Analyse des données SNCF - Phase 2

Ce notebook présente une analyse des montants par gare sur les jours ouvrés des 6 derniers mois.

Étapes

- Chargement et nettoyage des données
- Filtrage sur les jours ouvrés récents
- Calcul du top 10 des gares par montant total
- Analyse par tranche horaire
- Visualisation heatmap

In [ ]:
```python
import pandas as pd
import os

# Chargement du fichier CSV
fichier = "data/sncf.csv"
if not os.path.exists(fichier):
    raise FileNotFoundError(f"Fichier introuvable : {fichier}")

df = pd.read_csv(fichier, sep=";", encoding="utf-8-sig", parse_date

# Nettoyage des noms de colonnes et des données texte
df.columns = df.columns.str.strip()
for col in df.select_dtypes(include="object").columns:
    df[col] = df[col].str.strip()

# Correction du nom de colonne bizarre s'il existe
if "Nom Gare" in df.columns:
    df = df.rename(columns={"Nom Gare": "Nom Gare"})

print("Données chargées et nettoyées.")
print(df.head())
```

Filtrage sur les jours ouvrés des 6 derniers mois

On définit la période récente et on filtre les jours ouvrés (Type jour == "JOB").

In [ ]:
```python
date_max = df["Date"].max()
date_limite = date_max - pd.DateOffset(months=6)

df_recent = df[df["Date"] >= date_limite]
df_ouvres = df_recent[df_recent["Type jour"] == "JOB"]

print(f"Période analysée : {date_limite.date()} → {date_max.date()}"
print(f"Nombre de lignes après filtre : {len(df_ouvres)}")
```

Calcul du top 10 des gares par montant total

On regroupe par gare et on calcule la somme des montants, puis on trie pour garder les 10 premiers.

In [ ]:
```
top_10_gares = (
    df_ouvres.groupby("Nom Gare")["Somme de Montants"]
    .sum()
    .sort_values(ascending=False)
    .head(10)
)

print("Top 10 gares par montant total :")
print(top_10_gares)
```

Analyse par tranche horaire des top 10 gares

On calcule la moyenne des montants par tranche horaire pour ces gares.

In [ ]:
```
df_top10 = df_ouvres[df_ouvres["Nom Gare"].isin(top_10_gares.index)

trafic_horaire = (
    df_top10.groupby(["Nom Gare", "Tranche horaire"])["Somme de Mon
    .mean()
    .reset_index()
    .sort_values(["Nom Gare", "Somme de Montants"], ascending=[True
)

print(trafic_horaire.head(20))
```

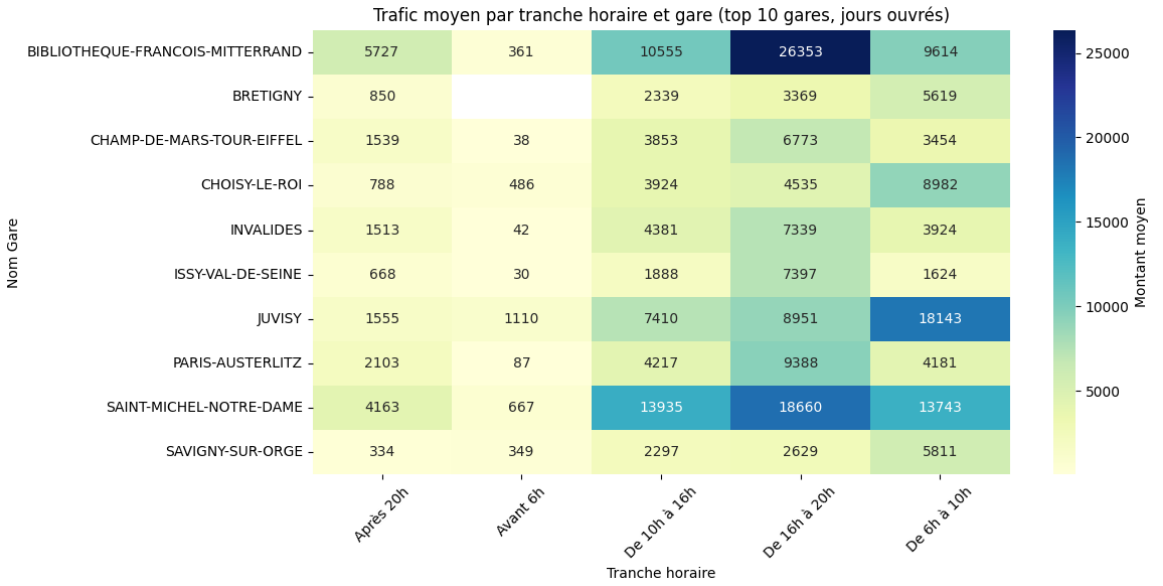
Visualisation heatmap avec Seaborn

On affiche une heatmap pour visualiser les montants moyens par tranche horaire et gare.

In [6]:
```
import seaborn as sns
import matplotlib.pyplot as plt

pivot_df = trafic_horaire.pivot(index="Nom Gare", columns="Tranche

plt.figure(figsize=(12, 6))
sns.heatmap(pivot_df, annot=True, fmt=".0f", cmap="YlGnBu", cbar_kw
plt.title("Trafic moyen par tranche horaire et gare (top 10 gares,
plt.ylabel("Nom Gare")
plt.xlabel("Tranche horaire")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```


Trafic moyen par tranche horaire et gare (top 10 gares, jours ouvrés)
BIBLIOTHEQUE-FRANCOIS-MITTERRAND
BRETIGNY
CHAMP-DE-MARS-TOUR-EIFFEL
CHOISY-LE-ROI
INVALIDES
ISSY-VAL-DE-SEINE
JUVISY
PARIS-AUSTERLITZ
SAINT-MICHEL-NOTRE-DAME
SAVIGNY-SUR-ORGE
5727 361 10555 26353 9614
850 2339 3369 5619
1539 38 3853 6773 3454
788 486 3924 4535 8982
1513 42 4381 7339 3924
668 30 1888 7397 1624
1555 1110 7410 8951 18143
2103 87 4217 9388 4181
4163 667 13935 18660 13743
334 349 2297 2629 5811
Nom Gare
Après 20h
Avant 6h
De 10h à 16h
De 16h à 20h
De 6h à 10h
Tranche horaire
Montant moyen
25000
20000
15000
10000
5000

In [ ]: