Projet 9 : Analysez les ventes d'une librairie avec R ou Python

Sommaire

Étape 1 - Importation des librairies et chargement des fichiers CSV

1.1 - Importation des librairies

In [5]:
```
#Importation de la librairie Pandas
import pandas as pd
#Importation de Numpy
import numpy as np
#Importation de matplotlib
import matplotlib.pyplot as plt
```

1.2 - Chargement des fichiers CSV

In [7]:
```
# Importation du fichier customers.csv
customers = pd.read_csv('customers.csv', sep=';')

# Importation du fichier products.csv
products = pd.read_csv('products.csv', sep=';')

# Importation du fichier Transactions.csv
transactions = pd.read_csv('Transactions.csv', sep=';')
```

Étape 2 - Analyse exploratoire des fichiers

2.1 - Analyse exploratoire du fichier customers

In [10]:
```
#Afficher les dimensions du dataset
print("Le tableau comporte {} observation(s) ou article(s)".format(customers.shape[0]))
print("Le tableau comporte {} colonne(s)".format(customers.shape[1]))
```

```
Le tableau comporte 8621 observation(s) ou article(s)
Le tableau comporte 3 colonne(s)
```

In [11]:
```
#Vérification des doublons sur toutes les colonnes
customers_doublons = customers[customers.duplicated()]

# Afficher les doublons (si existants)
print("Nombre de doublons :", len(customers_doublons))
```

```
Nombre de doublons : 0
```

In [12]:
```
#Affichage du dataset
customers.head()
```

Out[12]:

	client_id	sex	birth
0	c_4410	f	1967
1	c_7839	f	1975
2	c_1699	f	1984
3	c_5961	f	1962
4	c_5320	m	1943

In [13]:
```
#Vérifiaction des valeurs manquantes
customers.isna().any()
```

Out[13]:
```
client_id    False
sex          False
birth        False
dtype: bool
```

Il n'y a pas de valeurs manquantes dans la table customers

In [15]:
```
#Affichage du format des donnée du dataset
customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8621 entries, 0 to 8620
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   client_id  8621 non-null   object
 1   sex        8621 non-null   object
 2   birth      8621 non-null   int64 
dtypes: int64(1), object(2)
memory usage: 202.2+ KB
```

2.2 - Analyse exploratoire du fichier products

In [17]:
```
#Affichage du dataset
products.head()
```

Out[17]:

	id_prod	price	categ
0	0_1421	19.99	0
1	0_1368	5.13	0
2	0_731	17.99	0
3	1_587	4.99	1
4	0_1507	3.99	0

In [18]:
```
#Afficher les dimensions du dataset
print("Le tableau comporte {} observation(s) ou article(s)".format(products.shape[0]))
print("Le tableau comporte {} colonne(s)".format(products.shape[1]))
```

```
Le tableau comporte 3286 observation(s) ou article(s)
Le tableau comporte 3 colonne(s)
```

In [19]:
```
#Vérification des doublons sur toutes les colonnes
products_doublons = products[products.duplicated()]

# Afficher les doublons (si existants)
print("Nombre de doublons :", len(products_doublons))
```

```
Nombre de doublons : 0
```

In [20]:
```
#Vérifiaction des valeurs manquantes
customers.isna().any()
```

Out[20]:
```
client_id    False
sex          False
birth        False
dtype: bool
```

Il n'y a pas de valeurs manquantes dans la table products

In [22]:
```
#Affichage du format des donnée du dataset
products.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3286 entries, 0 to 3285
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   id_prod  3286 non-null   object
 1   price    3286 non-null   float64
 2   categ    3286 non-null   int64
dtypes: float64(1), int64(1), object(1)
memory usage: 77.1+ KB
```

2.3 - Analyse exploratoire du fichier transactions

In [24]:
```
transactions.head()
```

Out[24]:

	id_prod	date	session_id	client_id
0	0_1259	2021-03-01 00:01:07.843138	s_1	c_329
1	0_1390	2021-03-01 00:02:26.047414	s_2	c_664
2	0_1352	2021-03-01 00:02:38.311413	s_3	c_580
3	0_1458	2021-03-01 00:04:54.559692	s_4	c_7912
4	0_1358	2021-03-01 00:05:18.801198	s_5	c_2033

In [25]:
```
#Afficher les dimensions du dataset
print("Le tableau comporte {} observation(s) ou article(s)".format(transactions.shape[0]))
print("Le tableau comporte {} colonne(s)".format(transactions.shape[1]))
```

```
Le tableau comporte 687534 observation(s) ou article(s)
Le tableau comporte 4 colonne(s)
```

In [26]:
```
#Vérification des doublons sur toutes les colonnes
transactions_doublons =  transactions[transactions.duplicated()]

# Afficher les doublons (si existants)
print("Nombre de doublons :", len(transactions_doublons))
```

```
Nombre de doublons : 0
```

In [27]:
```
#Vérifiaction des valeurs manquantes
transactions.isna().any()
```

Out[27]:
```
id_prod       False
date          False
session_id    False
client_id     False
dtype: bool
```

Il n'y a pas de valeurs manquantes dans la table transactions

In [29]:
```
#Affichage du format des donnée du dataset
transactions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 687534 entries, 0 to 687533
Data columns (total 4 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   id_prod     687534 non-null  object
 1   date        687534 non-null  object
 2   session_id  687534 non-null  object
 3   client_id   687534 non-null  object
dtypes: object(4)
memory usage: 21.0+ MB
```

In [30]:
```
#Affichage du dataset
transactions.head()
```

Out[30]:

	id_prod	date	session_id	client_id
0	0_1259	2021-03-01 00:01:07.843138	s_1	c_329
1	0_1390	2021-03-01 00:02:26.047414	s_2	c_664
2	0_1352	2021-03-01 00:02:38.311413	s_3	c_580
3	0_1458	2021-03-01 00:04:54.559692	s_4	c_7912
4	0_1358	2021-03-01 00:05:18.801198	s_5	c_2033

2.4 - Jointure de la table customers et transactions

In [32]:
```
#Jointure des tables customers et transactions
customers_transactions = pd.merge(customers, transactions, how='inner', on=['client_id'])
```

In [33]:
```
#Affichage du nombre de lignes du dataset
customers_transactions.count()
```

Out[33]:
```
client_id     687534
sex           687534
birth         687534
id_prod       687534
date          687534
session_id    687534
dtype: int64
```

In [34]:
```
#Affichage du dataset crée
customers_transactions.head()
```

Out[34]:

	client_id	sex	birth	id_prod	date	session_id
0	c_4410	f	1967	1_483	2021-03-13 21:35:55.949042	s_5913
1	c_4410	f	1967	0_1111	2021-03-22 01:27:49.480137	s_9707
2	c_4410	f	1967	1_385	2021-03-22 01:40:22.782925	s_9707
3	c_4410	f	1967	0_1455	2021-03-22 14:29:25.189266	s_9942
4	c_4410	f	1967	0_1420	2021-03-22 22:31:25.825764	s_10092

2.5 - Jointure de la table customers_transactions et products

In [36]:
```
#Jointure de la table customers_transactions et products
sales = pd.merge(customers_transactions, products, how='inner', on=['id_prod'])
```

In [37]:
```
#Affichage
sales.head()
```

Out[37]:

	client_id	sex	birth	id_prod	date	session_id	price	categ
0	c_4410	f	1967	1_483	2021-03-13 21:35:55.949042	s_5913	15.99	1
1	c_4410	f	1967	0_1111	2021-03-22 01:27:49.480137	s_9707	19.99	0
2	c_4410	f	1967	1_385	2021-03-22 01:40:22.782925	s_9707	25.99	1
3	c_4410	f	1967	0_1455	2021-03-22 14:29:25.189266	s_9942	8.99	0
4	c_4410	f	1967	0_1420	2021-03-22 22:31:25.825764	s_10092	11.53	0

In [38]:
```
#Changement du type de donnée de la colonne categ en int
sales['categ'] = pd.to_numeric(sales['categ'], errors='coerce').astype('Int64')
```

In [39]:
```
# Changement du type de la colonne date au format datetime sans les millisecondes
sales['date'] = pd.to_datetime(sales['date']).dt.round('1s')
```

In [40]:
```
#Affichage du dataset sales
sales.head()
```

Out[40]:

	client_id	sex	birth	id_prod	date	session_id	price	categ
0	c_4410	f	1967	1_483	2021-03-13 21:35:56	s_5913	15.99	1
1	c_4410	f	1967	0_1111	2021-03-22 01:27:49	s_9707	19.99	0
2	c_4410	f	1967	1_385	2021-03-22 01:40:23	s_9707	25.99	1
3	c_4410	f	1967	0_1455	2021-03-22 14:29:25	s_9942	8.99	0
4	c_4410	f	1967	0_1420	2021-03-22 22:31:26	s_10092	11.53	0

Étape 3 - Représentations du CA, des ventes, des produits vendus et des clients

3.1 - Analyses du CA

In [43]:
```
#Calcul du chiffre d'affaires total pour 2021 et 2022
chiffre_affaires_total = int(sales['price'].sum())

print(f"Le chiffre d'affaires total est de: {chiffre_affaires_total:,} €".replace(",", " "))
```

```
Le chiffre d'affaires total est de: 12 027 663 €
```

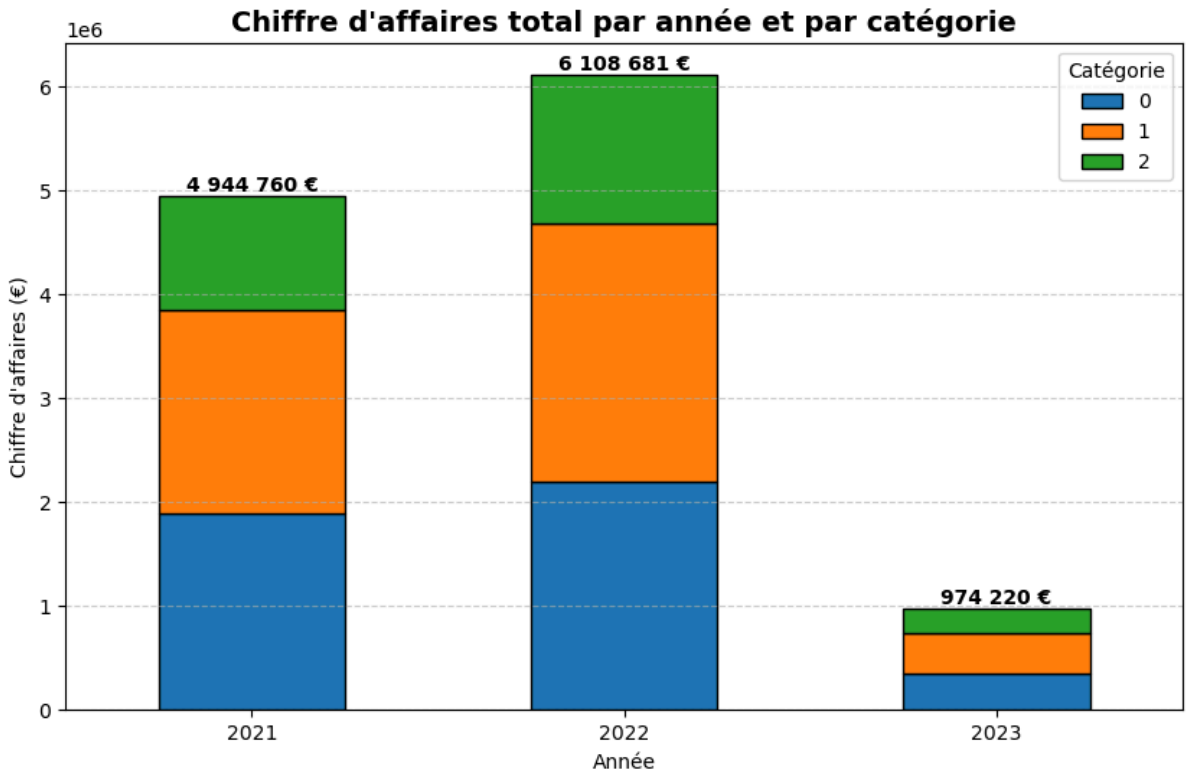
In [44]:
```
# Nombre de commandes
nb_commandes = sales['session_id'].nunique()

# Nombre total de produits vendus
nb_produits = len(sales)

# Nombre moyen de produits vendus par commande
moyenne_produits_par_commande = nb_produits / nb_commandes

# Panier moyen
panier_moyen = chiffre_affaires_total / nb_commandes

# Affichage
print("chiffre_affaires_total:", round(chiffre_affaires_total, 2))
print("Nombre de commandes :", nb_commandes)
print("Nombre de produits vendus :", nb_produits)
print("Nombre moyen de produits vendus par commande :", round(moyenne_produits_par_commande, 2))
print("Panier moyen :", round(panier_moyen, 2))
```

```
chiffre_affaires_total: 12027663
Nombre de commandes : 345505
Nombre de produits vendus : 687534
Nombre moyen de produits vendus par commande : 1.99
Panier moyen : 34.81
```

In [45]:
```
#Ajout de la colonne année
sales["date"] = pd.to_datetime(sales["date"], errors="coerce")

# Extraire l'année et créer une nouvelle colonne
sales["annee"] = sales["date"].dt.year
```

In [46]:
```
# CA par année et catégorie
ca_par_annee_categ = sales.groupby(["annee", "categ"])["price"].sum().unstack(fill_value=0)

# Couleurs pour catégories 0,1,2
couleurs = ["#1f77b4", "#ff7f0e", "#2ca02c"]

# Graphique empilé
ax = ca_par_annee_categ.plot(
    kind="bar",
    stacked=True,
    color=couleurs,
    figsize=(10,6),
    edgecolor="black"
)

# Titre et axes
plt.title("Chiffre d'affaires total par année et par catégorie", fontsize=14, fontweight="bold")
plt.xlabel("Année")
plt.ylabel("Chiffre d'affaires (€)")
plt.xticks(rotation=0)
plt.grid(axis="y", linestyle="--", alpha=0.6)

# Affichage totaux au-dessus des barres
for i, total in enumerate(ca_par_annee_categ.sum(axis=1)):
```

```
    plt.text(i, total, f"{int(total):,}".replace(",", " ") + " €",
             ha="center", va="bottom", fontsize=10, fontweight="bold")

plt.legend(title="Catégorie")
plt.show()
```

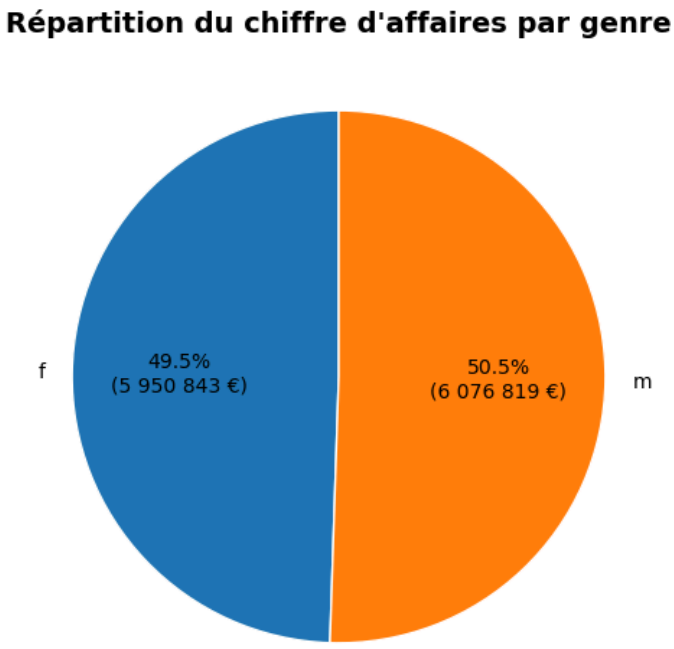


In [47]:
```
# Chiffres d'affaires en euros
ca_2021 = 4_944_760
ca_2022 = 6_108_681

# Calcul du pourcentage d'augmentation
augmentation_percent = ((ca_2022 - ca_2021) / ca_2021) * 100

# Affichage du résultat
print(f"Le chiffre d'affaires a augmenté de {augmentation_percent:.2f} % entre 2021 et 2022.")
```

```
Le chiffre d'affaires a augmenté de 23.54 % entre 2021 et 2022.
```

Entre 2021 et 2022, le chiffre d'affaires a augmenté de 23,5 %, passant de 4,94 M€ à 6,11 M€. Cette croissance reflète une forte dynamique commerciale.

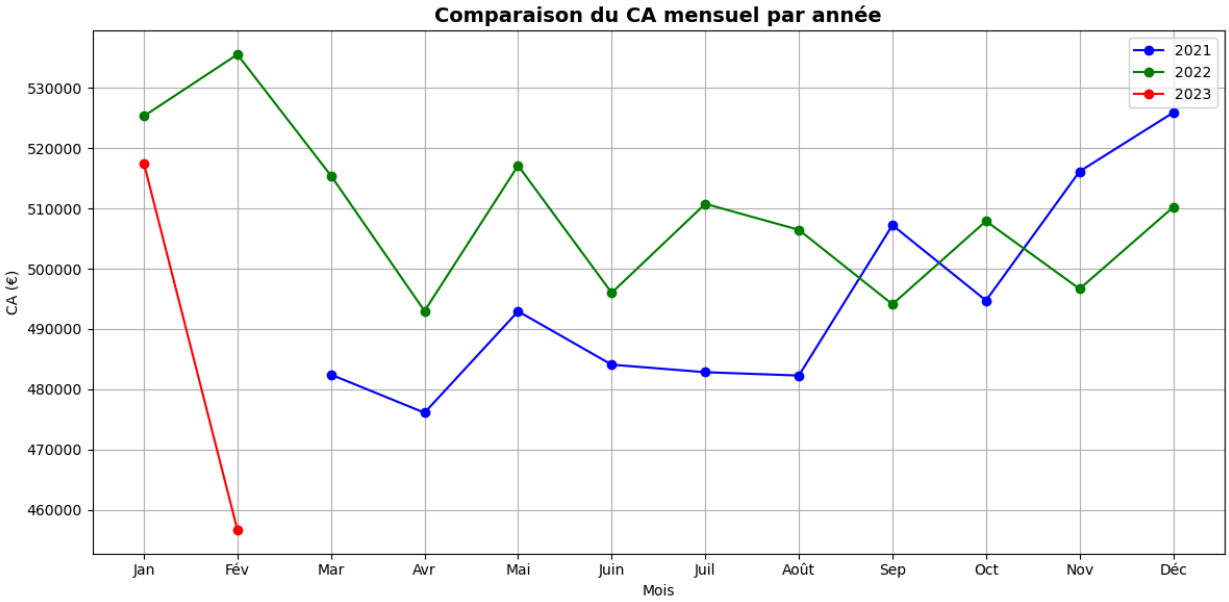
In [49]:
```
#Affichage de la répartition du chiffre d'affaires par genre

# Calcul du CA total par genre
ca_par_genre = sales.groupby("sex")["price"].sum()

# Pie chart
plt.figure(figsize=(6,6))
plt.pie(
    ca_par_genre,
    labels=ca_par_genre.index,
    autopct=lambda p: f"{p:.1f}%\n({int(p*ca_par_genre.sum()/100):,} €)".replace(",", " "),
    startangle=90,
    colors=["#1f77b4", "#ff7f0e"],
    wedgeprops={"edgecolor":"white"}
)

plt.title("Répartition du chiffre d'affaires par genre", fontsize=14, fontweight="bold")
plt.show()
```



Équilibre global : Le chiffre d'affaires est très équitablement réparti entre les deux genres.

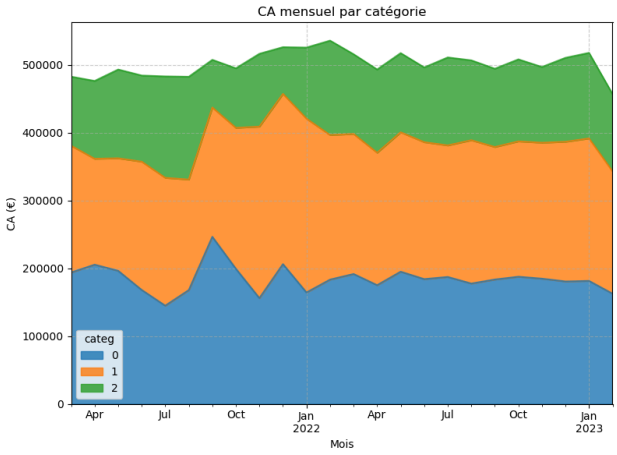
- Écart de seulement 0,9 point de pourcentage, soit 125 976 € de différence.
- Légère domination masculine :
- Les hommes génèrent légèrement plus de chiffre d'affaires, mais l'écart est faible et probablement non significatif statistiquement

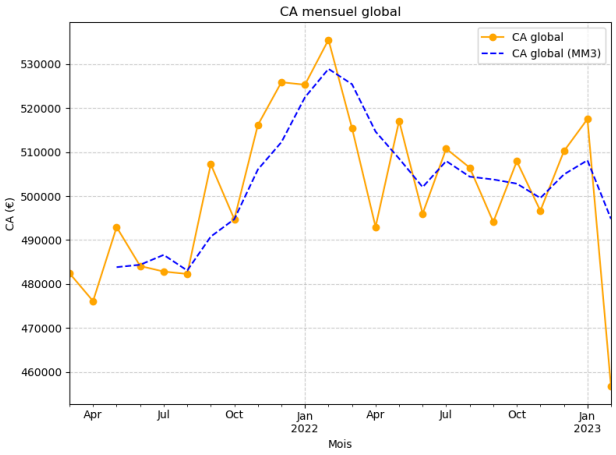
In [51]:
```
#Affichage de l'evolution du CA par année

# Grouper les données par année et mois (pas la date complète)
sales['year'] = sales['date'].dt.year
sales['month'] = sales['date'].dt.month

# Calcul du CA total par mois et par année
ca_mensuel = sales.groupby(['year', 'month'])['price'].sum().unstack(level=0)

# Affichage
plt.figure(figsize=(12, 6))
mois_labels = ['Jan', 'Fév', 'Mar', 'Avr', 'Mai', 'Juin',
               'Juil', 'Août', 'Sep', 'Oct', 'Nov', 'Déc']

# Tracage de chaque année avec couleur distincte
if 2021 in ca_mensuel.columns:
    plt.plot(ca_mensuel.index, ca_mensuel[2021], marker='o', label='2021', color='blue')
if 2022 in ca_mensuel.columns:
    plt.plot(ca_mensuel.index, ca_mensuel[2022], marker='o', label='2022', color='green')
if 2023 in ca_mensuel.columns:
    plt.plot(ca_mensuel.index, ca_mensuel[2023], marker='o', label='2023', color='red')

# Mise en forme
plt.xticks(ticks=range(1, 13), labels=mois_labels)
plt.title("Comparaison du CA mensuel par année", fontsize=14, fontweight='bold')
plt.xlabel("Mois")
plt.ylabel("CA (€)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



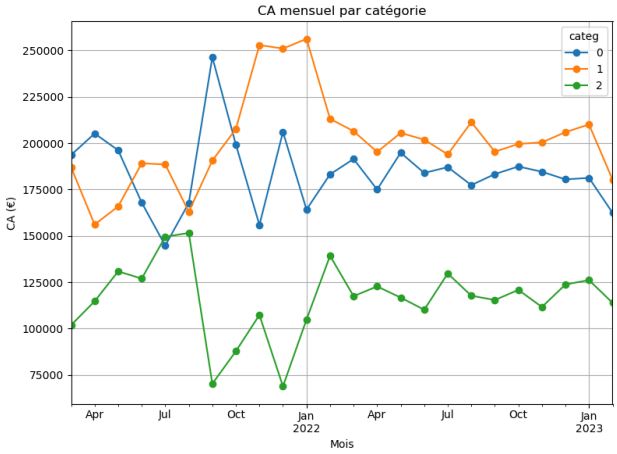
Le chiffre d'affaires mensuel en 2022 est supérieur à celui de 2021 sur l'ensemble de l'année. En 2021, le CA reste stable avec une légère hausse en fin d'année. L'année 2022 montre une meilleure performance globale, malgré des variations plus marquées entre les mois. En 2023, seuls les données de janvier et février sont disponibles, avec une forte baisse entre les deux mois.

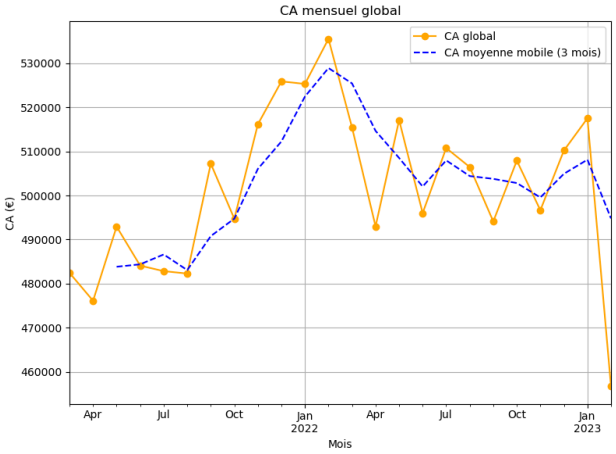
In [53]:
```
# CA mensuel par catégorie
ca_mensuel_categ = (
    sales.groupby([sales['date'].dt.to_period("M"), 'categ'])['price']
    .sum()
    .unstack(fill_value=0)
)
ca_mensuel_categ.index = ca_mensuel_categ.index.to_timestamp()

# CA global et moyenne mobile
ca_mensuel_global = ca_mensuel_categ.sum(axis=1)
ca_mensuel_global_mm3 = ca_mensuel_global.rolling(3).mean()

# Deux graphiques côte à côte
fig, axes = plt.subplots(1, 2, figsize=(16,6))

# Aire empilée par catégorie
ca_mensuel_categ.plot.area(
    ax=axes[0], stacked=True, alpha=0.8, title="CA mensuel par catégorie"
)
axes[0].set_xlabel("Mois"); axes[0].set_ylabel("CA (€)"); axes[0].grid(True, linestyle="--", alpha=0.6)

# Affichage CA mensuel global
ca_mensuel_global.plot(ax=axes[1], marker="o", color="orange", label="CA global")
ca_mensuel_global_mm3.plot(ax=axes[1], linestyle="--", color="blue", label="CA global (MM3)")
axes[1].set_title("CA mensuel global")
axes[1].set_xlabel("Mois"); axes[1].set_ylabel("CA (€)"); axes[1].grid(True, linestyle="--", alpha=0.6); axes[1].leger

plt.tight_layout()
plt.show()
```





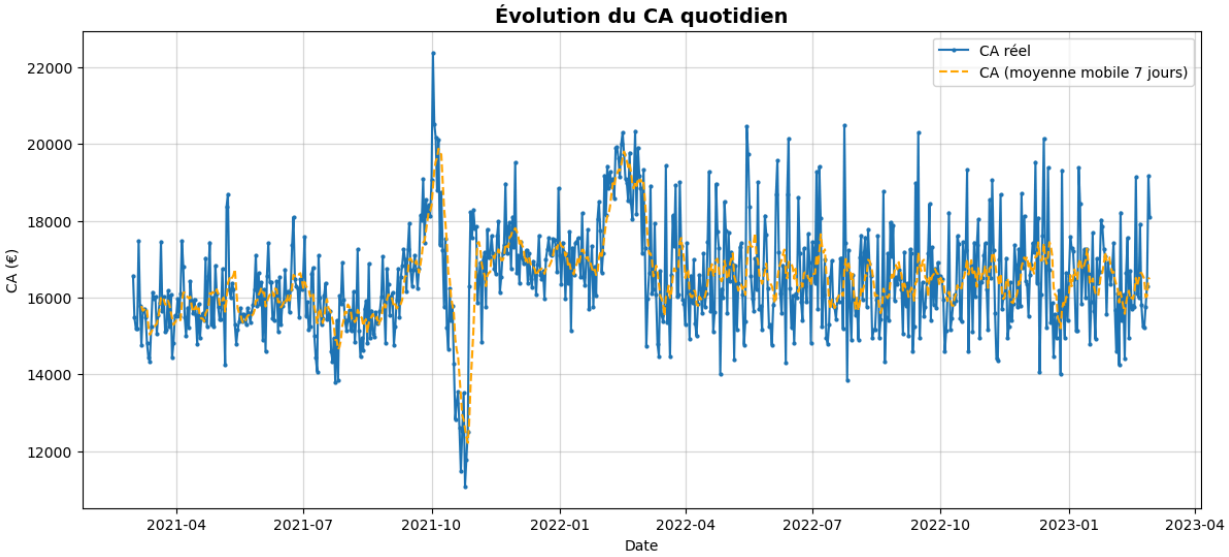
In [54]:
```
# Grouper par mois et catégorie pour calculer le CA
ca_mensuel_categ = (
    sales.groupby([sales['date'].dt.to_period("M"), 'categ'])['price']
    .sum()
    .unstack(fill_value=0)
)
ca_mensuel_categ.index = ca_mensuel_categ.index.to_timestamp()

# CA global + moyenne mobile
ca_mensuel_global = ca_mensuel_categ.sum(axis=1)
ca_mensuel_global_mm3 = ca_mensuel_global.rolling(3).mean()

# Couleurs
couleurs = ["#1f77b4", "#ff7f0e", "#2ca02c"]

# Deux graphiques côte à côte
fig, axes = plt.subplots(1, 2, figsize=(16,6))

# --- Par catégorie (sans MM3) ---
ca_mensuel_categ.plot(
    ax=axes[0], marker="o", color=couleurs, title="CA mensuel par catégorie"
)
axes[0].set_xlabel("Mois"); axes[0].set_ylabel("CA (€)"); axes[0].grid(True)

# --- Global (avec MM3) ---
ca_mensuel_global.plot(
    ax=axes[1], marker="o", color="orange", label="CA global"
)
ca_mensuel_global_mm3.plot(
    ax=axes[1], linestyle="--", color="blue", label="CA moyenne mobile (3 mois)"
)
axes[1].set_title("CA mensuel global")
axes[1].set_xlabel("Mois"); axes[1].set_ylabel("CA (€)"); axes[1].grid(True); axes[1].legend()

plt.tight_layout()
plt.show()
```





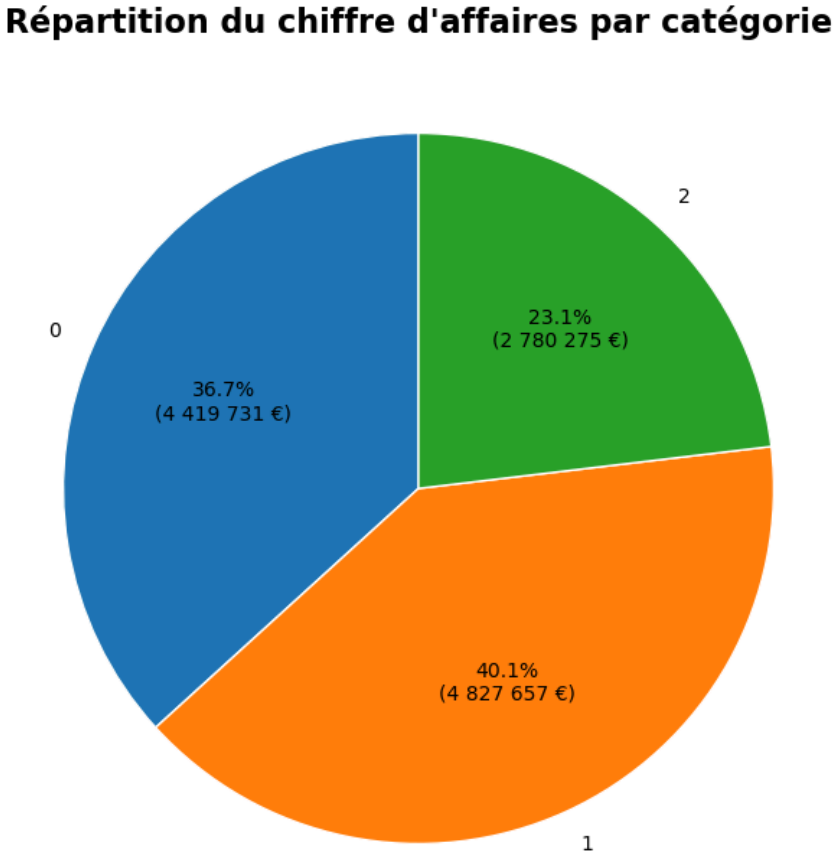
In [55]:
```
#Graphique représentant le CA quotidien et moyenne mobile sur 7 jours
# Grouper par jour et calculer le CA total
ca_quotidien = sales.groupby(sales['date'].dt.to_period("D"))['price'].sum()
ca_quotidien.index = ca_quotidien.index.to_timestamp()

# Calcul de la moyenne mobile sur 7 jours
ca_quotidien_mm7 = ca_quotidien.rolling(window=7).mean()
```

```
# Plot
plt.figure(figsize=(14,6))
plt.plot(ca_quotidien.index, ca_quotidien, marker='o', markersize=2, linestyle='-', label="CA réel")
plt.plot(ca_quotidien_mm7.index, ca_quotidien_mm7, linestyle="--", color="orange", label="CA (moyenne mobile 7 jours)"

plt.title("Évolution du CA quotidien", fontsize=14, fontweight="bold")
plt.xlabel("Date")
plt.ylabel("CA (€)")
plt.legend()
plt.grid(True, alpha=0.5)
plt.show()
```



In [56]:

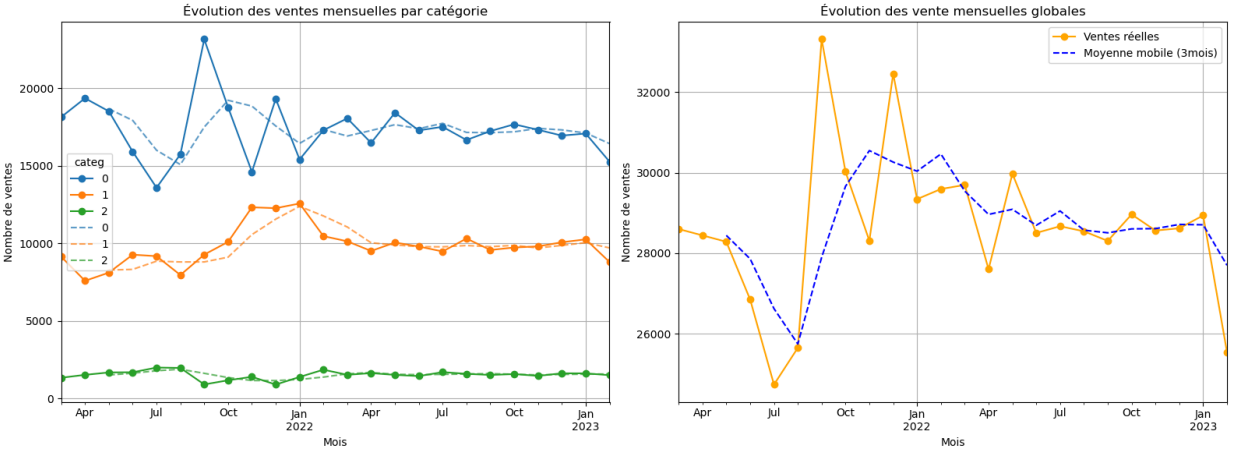
```
#CA par ctageories
ca_par_categ = sales.groupby("categ")["price"].sum()

plt.figure(figsize=(8,8))

# Fonction pour afficher % et montant
def autopct_format(values):
    def inner_autopct(pct):
        total = sum(values)
        val = int(round(pct*total/100.0))
        return f"{pct:.1f}%\n({val:,.0f} €)".replace(",", " ")
    return inner_autopct

#Affichage du pie chart
plt.pie(
    ca_par_categ,
    labels=ca_par_categ.index,
    autopct=autopct_format(ca_par_categ),
    startangle=90,
    colors=["#1f77b4", "#ff7f0e", "#2ca02c", "#d62728", "#9467bd"],
    wedgeprops={"edgecolor":"white"})
plt.title("Répartition du chiffre d'affaires par catégorie", fontsize=16, fontweight="bold")
plt.show()
```



3.2 - Les évolutions

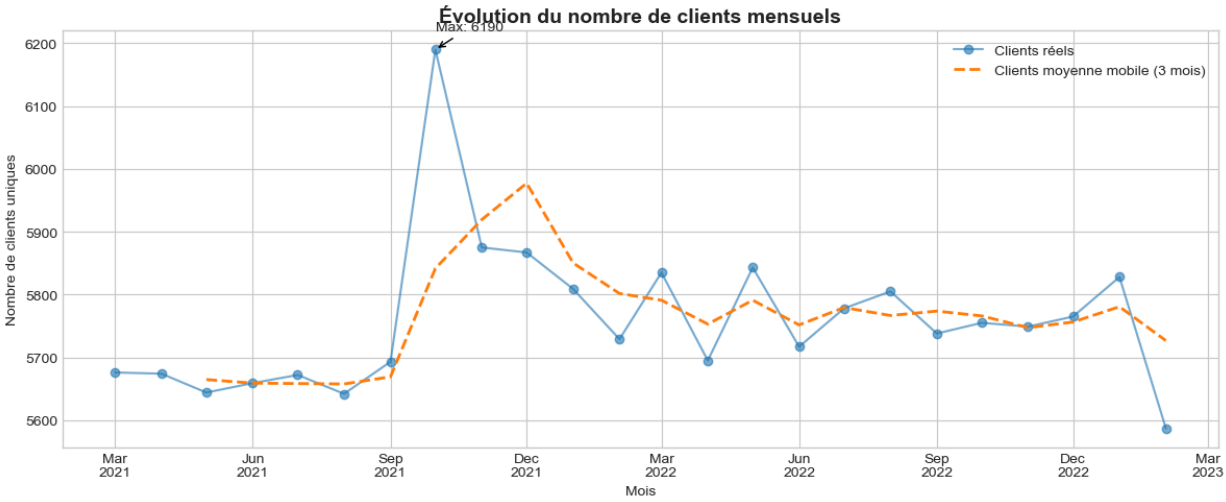
In [58]:
```
# Affichage de l'évolution du nombre de ventes mensuelles
# Grouper par mois et catégorie
ventes_mensuelles_categ = (
    sales.groupby([sales['date'].dt.to_period("M"), 'categ'])['id_prod']
    .count()
    .unstack(fill_value=0)
)
ventes_mensuelles_categ.index = ventes_mensuelles_categ.index.to_timestamp()

# Ventes globales
ventes_mensuelles_global = ventes_mensuelles_categ.sum(axis=1)

# Moyenne mobile (3 mois)
ventes_mensuelles_categ_mm3 = ventes_mensuelles_categ.rolling(3).mean()
ventes_mensuelles_global_mm3 = ventes_mensuelles_global.rolling(3).mean()

# Couleurs des catégories
couleurs = ["#1f77b4", "#ff7f0e", "#2ca02c"]

# Deux graphiques côte à côte
fig, axes = plt.subplots(1, 2, figsize=(16,6), sharey=False)

# Graphique gauche: Evolution par catégorie
ventes_mensuelles_categ.plot(
    ax=axes[0], marker="o", color=couleurs, title="Évolution des ventes mensuelles par catégorie")
ventes_mensuelles_categ_mm3.plot(
    ax=axes[0], linestyle="--", color=couleurs, alpha=0.7)
axes[0].set_xlabel("Mois")
axes[0].set_ylabel("Nombre de ventes")
axes[0].grid(True)

# Graphique droit: Evolution mensuelle globale
ventes_mensuelles_global.plot(
    ax=axes[1], marker="o", color="orange", label="Ventes réelles")

ventes_mensuelles_global_mm3.plot(
    ax=axes[1], linestyle="--", color="blue", label="Moyenne mobile (3mois)")
axes[1].set_title("Évolution des vente mensuelles globales")
axes[1].set_xlabel("Mois")
axes[1].set_ylabel("Nombre de ventes")
```

```
axes[1].grid(True)
axes[1].legend()

plt.tight_layout()
plt.show()
```

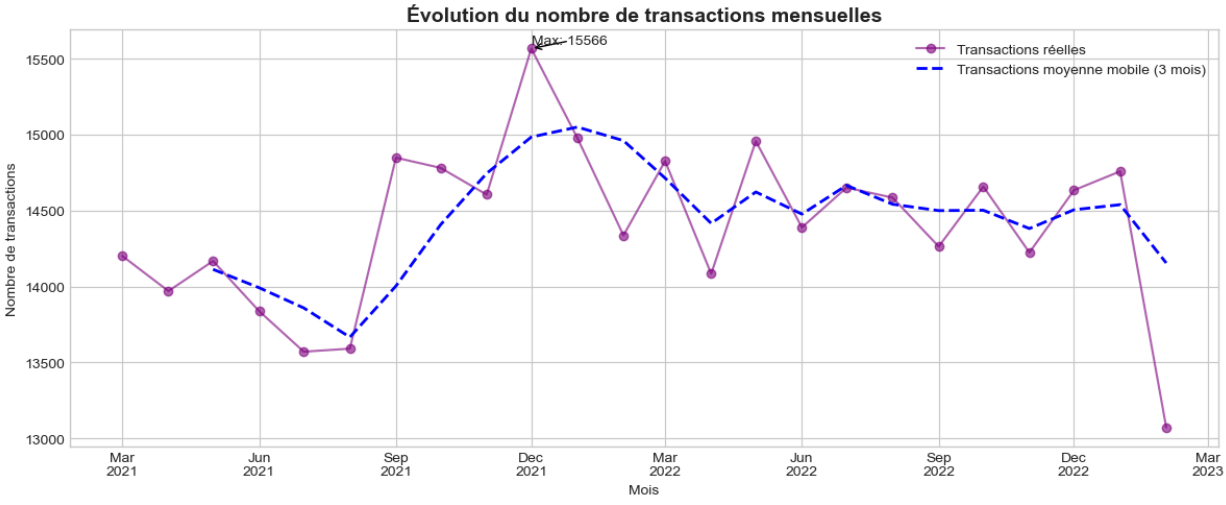


In [59]:

```
import matplotlib.dates as mdates

# Calcul du nombre de clients uniques par mois
clients_mensuels = sales.groupby(sales['date'].dt.to_period("M"))['client_id'].nunique()
clients_mensuels.index = clients_mensuels.index.to_timestamp()  # pour l'axe X

# Calcul de la moyenne mobile sur 3 mois
clients_mensuels_mm3 = clients_mensuels.rolling(window=3).mean()
# Évolution du nombre de clients mensuels
plt.style.use("seaborn-v0_8-whitegrid")
plt.figure(figsize=(12,5))

# Courbe réelle
plt.plot(clients_mensuels.index, clients_mensuels,
         marker='o', color="#1f77b4", alpha=0.6, label="Clients réels")

# Moyenne mobile
plt.plot(clients_mensuels_mm3.index, clients_mensuels_mm3,
         linestyle="--", color="#ff7f0e", linewidth=2, label="Clients moyenne mobile (3 mois)")

# Amélioration de l'axe X
plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=3))
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter("%b\n%Y"))

# Ajout des annotations
max_date, max_value = clients_mensuels.idxmax(), clients_mensuels.max()
plt.annotate(f"Max: {max_value}", xy=(max_date, max_value),
             xytext=(max_date, max_value+30),
             arrowprops=dict(arrowstyle="->", color="black"))

# Affichage du graphique
plt.title("Évolution du nombre de clients mensuels", fontsize=14, fontweight="bold")
plt.xlabel("Mois")
plt.ylabel("Nombre de clients uniques")
plt.legend()
plt.tight_layout()
plt.show()
```

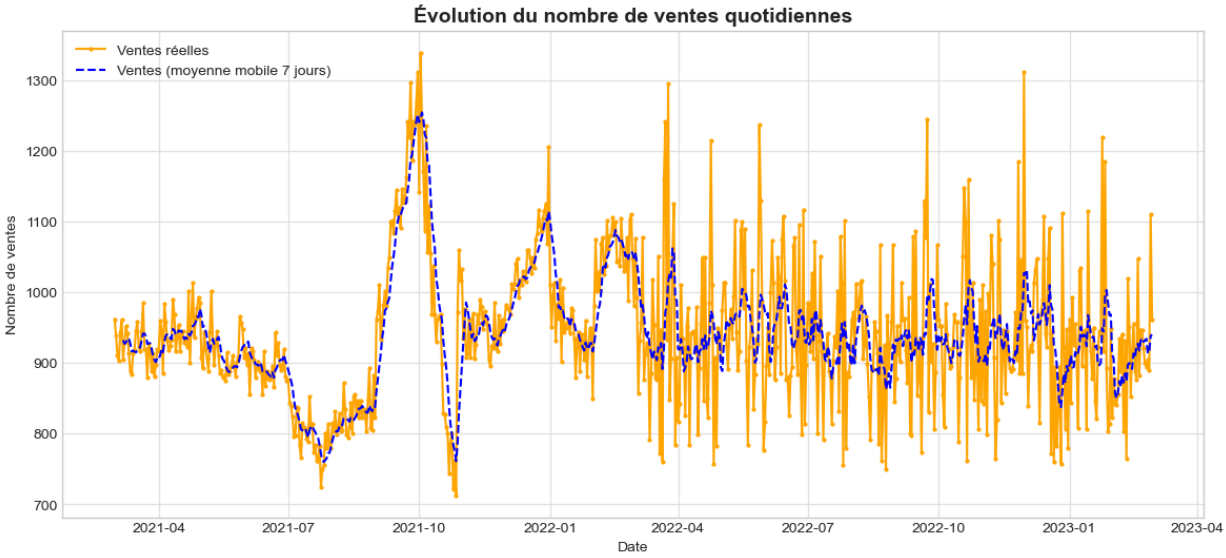


In [60]:
```
# Évolution du nombre de transactions mensuelles
# Calcul du nombre de transactions uniques par mois
transactions_mensuelles = sales.groupby(sales['date'].dt.to_period("M"))['session_id'].nunique()
transactions_mensuelles.index = transactions_mensuelles.index.to_timestamp()  # pour l'axe X

# Calcul de la moyenne mobile sur 3 mois
transactions_mensuelles_mm3 = transactions_mensuelles.rolling(window=3).mean()

# Style
plt.style.use("seaborn-v0_8-whitegrid")
plt.figure(figsize=(12,5))

# Transactions réelles
plt.plot(transactions_mensuelles.index, transactions_mensuelles,
         marker='o', color="purple", alpha=0.6, label="Transactions réelles")

# Moyenne mobile
plt.plot(transactions_mensuelles_mm3.index, transactions_mensuelles_mm3,
         linestyle="--", color="blue", linewidth=2, label="Transactions moyenne mobile (3 mois)")

# Amélioration de l'axe X
plt.gca().xaxis.set_major_locator(mdates.MonthLocator(interval=3))
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter("%b\n%Y"))

# Annotation du max
max_date, max_value = transactions_mensuelles.idxmax(), transactions_mensuelles.max()
plt.annotate(f"Max: {max_value}", xy=(max_date, max_value),
             xytext=(max_date, max_value+30),
             arrowprops=dict(arrowstyle="->", color="black"))

# Titres et labels
plt.title("Évolution du nombre de transactions mensuelles", fontsize=14, fontweight="bold")
plt.xlabel("Mois")
plt.ylabel("Nombre de transactions")
plt.legend()
plt.tight_layout()
plt.show()
```



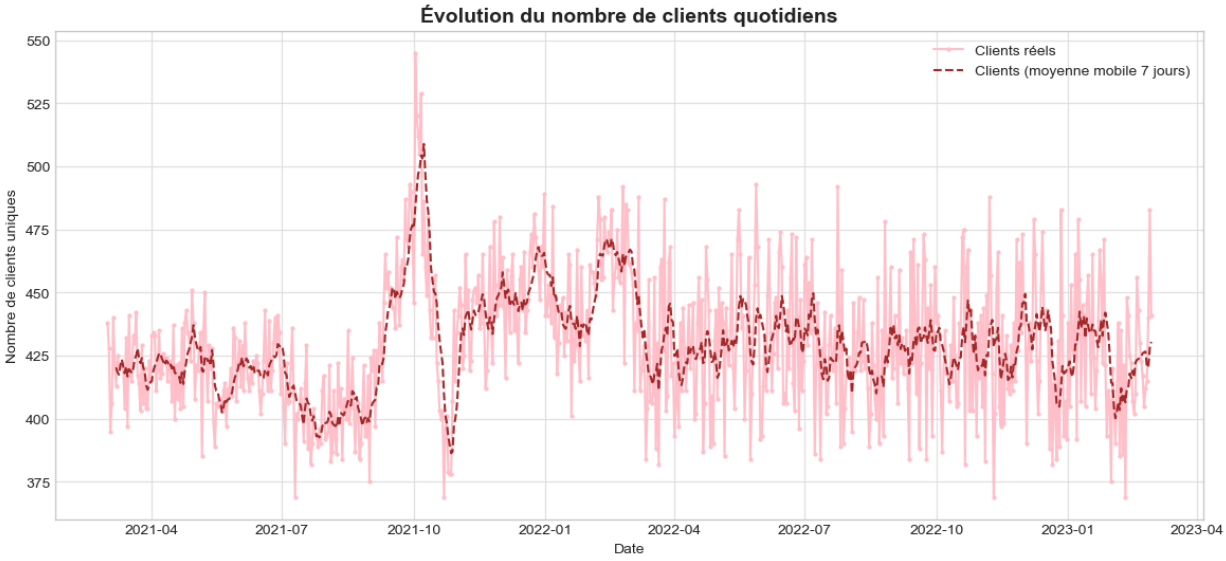
In [61]:
```
# Graphique représentant l'évolution des ventes quotidiennes et moyenne mobile sur 7 jours
# Grouper par jour et compter le nombre de ventes
```

```
ventes_quotidiennes = sales.groupby(sales['date'].dt.to_period("D"))['id_prod'].count()
ventes_quotidiennes.index = ventes_quotidiennes.index.to_timestamp()

# Calcul de la moyenne mobile sur 7 jours
ventes_quotidiennes_mm7 = ventes_quotidiennes.rolling(window=7).mean()

# Affichage du graphique
plt.figure(figsize=(14,6))
plt.plot(ventes_quotidiennes.index, ventes_quotidiennes, marker='o', markersize=2, linestyle='-', color="orange", labe
plt.plot(ventes_quotidiennes_mm7.index, ventes_quotidiennes_mm7, linestyle="--", color="blue", label="Ventes (moyenne

plt.title("Évolution du nombre de ventes quotidiennes", fontsize=14, fontweight="bold")
plt.xlabel("Date")
plt.ylabel("Nombre de ventes")
plt.legend()
plt.grid(True, alpha=0.5)
plt.show()
```

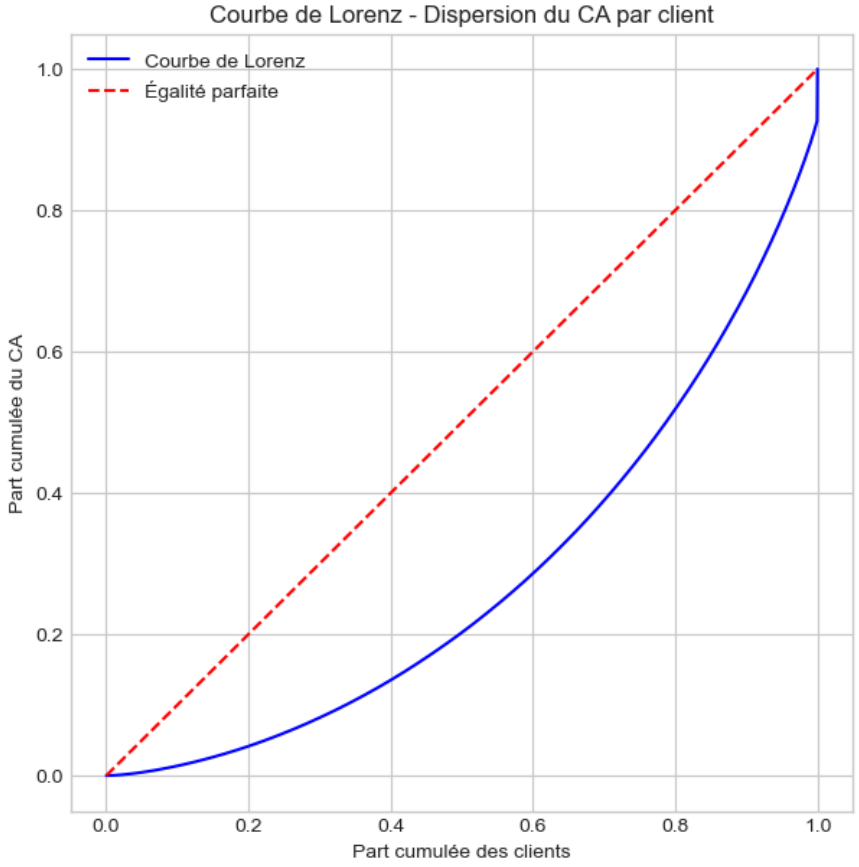


In [62]:

```
# Graphique représentant l'évolution du nombres de clients quotidiens
# Grouper par jour et compter le nombre de clients uniques
clients_quotidiens = sales.groupby(sales['date'].dt.to_period("D"))['client_id'].nunique()
clients_quotidiens.index = clients_quotidiens.index.to_timestamp()

# Calcul de la moyenne mobile sur 7 jours
clients_quotidiens_mm7 = clients_quotidiens.rolling(window=7).mean()

# Plot
plt.figure(figsize=(14,6))
plt.plot(clients_quotidiens.index, clients_quotidiens, marker='o', markersize=2, linestyle='-', color="pink", label="C
plt.plot(clients_quotidiens_mm7.index, clients_quotidiens_mm7, linestyle="--", color="brown", label="Clients (moyenne

plt.title("Évolution du nombre de clients quotidiens", fontsize=14, fontweight="bold")
plt.xlabel("Date")
plt.ylabel("Nombre de clients uniques")
plt.legend()
plt.grid(True, alpha=0.5)
plt.show()
```



In [63]:
```
# Courbe de Lorenz
# CA par client
ca_clients = sales.groupby('client_id')['price'].sum().reset_index()
ca_clients = ca_clients.sort_values('price')

# Part cumulée des clients (X)
ca_clients['part_clients'] = np.arange(1, len(ca_clients)+1) / len(ca_clients)

# Part cumulée du CA (Y)
ca_clients['part_ca'] = ca_clients['price'].cumsum() / ca_clients['price'].sum()

# Affichage de la courbe de Lorenz
plt.figure(figsize=(7,7))
plt.plot(ca_clients['part_clients'], ca_clients['part_ca'], label="Courbe de Lorenz", color="blue")
plt.plot([0,1], [0,1], linestyle="--", color="red", label="Égalité parfaite")

plt.title("Courbe de Lorenz - Dispersion du CA par client")
plt.xlabel("Part cumulée des clients")
plt.ylabel("Part cumulée du CA")
plt.legend()
plt.grid(True)
plt.show()
```

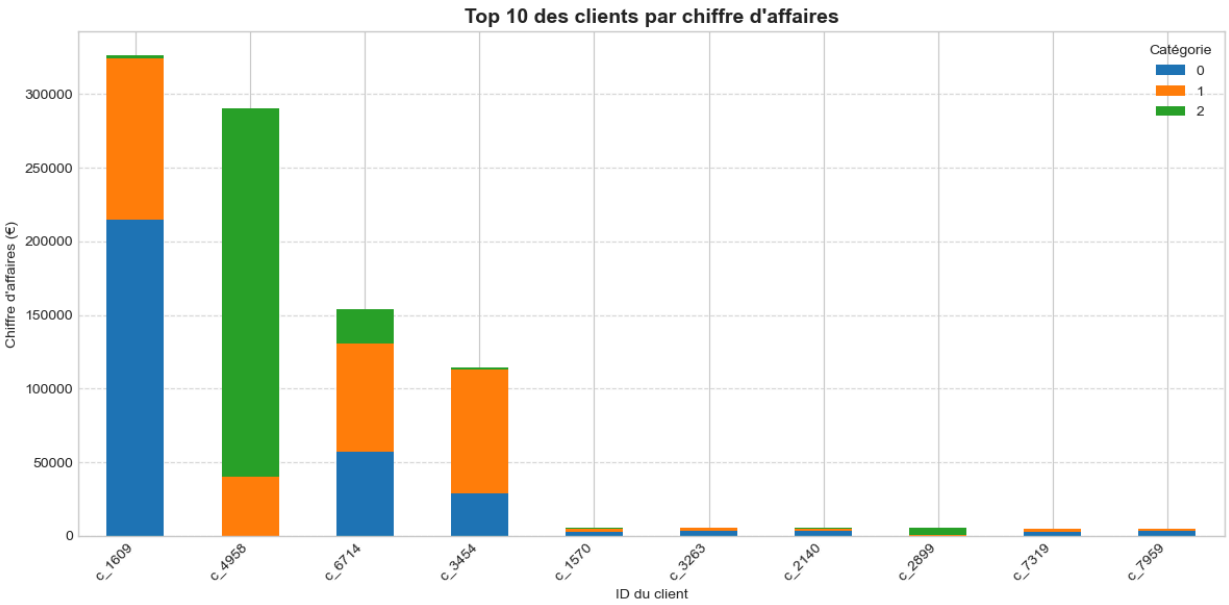


La courbe de Lorenz montre une forte inégalité dans la répartition du chiffre d'affaires entre les clients. Une petite proportion de clients génère la majorité du CA total. Plus la courbe s'éloigne de la diagonale, plus la concentration est élevée. Cela indique une forte dépendance à quelques clients clés. Cela peut s'expliquer par la présence de clients BTOB dans le jeu de données.

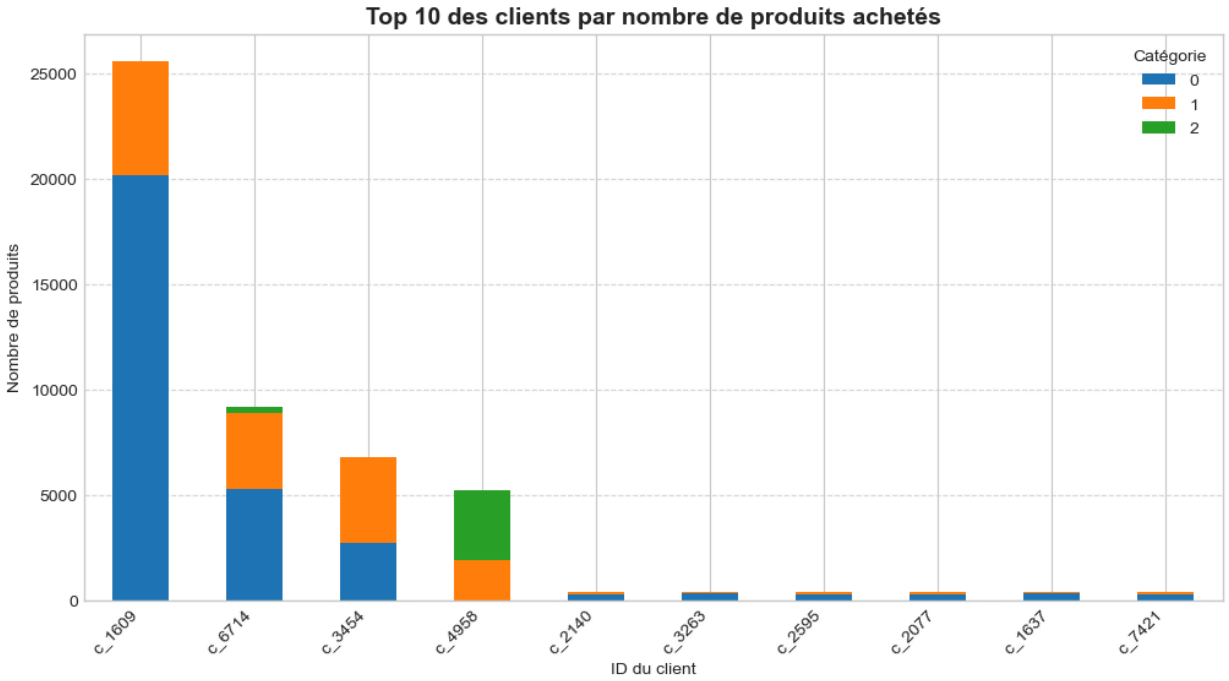
Étape 4 - Les TOPS ET FLOPS

4.1 - Les TOPS clients

In [67]:
```
#Afficher le TOP 10 des clients
from fonctions import top_clients
top_clients(sales)
```



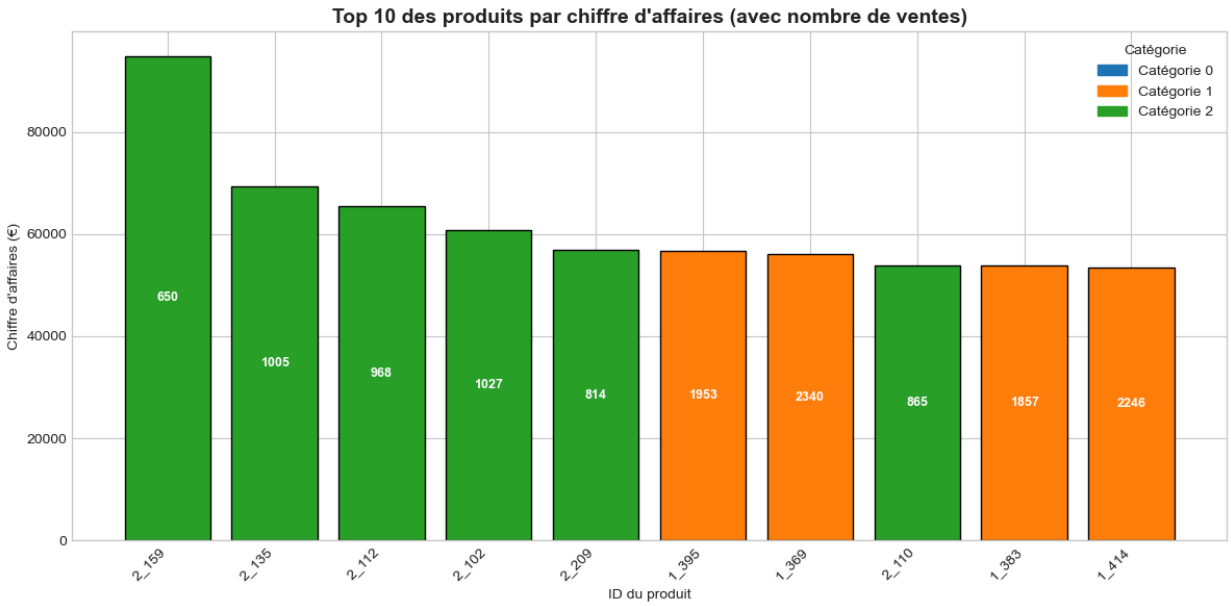
In [68]:
```
# Nombre de produits achetés par client et catégorie
top_clients_produits = sales.groupby(["client_id", "categ"])["id_prod"].count().reset_index(name="nb_produits")

# Top 10 clients par nombre total de produits
top10_clients_produits = (
    top_clients_produits.groupby("client_id")["nb_produits"].sum()
    .nlargest(10)
    .sort_values(ascending=False)
)

# Restreindre aux top 10
df_top10_prod = top_clients_produits[top_clients_produits["client_id"].isin(top10_clients_produits.index)]

# Transformer en format pivot pour barres empilées
df_pivot_prod = df_top10_prod.pivot(index="client_id", columns="categ", values="nb_produits").fillna(0)

# Réordonner les clients selon le nombre total de produits décroissant
df_pivot_prod = df_pivot_prod.loc[top10_clients_produits.index]

# Tracer avec couleurs personnalisées
df_pivot_prod.plot(
    kind="bar",
    stacked=True,
    figsize=(12,6),
    color=["#1f77b4", "#ff7f0e", "#2ca02c"]
)

plt.title("Top 10 des clients par nombre de produits achetés", fontsize=14, fontweight="bold")
plt.xlabel("ID du client")
plt.ylabel("Nombre de produits")
plt.xticks(rotation=45, ha="right")
plt.legend(title="Catégorie")
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```



On observe que quatre clients se distinguent nettement par un chiffre d'affaires bien supérieur à celui des autres. Le second graphique montre une tendance similaire en ce qui concerne le nombre de produits achetés. Cela suggère qu'il s'agit probablement de clients BtoB, en raison de leurs volumes d'achat importants

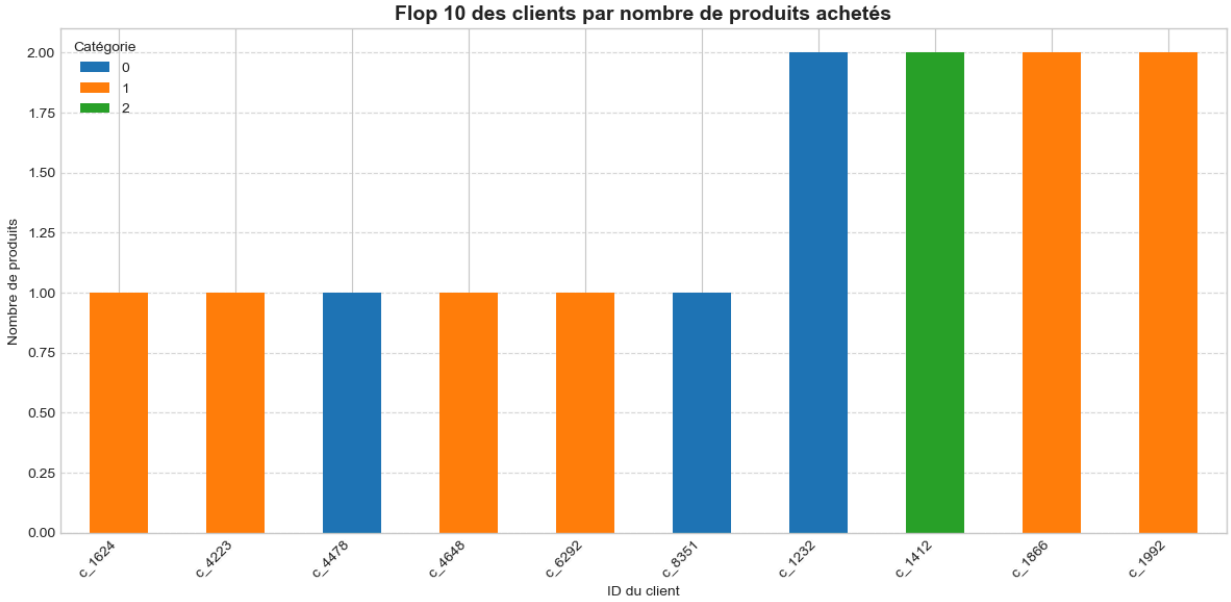
4.1.2 - Les TOPS produits

In [71]:
```
#Affichage du TOPS10 produits
# Palette de couleurs
couleurs = {0: "#1f77b4", 1: "#ff7f0e", 2: "#2ca02c"}

# Top 10 produits par CA
top10_ca = (
    sales.groupby(["id_prod", "categ"])
    .agg(ca_total=("price", "sum"), nb_ventes=("id_prod", "count"))
    .reset_index()
    .sort_values("ca_total", ascending=False)
    .head(10)
)

plt.figure(figsize=(12,6))
bars = plt.bar(
    top10_ca["id_prod"].astype(str),
    top10_ca["ca_total"],
    color=[couleurs[c] for c in top10_ca["categ"]],
    edgecolor="black"
)

plt.title("Top 10 des produits par chiffre d'affaires (avec nombre de ventes)", fontsize=14, fontweight="bold")
plt.xlabel("ID du produit")
plt.ylabel("Chiffre d'affaires (€)")
plt.xticks(rotation=45, ha="right", fontsize=10)
plt.legend(
    [plt.Rectangle((0,0),1,1,color=c) for c in couleurs.values()],
    [f"Catégorie {k}" for k in couleurs.keys()],
    title="Catégorie"
)

# Labels
for bar, ventes in zip(bars, top10_ca["nb_ventes"]):
    plt.text(bar.get_x() + bar.get_width()/2, bar.get_height()/2, ventes,
             ha="center", va="center", color="white", fontsize=9, fontweight="bold")

plt.tight_layout()
plt.show()
```

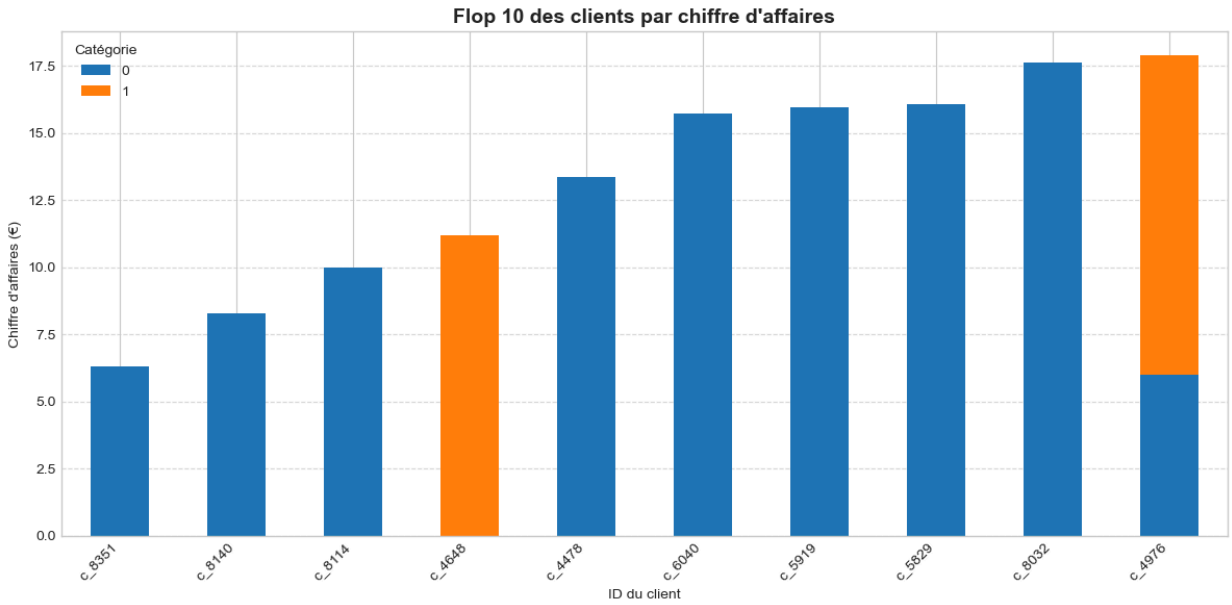


4.1.3 - Les FLOPS produits

In [73]:
```
# Flop 10 clients par nombre total de produits
flop10_clients_produits = (
    top_clients_produits.groupby("client_id")["nb_produits"].sum()
    .nsmallest(10)   # <-- changement ici
    .sort_values(ascending=True))

# Restreindre aux flop 10
df_flop10_prod = top_clients_produits[top_clients_produits["client_id"].isin(flop10_clients_produits.index)]

# Transformer en format pivot (pour barres empilées)
df_pivot_flop = df_flop10_prod.pivot(index="client_id", columns="categ", values="nb_produits").fillna(0)

# Réordonner les clients selon le nombre total de produits croissant
df_pivot_flop = df_pivot_flop.loc[flop10_clients_produits.index]

# Tracer le graphique
df_pivot_flop.plot(
    kind="bar",
    stacked=True,
    figsize=(12,6),
    color=["#1f77b4", "#ff7f0e", "#2ca02c"]
)

plt.title("Flop 10 des clients par nombre de produits achetés", fontsize=14, fontweight="bold")
plt.xlabel("ID du client")
plt.ylabel("Nombre de produits")
plt.xticks(rotation=45, ha="right")
plt.legend(title="Catégorie")
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.tight_layout()
plt.show()
```



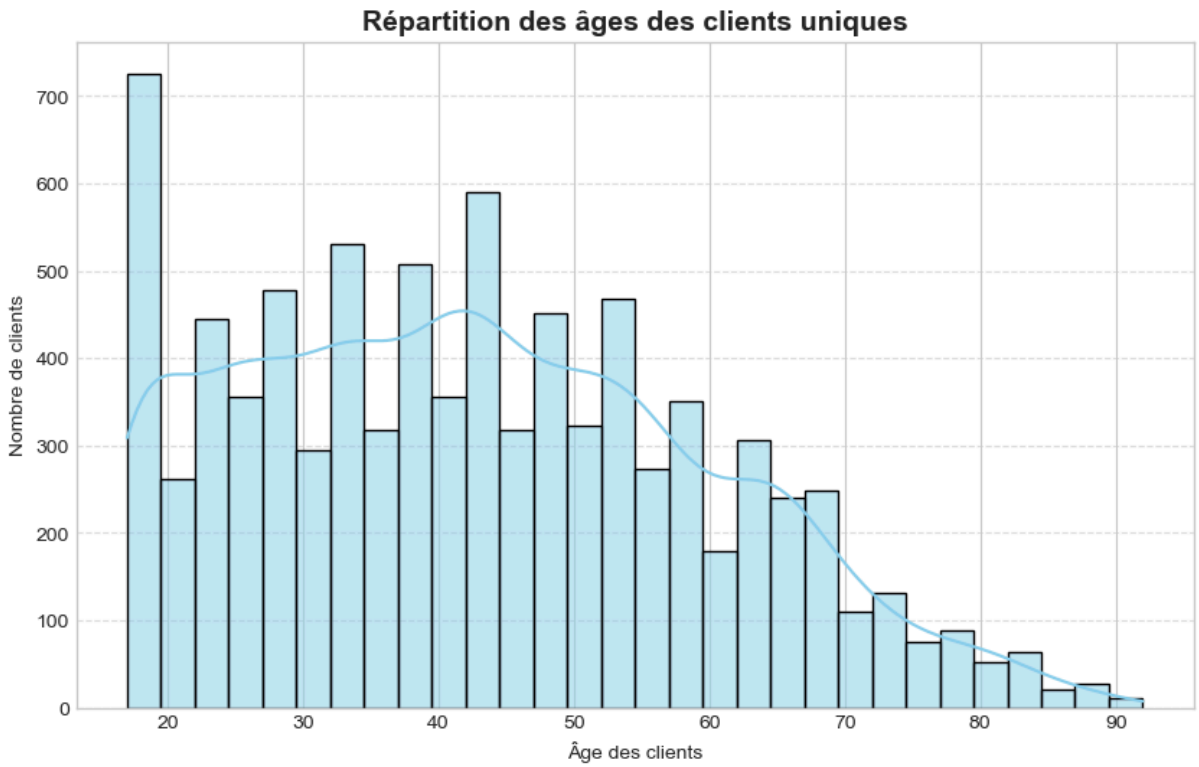
In [74]:
```
# Flop 10 des clients par chiffre d'affaires
# CA par client et catégorie
top_clients_categ = sales.groupby(["client_id", "categ"])["price"].sum().reset_index()

# Flop 10 clients par CA total
flop10_clients_CA = (
    top_clients_categ.groupby("client_id")["price"].sum()
    .nsmallest(10)   # <-- ici on prend les plus petits
    .sort_values(ascending=True)
)

# flop 10
df_flop10_CA = top_clients_categ[top_clients_categ["client_id"].isin(flop10_clients_CA.index)]

# format pivot (pour barres empilées)
df_pivot_flop_CA = df_flop10_CA.pivot(index="client_id", columns="categ", values="price").fillna(0)

# Réordonner les clients selon le CA croissant
df_pivot_flop_CA = df_pivot_flop_CA.loc[flop10_clients_CA.index]

# Affichage du graphique
df_pivot_flop_CA.plot(
    kind="bar",
    stacked=True,
    figsize=(12,6),
    color=["#1f77b4", "#ff7f0e", "#2ca02c"]
)

plt.title("Flop 10 des clients par chiffre d'affaires", fontsize=14, fontweight="bold")
plt.xlabel("ID du client")
plt.ylabel("Chiffre d'affaires (€)")
plt.xticks(rotation=45, ha="right")
plt.legend(title="Catégorie")
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.tight_layout()
plt.show()
```



4.2 - Âge

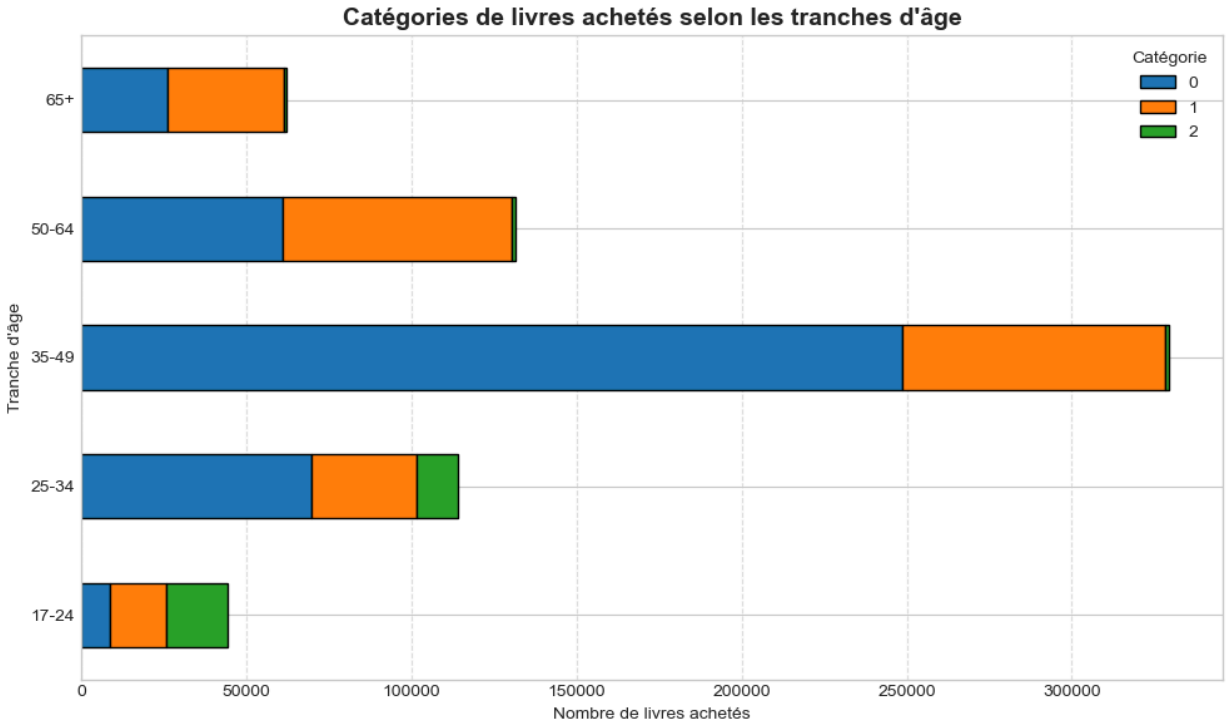
In [76]:
```
# Calcul de l'âge en années
sales["age"] = sales["annee"] - sales["birth"]
```

In [77]:
```
#Import de la librairie
import seaborn as sns

# Un seul enregistrement par client
clients_uniques = sales.drop_duplicates(subset="client_id")

# Distribution des âges
plt.figure(figsize=(10,6))
sns.histplot(clients_uniques["age"], bins=30, kde=True, color="skyblue", edgecolor="black")

plt.title("Répartition des âges des clients uniques", fontsize=14, fontweight="bold")
plt.xlabel("Âge des clients")
plt.ylabel("Nombre de clients")
plt.grid(axis="y", linestyle="--", alpha=0.6)
plt.show()
```



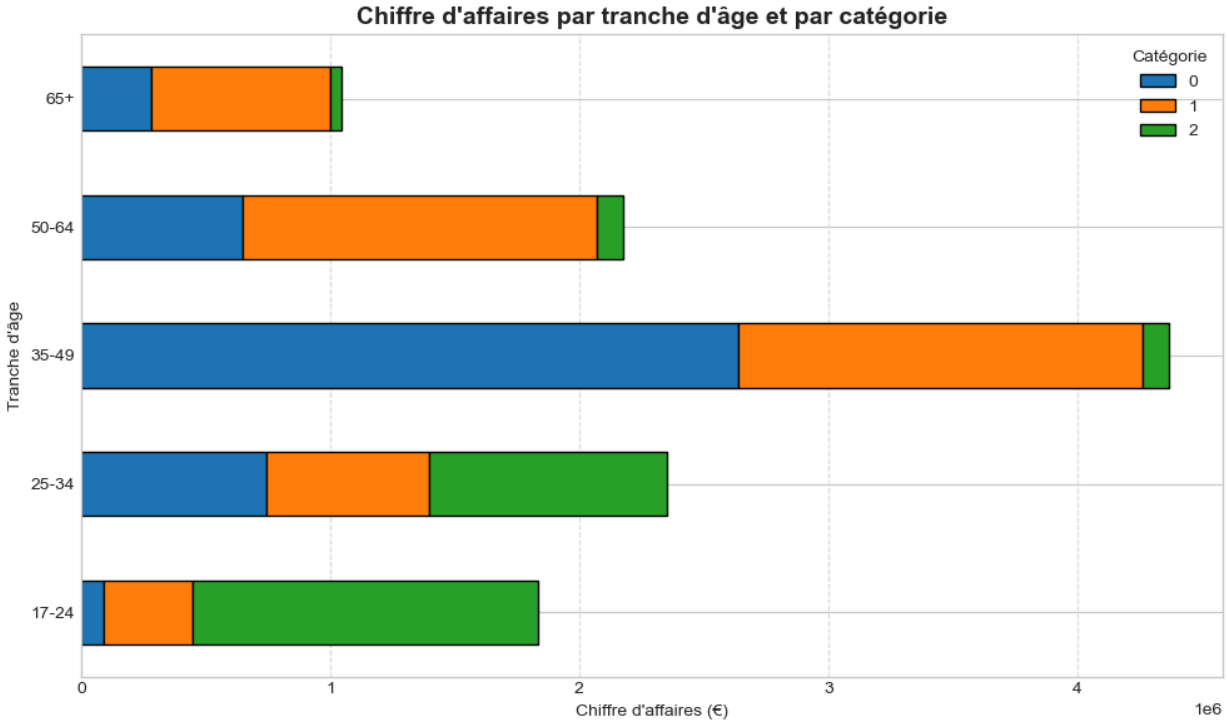
La majorité des clients ont entre 18 et 55 ans, avec un pic notable autour de 20 ans. La répartition reste relativement stable entre 25 et 55 ans, ce qui indique une base client principalement active. Au-delà de 60 ans, le nombre de clients diminue progressivement. Les personnes âgées de plus de 80 ans sont très peu représentées. L'offre semble donc principalement attirer une clientèle jeune et adulte

In [79]:
```
# Catégories de livres achetés selon les tranches d'âge
# Définir les tranches d'âge
bins = [17, 24, 34, 49, 64, 120]
labels = ["17-24", "25-34", "35-49", "50-64", "65+"]

sales["tranche_age"] = pd.cut(sales["age"], bins=bins, labels=labels, right=True)

# Regroupement par tranche d'âge et catégorie
age_categ = sales.groupby(
    ["tranche_age", "categ"], observed=True
)["id_prod"].count().unstack(fill_value=0)


# Traçage du graphique en barres empilées
ax = age_categ.plot(
    kind="barh",
    stacked=True,
    figsize=(10,6),
    color=["#1f77b4", "#ff7f0e", "#2ca02c"],
    edgecolor="black"
)

# Affichage du graphique
plt.title("Catégories de livres achetés selon les tranches d'âge", fontsize=14, fontweight="bold")
plt.xlabel("Nombre de livres achetés")
plt.ylabel("Tranche d'âge")
plt.legend(title="Catégorie")
plt.grid(axis="x", linestyle="--", alpha=0.6)

plt.tight_layout()
plt.show()
```



In [80]:
```
# Chiffre d'affaires par tranche d'âge et par catégorie
# Les tranches d'âge
bins = [17, 24, 34, 49, 64, 120]
labels = ["17-24", "25-34", "35-49", "50-64", "65+"]

sales["tranche_age"] = pd.cut(sales["age"], bins=bins, labels=labels, right=True)

# Calcul du CA par tranche d'âge et par catégorie
ca_age_categ = sales.groupby(["tranche_age", "categ"], observed=True)["price"].sum().unstack(fill_value=0)
```

```
# Graphique en barres empilées
ax = ca_age_categ.plot(
    kind="barh",
    stacked=True,
    figsize=(10,6),
    color=["#1f77b4", "#ff7f0e", "#2ca02c"],  # adapte selon tes catégories
    edgecolor="black"
)

# Affichage du graphique
plt.title("Chiffre d'affaires par tranche d'âge et par catégorie", fontsize=14, fontweight="bold")
plt.xlabel("Chiffre d'affaires (€)")
plt.ylabel("Tranche d'âge")
plt.legend(title="Catégorie")
plt.grid(axis="x", linestyle="--", alpha=0.6)

plt.tight_layout()
plt.show()
```



- La tranche d'âge 35-49 ans génère le plus grand chiffre d'affaires, toutes catégories confondues.
- La catégorie 0 (bleue) domine clairement dans les tranches 35-49 et 50-64 ans.
- La catégorie 2 (verte) est majoritaire chez les 17-24 ans et significative chez les 25-34 ans.
- Les 65 ans et plus ont une contribution globale plus faible au chiffre d'affaires.
- Le chiffre d'affaires diminue globalement aux extrêmes d'âge (17-24 et 65+), avec une concentration sur les 25 à 64 ans

Nous allons maintenant analyser les prix et les paniers moyens selon l'âge des clients

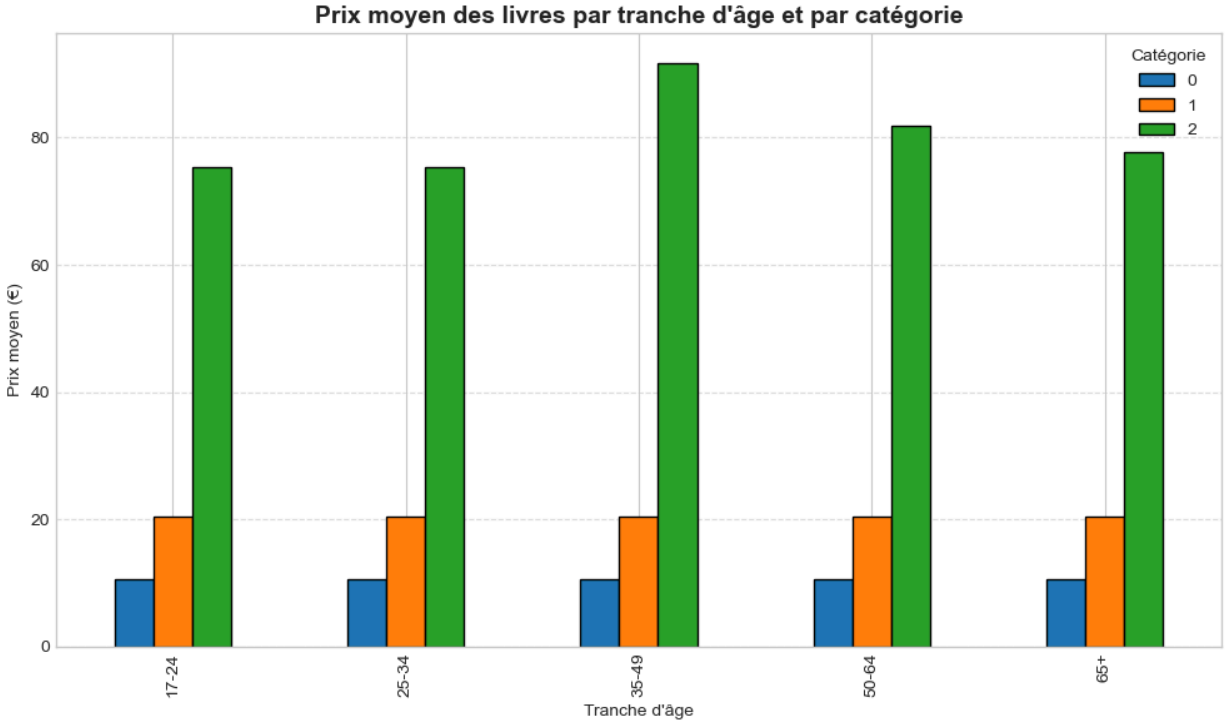
ANALYSE DES PRIX ET PANIER MOYEN SELON L'AGE

In [195…

```
#Prix moyen des livres par tranche d'âge et par catégorie
#Prix moyen des livres achetés et panier moyen
# Définition des tranches d'âge
bins = [17, 24, 34, 49, 64, 120]
labels = ["17-24", "25-34", "35-49", "50-64", "65+"]
sales["tranche_age"] = pd.cut(sales["age"], bins=bins, labels=labels, right=True)

# Panier moyen dépensé par tranche d'âge
ca_par_age = sales.groupby("tranche_age", observed=True)["price"].sum()
sessions_par_age = sales.groupby("tranche_age", observed=True)["session_id"].nunique()
panier_moyen = ca_par_age / sessions_par_age

print("Panier moyen dépensé par tranche d'âge (€):")
```

```
print(panier_moyen)

# Prix moyen des livres achetés par catégorie et tranche d'âge
prix_moyen = sales.groupby(["tranche_age", "categ"], observed=True)["price"].mean().unstack(fill_value=0)

print("\nPrix moyen des livres achetés (€):")
print(prix_moyen)

# Visualisation du prix moyen par tranche d'âge et catégorie
ax = prix_moyen.plot(
    kind="bar",
    figsize=(10,6),
    color=["#1f77b4", "#ff7f0e", "#2ca02c"],
    edgecolor="black"
)

#Affichage du graphique
plt.title("Prix moyen des livres par tranche d'âge et par catégorie", fontsize=14, fontweight="bold")
plt.xlabel("Tranche d'âge")
plt.ylabel("Prix moyen (€)")
plt.legend(title="Catégorie")
plt.grid(axis="y", linestyle="--", alpha=0.6)

plt.tight_layout()
plt.show()
```

```
Panier moyen dépensé par tranche d'âge (€):
tranche_age
17-24    70.305281
25-34    45.267646
35-49    32.036821
50-64    24.957446
65+      24.709863
dtype: float64

Prix moyen des livres achetés (€):
categ               0          1          2
tranche_age
17-24        10.680008  20.548919  75.359788
25-34        10.667031  20.541749  75.411687
35-49        10.616996  20.426650  91.766442
50-64        10.661247  20.518254  81.914684
65+          10.662923  20.476535  77.672504
```



Globalement, le panier moyen dépensé diminue avec l'âge : il est le plus élevé chez les 17-24 ans (~75 €) et baisse progressivement pour atteindre ~25 € chez les plus de 65 ans.

En revanche, le prix moyen des livres achetés reste assez stable dans les catégories 0 et 1 (autour de 10 € et 20 €). La catégorie 2 est nettement plus chère, avec un prix moyen autour de 75–90 €, quel que soit l'âge.

In [82]:
```
# Extraire les clients uniques
clients_uniques = sales.drop_duplicates(subset="client_id")

# Sélectionner les clients de moins de 20 ans
clients_moins_20 = clients_uniques[clients_uniques["age"] < 20]

# Afficher un aperçu
print(clients_moins_20[["client_id", "birth", "age"]].head(20))

# Vérifier combien ils sont
print(f"Nombre de clients de moins de 20 ans : {clients_moins_20.shape[0]}")

# Distribution des années de naissance (ça peut montrer un bug de saisie)
print(clients_moins_20['birth'].value_counts().head(10))
```

```
      client_id  birth  age
721       c_2998   2003   18
1356      c_3213   2003   18
2599       c_125   2002   19
2683      c_1046   2004   17
3233      c_3424   2003   18
3359      c_8413   2003   18
4424      c_3081   2004   17
6182      c_1418   2004   17
6348      c_1821   2003   18
6451      c_3338   2004   17
6793      c_1774   2004   17
7022      c_8357   2004   17
7592      c_6276   2004   17
7707      c_1934   2004   17
9414      c_5059   2004   17
9424      c_6845   2002   19
11616     c_1859   2003   18
12110     c_1561   2004   17
14022     c_4512   2003   18
15080     c_7795   2003   18
Nombre de clients de moins de 20 ans : 726
birth
2004    437
2003    145
2002    144
Name: count, dtype: int64
```

In [83]:
```
#Affichage du nombre de clients total
nb_clients_uniques = sales["client_id"].nunique()
print(f"Nombre total de clients uniques : {nb_clients_uniques}")
```

```
Nombre total de clients uniques : 8600
```

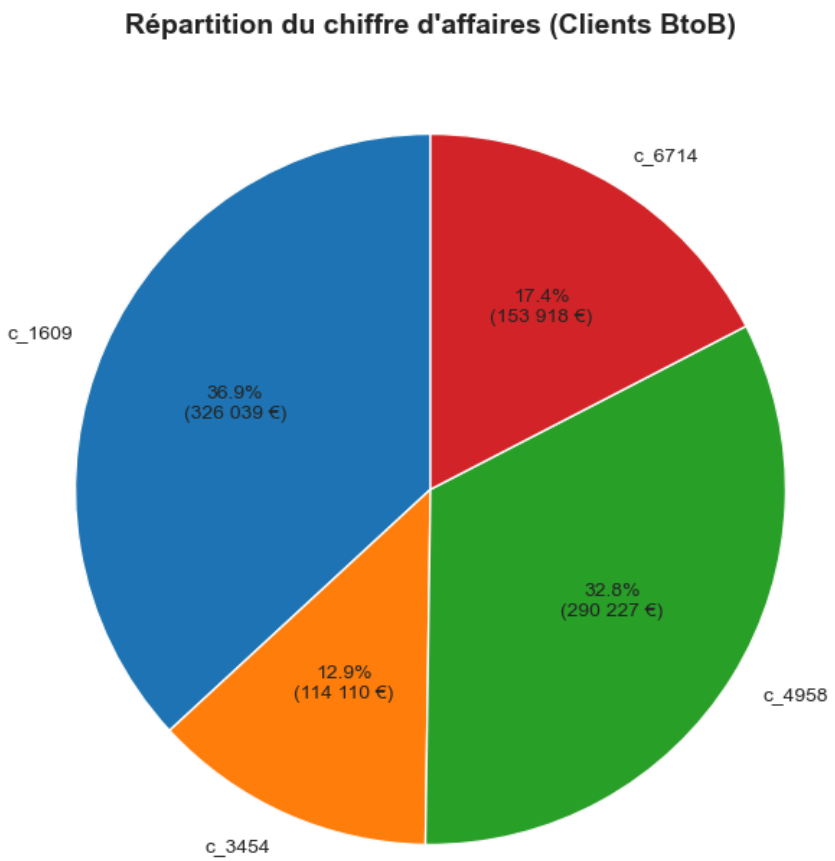
In [84]:
```
#Répartion selon l'âge
nb_clients_uniques= sales["client_id"].nunique()
sales["age"].describe()
```

Out[84]:
```
count    687534.000000
mean         43.844898
std          13.623882
min          17.000000
25%          35.000000
50%          42.000000
75%          51.000000
max          94.000000
Name: age, dtype: float64
```

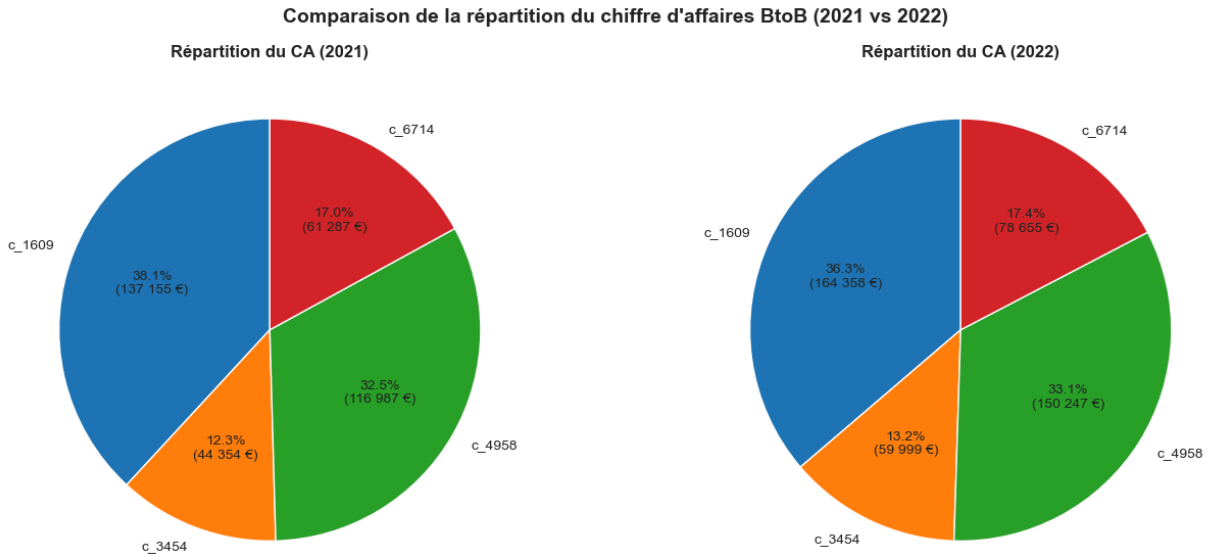
ANALYSE DES CLIENTS BTOB

In [85]:
```
# Sélection des clients BtoB
clients_selection = ["c_1609", "c_4958", "c_6714", "c_3454"]
sales_sel = sales[sales["client_id"].isin(clients_selection)].copy()

# CA total par client
ca_repartition = sales_sel.groupby("client_id")["price"].sum()

# Graphique camembert
plt.figure(figsize=(8,8))
plt.pie(
    ca_repartition,
    labels=ca_repartition.index,
    autopct=lambda p: f"{p:.1f}%\n({int(p*ca_repartition.sum()/100):,} €)".replace(",", " "),
    startangle=90,
    wedgeprops={"edgecolor":"white"}
)
```

```
plt.title("Répartition du chiffre d'affaires (Clients BtoB)", fontsize=14, fontweight="bold")
plt.show()
```



In [86]:
```
# Sélection des clients BtoB
clients_selection = ["c_1609", "c_4958", "c_6714", "c_3454"]
sales_sel = sales[sales["client_id"].isin(clients_selection)].copy()
sales_sel["date"] = pd.to_datetime(sales_sel["date"])
sales_sel["year"] = sales_sel["date"].dt.year

# CA total par client et par année
ca_repartition = sales_sel.groupby(["year", "client_id"])["price"].sum().unstack(fill_value=0)

# --- Graphiques côte à côte ---
fig, axes = plt.subplots(1, 2, figsize=(14,6))

# Camembert 2021
axes[0].pie(
    ca_repartition.loc[2021],
    labels=ca_repartition.columns,
    autopct=lambda p: f"{p:.1f}%\n({int(p*ca_repartition.loc[2021].sum()/100):,} €)".replace(",", " "),
    startangle=90,
    wedgeprops={"edgecolor":"white"}
)
axes[0].set_title("Répartition du CA (2021)", fontsize=12, fontweight="bold")

# Camembert 2022
axes[1].pie(
    ca_repartition.loc[2022],
    labels=ca_repartition.columns,
    autopct=lambda p: f"{p:.1f}%\n({int(p*ca_repartition.loc[2022].sum()/100):,} €)".replace(",", " "),
    startangle=90,
    wedgeprops={"edgecolor":"white"}
)
axes[1].set_title("Répartition du CA (2022)", fontsize=12, fontweight="bold")

plt.suptitle("Comparaison de la répartition du chiffre d'affaires BtoB (2021 vs 2022)", fontsize=14, fontweight="bold"
plt.tight_layout()
plt.show()
```



Étape 5 - Les corrélations (Tests statistiques)

In [88]:
```
#Suppression des clients BTOB
clients_BtoB = ["c_1609", "c_4958", "c_6714", "c_3454"]

sales = sales[~sales["client_id"].isin(clients_BtoB)]
```

Pour l'analyse des corrélations nous avons choisis de supprimer les clients BTOB

5.1 - Corrélation entre le genre d'un client et les catégories de livres achetés

Dans cette première analyse des corrélations nous allons voir s'il existe un lien entre le genre d'un client et la catégorie de livres achetés. On a 2 variables qualitatives: le sexe des clients et la catégorie

Le test adapté ici est le test de Chi-2

In [92]:
```
table_contingence = pd.crosstab(sales['sex'], sales['categ'])
print(table_contingence)
```

```
categ       0       1      2
sex
f      200793  115721  16980
m      186488  104884  15868
```

In [93]:
```
from scipy.stats import chi2_contingency

# Test du Chi²
chi2, p, dof, expected = chi2_contingency(table_contingence)

print("Statistique de Chi² :", round(chi2, 3))
print("Degrés de liberté :", dof)
print("p-value :", format(p, ".6f"))
```

```
Statistique de Chi² : 22.669
Degrés de liberté : 2
p-value : 0.000012
```

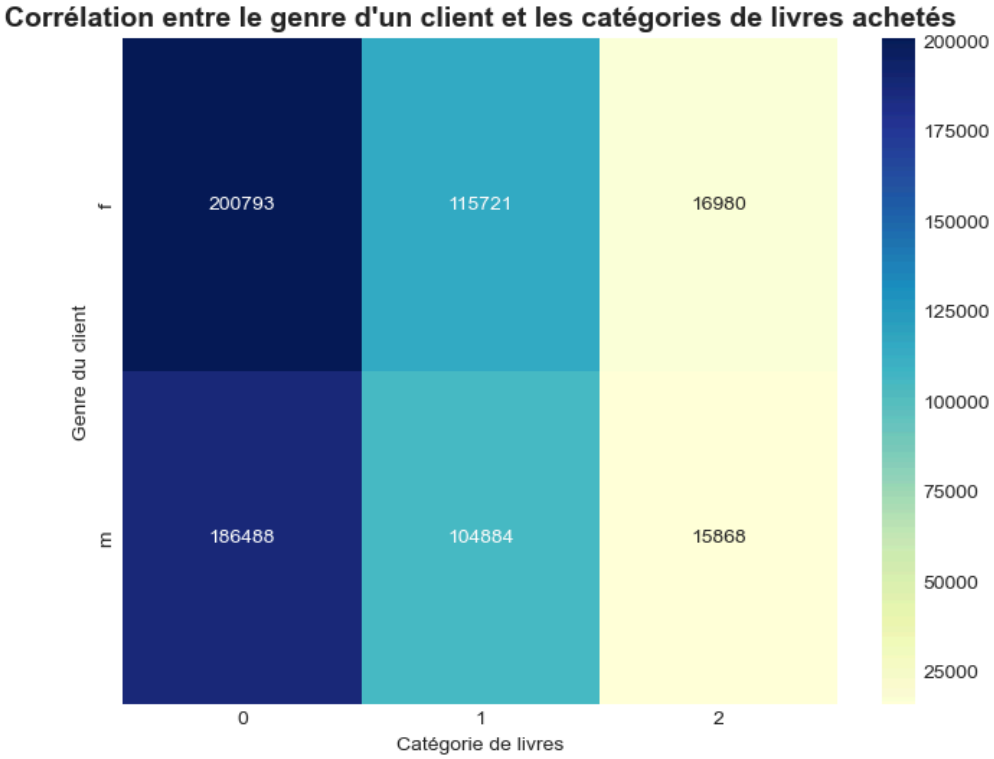
Résultats du test du Chi² d'indépendance

- **Statistique de Chi²** : 22.67
- **Degrés de liberté** : 2
- **p-value** : ≈ 0.00001

Comme la p-value est très inférieure à 0.05, on **rejette l'hypothèse d'indépendance**.

Conclusion : il existe une relation statistiquement significative entre le genre du client et les catégories de livres achetées.

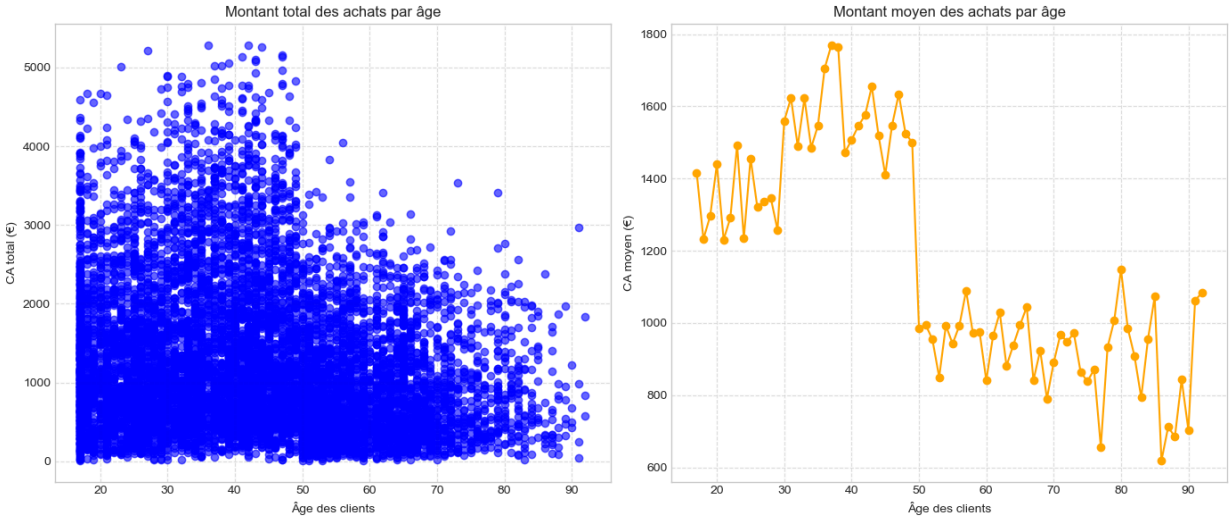
In [95]:
```
import seaborn as sns

#Affichage de la corrélation entre le genre d'un client et les catégories de livres achetés

# Table de contingence
table_contingence = pd.crosstab(sales['sex'], sales['categ'])

# Heatmap
plt.figure(figsize=(8,6))
sns.heatmap(table_contingence, annot=True, fmt="d", cmap="YlGnBu")

plt.title("Corrélation entre le genre d'un client et les catégories de livres achetés", fontsize=14, fontweight="bold"
plt.xlabel("Catégorie de livres")
plt.ylabel("Genre du client")
plt.show()
```



5.2 - Corrélation entre l'âge des clients et le montant total des achats

In [97]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Chiffre d'affaires total par client
df_age_ca = sales.groupby("client_id", as_index=False).agg({
    "age": "first",   # âge du client
    "price": "sum"    # total des achats
})
df_age_ca = df_age_ca.rename(columns={"price": "ca_total"})

# Montant moyen par âge
ca_moyen_par_age = df_age_ca.groupby("age", as_index=False)["ca_total"].mean()

fig, axes = plt.subplots(1, 2, figsize=(14, 6))

# 1. Affichage Montant total des achats par client
axes[0].scatter(df_age_ca["age"], df_age_ca["ca_total"], alpha=0.6, color="blue")
axes[0].set_title("Montant total des achats par âge")
axes[0].set_xlabel("Âge des clients")
axes[0].set_ylabel("CA total (€)")
axes[0].grid(True, linestyle="--", alpha=0.6)

# 2. Affichage Montant moyen des achats par âge
axes[1].plot(ca_moyen_par_age["age"], ca_moyen_par_age["ca_total"], marker="o", color="orange")
axes[1].set_title("Montant moyen des achats par âge")
axes[1].set_xlabel("Âge des clients")
axes[1].set_ylabel("CA moyen (€)")
axes[1].grid(True, linestyle="--", alpha=0.6)
```

```
plt.tight_layout()
plt.show()
```



Le test de Spearman serait plus approprié ici car :

- La relation entre l'âge et le montant total des achats n'est pas linéaire.
- Il y a des outliers visibles

In [99]:
```
from scipy.stats import spearmanr

# Calcul de la corrélation de Spearman
r_spearman, p_spearman = spearmanr(df_age_ca["age"], df_age_ca["ca_total"])

# Affichage du résultat
print(f"Corrélation de Spearman : r = {r_spearman:.3f}, p-value = {p_spearman:.6f}")
```

```
Corrélation de Spearman : r = -0.185, p-value = 0.000000
```

Le coefficient de Spearman r = -0.185 indique une corrélation monotone négative faible : plus l'âge augmente, moins le montant total des achats est élevé.

La p-value = 0.00 (inférieure à 0.05) signifie que cette relation est statistiquement significative. Elle n'est pas due au hasard

Conclusion: Il existe une relation négative faible et significative entre l'âge et le montant total des achats (l'âge explique peu la variation du montant total des achats)

5.3 - Corrélation entre l'âge et la fréquence d'achat

In [102…
```
sales_no_annee = sales.loc[:, sales.columns != "annee"]
sales_no_annee
```

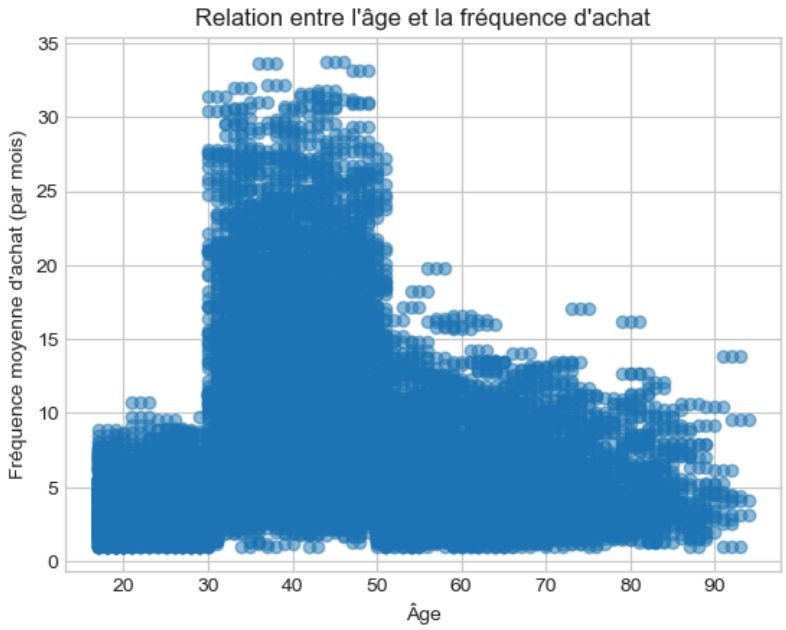
Out[102…

	client_id	sex	birth	id_prod	date	session_id	price	categ	year	month	age	tranche_age
0	c_4410	f	1967	1_483	2021-03-13 21:35:56	s_5913	15.99	1	2021	3	54	50-64
1	c_4410	f	1967	0_1111	2021-03-22 01:27:49	s_9707	19.99	0	2021	3	54	50-64
2	c_4410	f	1967	1_385	2021-03-22 01:40:23	s_9707	25.99	1	2021	3	54	50-64
3	c_4410	f	1967	0_1455	2021-03-22 14:29:25	s_9942	8.99	0	2021	3	54	50-64
4	c_4410	f	1967	0_1420	2021-03-22 22:31:26	s_10092	11.53	0	2021	3	54	50-64
...	...	...	...	...	...	...	...	...	...	...	...	...
687529	c_84	f	1982	0_1472	2022-05-14 00:24:49	s_208110	12.49	0	2022	5	40	35-49
687530	c_84	f	1982	0_1438	2022-05-29 06:11:50	s_215697	9.31	0	2022	5	40	35-49
687531	c_84	f	1982	1_459	2022-12-17 00:16:57	s_313173	15.99	1	2022	12	40	35-49
687532	c_84	f	1982	0_1104	2022-12-17 00:24:14	s_313173	13.21	0	2022	12	40	35-49
687533	c_84	f	1982	1_688	2022-12-17 00:36:24	s_313173	18.19	1	2022	12	40	35-49

640734 rows × 12 columns

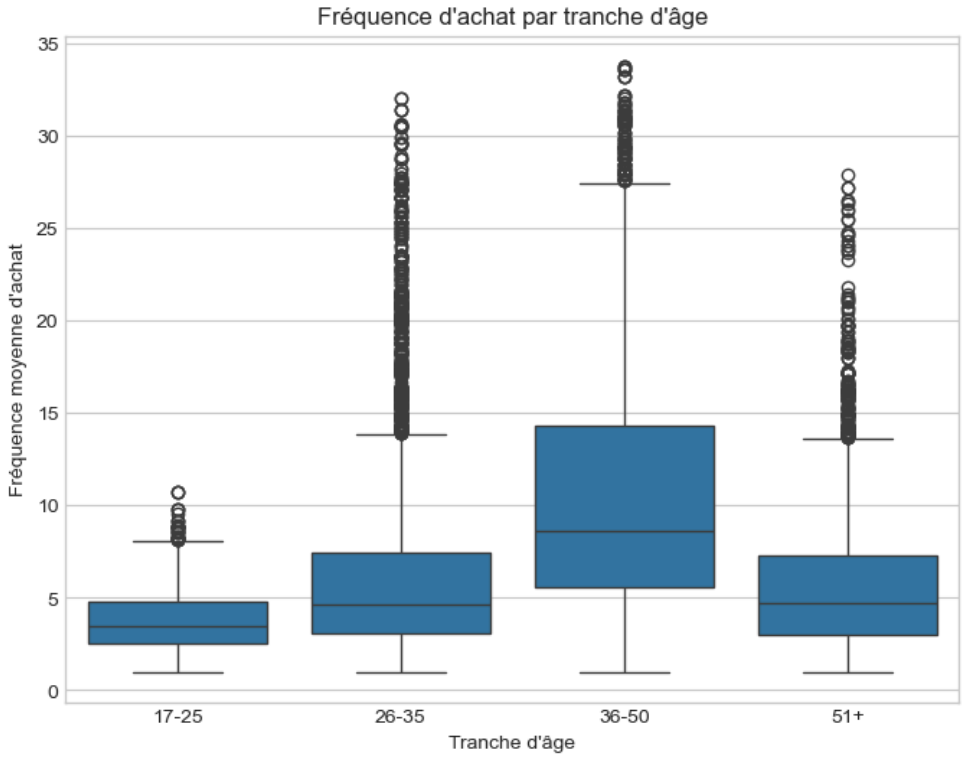
In [103…

```
# Compter le nombre d'achats par client et par mois

# Calcul du nombre d'achats par client et par mois
freq_mensuelle = sales.groupby(['client_id', 'month']).size().reset_index(name='nb_achats')

# Calcul de la fréquence moyenne par client (moyenne des nb_achats par mois)
freq_client = freq_mensuelle.groupby('client_id')['nb_achats'].mean().reset_index(name='freq_mensuelle')

# Joindre avec l'âge du client
data_corr = pd.merge(sales[['client_id', 'age']].drop_duplicates(), freq_client, on='client_id')

# Scatter plot âge vs fréquence
plt.scatter(data_corr['age'], data_corr['freq_mensuelle'], alpha=0.5)
plt.xlabel("Âge")
plt.ylabel("Fréquence moyenne d'achat (par mois)")
plt.title("Relation entre l'âge et la fréquence d'achat")
plt.show()
```



In [104…

```
import seaborn as sns

# Créer des tranches d'âge
bins = [17, 25, 35, 50, 100]
labels = ["17-25", "26-35", "36-50", "51+"]
data_corr['tranche_age'] = pd.cut(data_corr['age'], bins=bins, labels=labels, right=False)

plt.figure(figsize=(8,6))
sns.boxplot(x="tranche_age", y="freq_mensuelle", data=data_corr)
plt.title("Fréquence d'achat par tranche d'âge")
plt.xlabel("Tranche d'âge")
plt.ylabel("Fréquence moyenne d'achat")
plt.show()
```



Le test de Spearman est plus adapté car il détecte des relations monotones, pas forcément linéaires.

Car La relation n'a pas l'air linéaire, les jeunes et les plus âgés achètent moins, les 30–50 ans un peu plus puis ça redescend

In [106…
```
from scipy.stats import spearmanr

# Test de Spearman : âge vs fréquence d'achat
rho, p_spear = spearmanr(data_corr['age'], data_corr['freq_mensuelle'])

print("Corrélation de Spearman :", format(rho, ".4f"))
print("p-value :", format(p_spear, ".6f"))
```

```
Corrélation de Spearman : 0.0899
p-value : 0.000000
```

In [107…
```
from scipy.stats import f_oneway

# Préparatoion des groupes pour l'ANOVA
groupes = [data_corr[data_corr['tranche_age'] == g]['freq_mensuelle'] for g in labels]

# Test ANOVA
anova_stat, p_anova = f_oneway(*groupes)

print("Statistique ANOVA :", format(anova_stat, ".4f"))
print("p-value :", format(p_anova, ".6f"))
```

```
Statistique ANOVA : 2198.2394
p-value : 0.000000
```

- L'ANOVA compare les moyennes de fréquence d'achat entre les tranches d'âge.
- La statistique obtenue est très élevée (2198.24), ce qui montre une grande différence entre groupes.
- La p-value est quasiment nulle (< 0.000001).

=> Cela signifie que l'âge a un impact significatif sur la fréquence d'achat.

- Les moyennes de fréquence d'achat ne sont pas identiques selon les tranches d'âge

Conclusion : L'ANOVA met en évidence des différences significatives de fréquence d'achat selon les tranches d'âge. On rejette donc l'hypothèse d'égalité des moyennes : l'âge influence de manière significative la fréquence d'achat, et toutes les tranches d'âge n'achètent pas avec la même régularité

In [109…
```
from statsmodels.stats.multicomp import pairwise_tukeyhsd

# Test de Tukey
tukey = pairwise_tukeyhsd(endog=data_corr['freq_mensuelle'],
                          groups=data_corr['tranche_age'],
                          alpha=0.05)

print(tukey)
```

```
Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================
group1 group2 meandiff p-adj  lower   upper  reject
-----------------------------------------------------
 17-25  26-35   2.8234   0.0  2.5428   3.104   True
 17-25  36-50   6.8752   0.0  6.6207  7.1297   True
 17-25    51+   1.8039   0.0  1.5557  2.0522   True
 26-35  36-50   4.0518   0.0  3.8166  4.2871   True
 26-35    51+  -1.0195   0.0 -1.2479  -0.791   True
 36-50    51+  -5.0713   0.0 -5.2668 -4.8757   True
-----------------------------------------------------
```

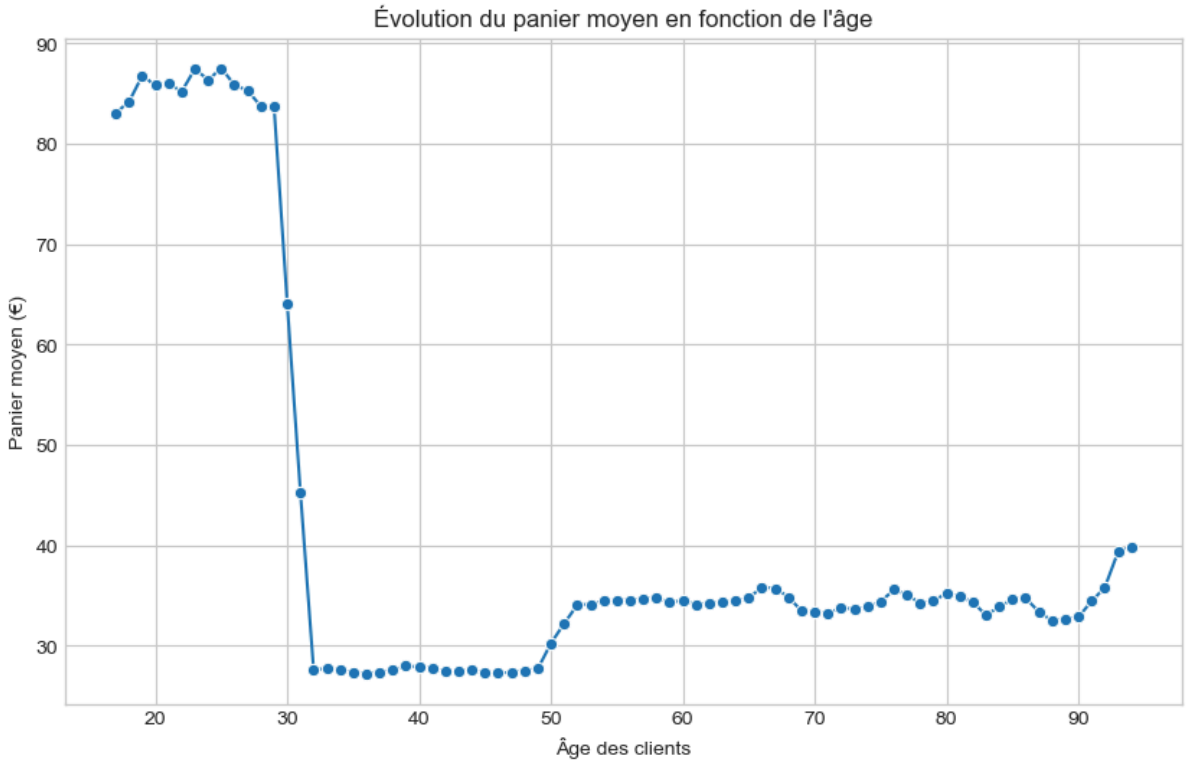
5.4 - Corrélation entre l'âge des clients et la taille du panier moyen

Ici nous avons:

- Âge → variable quantitative continue
- Panier moyen → variable quantitative continue (montant moyen des achats d'un client

Pour le choix du test on va vérifier si la relation est linéaire

In [112…
```
# 1. Calcul du panier moyen par client
depenses_totales = sales.groupby('client_id')['price'].sum()
nb_commandes = sales.groupby('client_id')['id_prod'].nunique()
panier_moyen = (depenses_totales / nb_commandes).reset_index(name='panier_moyen')

# 2. Ajouter l'âge du client
data_corr = pd.merge(sales[['client_id', 'age']].drop_duplicates(), panier_moyen, on='client_id')

# 3. Calcul du panier moyen par âge exact
panier_par_age = data_corr.groupby('age')['panier_moyen'].mean().reset_index()

# 4. Courbe lissée
plt.figure(figsize=(10,6))
sns.lineplot(x="age", y="panier_moyen", data=panier_par_age, marker="o")
plt.xlabel("Âge des clients")
plt.ylabel("Panier moyen (€)")
plt.title("Évolution du panier moyen en fonction de l'âge")
plt.grid(True)
plt.show()
```



La courbe montre clairement que la relation n'est pas linéaire (forte chute après 30 ans, puis stabilisation, légère remontée après 60 ans

In [114…
```
from scipy.stats import f_oneway

# Définir les tranches d'âge (
bins = [17, 25, 35, 50, 65, 100]
labels = ["17-24", "25-34", "35-49", "50-64", "65+"]

# Ajouter une colonne tranche d'âge
data_corr['tranche_age'] = pd.cut(data_corr['age'], bins=bins, labels=labels, right=False)

# Créer les groupes pour l'ANOVA
groupes = [data_corr[data_corr['tranche_age'] == g]['panier_moyen'] for g in labels]

# Test ANOVA
anova_stat, p_value = f_oneway(*groupes)

print("Statistique ANOVA :", format(anova_stat, ".4f"))
print("p-value :", format(p_value, ".6f"))
```

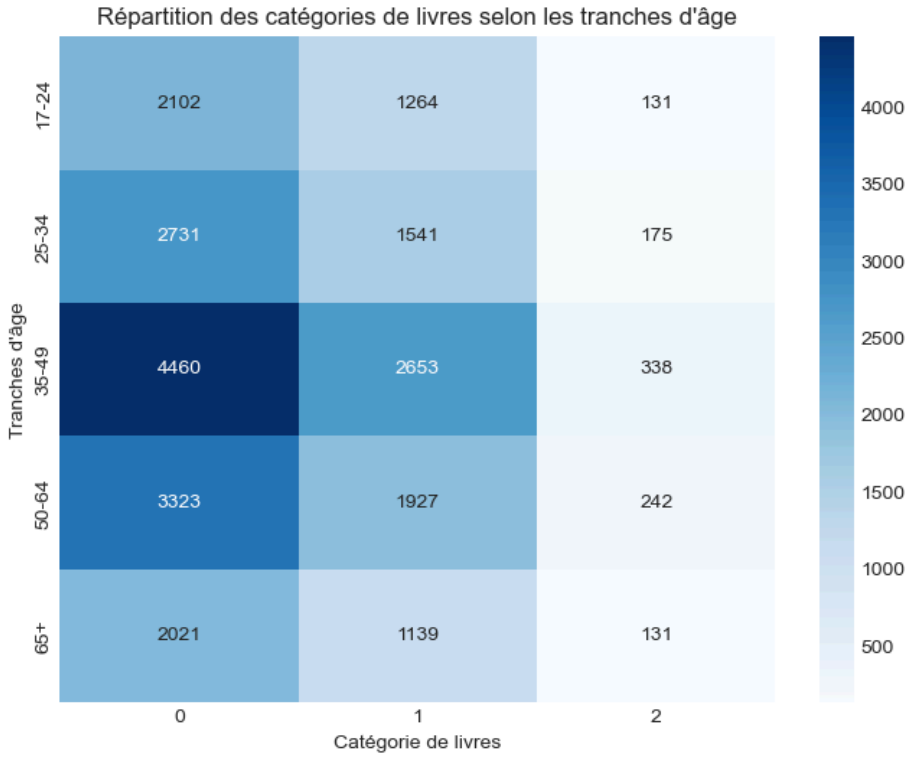
```
Statistique ANOVA : 4358.1033
p-value : 0.000000
```

- On rejette très fortement l'hypothèse d'égalité des moyennes
- Le panier moyen diffère significativement selon les tranches d'âge

Conclusion l'âge influence clairement le panier moyen

5.5 - Corrélation entre l'âge et la catégorie de livres achetés

In [117…
```
# Table de contingence
contingency_table = pd.crosstab(data_corr['tranche_age'], sales['categ'])

# Heatmap
plt.figure(figsize=(8,6))
sns.heatmap(contingency_table, annot=True, fmt="d", cmap="Blues")
plt.title("Répartition des catégories de livres selon les tranches d'âge")
plt.ylabel("Tranches d'âge")
plt.xlabel("Catégorie de livres")
plt.show()
```



In [118…

```python
import pandas as pd
from scipy.stats import chi2_contingency

# Supposons que ton DataFrame s'appelle sales

# 1. Créer des tranches d'âges
bins = [17, 25, 35, 50, 65, 100]  # bornes des tranches
labels = ['17-25', '26-35', '36-50', '51-65', '65+']
sales['age_group'] = pd.cut(sales['age'], bins=bins, labels=labels, right=False)

# 2. Créer une table de contingence (âge groupé x catégorie)
contingency_table = pd.crosstab(sales['age_group'], sales['categ'])
print("Table de contingence :\n", contingency_table)

# 3. Test du Chi-2
chi2, p, dof, expected = chi2_contingency(contingency_table)

print("\nStatistique Chi-2 :", chi2)
print("Degrés de liberté :", dof)
print("P-value :", p)

if p < 0.05:
    print("→ Les catégories de livres dépendent significativement des tranches d'âge.")
else:
    print("→ Pas de dépendance significative entre les catégories de livres et les tranches d'âge.")
```

```
Table de contingence :
 categ           0      1      2
age_group
17-25       10008  17731  17533
26-35       69601  31899  12677
36-50      228458  74089   1099
51-65       52915  61655    980
65+         26299  35231    559

Statistique Chi-2 : 188433.8753665135
Degrés de liberté : 8
P-value : 0.0
→ Les catégories de livres dépendent significativement des tranches d'âge.
```

Ici la valeur du Chi-2 est grande, plus l'écart entre observé et attendu est fort, donc plus il est probable que les deux variables soient liées. La p-value étant très largement inférieure à 0.05, on rejette l'hypothèse d'indépendance.

Conclusion : la catégorie de livres choisie dépend significativement de la tranche d'âge.