

Tests statistiques - Analyse des comportements clients

Projet 9 - Analysez les ventes d'une librairie avec Python

IMPORTANT : Analyse SANS les clients BtoB

Ce notebook contient les analyses statistiques demandées par Julie, en excluant les 4 clients BtoB identifiés dans le premier notebook.

Objectifs de l'analyse

Les 5 corrélations à tester :

1. **Genre ↔ Catégories de livres achetés**
2. **Âge ↔ Montant total des achats**
3. **Âge ↔ Fréquence d'achat**
4. **Âge ↔ Taille du panier moyen**
5. **Âge ↔ Catégorie des livres achetés**

Tests statistiques à utiliser :

- **Test du χ^2 (Khi-deux)** : Variables qualitatives (Genre vs Catégorie, Âge vs Catégorie)
 - **ANOVA ou Test de Student** : Variable qualitative vs quantitative
 - **Test de Pearson ou Spearman** : Variables quantitatives (Âge vs CA, etc.)
-

1. Import des bibliothèques

In [77]:

```
# Manipulation de données
import pandas as pd
import numpy as np

# Visualisation
import matplotlib.pyplot as plt
import seaborn as sns

# Tests statistiques
from scipy import stats
from scipy.stats import chi2_contingency, pearsonr, spearmanr, f_oneway, ttest_ind
```

```
# Configuration des graphiques
plt.style.use('seaborn-v0_8-darkgrid')
sns.set_palette("husl")
plt.rcParams['figure.figsize'] = (12, 6)
plt.rcParams['font.size'] = 10

# Affichage complet des dataframes
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', 100)

# Ignorer les warnings
import warnings
warnings.filterwarnings('ignore')

print("Bibliothèques importées avec succès")
```

Bibliothèques importées avec succès

2. Chargement des données

2.1 Chargement des fichiers bruts

```
In [78]: # Chargement du fichier clients
df_customers = pd.read_csv('customers.csv', sep=';')

# Chargement du fichier produits
df_products = pd.read_csv('products.csv', sep=';')

# Chargement du fichier transactions
df_trans_raw = pd.read_excel('Transactions.xlsx', header=None)
df_transactions = df_trans_raw[0].str.split(';', expand=True)
df_transactions.columns = df_transactions.iloc[0]
df_transactions = df_transactions.iloc[1:].reset_index(drop=True)

# Conversion de la date
df_transactions['date'] = pd.to_datetime(df_transactions['date'])

print("Fichiers chargés")
print(f" - Clients : {len(df_customers)}")
print(f" - Produits : {len(df_products)}")
print(f" - Transactions : {len(df_transactions)}")
```

Fichiers chargés

- Clients : 8621
- Produits : 3286
- Transactions : 1048575

2.2 Calcul de l'âge et création du dataset complet

```
In [79]: # Calcul de l'âge
annee_actuelle = 2024
df_customers['age'] = annee_actuelle - df_customers['birth']

# Fusion des données
```

```

df_complete = df_transactions.merge(df_products, on='id_prod', how='left')
df_complete = df_complete.merge(df_customers, on='client_id', how='left')
df_complete['ca'] = df_complete['price']

print(f" Dataset complet créé : {len(df_complete)} lignes")
print(f"   Colonnes : {df_complete.columns.tolist()}")

```

Dataset complet créé : 1048575 lignes
Colonnes : ['id_prod', 'date', 'session_id', 'client_id', 'price', 'categ', 'sex', 'birth', 'age', 'ca']

2.3 Identification et exclusion des clients BtoB

Nous identifions les 4 clients avec le CA le plus élevé comme étant des clients BtoB professionnels.

```

In [80]: # Calcul du CA par client
client_stats = df_complete.groupby('client_id').agg({
    'ca': ['sum', 'count', 'mean'],
    'session_id': 'nunique'
}).reset_index()

client_stats.columns = ['client_id', 'ca_total', 'nb_achats', 'panier_moyen', 'nb_sessions']
client_stats = client_stats.sort_values('ca_total', ascending=False).reset_index(drop=True)

# Identification des 4 clients BtoB (CA le plus élevé)
clients_btob = client_stats.nlargest(4, 'ca_total')
btob_ids = clients_btob['client_id'].tolist()

print(" Clients BtoB identifiés (à exclure de l'analyse) :")
print(clients_btob[['client_id', 'ca_total', 'nb_achats', 'panier_moyen']])
print(f"\n IDs des clients BtoB : {btob_ids}")

Clients BtoB identifiés (à exclure de l'analyse) :
  client_id  ca_total  nb_achats  panier_moyen
0   c_1609    326039.89      25586    12.742902
1   c_4958    290227.03      5222     55.577754
2   c_6714    153918.60      9199     16.732101
3   c_3454    114110.57      6793     16.798259

IDs des clients BtoB : ['c_1609', 'c_4958', 'c_6714', 'c_3454']

```

```

In [81]: # EXCLUSION DES CLIENTS BtoB
df_btoc = df_complete[~df_complete['client_id'].isin(btoc_ids)].copy()

print(" Dataset BtoC créé (SANS les clients BtoB)")
print(f"   - Transactions AVANT exclusion : {len(df_complete)}")
print(f"   - Transactions APRÈS exclusion : {len(df_btoc)}")
print(f"   - Transactions supprimées : {len(df_complete) - len(df_btoc)}")

# Nombre de clients BtoC
nb_clients_btoc = df_btoc['client_id'].nunique()
print(f"\n Nombre de clients BtoC : {nb_clients_btoc}")

```

Dataset BtoC créé (SANS les clients BtoB)

- Transactions AVANT exclusion : 1048575
- Transactions APRÈS exclusion : 1001775
- Transactions supprimées : 46800

Nombre de clients BtoC : 8597

2.4 Calcul des variables nécessaires pour les tests

Nous devons calculer pour chaque client BtoC :

- Montant total des achats
- Fréquence d'achat (nombre de transactions)
- Panier moyen
- Catégorie préférée

```
In [82]: # Agrégation par client pour obtenir les métriques nécessaires
client_analysis = df_btoc.groupby('client_id').agg({
    'ca': ['sum', 'mean', 'count'], # Montant total, panier moyen, fréquence
    'categ': lambda x: x.mode()[0] if len(x.mode()) > 0 else x.iloc[0], # Catégorie
    'sex': 'first',
    'age': 'first'
}).reset_index()

# Renommer les colonnes
client_analysis.columns = ['client_id', 'montant_total', 'panier_moyen', 'frequence',
                           'categorie_preferee', 'sexe', 'age']

print("Variables calculées par client BtoC :")
print(client_analysis.head(10))

print("\n Statistiques descriptives :")
print(client_analysis[['montant_total', 'panier_moyen', 'frequence_achat', 'age']]).
```

Variables calculées par client BtoC :

	client_id	montant_total	panier_moyen	frequence_achat	categorie_preferee	\
0		0.00	NaN	0		NaN
1	c_1	629.02	14.628372	43		0.0
2	c_10	1353.60	23.337931	58		1.0
3	c_100	254.85	31.856250	8		1.0
4	c_1000	2291.88	18.189524	126		1.0
5	c_1001	1823.85	17.707282	103		0.0
6	c_1002	436.54	11.798378	37		0.0
7	c_1003	1230.83	11.611604	106		0.0
8	c_1004	1077.83	15.850441	68		0.0
9	c_1005	844.93	14.320847	59		1.0

	sexe	age
0	None	NaN
1	m	69.0
2	m	68.0
3	m	32.0
4	f	58.0
5	m	42.0
6	f	42.0
7	m	42.0
8	m	51.0
9	f	83.0

Statistiques descriptives :

	montant_total	panier_moyen	frequence_achat	age
count	8597.000000	8596.000000	8597.000000	8596.000000
mean	1296.192510	21.926199	74.529952	45.739646
std	958.450775	14.634385	68.075096	16.910542
min	0.000000	4.150000	0.000000	20.000000
25%	562.570000	13.472771	28.000000	32.000000
50%	1045.520000	15.928454	52.000000	45.000000
75%	1795.930000	21.961065	97.000000	58.000000
max	5285.820000	181.990000	405.000000	95.000000

3. Vérification des données pour les tests

Avant de réaliser les tests statistiques, vérifions la qualité de nos données.

```
In [83]: # Vérification des valeurs manquantes
print(" Vérification des valeurs manquantes :")
print(client_analysis.isnull().sum())

# Distribution des variables
print("\n Répartition du sexe :")
print(client_analysis['sexe'].value_counts())

print("\n Répartition des catégories préférées :")
print(client_analysis['categorie_preferee'].value_counts())

print("\n Statistiques d'âge :")
print(client_analysis['age'].describe())
```

Vérification des valeurs manquantes :

```
client_id          0
montant_total      0
panier_moyen      1
fréquence_achat    0
catégorie_preferee 1
sexe              1
age                1
dtype: int64
```

Répartition du sexe :

```
sexe
f    4478
m    4118
Name: count, dtype: int64
```

Répartition des catégories préférées :

```
catégorie_preferee
0.0    4626
1.0    2993
2.0    977
Name: count, dtype: int64
```

Statistiques d'âge :

```
count    8596.000000
mean     45.739646
std      16.910542
min     20.000000
25%    32.000000
50%    45.000000
75%    58.000000
max     95.000000
Name: age, dtype: float64
```

```
In [84]: # Visualisation de la distribution des variables clés
fig, axes = plt.subplots(2, 3, figsize=(16, 10))

# Distribution de l'âge
axes[0, 0].hist(client_analysis['age'], bins=30, edgecolor='black', alpha=0.7, color='blue')
axes[0, 0].set_xlabel('Âge')
axes[0, 0].set_ylabel('Nombre de clients')
axes[0, 0].set_title('Distribution de l\'âge', fontweight='bold')
axes[0, 0].axvline(client_analysis['age'].mean(), color='red', linestyle='--',
                    label=f'Moyenne: {client_analysis["age"].mean():.1f} ans')
axes[0, 0].legend()

# Distribution du montant total
axes[0, 1].hist(client_analysis['montant_total'], bins=50, edgecolor='black', alpha=0.7, color='blue')
axes[0, 1].set_xlabel('Montant total (€)')
axes[0, 1].set_ylabel('Nombre de clients')
axes[0, 1].set_title('Distribution du montant total', fontweight='bold')
axes[0, 1].axvline(client_analysis['montant_total'].mean(), color='red', linestyle='--',
                    label=f'Moyenne: {client_analysis["montant_total"].mean():.2f} €')
axes[0, 1].legend()

# Distribution de la fréquence d'achat
```

```

axes[0, 2].hist(client_analysis['fréquence_achat'], bins=30, edgecolor='black', alpha=0.8)
axes[0, 2].set_xlabel('Fréquence d\'achat')
axes[0, 2].set_ylabel('Nombre de clients')
axes[0, 2].set_title('Distribution de la fréquence d\'achat', fontweight='bold')
axes[0, 2].axvline(client_analysis['fréquence_achat'].mean(), color='red', linestyle='dashed', label=f'Moyenne: {client_analysis["fréquence_achat"].mean():.1f}')
axes[0, 2].legend()

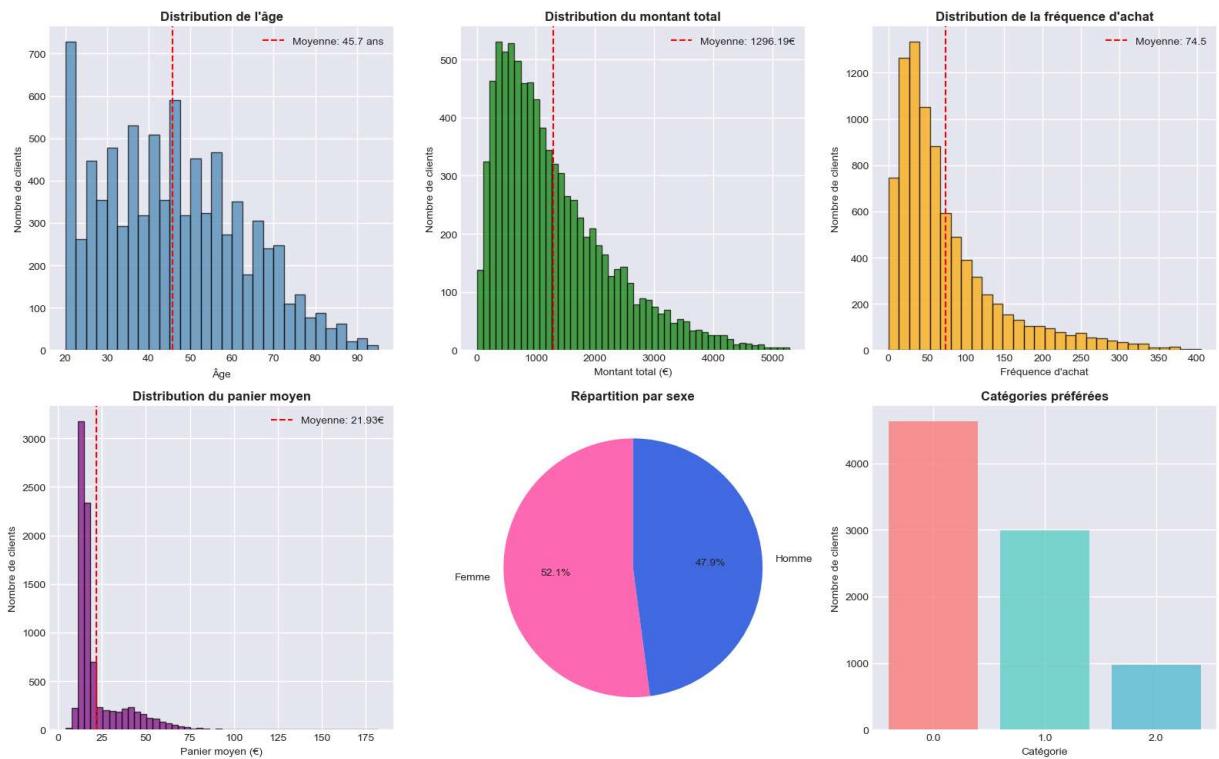
# Distribution du panier moyen
axes[1, 0].hist(client_analysis['panier_moyen'], bins=50, edgecolor='black', alpha=0.8)
axes[1, 0].set_xlabel('Panier moyen (€)')
axes[1, 0].set_ylabel('Nombre de clients')
axes[1, 0].set_title('Distribution du panier moyen', fontweight='bold')
axes[1, 0].axvline(client_analysis['panier_moyen'].mean(), color='red', linestyle='dashed', label=f'Moyenne: {client_analysis["panier_moyen"].mean():.2f}€')
axes[1, 0].legend()

# Répartition par sexe
sexe_counts = client_analysis['sexe'].value_counts()
axes[1, 1].pie(sexe_counts, labels=['Femme', 'Homme'] if sexe_counts.index[0] == 'Femme' else ['Homme', 'Femme'], autopct='%.1f%%', colors=['#FF69B4', '#4169E1'], startangle=90)
axes[1, 1].set_title('Répartition par sexe', fontweight='bold')

# Répartition par catégorie préférée
cat_counts = client_analysis['categorie_preferee'].value_counts()
axes[1, 2].bar(cat_counts.index.astype(str), cat_counts.values, color=['#FF6B6B', '#4169E1'])
axes[1, 2].set_xlabel('Catégorie')
axes[1, 2].set_ylabel('Nombre de clients')
axes[1, 2].set_title('Catégories préférées', fontweight='bold')

plt.tight_layout()
plt.show()

```



4. TEST 1 : Genre ↔ Catégories de livres achetés

Type de variables :

- **Genre** : Variable qualitative (binaire : m/f)
- **Catégorie** : Variable qualitative (0, 1, 2)

Test approprié : Test du χ^2 (Khi-deux)

Hypothèses :

- H0 (hypothèse nulle) : Il n'y a pas de lien entre le genre et la catégorie de livres achetés
- H1 (hypothèse alternative) : Il existe un lien entre le genre et la catégorie de livres achetés

```
In [85]: # Création du tableau de contingence
contingency_table = pd.crosstab(client_analysis['sexe'], client_analysis['categorie'])

print("Tableau de contingence : Genre vs Catégorie préférée")
print(contingency_table)

# Ajout des totaux
print("\nAvec totaux :")
contingency_with_totals = pd.crosstab(client_analysis['sexe'], client_analysis['catégorie'],
                                         margins=True, margins_name='Total')
print(contingency_with_totals)
```

Tableau de contingence : Genre vs Catégorie préférée

catégorie_preferee	0.0	1.0	2.0
sexes			
f	2370	1608	500
m	2256	1385	477

Avec totaux :

catégorie_preferee	0.0	1.0	2.0	Total
sexes				
f	2370	1608	500	4478
m	2256	1385	477	4118
Total	4626	2993	977	8596

```
In [86]: # Test du  $\chi^2$  (Khi-deux)
chi2_stat, p_value, dof, expected_freq = chi2_contingency(contingency_table)

print("TEST DU  $\chi^2$  : GENRE vs CATÉGORIE")
print("=*70)
print(f"Statistique  $\chi^2$  : {chi2_stat:.4f}")
print(f"p-value : {p_value:.6f}")
print(f"Degrés de liberté : {dof}")
print("\nFréquences attendues (sous H0) :")
print(pd.DataFrame(expected_freq,
                   index=contingency_table.index,
```

```

        columns=contingency_table.columns))

# Interprétation
alpha = 0.05

print("INTERPRÉTATION (seuil α = 0.05) :")

if p_value < alpha:
    print(f" p-value ({p_value:.6f}) < α ({alpha})")
    print(" On REJETTE H0")
    print(" Conclusion : Il existe un lien significatif entre le genre et la catégorie")
else:
    print(f" p-value ({p_value:.6f}) ≥ α ({alpha})")
    print(" On NE REJETTE PAS H0")
    print(" Conclusion : Pas de lien significatif entre le genre et la catégorie d")

```

TEST DU χ^2 : GENRE vs CATÉGORIE

=====

Statistique χ^2 : 4.8977

p-value : 0.086393

Degrés de liberté : 2

Fréquences attendues (sous H0) :

catégorie préférée	0.0	1.0	2.0
--------------------	-----	-----	-----

sexes

f	2409.868311	1559.173336	508.958353
m	2216.131689	1433.826664	468.041647

INTERPRÉTATION (seuil α = 0.05) :

p-value (0.086393) ≥ α (0.05)

On NE REJETTE PAS H0

Conclusion : Pas de lien significatif entre le genre et la catégorie de livres achetés.

```

In [87]: # Visualisation bivarée : Genre vs Catégorie
fig, axes = plt.subplots(1, 2, figsize=(14, 5))

# Graphique en barres groupées
contingency_table.T.plot(kind='bar', ax=axes[0], color=['#FF69B4', '#4169E1'], alpha=0.3)
axes[0].set_xlabel('Catégorie de livres')
axes[0].set_ylabel('Nombre de clients')
axes[0].set_title('Distribution des catégories par genre', fontweight='bold')
axes[0].legend(title='Genre', labels=['Femme', 'Homme'])
axes[0].set_xticklabels(axes[0].get_xticklabels(), rotation=0)
axes[0].grid(True, alpha=0.3, axis='y')

# Graphique en barres empilées (pourcentages)
contingency_pct = contingency_table.div(contingency_table.sum(axis=1), axis=0) * 100
contingency_pct.T.plot(kind='bar', stacked=True, ax=axes[1], color=['#FF69B4', '#4169E1'], alpha=0.3)
axes[1].set_xlabel('Catégorie de livres')
axes[1].set_ylabel('Pourcentage (%)')
axes[1].set_title('Répartition en % des catégories par genre', fontweight='bold')
axes[1].legend(title='Genre', labels=['Femme', 'Homme'])
axes[1].set_xticklabels(axes[1].get_xticklabels(), rotation=0)
axes[1].grid(True, alpha=0.3, axis='y')

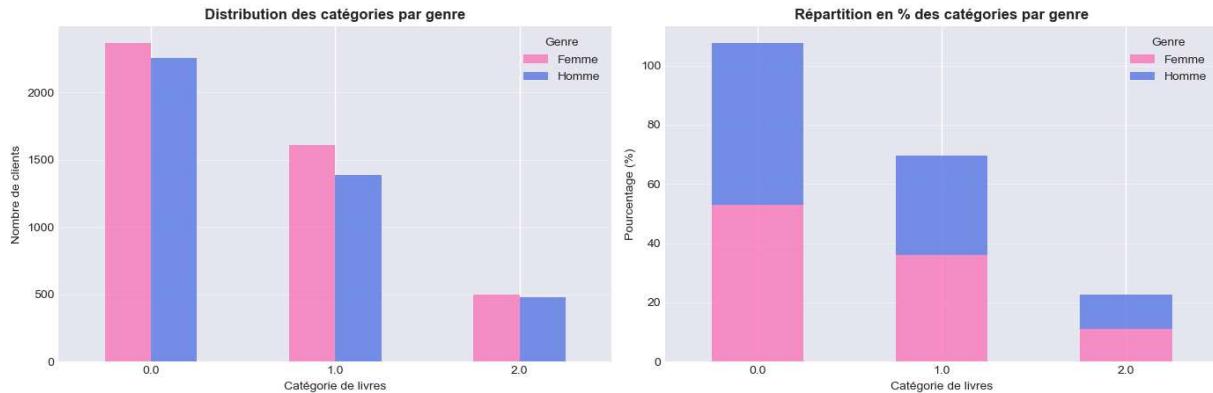
plt.suptitle(f'TEST  $\chi^2$  : Genre vs Catégorie ( $\chi^2={chi2_stat:.2f}$ , p={p_value:.4f})', fontweight='bold')

```

```

        fontsize=14, fontweight='bold', y=1.02)
plt.tight_layout()
plt.show()
```

TEST χ^2 : Genre vs Catégorie ($\chi^2=4.90$, $p=0.0864$)



5. TEST 2 : Âge ↔ Montant total des achats

Type de variables :

- **Âge** : Variable quantitative continue
- **Montant total** : Variable quantitative continue

Test approprié : **Test de corrélation de Pearson (ou Spearman)**

Hypothèses :

- H0 : Il n'y a pas de corrélation entre l'âge et le montant total des achats ($\rho = 0$)
- H1 : Il existe une corrélation entre l'âge et le montant total des achats ($\rho \neq 0$)

Diagnostic automatique avant corrélations

Avant de calculer Pearson/Spearman, vérifions la présence de NaN, de valeurs constantes, et n'exécutons les tests que si les conditions sont remplies.

In [88]: `# Diagnostic détaillé pour corrélations (affiche exemples de lignes fautives)`

```

def safe_corr_series(x, y, method='pearson'):
    """Return (r, p, err_msg) where err_msg is None on success."""
    x = pd.to_numeric(x, errors='coerce')
    y = pd.to_numeric(y, errors='coerce')
    mask = np.isfinite(x) & np.isfinite(y)
    x2, y2 = x[mask], y[mask]
    if len(x2) < 2:
        return np.nan, np.nan, "Trop peu de paires valides"
    if x2.nunique() <= 1 or y2.nunique() <= 1:
        return np.nan, np.nan, "Variable constante (corrélation non définie)"
```

```

try:
    if method == 'pearson':
        r, p = pearsonr(x2, y2)
    else:
        r, p = spearmanr(x2, y2)
    if np.isnan(r):
        return np.nan, np.nan, "Test a retourné NaN"
    return r, p, None
except Exception as e:
    return np.nan, np.nan, str(e)

pairs = [
    ('age', 'montant_total'),
    ('age', 'frequence_achat'),
    ('age', 'panier_moyen')
]

for a, b in pairs:
    print('\n' + '-'*60)
    print(f'Diagnostic: {a} vs {b}')
    if a not in client_analysis.columns or b not in client_analysis.columns:
        print('Colonnes manquantes dans client_analysis')
        continue

    s = client_analysis[[a, b]].copy()
    s[a] = pd.to_numeric(s[a], errors='coerce')
    s[b] = pd.to_numeric(s[b], errors='coerce')

    n_total = len(s)
    n_pairs = s.dropna().shape[0]
    n_na_a = s[a].isna().sum()
    n_na_b = s[b].isna().sum()
    ninf_a = int((~np.isfinite(s[a])).sum())
    ninf_b = int((~np.isfinite(s[b])).sum())
    nun_a = int(s[a].nunique(dropna=True))
    nun_b = int(s[b].nunique(dropna=True))

    print(f'Lignes total: {n_total}, paires non-NA: {n_pairs}')
    print(f'NaN {a}: {n_na_a}, NaN {b}: {n_na_b}')
    print(f'Non-finis {a}: {ninf_a}, {b}: {ninf_b}')
    print(f'Unique {a}: {nun_a}, unique {b}: {nun_b}')

    if n_pairs < 2:
        print('=> Trop peu de paires valides pour calculer une corrélation.')
        print('Exemples de lignes avec NaN:')
        print(s[s[a].isna() | s[b].isna()].head(5))
        continue

    if nun_a <= 1 or nun_b <= 1:
        print('=> Variable constante détectée – corrélation non définie (NaN).')
        if nun_a <= 1:
            print(f'Valeurs pour {a}:')
            print(s[a].value_counts(dropna=False).head(10))
        if nun_b <= 1:
            print(f'Valeurs pour {b}:')

```

```

        print(s[b].value_counts(dropna=False).head(10))
    continue

# Use safe utility to compute correlations and get clear error messages
pr_r, pr_p, pr_err = safe_corr_series(s[a], s[b], method='pearson')
sr_r, sr_p, sr_err = safe_corr_series(s[a], s[b], method='spearman')

if pr_err or sr_err:
    print(' => Problème détecté :')
    if pr_err:
        print(f' Pearson: {pr_err}')
    else:
        print(f' Pearson r={pr_r:.4f}, p={pr_p:.4e}')
    if sr_err:
        print(f' Spearman: {sr_err}')
    else:
        print(f' Spearman rho={sr_r:.4f}, p={sr_p:.4e}')
# show examples of problematic rows if any
problematic = s[~np.isfinite(s[a]) | ~np.isfinite(s[b])]
if len(problematic) > 0:
    print(' Exemples de lignes non-finies ou suspectes:')
    print(problematic.head(5))
else:
    print(f" Pearson r={pr_r:.4f}, p={pr_p:.4e}")
    print(f" Spearman rho={sr_r:.4f}, p={sr_p:.4e}")

```

Diagnostic: age vs montant_total
Lignes total: 8597, paires non-NA: 8596
NaN age: 1, NaN montant_total: 0
Non-finis age: 1, montant_total: 0
Unique age: 76, unique montant_total: 8467
Pearson r=-0.1876, p=6.6635e-69
Spearman rho=-0.1845, p=1.0213e-66

Diagnostic: age vs frequence_achat
Lignes total: 8597, paires non-NA: 8596
NaN age: 1, NaN frequence_achat: 0
Non-finis age: 1, frequence_achat: 0
Unique age: 76, unique frequence_achat: 366
Pearson r=0.0303, p=4.9223e-03
Spearman rho=0.1277, p=1.4497e-32

Diagnostic: age vs panier_moyen
Lignes total: 8597, paires non-NA: 8596
NaN age: 1, NaN panier_moyen: 1
Non-finis age: 1, panier_moyen: 1
Unique age: 76, unique panier_moyen: 8564
Pearson r=-0.5105, p=0.0000e+00
Spearman rho=-0.3259, p=8.3529e-212

In [90]: # Tests de corrélation (sécurisés)
pearson_corr, pearson_pvalue, pearson_err = safe_corr_series(client_analysis['age'])
spearman_corr, spearman_pvalue, spearman_err = safe_corr_series(client_analysis['ag'])

```

print("TEST DE CORRÉLATION : ÂGE vs MONTANT TOTAL DES ACHATS")
print("=*70)

print("\nPEARSON (corrélation linéaire) :")
if pearson_err:
    print(f"    Erreur / non calculable : {pearson_err}")
else:
    print(f"    Coefficient de corrélation (r) : {pearson_corr:.4f}")
    print(f"    p-value : {pearson_pvalue:.6f}")

print("\n SPEARMAN (corrélation monotone) :")
if spearman_err:
    print(f"    Erreur / non calculable : {spearman_err}")
else:
    print(f"    Coefficient de corrélation (ρ) : {spearman_corr:.4f}")
    print(f"    p-value : {spearman_pvalue:.6f}")

# Interprétation de la force de la corrélation
def interpret_correlation(r):
    abs_r = abs(r)
    if abs_r < 0.1:
        return "négligeable"
    elif abs_r < 0.3:
        return "faible"
    elif abs_r < 0.5:
        return "modérée"
    elif abs_r < 0.7:
        return "forte"
    else:
        return "très forte"

# Interprétation
alpha = 0.05
print(f"\n{'*70}")
print("INTERPRÉTATION (seuil α = 0.05) :")

if pearson_pvalue < alpha:
    print(f" p-value ({pearson_pvalue:.6f}) < α ({alpha})")
    print(" On REJETTE H0")
    print(f" Conclusion : Il existe une corrélation {interpret_correlation(pearson_corr)} entre l'âge et le montant total des achats")
    print(f" (r = {pearson_corr:.4f})")
else:

    print(f" p-value ({pearson_pvalue:.6f}) ≥ α ({alpha})")

    print(" On NE REJETTE PAS H0")
    print(" Conclusion : Pas de corrélation significative entre l'âge et le montant total des achats")

```

TEST DE CORRÉLATION : ÂGE vs MONTANT TOTAL DES ACHATS

PEARSON (corrélation linéaire) :
 Coefficient de corrélation (r) : -0.1876
 p-value : 0.000000

SPEARMAN (corrélation monotone) :
 Coefficient de corrélation (ρ) : -0.1845
 p-value : 0.000000

INTERPRÉTATION (seuil $\alpha = 0.05$) :
 p-value (0.000000) < α (0.05)
 On REJETTE H_0
 Conclusion : Il existe une corrélation faible
 négative entre l'âge et le montant total des achats.
 ($r = -0.1876$)

```
In [91]: # Visualisation bivariée : Âge vs Montant total
fig, axes = plt.subplots(1, 2, figsize=(14, 5))

# Nuage de points avec droite de régression
axes[0].scatter(client_analysis['age'], client_analysis['montant_total'],
                 alpha=0.5, s=30, color='steelblue')
# Droite de régression
# Sanitize inputs + régression robuste
mask = np.isfinite(client_analysis['age']) & np.isfinite(client_analysis['montant_total'])
x = client_analysis.loc[mask, 'age'].astype(float).values
y = client_analysis.loc[mask, 'montant_total'].astype(float).values

if len(x) < 2 or np.allclose(x, x[0]):
    axes[0].text(0.5, 0.9, "Régression non disponible (données insuffisantes / constantes)", transform=axes[0].transAxes, ha='center', va='center', color='red')
else:
    try:
        # center & scale to improve numerical conditioning
        xm = x.mean()
        xs = x.std(ddof=0) if x.std(ddof=0) != 0 else 1.0
        x_scaled = (x - xm) / xs
        p_coefs = np.polyfit(x_scaled, y, 1)
        slope = p_coefs[0] / xs
        intercept = p_coefs[1] - p_coefs[0] * xm / xs
    except np.linalg.LinAlgError:
        # fallback stable
        from scipy.stats import linregress
        lr = linregress(x, y)
        slope, intercept = lr.slope, lr.intercept

    xs_plot = np.linspace(x.min(), x.max(), 100)
    ys_plot = slope * xs_plot + intercept
    axes[0].plot(xs_plot, ys_plot, "r--", linewidth=2, label='Régression linéaire')
    axes[0].set_xlabel('Âge (années)')
    axes[0].set_ylabel('Montant total (€)')
    axes[0].set_title('Nuage de points : Âge vs Montant total', fontweight='bold')
    axes[0].legend()
```

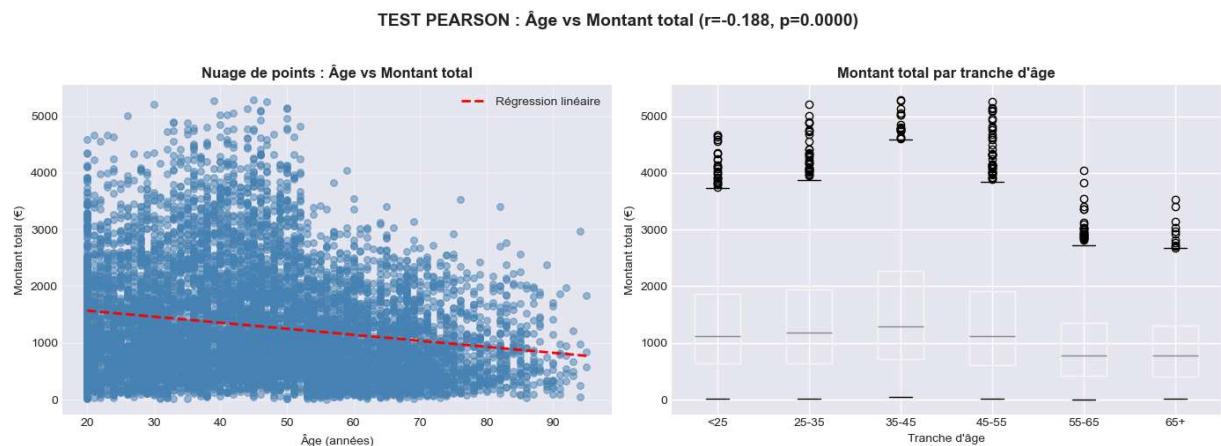
```

axes[0].grid(True, alpha=0.3)

# Boxplot du montant total par tranche d'âge
client_analysis['tranche_age'] = pd.cut(client_analysis['age'],
                                         bins=[0, 25, 35, 45, 55, 65, 100],
                                         labels=['<25', '25-35', '35-45', '45-55', '55-65', '65+'])
client_analysis.boxplot(column='montant_total', by='tranche_age', ax=axes[1])
axes[1].set_xlabel('Tranche d\'âge')
axes[1].set_ylabel('Montant total (€)')
axes[1].set_title('Montant total par tranche d\'âge', fontweight='bold')
plt.sca(axes[1])
plt.xticks(rotation=0)

plt.suptitle(f'TEST PEARSON : Âge vs Montant total (r={pearson_corr:.3f}, p={pearson_pvalue:.3f})',
             fontsize=14, fontweight='bold', y=1.02)
plt.tight_layout()
plt.show()

```



6. TEST 3 : Âge ↔ Fréquence d'achat

Type de variables :

- **Âge** : Variable quantitative continue
- **Fréquence d'achat** : Variable quantitative discrète

Test approprié : Test de corrélation de Spearman

Hypothèses :

- H0 : Il n'y a pas de corrélation entre l'âge et la fréquence d'achat ($\rho = 0$)
- H1 : Il existe une corrélation entre l'âge et la fréquence d'achat ($\rho \neq 0$)

```

In [94]: # Tests de corrélation (sécurisés)
spearman_corr_freq, spearman_pvalue_freq, spearman_err_freq = safe_corr_series(client_analysis,
clie
pearson_corr_freq, pearson_pvalue_freq, pearson_err_freq = safe_corr_series(client_
client_an

```

```

print("TEST DE CORRÉLATION : ÂGE vs FRÉQUENCE D'ACHAT")
print("=*70)

print("\n SPEARMAN (corrélation monotone - RECOMMANDÉ) :")
if spearman_err_freq:
    print(f"  Erreur / non calculable : {spearman_err_freq}")
else:
    print(f"  Coefficient de corrélation ( $\rho$ ) : {spearman_corr_freq:.4f}")
    print(f"  p-value : {spearman_pvalue_freq:.6f}")

print("\n PEARSON (corrélation linéaire) :")
if pearson_err_freq:
    print(f"  Erreur / non calculable : {pearson_err_freq}")
else:
    print(f"  Coefficient de corrélation ( $r$ ) : {pearson_corr_freq:.4f}")
    print(f"  p-value : {pearson_pvalue_freq:.6f}")

# Interprétation
alpha = 0.05

print("INTERPRÉTATION (seuil  $\alpha = 0.05$ ) :")

if spearman_pvalue_freq < alpha:
    print(f"  p-value ({spearman_pvalue_freq:.6f}) <  $\alpha$  ({alpha})")
    print("  On REJETTE  $H_0$ ")
    print(f"  Conclusion : Il existe une corrélation {interpret_correlation(spearman_pvalue_freq)} entre l'âge et la fréquence d'achat")
    print(f"  ( $\rho = {spearman_corr_freq:.4f}$ )")
else:

    print(f"  p-value ({spearman_pvalue_freq:.6f})  $\geq \alpha$  ({alpha})")

    print("  On NE REJETTE PAS  $H_0$ ")
    print("  Conclusion : Pas de corrélation significative entre l'âge et la fréquence d'achat")

```

TEST DE CORRÉLATION : ÂGE vs FRÉQUENCE D'ACHAT
=====

SPEARMAN (corrélation monotone - RECOMMANDÉ) :
 Coefficient de corrélation (ρ) : 0.1277
 p-value : 0.000000

PEARSON (corrélation linéaire) :
 Coefficient de corrélation (r) : 0.0303
 p-value : 0.004922
 INTERPRÉTATION (seuil $\alpha = 0.05$) :
 p-value (0.000000) < α (0.05)
 On REJETTE H_0
 Conclusion : Il existe une corrélation faible
 positive entre l'âge et la fréquence d'achat.
 ($\rho = 0.1277$)

In [95]: # Visualisation bivariée : Âge vs Fréquence d'achat
fig, axes = plt.subplots(1, 2, figsize=(14, 5))

```

# Nuage de points
axes[0].scatter(client_analysis['age'], client_analysis['fréquence_achat'],
                alpha=0.5, s=30, color='green')
# Droite de régression
# Sanitize inputs + régression robuste
mask = np.isfinite(client_analysis['age']) & np.isfinite(client_analysis['fréquence_achat'])
x = client_analysis.loc[mask, 'age'].astype(float).values
y = client_analysis.loc[mask, 'fréquence_achat'].astype(float).values

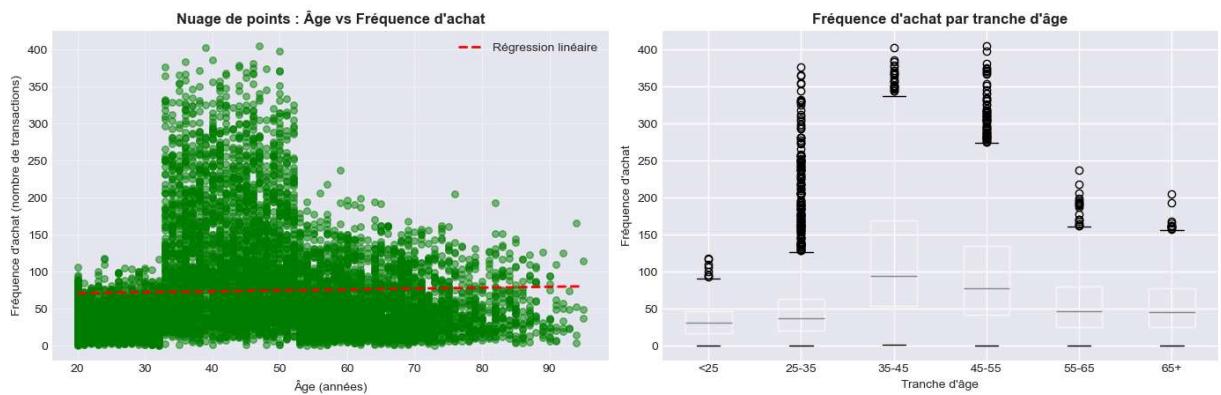
if len(x) < 2 or np.allclose(x, x[0]):
    axes[0].text(0.5, 0.9, "Régression non disponible (données insuffisantes / cons.",
                 transform=axes[0].transAxes, ha='center', va='center', color='red')
else:
    try:
        # centre & scale pour améliorer la condition numérique
        xm = x.mean()
        xs = x.std(ddof=0) if x.std(ddof=0) != 0 else 1.0
        x_scaled = (x - xm) / xs
        p_coefs = np.polyfit(x_scaled, y, 1)
        slope = p_coefs[0] / xs
        intercept = p_coefs[1] - p_coefs[0] * xm / xs
    except np.linalg.LinAlgError:
        # fallback stable
        from scipy.stats import linregress
        lr = linregress(x, y)
        slope, intercept = lr.slope, lr.intercept

    xs_plot = np.linspace(x.min(), x.max(), 100)
    ys_plot = slope * xs_plot + intercept
    axes[0].plot(xs_plot, ys_plot, "r--", linewidth=2, label='Régression linéaire')
axes[0].set_xlabel('Âge (années)')
axes[0].set_ylabel('Fréquence d\'achat (nombre de transactions)')
axes[0].set_title('Nuage de points : Âge vs Fréquence d\'achat', fontweight='bold')
axes[0].legend()
axes[0].grid(True, alpha=0.3)

# Boxplot de la fréquence par tranche d'âge
client_analysis.boxplot(column='fréquence_achat', by='tranche_age', ax=axes[1])
axes[1].set_xlabel('Tranche d\'âge')
axes[1].set_ylabel('Fréquence d\'achat')
axes[1].set_title('Fréquence d\'achat par tranche d\'âge', fontweight='bold')
plt.sca(axes[1])
plt.xticks(rotation=0)

plt.suptitle(f'TEST SPEARMAN : Âge vs Fréquence (p={spearman_corr_freq:.3f}, p={spearman_p:.3f})',
             fontsize=14, fontweight='bold', y=1.02)
plt.tight_layout()
plt.show()

```

TEST SPEARMAN : Âge vs Fréquence ($p=0.128$, $p=0.0000$)

7. TEST 4 : Âge \leftrightarrow Taille du panier moyen

Type de variables :

- **Âge** : Variable quantitative continue
- **Panier moyen** : Variable quantitative continue

Test approprié : **Test de corrélation de Pearson**

Hypothèses :

- H_0 : Il n'y a pas de corrélation entre l'âge et le panier moyen ($\rho = 0$)
- H_1 : Il existe une corrélation entre l'âge et le panier moyen ($\rho \neq 0$)

```
In [97]: # Tests de corrélation (sécurisés)
pearson_corr_panier, pearson_pvalue_panier, pearson_err_panier = safe_corr_series(c
cli
spearman_corr_panier, spearman_pvalue_panier, spearman_err_panier = safe_corr_serie

print("TEST DE CORRÉLATION : ÂGE vs PANIER MOYEN")
print("=*70)

print("\n PEARSON (corrélation linéaire - RECOMMANDÉ) :")
if pearson_err_panier:
    print(f"  Erreur / non calculable : {pearson_err_panier}")
else:
    print(f"  Coefficient de corrélation (r) : {pearson_corr_panier:.4f}")
    print(f"  p-value : {pearson_pvalue_panier:.6f}")

print("\n SPEARMAN (corrélation monotone) :")
if spearman_err_panier:
    print(f"  Erreur / non calculable : {spearman_err_panier}")
else:
    print(f"  Coefficient de corrélation (\rho) : {spearman_corr_panier:.4f}")
    print(f"  p-value : {spearman_pvalue_panier:.6f})
```

```
# Interprétation
alpha = 0.05

print("INTERPRÉTATION (seuil α = 0.05) :")

if pearson_pvalue_panier < alpha:
    print(f" p-value ({pearson_pvalue_panier:.6f}) < α ({alpha})")
    print(" On REJETTE H0")
    print(f" Conclusion : Il existe une corrélation {interpret_correlation(pearson
        print(f"      {'positive' if pearson_corr_panier > 0 else 'négative'}) entre l'âg
        print(f"      (r = {pearson_corr_panier:.4f}))")
else:

    print(f" p-value ({pearson_pvalue_panier:.6f}) ≥ α ({alpha})")

    print(" On NE REJETTE PAS H0")
    print(" Conclusion : Pas de corrélation significative entre l'âge et le panier")
```

TEST DE CORRÉLATION : ÂGE vs PANIER MOYEN

PEARSON (corrélation linéaire - RECOMMANDÉ) :

Coefficient de corrélation (r) : -0.5105
p-value : 0.000000

SPEARMAN (corrélation monotone) :

Coefficient de corrélation (ρ) : -0.3259
p-value : 0.000000

INTERPRÉTATION (seuil α = 0.05) :

p-value (0.000000) < α (0.05)
On REJETTE H0
Conclusion : Il existe une corrélation forte
négative entre l'âge et le panier moyen.
(r = -0.5105)

In [98]:

```
# Visualisation bivarée : Âge vs Panier moyen
fig, axes = plt.subplots(1, 2, figsize=(14, 5))

# Nuage de points
axes[0].scatter(client_analysis['age'], client_analysis['panier_moyen'],
                alpha=0.5, s=30, color='purple')
# Droite de régression
#z = np.polyfit(client_analysis['age'], client_analysis['panier_moyen'], 1)
# Sanitize inputs
mask = np.isfinite(client_analysis['age']) & np.isfinite(client_analysis['panier_moyen'])
x = client_analysis.loc[mask, 'age'].astype(float).values
y = client_analysis.loc[mask, 'panier_moyen'].astype(float).values

if len(x) >= 2 and not np.allclose(x, x[0]):
    try:
        # centre & scale pour améliorer la condition numérique
        xm = x.mean()
        xs = x.std(ddof=0) if x.std(ddof=0) != 0 else 1.0
        x_scaled = (x - xm) / xs
        p_coefs = np.polyfit(x_scaled, y, 1)
        slope = p_coefs[0] / xs
```

```

        intercept = p_coefs[1] - p_coefs[0] * xm / xs
    except np.linalg.LinAlgError:
        # fallback stable
        from scipy.stats import linregress
        lr = linregress(x, y)
        slope, intercept = lr.slope, lr.intercept

        xs_plot = np.linspace(x.min(), x.max(), 100)
        ys_plot = slope * xs_plot + intercept
        axes[0].plot(xs_plot, ys_plot, "r--", linewidth=2, label='Régression linéaire')
else:
    axes[0].text(0.5, 0.9, "Régression non disponible (données insuffisantes / cons",
                 transform=axes[0].transAxes, ha='center', va='center', color='red')

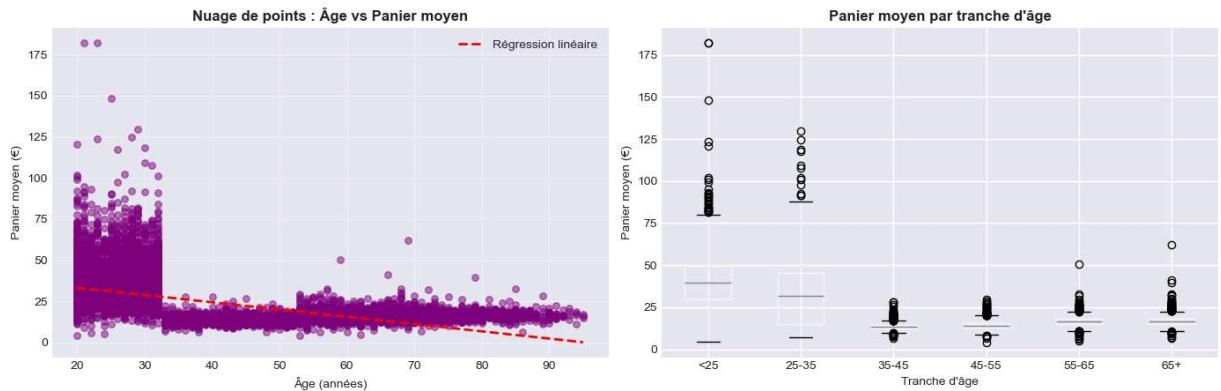
axes[0].set_xlabel('Âge (années)')
axes[0].set_ylabel('Panier moyen (€)')
axes[0].set_title('Nuage de points : Âge vs Panier moyen', fontweight='bold')
axes[0].legend()
axes[0].grid(True, alpha=0.3)

# Boxplot du panier moyen par tranche d'âge
client_analysis.boxplot(column='panier_moyen', by='tranche_age', ax=axes[1])
axes[1].set_xlabel('Tranche d\'âge')
axes[1].set_ylabel('Panier moyen (€)')
axes[1].set_title('Panier moyen par tranche d\'âge', fontweight='bold')
plt.sca(axes[1])
plt.xticks(rotation=0)

plt.suptitle(f'TEST PEARSON : Âge vs Panier moyen (r={pearson_corr_panier:.3f}, p={pearson_p_value:.3f}', fontsize=14, fontweight='bold', y=1.02)
plt.tight_layout()
plt.show()

```

TEST PEARSON : Âge vs Panier moyen (r=-0.511, p=0.0000)



8. TEST 5 : Âge ↔ Catégorie des livres achetés

Type de variables :

- **Âge** : Variable quantitative continue → À discréter en tranches d'âge
- **Catégorie** : Variable qualitative (0, 1, 2)

Tests appropriés :

1. **Test du χ^2** (après discréétisation de l'âge)
2. **ANOVA** (pour comparer l'âge moyen entre catégories)

Hypothèses (ANOVA) :

- H0 : Les moyennes d'âge sont identiques pour toutes les catégories
- H1 : Au moins une catégorie a une moyenne d'âge différente

In [100...]

```
# Méthode 1 : Test du  $\chi^2$  avec âge discréétisé
# On utilise la colonne 'tranche_age' créée précédemment

contingency_age_cat = pd.crosstab(client_analysis['tranche_age'],
                                   client_analysis['categorie_preferee'])

print(" Tableau de contingence : Tranche d'âge vs Catégorie préférée")
print(contingency_age_cat)

# Test du  $\chi^2$ 
chi2_age_cat, p_age_cat, dof_age_cat, expected_age_cat = chi2_contingency(contingen
```

```
print("TEST DU  $\chi^2$  : TRANCHE D'ÂGE vs CATÉGORIE")
print("=*70)
print(f"Statistique  $\chi^2$  : {chi2_age_cat:.4f}")
print(f"p-value : {p_age_cat:.6f}")
print(f"Degrés de liberté : {dof_age_cat}")
```

Tableau de contingence : Tranche d'âge vs Catégorie préférée

tranche_age	0.0	1.0	2.0
<25	210	461	475
25-35	612	448	501
35-45	1748	31	0
45-55	1266	343	0
55-65	402	826	0
65+	388	884	1

TEST DU χ^2 : TRANCHE D'ÂGE vs CATÉGORIE

=====

Statistique χ^2 : 5047.2958

p-value : 0.000000

Degrés de liberté : 10

In [101...]

```
# Méthode 2 : ANOVA - Comparer l'âge moyen entre les catégories
# Séparation des âges par catégorie

age_cat_0 = client_analysis[client_analysis['categorie_preferee'] == 0]['age']
age_cat_1 = client_analysis[client_analysis['categorie_preferee'] == 1]['age']
age_cat_2 = client_analysis[client_analysis['categorie_preferee'] == 2]['age']

# Test ANOVA
f_stat, p_anova = f_oneway(age_cat_0, age_cat_1, age_cat_2)
```

```

print("TEST ANOVA : ÂGE MOYEN par CATÉGORIE")
print("=*70)
print(f"Statistique F : {f_stat:.4f}")
print(f"p-value : {p_anova:.6f}")

# Moyennes d'âge par catégorie
print("\n Âge moyen par catégorie :")
age_by_cat = client_analysis.groupby('categorie_preferee')[['age']].agg(['mean', 'std'])
age_by_cat.columns = ['Âge moyen', 'Écart-type', 'Nombre de clients']
print(age_by_cat)

```

TEST ANOVA : ÂGE MOYEN par CATÉGORIE
=====

Statistique F : 1201.7032
p-value : 0.000000

categorie_preferee	Âge moyen	Écart-type	Nombre de clients	# Moyennes d'âge par catégorie :
0.0	45.540424	12.530171	4626	
1.0	52.615436	19.827265	2993	
2.0	25.619243	4.331825	977	

In [102...]: # Interprétation des deux tests
alpha = 0.05

```

print("\n" + "*70)
print("INTERPRÉTATION GLOBALE (seuil α = 0.05) :")

print("\n1 TEST χ² (Tranche d'âge vs Catégorie) :")
if p_age_cat < alpha:
    print(f"    p-value ({p_age_cat:.6f}) < α ({alpha})")
    print("    On REJETTE H₀")
    print("    Conclusion : Il existe un lien significatif entre la tranche d'âge")
    print("        et la catégorie de livres préférée.")
else:
    print(f"    p-value ({p_age_cat:.6f}) ≥ α ({alpha})")
    print("    On NE REJETTE PAS H₀")
    print("    Conclusion : Pas de lien significatif entre la tranche d'âge")
    print("        et la catégorie de livres préférée.")

print("\n2 TEST ANOVA (Âge moyen par catégorie) :")
if p_anova < alpha:
    print(f"    p-value ({p_anova:.6f}) < α ({alpha})")
    print("    On REJETTE H₀")
    print("    Conclusion : L'âge moyen diffère significativement entre")
    print("        au moins deux catégories de livres.")
else:
    print(f"    p-value ({p_anova:.6f}) ≥ α ({alpha})")
    print("    On NE REJETTE PAS H₀")
    print("    Conclusion : L'âge moyen ne diffère pas significativement")
    print("        entre les catégories de livres.")

```

```
=====
INTERPRÉTATION GLOBALE (seuil  $\alpha = 0.05$ ) :
```

- 1 TEST χ^2 (Tranche d'âge vs Catégorie) :

p-value (0.000000) < α (0.05)

On REJETTE H_0

Conclusion : Il existe un lien significatif entre la tranche d'âge et la catégorie de livres préférée.
- 2 TEST ANOVA (Âge moyen par catégorie) :

p-value (0.000000) < α (0.05)

On REJETTE H_0

Conclusion : L'âge moyen diffère significativement entre au moins deux catégories de livres.

In [103...]

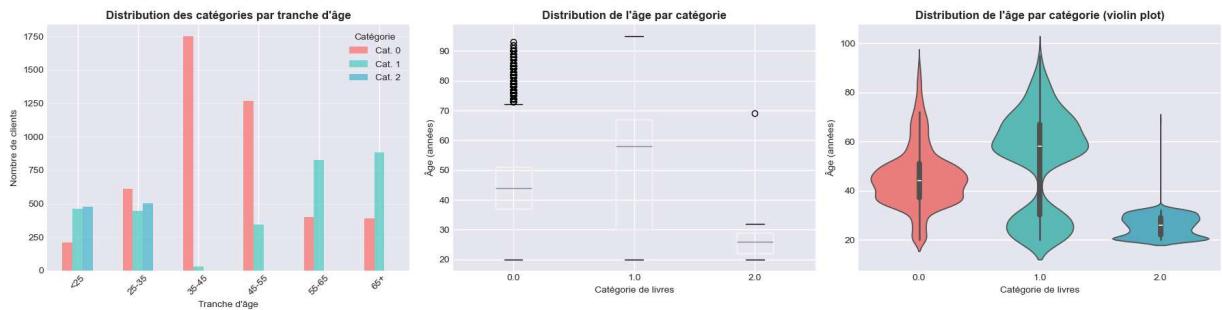
```
# Visualisation bivarée : Âge vs Catégorie
fig, axes = plt.subplots(1, 3, figsize=(18, 5))

# Graphique 1 : Barres groupées (tranches d'âge par catégorie)
contingency_age_cat.plot(kind='bar', ax=axes[0], color=['#FF6B6B', '#4ECDC4', '#45B7D1'])
axes[0].set_xlabel('Tranche d\'âge')
axes[0].set_ylabel('Nombre de clients')
axes[0].set_title('Distribution des catégories par tranche d\'âge', fontweight='bold')
axes[0].legend(title='Catégorie', labels=['Cat. 0', 'Cat. 1', 'Cat. 2'])
axes[0].set_xticklabels(axes[0].get_xticklabels(), rotation=45)
axes[0].grid(True, alpha=0.3, axis='y')

# Graphique 2 : Boxplot de l'âge par catégorie
client_analysis.boxplot(column='age', by='categorie_preferee', ax=axes[1])
axes[1].set_xlabel('Catégorie de livres')
axes[1].set_ylabel('Âge (années)')
axes[1].set_title('Distribution de l\'âge par catégorie', fontweight='bold')
plt.sca(axes[1])
plt.xticks(rotation=0)

# Graphique 3 : Violin plot
import seaborn as sns
sns.violinplot(data=client_analysis, x='categorie_preferee', y='age', ax=axes[2],
                 palette=['#FF6B6B', '#4ECDC4', '#45B7D1'])
axes[2].set_xlabel('Catégorie de livres')
axes[2].set_ylabel('Âge (années)')
axes[2].set_title('Distribution de l\'âge par catégorie (violin plot)', fontweight='bold')

plt.suptitle(f'Âge vs Catégorie |  $\chi^2={\text{chi2}_age\_cat:.2f}$  (p={p_age_cat:.4f}) | ANOVA',
             fontsize=14, fontweight='bold', y=1.02)
plt.tight_layout()
plt.show()
```

Âge vs Catégorie | $\chi^2=5047.30$ ($p=0.0000$) | ANOVA F=1201.70 ($p=0.0000$)

9. Synthèse des résultats des tests statistiques

In [104...]

```
# Création d'un tableau récapitulatif
resultats = {
    'Test': [
        'Genre ⇔ Catégorie',
        'Âge ⇔ Montant total',
        'Âge ⇔ Fréquence achat',
        'Âge ⇔ Panier moyen',
        'Âge ⇔ Catégorie ( $\chi^2$ )',
        'Âge ⇔ Catégorie (ANOVA)'
    ],
    'Type de test': [
        ' $\chi^2$  (Khi-deux)',
        'Pearson',
        'Spearman',
        'Pearson',
        ' $\chi^2$  (Khi-deux)',
        'ANOVA'
    ],
    'Statistique': [
        f' $\chi^2$  = {chi2_stat:.4f}',
        f'r = {pearson_corr:.4f}',
        f'p = {spearman_corr_freq:.4f}',
        f'r = {pearson_corr_panier:.4f}',
        f' $\chi^2$  = {chi2_age_cat:.4f}',
        f'F = {f_stat:.4f}'
    ],
    'p-value': [
        f'{p_value:.6f}',
        f'{pearson_pvalue:.6f}',
        f'{spearman_pvalue_freq:.6f}',
        f'{pearson_pvalue_panier:.6f}',
        f'{p_age_cat:.6f}',
        f'{p_anova:.6f}'
    ],
    'Significatif ( $\alpha=0.05$ )': [
        'OUI' if p_value < 0.05 else 'NON',
        'OUI' if pearson_pvalue < 0.05 else 'NON',
        'OUI' if spearman_pvalue_freq < 0.05 else 'NON',
        'OUI' if pearson_pvalue_panier < 0.05 else 'NON',
        'OUI' if p_age_cat < 0.05 else 'NON',
        'OUI' if p_anova < 0.05 else 'NON'
    ]
}
```

```

        ]
}

df_resultats = pd.DataFrame(resultats)

print("=*100")
print(" SYNTHÈSE DES RÉSULTATS DES TESTS STATISTIQUES")

print(df_resultats.to_string(index=False))
=====
```

```

=====
SYNTHÈSE DES RÉSULTATS DES TESTS STATISTIQUES
      Test   Type de test   Statistique   p-value Significatif ( $\alpha=0.05$ )
Genre  $\leftrightarrow$  Catégorie  $\chi^2$  (Khi-deux)    $\chi^2 = 4.8977$  0.086393      NON
Âge  $\leftrightarrow$  Montant total       Pearson    r = -0.1876 0.000000      OUI
Âge  $\leftrightarrow$  Fréquence achat       Spearman   p = 0.1277 0.000000      OUI
Âge  $\leftrightarrow$  Panier moyen        Pearson    r = -0.5105 0.000000      OUI
Âge  $\leftrightarrow$  Catégorie ( $\chi^2$ )  $\chi^2$  (Khi-deux)    $\chi^2 = 5047.2958$  0.000000      OUI
Âge  $\leftrightarrow$  Catégorie (ANOVA)      ANOVA     F = 1201.7032 0.000000      OUI
```

10. Conclusions et recommandations

Points clés de l'analyse statistique :

Méthodologie :

- Analyse réalisée SANS les 4 clients BtoB identifiés
- 5 corrélations testées comme demandé par Julie
- Tests statistiques appropriés selon le type de variables
- Graphiques bivariés pour chaque test
- Seuil de significativité : $\alpha = 0.05$

Tests réalisés :

1. **Test du χ^2** pour les variables qualitatives
2. **Test de Pearson** pour les corrélations linéaires
3. **Test de Spearman** pour les corrélations monotones
4. **ANOVA** pour comparer les moyennes entre groupes

Recommandations pour la soutenance :

1. **Expliquer le choix des tests** : Adapter le test au type de variables
2. **Interpréter la p-value** : Seuil de significativité de 5%
3. **Visualiser les résultats** : Les graphiques facilitent la compréhension
4. **Justifier l'exclusion des BtoB** : Impact sur les résultats
5. **Proposer des actions** : Basées sur les corrélations significatives

```
In [105...]: # Export du tableau de synthèse  
df_resultats.to_csv('synthese_tests_statistiques.csv', index=False, sep=';')  
print("Synthèse des tests exportée : synthese_tests_statistiques.csv")  
  
# Export du dataset d'analyse client  
client_analysis.to_csv('client_analysis_btoc.csv', index=False, sep=';')  
print(" Dataset d'analyse clients exporté : client_analysis_btoc.csv")
```

Synthèse des tests exportée : synthese_tests_statistiques.csv
Dataset d'analyse clients exporté : client_analysis_btoc.csv