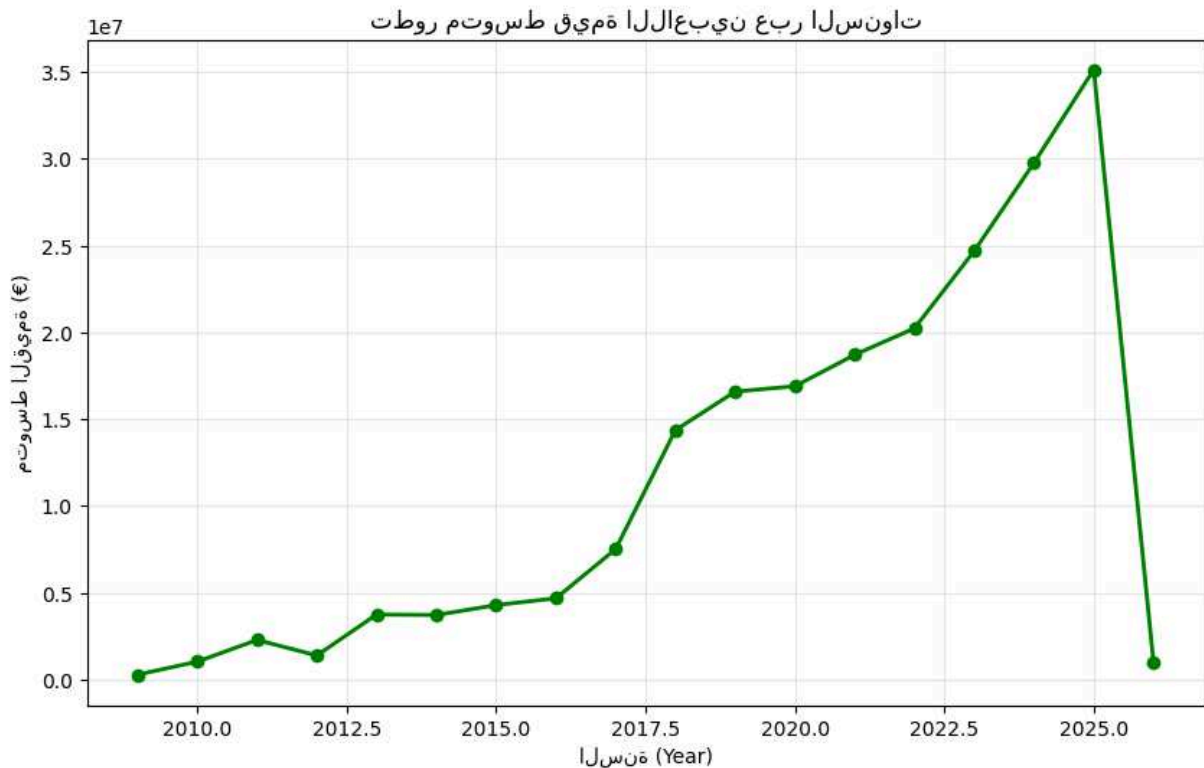
# 1. تجهيز البيانات: حساب متوسط السعر لكل سنة
# تجميع البيانات حسب السنة ثم نحسب المتوسط groupby نستخدم
yearly_avg = df.groupby('year')['value_eur'].mean()

# 2. Line Graph رسم ال
plt.figure(figsize=(10, 6))
plt.plot(yearly_avg.index, yearly_avg.values, color='green', marker='o', linestyle=

# 3. إضافة العناوين (بنفس أسلوب أمثلتك)
plt.title("تطور متوسط قيمة اللاعبين عبر السنوات")
plt.xlabel("السنة (Year)")
plt.ylabel("متوسط القيمة (€)")

# 4. كما في المثال السابق (Grid) إضافة الشبكة
plt.grid(True, alpha=0.3)

# 5. عرض الرسم
plt.show()
```



```
In [53]: import matplotlib.pyplot as plt

# 1. تحديد الناديين اللذين نريد مقارنتهما (تأكد من الأسماء كما تظهر في بياناتك)
club_a_name = 'Real Madrid'
club_b_name = 'FC Barcelona'

# 2. حساب متوسط القيمة السنوية لكل نادي
# تصفية البيانات لكل نادي ثم التجميع حسب السنة وحساب المتوسط
club_a_data = df[df['club'] == club_a_name].groupby('year')['value_eur'].mean()
club_b_data = df[df['club'] == club_b_name].groupby('year')['value_eur'].mean()

# 3. رسم الخطوط المتعددة (اتباعاً لنفس أسلوب مثالك)
plt.figure(figsize=(10, 6))

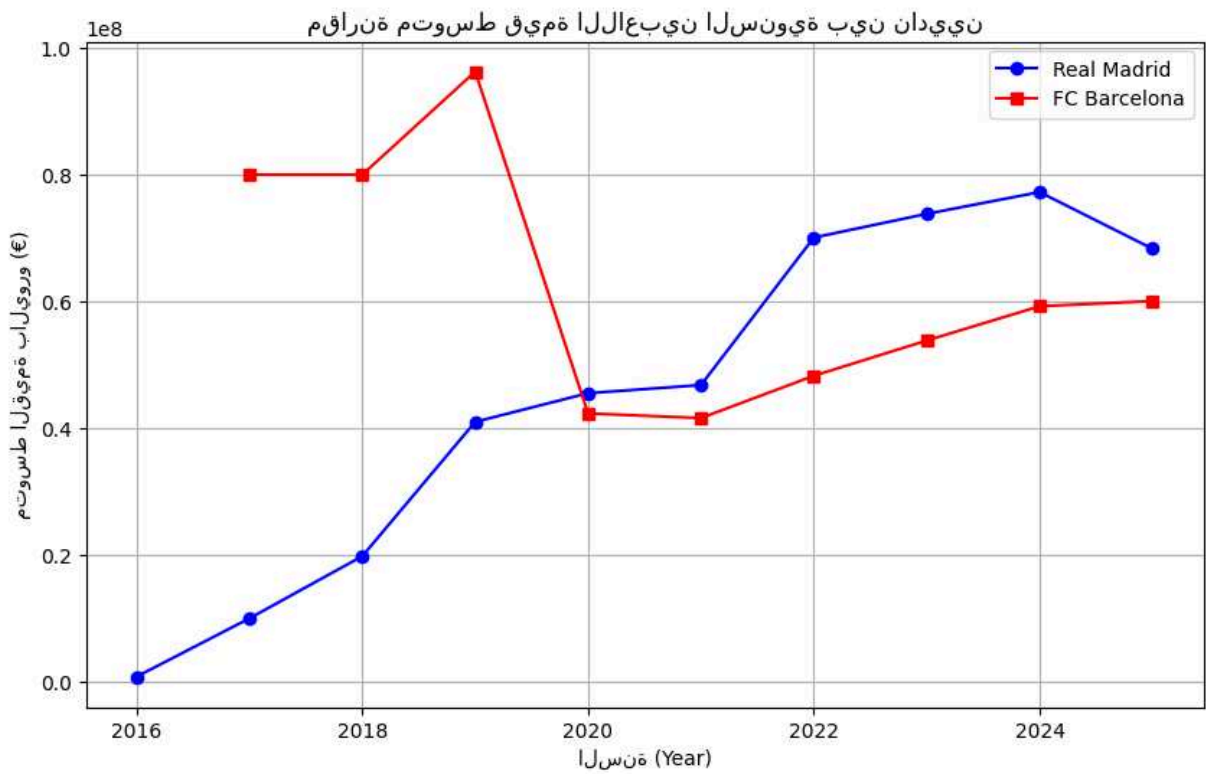
# الخط الأول للنادي A
plt.plot(club_a_data.index, club_a_data.values, marker='o', label=club_a_name, color='blue')

# الخط الثاني للنادي B
plt.plot(club_b_data.index, club_b_data.values, marker='s', label=club_b_name, color='red')

# 4. إضافة العناوين والأسماء (Labels)
plt.title('مقارنة متوسط قيمة اللاعبين السنوية بين ناديين')
plt.xlabel('السنة (Year)')
plt.ylabel('متوسط القيمة باليورو (€)')

# 5. إضافة وسيلة الإيضاح والشبكة
plt.legend() # لإظهار أسماء الأندية المرتبطة بكل لون
plt.grid(True)

# 6. عرض الرسم
plt.show()
```



```
In [54]: #Examples from datasets
df = pd.read_csv('transfermarkt_value_history.csv')
df.head()
```

Out[54]:

	player_id	valuation_date	value_eur	value_display	club	age_at_date	year	month
0	108390	01/06/2016	35000000	€35.00m	Chelsea FC	24	2016	5
1	108390	01/07/2015	35000000	€35.00m	Chelsea FC	23	2015	6
2	108390	01/08/2012	15000000	€15.00m	Atlético de Madrid	20	2012	7
3	108390	01/08/2016	35000000	€35.00m	Chelsea FC	24	2016	7
4	108390	02/01/2018	50000000	€50.00m	Chelsea FC	25	2018	1

```
In [55]: df.describe()
```

Out[55]:

	player_id	value_eur	age_at_date	year	month
count	9.764000e+03	9.764000e+03	9764.000000	9764.000000	9764.000000
mean	4.848166e+05	2.212540e+07	21.615629	2021.587567	7.629865
std	2.070340e+05	2.586825e+07	2.969034	2.767403	3.546509
min	1.083900e+05	0.000000e+00	15.000000	2009.000000	1.000000
25%	3.393140e+05	3.000000e+06	19.000000	2020.000000	5.000000
50%	4.591510e+05	1.500000e+07	21.000000	2022.000000	7.000000
75%	6.081720e+05	3.000000e+07	24.000000	2024.000000	12.000000
max	1.390649e+06	2.000000e+08	33.000000	2026.000000	12.000000

In [56]:

```

#Correlation heatmap
import seaborn as sns
import matplotlib.pyplot as plt

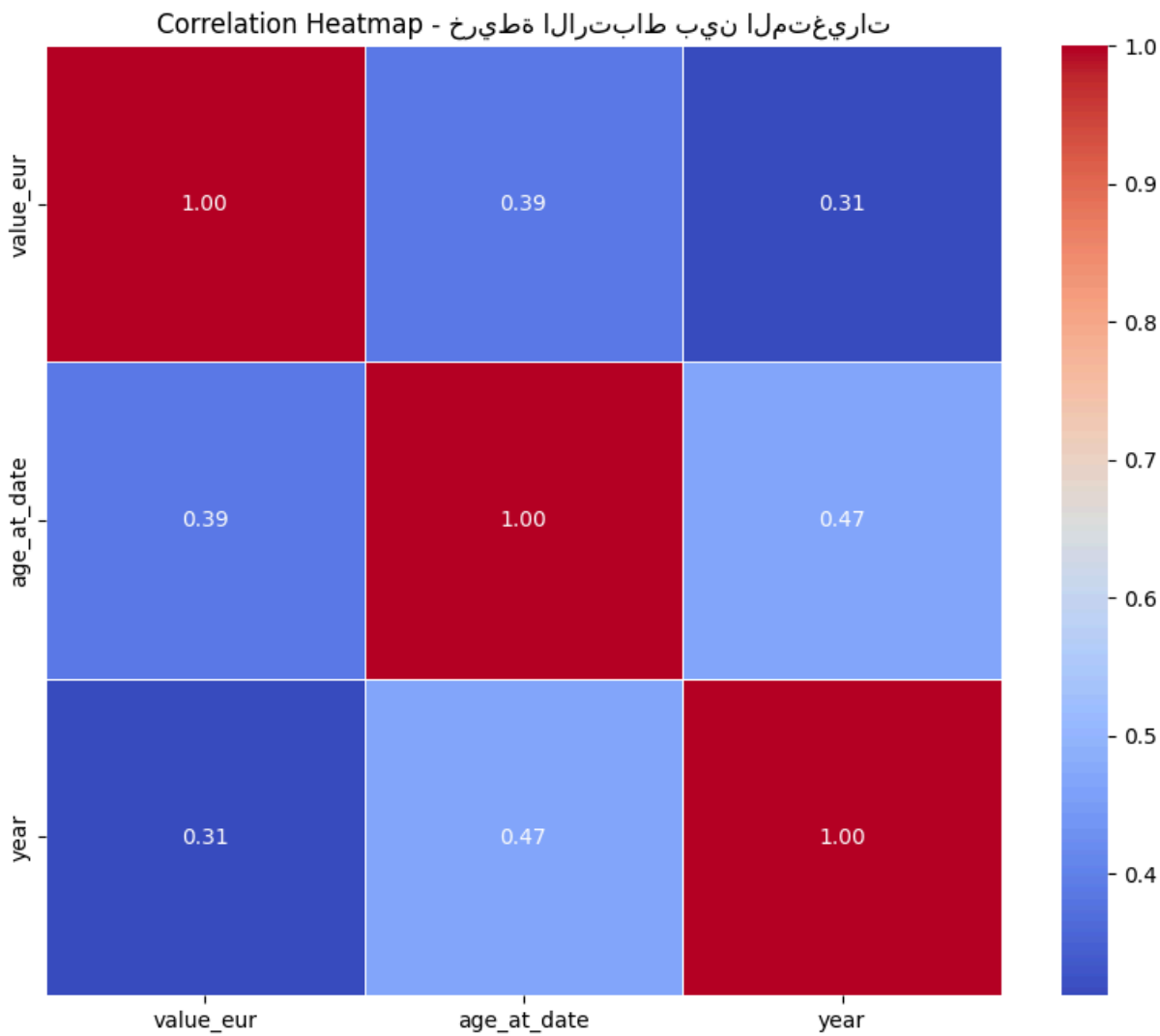
# 1. اختيار الأعمدة الرقمية فقط (لأن الارتباط لا يحسب إلا للأرقام)
# سنختار: القيمة، العمر، السنة
numeric_df = df[['value_eur', 'age_at_date', 'year']]

# 2. حساب مصفوفة الارتباط
corr_matrix = numeric_df.corr()

# 3. رسم الخريطة الحرارية باستخدام Seaborn
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)

# 4. إضافة العناوين
plt.title('Correlation Heatmap - بين المتغيرات')
plt.show()

```



```
In [ ]:
In [ ]:
In [ ]:
In [ ]:
```