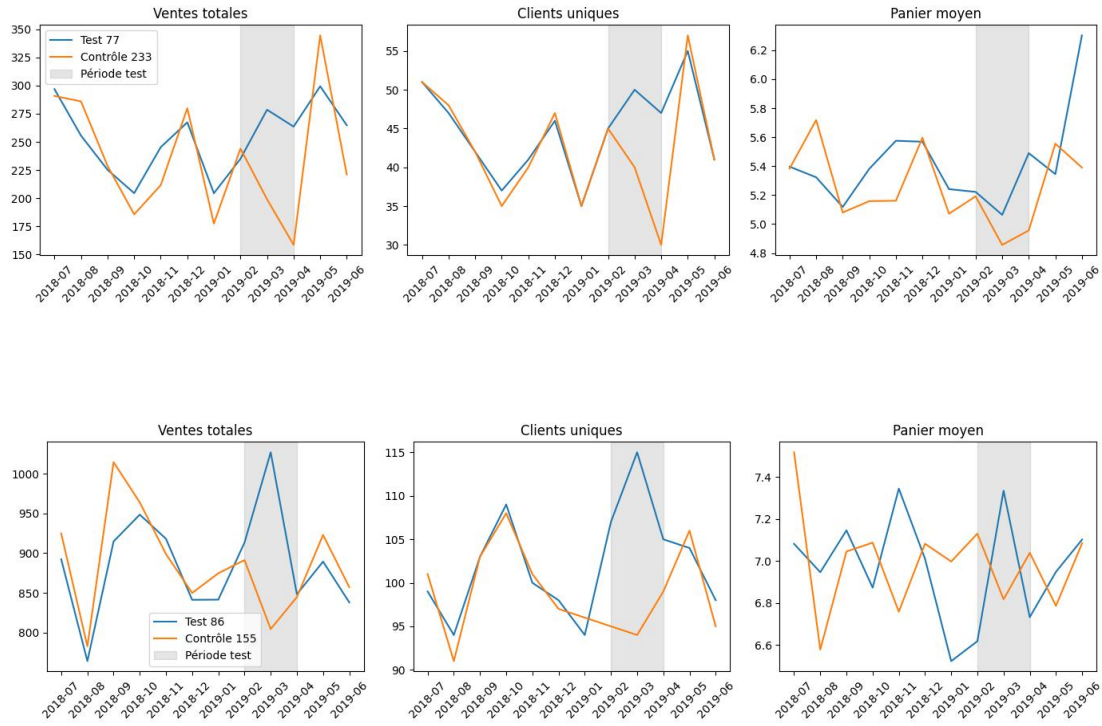
TASK 2

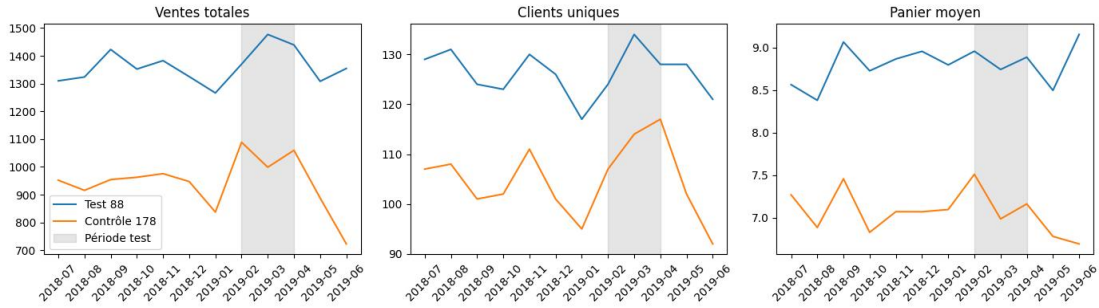
```
import pandas as pd

df = pd.read_csv('QVI_data.csv')
df.head()
df['DATE'] = pd.to_datetime(df['DATE'])
df['MONTH'] = df['DATE'].dt.to_period('M')
grouped = df.groupby(['STORE_NBR', 'MONTH']).agg({'TOT_SALES': 'sum', 'PROD_QTY': 'sum'}).reset_index()
grouped.head()
monthly_store_stats = df.groupby(['STORE_NBR', 'MONTH']).agg(
   TOT_SALES=('TOT_SALES', 'sum'),
   UNIQUE_CUSTOMERS=('LYLTY_CARD_NBR', 'nunique'),
   TOTAL_TRANSACTIONS=('TXN_ID', 'nunique')
).reset_index()

monthly_store_stats['AVG_TXN_PER_CUSTOMER'] = (
   monthly_store_stats['TOTAL_TRANSACTIONS'] / monthly_store_stats['UNIQUE_CUSTOMERS']
)

monthly_store_stats.head()
# Définir les magasins test
test_stores = [77, 86, 88]

# Période avant le test (par exemple, avant 2019-02)
pre_test_period = monthly_store_stats['MONTH'] < '2019-02'

# Filtrer les données avant le test
pre_test_data = monthly_store_stats[pre_test_period]

# Fonction pour calculer la similarité (corrélation de Pearson) entre deux magasins
def store_similarity(test_store, candidate_store, metric='TOT_SALES'):
   test_series = pre_test_data[pre_test_data['STORE_NBR'] == test_store].set_index('MONTH')[metric]
   candidate_series = pre_test_data[pre_test_data['STORE_NBR'] == candidate_store].set_index('MONTH')[metric]
   # Aligner les index
   aligned = test_series.align(candidate_series, join='inner')
   if len(aligned[0]) < 6:  # Trop peu de points pour comparer
      return None
   return aligned[0].corr(aligned[1])
```

```
# Pour chaque magasin test, trouver les magasins les plus similaires
control_store_candidates = {}
for test_store in test_stores:
    similarities = []
    for candidate_store in monthly_store_stats['STORE_NBR'].unique():
        if candidate_store == test_store:
            continue
        corr_sales = store_similarity(test_store, candidate_store, metric='TOT_SALES')
        corr_customers = store_similarity(test_store, candidate_store, metric='UNIQUE_CUSTOMERS')
        if corr_sales is not None and corr_customers is not None:
            # Moyenne des corrélations ventes et clients
            mean_corr = (corr_sales + corr_customers) / 2
            similarities.append((candidate_store, mean_corr))
    # Trier par similarité décroissante
    similarities.sort(key=lambda x: x[1], reverse=True)
    # Garder les 3 meilleurs candidats
    control_store_candidates[test_store] = similarities[:3]

control_store_candidates
from scipy.stats import ttest_ind

# Définir la période du test (par exemple, de 2019-02 à 2019-04 inclus)
test_period = (monthly_store_stats['MONTH'] >= '2019-02') & (monthly_store_stats['MONTH'] <= '2019-04')

results = {}

for test_store in test_stores:
    # Prendre le meilleur control store pour chaque test store
    control_store = control_store_candidates[test_store][0][0]

    # Extraire les données pour la période du test
    test_data = monthly_store_stats[(monthly_store_stats['STORE_NBR'] == test_store) & test_period]
    control_data = monthly_store_stats[(monthly_store_stats['STORE_NBR'] == control_store) & test_period]

    # Comparer les ventes totales
    sales_test = test_data['TOT_SALES']
    sales_control = control_data['TOT_SALES']
    t_stat_sales, p_value_sales = ttest_ind(sales_test, sales_control, equal_var=False)

    # Comparer le nombre de clients uniques
```

```
    customers_test = test_data['UNIQUE_CUSTOMERS']
    customers_control = control_data['UNIQUE_CUSTOMERS']
    t_stat_customers, p_value_customers = ttest_ind(customers_test, customers_control, equal_var=False)

    # Comparer le nombre de transactions par client
    avg_txn_test = test_data['AVG_TXN_PER_CUSTOMER']
    avg_txn_control = control_data['AVG_TXN_PER_CUSTOMER']
    t_stat_txn, p_value_txn = ttest_ind(avg_txn_test, avg_txn_control, equal_var=False)

    # Comparer le panier moyen (ventes / transactions)
    basket_test = test_data['TOT_SALES'] / test_data['TOTAL_TRANSACTIONS']
    basket_control = control_data['TOT_SALES'] / control_data['TOTAL_TRANSACTIONS']
    t_stat_basket, p_value_basket = ttest_ind(basket_test, basket_control, equal_var=False)

    results[test_store] = {
        'control_store': control_store,
        'sales': {'t_stat': t_stat_sales, 'p_value': p_value_sales},
        'unique_customers': {'t_stat': t_stat_customers, 'p_value': p_value_customers},
        'avg_txn_per_customer': {'t_stat': t_stat_txn, 'p_value': p_value_txn},
        'basket_size': {'t_stat': t_stat_basket, 'p_value': p_value_basket}
    }

results
# Visualisations avant/après et test/control
import matplotlib.pyplot as plt

for test_store in test_stores:
    control_store = results[test_store]['control_store']
    test_data = monthly_store_stats[(monthly_store_stats['STORE_NBR'] == test_store)]
    control_data = monthly_store_stats[(monthly_store_stats['STORE_NBR'] == control_store)]

    plt.figure(figsize=(14, 4))
    # Ventes
    plt.subplot(1, 3, 1)
    plt.plot(test_data['MONTH'].astype(str), test_data['TOT_SALES'], label=f'Test {test_store}')
    plt.plot(control_data['MONTH'].astype(str), control_data['TOT_SALES'], label=f'Contrôle {control_store}')
    plt.axvspan('2019-02', '2019-04', color='grey', alpha=0.2, label='Période test')
    plt.title('Ventes totales')
    plt.xticks(rotation=45)
    plt.legend()
    # Clients
    plt.subplot(1, 3, 2)
```

```
    plt.plot(test_data['MONTH'].astype(str), test_data['UNIQUE_CUSTOMERS'], label=f'Test {test_store}')
    plt.plot(control_data['MONTH'].astype(str), control_data['UNIQUE_CUSTOMERS'], label=f'Contrôle {control_store}')
    plt.axvspan('2019-02', '2019-04', color='grey', alpha=0.2)
    plt.title('Clients uniques')
    plt.xticks(rotation=45)
    # Panier moyen
    plt.subplot(1, 3, 3)
    plt.plot(test_data['MONTH'].astype(str), test_data['TOT_SALES']/test_data['TOTAL_TRANSACTIONS'], label=f'Test {test_store}')
    plt.plot(control_data['MONTH'].astype(str), control_data['TOT_SALES']/control_data['TOTAL_TRANSACTIONS'], label=f'Contrôle {control_store}')
    plt.axvspan('2019-02', '2019-04', color='grey', alpha=0.2)
    plt.title('Panier moyen')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```





Rapport d’analyse de la performance du layout test - Task 2 (Quantium)

Contexte
Julia, la Category Manager, nous a demandé d'évaluer l'impact de nouveaux aménagements en magasin (layouts) sur les ventes de chips dans trois magasins tests (77, 86, 88). L’objectif est de déterminer si ces changements doivent être déployés à l’ensemble des magasins.

Période test
Février 2019 à avril 2019 (incluse).

Magasin Test 77 vs Contrôle 233

Ventes totales : Le magasin test dépasse les ventes du magasin contrôle durant la période du test, sauf en avril.

Clients uniques : Les deux courbes suivent une dynamique similaire, indiquant un comportement client stable.

Panier moyen : Hausse marquée dans le magasin test pendant la période d’essai.

Conclusion : Effet positif du layout, porté par une augmentation du panier moyen.

Recommandation : Déploiement recommandé.

Magasin Test 86 vs Contrôle 155

Ventes totales : Forte augmentation en février, puis retour à des niveaux comparables.

Clients uniques : Hausse du nombre de clients pendant le test.

Panier moyen : Pas de différence significative avec le magasin contrôle.

Conclusion : Impact modéré du layout, probablement dû à une hausse temporaire de fréquentation.

Recommandation : Déploiement possible mais à confirmer par d'autres tests ou sur une plus longue durée.

Magasin Test 88 vs Contrôle 178

Ventes totales : Le magasin test est systématiquement au-dessus, mais aucune variation significative pendant la période test.

Clients uniques : Stable et élevés dès le départ dans le magasin test.

Panier moyen : Légère hausse mais tendanciellement stable.

Conclusion : Aucune rupture dans la tendance des ventes ou du comportement client. Pas d'impact mesurable du layout.

Recommandation : Ne pas déployer dans ce type de magasin.

Synthèse finale

* **Succès net** pour le magasin 77
* **Impact modéré** pour le magasin 86
* **Pas d'effet notable** pour le magasin 88

Proposition : Déployer le layout dans les magasins similaires à 77 (panier moyen sensible), réaliser des tests complémentaires sur les profils similaires à 86, et exclure ceux similaires à 88 des prochaines phases de déploiement.