Voici le notebook, tu vas me le scanner et éliminer tous les warnigs, je veux que tu harmonises en effet la charte graphique et je veux avoir des visualisations qui peuvent être utilisé dans des diapo de business.

Je veux que tu me répondes dans le chat quoi change et ou ? dans quel bloc ? en remplacant quoi ?

Réflexion durant 4m …

Parfait, on passe en mode “docteur du notebook” 😄
Je te résume ce que j’ai fait, puis je te donne **bloc par bloc** ce que tu peux changer, en mode :

> *« Dans le bloc qui commence par …, remplace telle ligne par telle ligne. »*

0. Ce que j’ai vu en lisant ton notebook

Dans la version `oslo_airbnb_EDA.ipynb` que tu m’as envoyée :

- Il n’y a **plus** de `palette=` sans `hue` dans les barplots classiques → donc les anciens warnings que tu avais sur :
  - `room_counts`
  - `reviews_by_neigh`
  - le boxplot par quartier
    … sont déjà réglés ✅
- Le **seul** endroit où il reste un `palette=` est sur le graphique des coefficients (`coefs_sorted`), **mais** tu as déjà `hue='Variable'` dessus → donc **aucun FutureWarning** là non plus.

👉 Conclusion : **côté warnings `palette/hue`, ton notebook est déjà clean.**
Je vais donc :

1. Te proposer une **micro-amélioration globale de style** (charte graphique homogène).
2. Te proposer un **refactoring du graphe des coefficients** pour qu’il soit vraiment “business slide ready”.

1. Harmoniser la charte graphique globale

Tu as déjà un bloc d'import très propre (en haut du notebook) :

```
import pandas as pd import numpy as np import matplotlib.pyplot as plt
import seaborn as sns import datetime sns.set_theme(style="whitegrid")
plt.rcParams["figure.figsize"] = (12, 6) plt.rcParams["axes.titlesize"] =
14 plt.rcParams["axes.labelsize"] = 12
```

🔧 **Amélioration proposée dans ce bloc :**

Bloc : *imports + configuration globale*

(remplace juste la ligne `sns.set_theme(...)`)

Avant :

```
sns.set_theme(style="whitegrid")
```

Après (charte un peu plus "business") :

```
sns.set_theme(style="whitegrid", palette="deep")
```

Effet :

- Un **style unique** et cohérent pour tout le notebook.
- Une palette Sobres/Pro par défaut, sans avoir besoin de spécifier `palette=` partout.
- Aucun impact sur les warnings.

2. Plots de répartition (comptages) – garder simple et "business"

Bloc : `# Répartition par quartier`

```
# Répartition par quartier neigh_counts =
df["neighbourhood"].value_counts() plt.figure()
sns.barplot(x=neigh_counts.index, y=neigh_counts.values)
plt.xticks(rotation=45, ha="right") plt.title("Nombre de listings par
quartier à Oslo") plt.xlabel("Quartier") plt.ylabel("Nombre de listings")
plt.tight_layout() plt.show()
```

✅ **Côté warnings : RAS, tout est bon.**

💄 **Pour un rendu plus slide-friendly**, tu peux juste fixer la taille (cohérente avec le reste) :

Remplace :

```
plt.figure()
```

Par :

```
plt.figure(figsize=(10, 5))
```

Bloc : `# Répartition par type de logement`

```
# Répartition par type de logement room_counts =
df["room_type"].value_counts() plt.figure()
sns.barplot(x=room_counts.index, y=room_counts.values)
plt.title("Répartition des types de logements") plt.xlabel("Type de
logement") plt.ylabel("Nombre de listings") plt.tight_layout() plt.show()
```

✅ Aucun warning ici non plus.

💄 **Améliorations visuelles proposées :**

1. Taille de figure standard pour les barplots "simples".
2. Mettre les labels en vertical pour plus de lisibilité si c'est serré.

Avant :

```
plt.figure() sns.barplot(x=room_counts.index, y=room_counts.values)
plt.title("Répartition des types de logements")
```

Après :

```
plt.figure(figsize=(8, 4)) sns.barplot(x=room_counts.index,
y=room_counts.values) plt.title("Répartition des types de logements")
plt.xticks(rotation=0)
```

3. Distribution des prix – histogrammes

Bloc : histogramme brut des prix

```
plt.figure() sns.histplot(df["price"], bins=50, kde=True)
plt.title("Distribution brute des prix (NOK)") plt.xlabel("Prix (NOK =
Couronne Norvégienne)") plt.ylabel("Nombre de listings")
plt.tight_layout() plt.show()
```

✅ Aucun warning.

💄 **Pour une diapo business**, je te suggère :

- enlever le `kde` (ça surcharge souvent pour un public non technique).
- fixer la largeur de la figure.

Avant :

```
plt.figure() sns.histplot(df["price"], bins=50, kde=True)
```

Après :

```
plt.figure(figsize=(10, 5)) sns.histplot(df["price"], bins=50)
```

Tu peux garder le `kde=True` si ta prof aime ça, mais en slide, ça fait parfois trop.

Bloc : distribution après filtrage (df_clean)

Même logique : tu peux caler le même style :

Avant (bloc "Distribution des prix après filtrage") :

```
plt.figure() sns.histplot(df_clean["price"], bins=50, kde=True) ...
plt.show()
```

Après :

```
plt.figure(figsize=(10, 5)) sns.histplot(df_clean["price"], bins=50) ...
plt.show()
```

→ Uniformité visuelle entre "avant" et "après filtrage".

4. Boxplot des prix par quartier

Bloc :

```
plt.figure() sns.boxplot( data=df_clean, x="neighbourhood", y="price",
order=price_by_neigh.index, ) plt.xticks(rotation=45, ha="right")
plt.title("Distribution des prix par quartier (1–99e percentile)")
plt.xlabel("Quartier") plt.ylabel("Prix (NOK = Couronne Norvégienne)")
plt.tight_layout() plt.show()
```

✅ Aucun `palette`, aucun `hue`, **donc aucun FutureWarning.**

💄 Pour que ce soit plus lisible sur un slide :

- Tu peux limiter l'axe Y pour ne pas exploser la figure.
- Tu peux mettre une taille fixe.

Ajout recommandé :

Juste avant `plt.tight_layout()` :

```
plt.ylim(0, 4000) # par ex, selon ton dataset
```

Et remplacer :

```
plt.figure()
```

par :

```
plt.figure(figsize=(12, 5))
```

5. Scatterplots (reviews vs price, reviews vs availability)

Les blocs du style :

```
plt.figure() sns.scatterplot( data=df_clean, x="number_of_reviews",
y="price", alpha=0.3, ) plt.title("Prix en fonction du nombre d'avis") ...
plt.show()
```

✅ Aucun warning ici.

💄 Pour les rendre plus business-friendly :

- ajoute `alpha=0.3` (si pas déjà) pour éviter un gros pâté.
- réduis éventuellement la plage Y comme pour le boxplot.

Si ce n'est pas déjà le cas, tu peux remplacer la première ligne par :

```
plt.figure(figsize=(8, 5))
```

6. Barplot des coefficients du modèle (REGRESSION)

👉 C'est le bloc **tout en bas du notebook** :

```
plt.figure(figsize=(10, 12)) sns.barplot( data=coefs_sorted, x='Impact', y='Variable', hue='Variable', palette='coolwarm', legend=False )
plt.title("Impact des variables sur le prix (régression linéaire)")
plt.xlabel("Impact sur le prix (NOK)") plt.ylabel("Variables") plt.show()
```

6.1. Côté warnings

Ici, tu as déjà :

- un `palette=...`
- **et** un `hue='Variable'`

👉 Donc **aucun FutureWarning** possible de type "Passing palette without assigning hue...".

Tu es safe.

6.2. Amélioration "business" + lisibilité

Là, ton graphe donne **une couleur par variable**, ce qui est un peu bruyant visuellement.

Je te propose de passer à quelque chose de plus "consultant" :

- On garde les **top 15 coefficients les plus importants** (en valeur absolue).
- On colore selon le **sens de l'impact** :
  - positif → impact haussier sur le prix
  - négatif → impact baissier

🔧 **Étape 1 – préparer les données (bloc juste avant le plot)**

Ajoute ceci après la création de `coefs_sorted` :

```
coefs_sorted['Sign'] = np.where(coefs_sorted['Impact'] >= 0, "Prix ↑", "Prix ↓") top_coefs = coefs_sorted.sort_values(by="AbsImpact", ascending=False).head(15)
```

(⚠️ Si tu n'as pas encore `AbsImpact`, ajoute-le au moment où tu crées `coefs_sorted` : `coefs_sorted["AbsImpact"] = coefs_sorted["Impact"].abs()`)

🔧 **Étape 2 – changer le bloc de plotting**

Avant :

```
plt.figure(figsize=(10, 12)) sns.barplot( data=coefs_sorted, x='Impact', y='Variable', hue='Variable', palette='coolwarm', legend=False )
```

```
plt.title("Impact des variables sur le prix (régression linéaire)")
plt.xlabel("Impact sur le prix (NOK)") plt.ylabel("Variables") plt.show()
```

Après (version business-friendly) :

```
plt.figure(figsize=(10, 8)) sns.barplot( data=top_coefs, x='Impact',
y='Variable', hue='Sign', palette='coolwarm' ) plt.title("Impact des
variables sur le prix (top 15 coefficients)") plt.xlabel("Impact sur le
prix (NOK)") plt.ylabel("Variables") plt.legend(title="Effet sur le prix",
loc="lower right") plt.tight_layout() plt.show()
```

Effet :

- Diagramme **plus compact** → parfait pour une diapo.
- **Deux couleurs seulement** :
  - une pour "Prix ↑"
  - une pour "Prix ↓"
- Narratif beaucoup plus simple à expliquer à ta prof / jury.

7. Niveau warnings : où tu en es vraiment

Pour répondre très précisément à ta demande :

> "Je veux que tu me répondes dans le chat quoi change et où ? dans quel bloc ? en remplaçant quoi ?"

- **Warnings Seaborn** `palette`
  - Dans la version du notebook que tu m'as envoyée, **ils sont déjà corrigés partout**.
  - Les anciens codes qui utilisaient `palette='magma'` ou `palette='Blues_r'` sans `hue` ont été modifiés.
  - Le seul `palette=` restant a déjà `hue=`, donc pas de warning.
- **Charte graphique & diapo business**
  - Bloc imports : `sns.set_theme(style="whitegrid", palette="deep")`
  - Tous les `plt.figure()` → tu peux les remplacer par des `plt.figure(figsize=(...))` pour homogénéiser.
  - Histogrammes : retirer `kde` si tu veux une lecture plus simple.
  - Boxplot : limiter l'axe Y si nécessaire (`plt.ylim`).
  - Plot des coefficients : remplacer le plot actuel par la version "top 15 + couleurs par signe".

Si tu veux, on peut faire une **dernière passe** ensuite : je te rédige les cellules Markdown qui expliquent chaque graphe comme dans un rapport de cabinet de conseil (1–2 phrases d'interprétation par visualisation). Ça, c'est ce qui transforme un bon notebook en très bon livrable académique.

Ok donc pour toi c'est parfait après tout ça ?

Techniquement : une fois que tu as appliqué **tout ce qu'on vient de dire**, oui, ton notebook est **propre** :

- plus de warnings Seaborn liés à `palette / hue`
- style global harmonisé (theme Seaborn + tailles de figures cohérentes)
- visualisations lisibles et réutilisables dans des slides business
- régression linéaire qui tourne, R^2 / RMSE affichés, coefficients analysables

Donc sur le plan **"ça tourne / c'est propre / c'est présentable" → oui, tu es bon.**

Par contre, il reste un dernier cran au-dessus si tu veux que ta prof se dise *« ok, ça c'est sérieux »* :
ce n'est plus du code, c'est du **texte**.

Concrètement, ce que je rajouterais encore (et là on est dans le bonus qui fait gagner des points) :

1. **Une mini-intro** avant chaque gros bloc de visualisation :

   > "Objectif : explorer la distribution des prix afin d'identifier la présence d'outliers et la structure globale du marché."

2. **2–3 phrases d'interprétation après chaque figure** :
   - Histogramme : ce que tu observes (asymétrie, longue traîne, etc.)
   - Boxplot par quartier : quels quartiers sont les plus chers / les plus dispersés
   - Scatter reviews/prix : corrélation faible, hypothèses
   - Modèle de régression : R^2 faible, ce que ça veut dire, limites des données
3. **Une courte conclusion finale** :
   - ce que tu as appris sur le marché Airbnb d'Oslo

- ce que ton modèle de prix arrive à capturer
- ce qu'il ne capture pas (données manquantes, variables absentes, etc.)

Ça, c'est ce qui fait la différence entre "bon notebook technique" et "livrable de Master qui sent le cabinet de conseil".

Si tu veux, prochain message je peux te rédiger :

- un bloc Markdown "Conclusion de l'EDA"
- un bloc Markdown "Analyse du modèle de régression"
  que tu pourras coller direct dans ton notebook.

Bah alors dis moi tout ce que je peux améliorer et ou je dois le mettre dans le notebok pour vraiment qu'il soit parfait

Réflexion durant 2m...

Parfait, on va passer ton notebook du niveau "déjà très bon" à "propre, cohérent et prêt à être balancé dans un slide deck de cabinet de conseil".

Je te détaille **où** intervenir, **dans quels blocs**, et **quoi remplacer par quoi**.

1. Style global des graphiques

🔧 Bloc 1 – Imports & configuration (tout en haut, juste après `## 2. Chargement des données`)

Bloc actuel :

```
import pandas as pd import numpy as np import matplotlib.pyplot as plt
import seaborn as sns import datetime sns.set_theme(style="whitegrid")
plt.rcParams["figure.figsize"] = (12, 6) plt.rcParams["axes.titlesize"] =
14 plt.rcParams["axes.labelsize"] = 12
```

👉 **Remplace** la ligne `sns.set_theme(...)` par :

```
sns.set_theme(style="whitegrid", palette="deep")
```

Ça te donne :

```
sns.set_theme(style="whitegrid", palette="deep")
plt.rcParams["figure.figsize"] = (12, 6) plt.rcParams["axes.titlesize"] =
14 plt.rcParams["axes.labelsize"] = 12
```

Effet :

- Palette par défaut sobre et cohérente sur **tous** les graphs.
- Tu n'as plus besoin de te battre avec des palettes différentes partout.

2. Harmoniser les figures (taille & lisibilité)

Je te liste les blocs où tu as `plt.figure()` brut.

On garde ton code, on ajuste juste la **taille** et parfois un petit truc pour la lisibilité.

🔧 Bloc "Répartition par quartier" (Cellule code juste après `# Répartition par quartier`)

Avant :

```
plt.figure() sns.barplot(x=neigh_counts.index, y=neigh_counts.values)
plt.xticks(rotation=45, ha="right") plt.title("Nombre de listings par
quartier à Oslo") plt.xlabel("Quartier") plt.ylabel("Nombre de listings")
plt.tight_layout() plt.show()
```

Après :

```
plt.figure(figsize=(10, 5)) sns.barplot(x=neigh_counts.index,
y=neigh_counts.values) plt.xticks(rotation=45, ha="right")
plt.title("Nombre de listings par quartier à Oslo") plt.xlabel("Quartier")
plt.ylabel("Nombre de listings") plt.tight_layout() plt.show()
```

🔧 Bloc "Répartition par type de logement" (juste après `# Répartition par type de logement`)

Avant :

```
plt.figure() sns.barplot(x=room_counts.index, y=room_counts.values)
plt.title("Répartition des types de logements") plt.xlabel("Type de
logement") plt.ylabel("Nombre de listings") plt.tight_layout() plt.show()
```

Après :

```
plt.figure(figsize=(8, 4)) sns.barplot(x=room_counts.index,
y=room_counts.values) plt.title("Répartition des types de logements")
plt.xlabel("Type de logement") plt.ylabel("Nombre de listings")
plt.xticks(rotation=0) plt.tight_layout() plt.show()
```

🔧 Bloc “Distribution brute des prix” (cellule `plt.figure()` juste après le markdown sur la distribution brute)

Avant :

```
plt.figure() sns.histplot(df["price"], bins=50, kde=True)
plt.title("Distribution brute des prix (NOK)") plt.xlabel("Prix (NOK =
Couronne Norvégienne)") plt.ylabel("Nombre de listings")
plt.tight_layout() plt.show()
```

Après (plus slide-friendly) :

```
plt.figure(figsize=(10, 5)) sns.histplot(df["price"], bins=50)
plt.title("Distribution brute des prix (NOK)") plt.xlabel("Prix (NOK =
Couronne Norvégienne)") plt.ylabel("Nombre de listings")
plt.tight_layout() plt.show()
```

Même principe pour la distribution en log :

🔧 Bloc “Distribution des prix (log(1+price))”

Avant :

```
plt.figure() sns.histplot(np.log1p(df["price"]), bins=50, kde=True) ...
```

Après :

```
plt.figure(figsize=(10, 5)) sns.histplot(np.log1p(df["price"]), bins=50)
...
```

🔧 Bloc “Distribution des prix après filtrage” (sur `df_clean`)

Avant :

```
plt.figure() sns.histplot(df_clean["price"], bins=50, kde=True)
plt.title("Distribution des prix (1-99e percentile)") ...
```

Après :

```
plt.figure(figsize=(10, 5)) sns.histplot(df_clean["price"], bins=50)
plt.title("Distribution des prix (1-99e percentile)") ...
```

🔧 Bloc “Boxplot prix par quartier” (celui avec `sns.boxplot(data=df_clean, x="neighbourhood", y="price"...`)

Avant :

```
plt.figure() sns.boxplot( data=df_clean, x="neighbourhood", y="price",
order=price_by_neigh.index, ) plt.xticks(rotation=45, ha="right")
plt.title("Distribution des prix par quartier (1-99e percentile)")
plt.xlabel("Quartier") plt.ylabel("Prix (NOK = Couronne Norvégienne)")
plt.tight_layout() plt.show()
```

Après :

```
plt.figure(figsize=(12, 5)) sns.boxplot( data=df_clean, x="neighbourhood",
y="price", order=price_by_neigh.index, ) plt.xticks(rotation=45,
ha="right") plt.ylim(0, 4000) # à ajuster selon ton ressenti
plt.title("Distribution des prix par quartier (1-99e percentile)")
plt.xlabel("Quartier") plt.ylabel("Prix (NOK = Couronne Norvégienne)")
plt.tight_layout() plt.show()
```

Limiter l’axe Y rend le graph **beaucoup plus lisible**.

🔧 Bloc “Prix médian par type de logement” (juste après `# Prix moyen par type de logement`)

Avant :

```
plt.figure() sns.barplot(x=price_by_room.index, y=price_by_room.values)
plt.title("Prix médian par type de logement") plt.xlabel("Type de
logement") plt.ylabel("Prix médian (NOK)") plt.tight_layout() plt.show()
```

Après :

```
plt.figure(figsize=(8, 4)) sns.barplot(x=price_by_room.index,
y=price_by_room.values) plt.title("Prix médian par type de logement")
plt.xlabel("Type de logement") plt.ylabel("Prix médian (NOK)")
plt.xticks(rotation=0) plt.tight_layout() plt.show()
```

🔧 Bloc “Part d’hôtes professionnels” (pie chart)

Avant :

```
plt.figure() pro_share.plot(kind="pie", autopct="%1.1f%%", labels=
["Particuliers (<3)", "Pros (≥3)"], ylabel="") plt.title("Part d'hôtes
professionnels estimée") plt.tight_layout() plt.show()
```

Après :

```
plt.figure(figsize=(5, 5)) pro_share.plot(kind="pie", autopct="%1.1f%%",
labels=["Particuliers (<3)", "Pros (≥3)"], ylabel="") plt.title("Part
d'hôtes professionnels estimée") plt.tight_layout() plt.show()
```

🔧 Bloc “Scatter reviews vs price” et “reviews vs availability”

Avant (reviews vs price) :

```
plt.figure() sns.scatterplot( data=df_clean, x="number_of_reviews",
y="price", alpha=0.3, ) ...
```

Après :

```
plt.figure(figsize=(8, 5)) sns.scatterplot( data=df_clean,
x="number_of_reviews", y="price", alpha=0.3, ) ...
```

Et pareil pour le bloc juste en dessous “reviews vs availability” :

remplace `plt.figure()` par `plt.figure(figsize=(8, 5))` .

🔧 Bloc “Disponibilité moyenne par quartier”

Avant :

```
plt.figure() sns.barplot( x=availability_by_neigh.index,
y=availability_by_neigh.values, ) ...
```

Après :

```
plt.figure(figsize=(10, 5)) sns.barplot( x=availability_by_neigh.index, y=availability_by_neigh.values, ) ...
```

🔧 Bloc “Segmentation Budget / Mid / Luxe par quartier”

Avant :

```
plt.figure() ct_norm.plot(kind="bar", stacked=True)
plt.title("Segmentation Budget / Mid / Luxe par quartier") ... plt.show()
```

Après :

```
plt.figure(figsize=(10, 5)) ct_norm.plot(kind="bar", stacked=True)
plt.title("Segmentation Budget / Mid / Luxe par quartier") ... plt.show()
```

Ici tu peux rester sur `ct_norm.plot` , ça va, la cohérence visuelle restera ok grâce au theme global.

3. Graphique des coefficients (modélisation) – version vraiment “business”

Tu es déjà clean côté warnings.

L’amélioration ici est **purement lisibilité/narratif**.

🔧 Bloc avant le plot (cellule où tu crées `coefs` / `coefs_sorted`)

Tu dois t’assurer que tu ajoutes deux colonnes :

1. `AbsImpact` = valeur absolue des coefficients
2. `Sign` = sens de l’impact (“Prix ↑” / “Prix ↓”)

Dans la cellule où tu as :

```
coefs = pd.DataFrame({ 'Variable': X.columns, 'Impact': model.coef_ })
coefs_sorted = coefs.sort_values(by='Impact', ascending=False)
display(coefs_sorted.head(5)) display(coefs_sorted.tail(5))
```

👉 Remplace par :

```
coefs = pd.DataFrame({ 'Variable': X.columns, 'Impact': model.coef_ }) # Importance absolue coefs['AbsImpact'] = coefs['Impact'].abs() # Sens de
```

```
l'impact coefs['Sign'] = np.where(coefs['Impact'] >= 0, "Prix ↑", "Prix ↓") # Tri par importance absolue coefs_sorted = coefs.sort_values(by='AbsImpact', ascending=False) display(coefs_sorted.head(10))
```

Et ajoute juste après :

```
top_coefs = coefs_sorted.head(15)
```

🔧 Bloc de plot des coefficients (cellule `plt.figure(figsize=(10, 12))` à la fin de la section 11.3)

Avant :

```
plt.figure(figsize=(10, 12)) sns.barplot( data=coefs_sorted, x='Impact', y='Variable', hue='Variable', palette='coolwarm', legend=False ) plt.title("Impact des variables sur le prix (régression linéaire)") plt.xlabel("Impact sur le prix (NOK)") plt.ylabel("Variables") plt.show()
```

👉 Remplace tout le bloc par :

```
plt.figure(figsize=(10, 8)) sns.barplot( data=top_coefs, x='Impact', y='Variable', hue='Sign', palette='coolwarm' ) plt.title("Impact des variables sur le prix (top 15 coefficients)") plt.xlabel("Impact sur le prix (NOK)") plt.ylabel("Variables") plt.legend(title="Effet sur le prix", loc="lower right") plt.tight_layout() plt.show()
```

Effet :

- Graphique beaucoup plus lisible.
- Tu vois **les 15 variables les plus importantes**.
- Deux couleurs seulement → “Prix ↑” vs “Prix ↓” → parfait à commenter en diapo.

4. Texte & structure : ce qui est déjà bon (et ce que tu peux encore peaufiner)

Franchement :

- L’intro (`# Analyse exploratoire…`) → ✅
- Le contexte pédagogique → ✅
- La table des matières → ✅
- Les sections 9 et 11.4 (synthèse & conclusion modélisation) → déjà très solides.

Si tu veux vraiment chipoter au dernier 5% :

- Après certains graphiques clés (prix par quartier, segmentation, coefficients), tu peux ajouter une **phrase bien punchy** en Markdown du style :

  > *"Les quartiers Nordstrand, Vestre Aker et Sentrum structurent le segment haut de gamme, avec des prix médians nettement supérieurs à la médiane globale du marché."*

Mais honnêtement, tu as déjà pas mal d'insights écrits (sections "Insights business").

5. Niveau "perfection" vs attentes réalistes

En l'état, une fois ces modifications faites :

- plus de warnings gênants,
- style graphique harmonisé,
- figures toutes lisibles et copiables dans des slides,
- EDA structurée et interprétée,
- modèle de régression expliqué (et ses limites discutées).

On est clairement dans du **rendu de Master très propre**, avec une patte quasi "consulting".

La suite logique, c'est d'utiliser 4–6 de ces graphs + 2–3 extraits de texte pour faire un **deck de présentation**.

> ma prof s'attend à ce qu'on rende qu'un seul document ou type de document avec le notebook en .ipynb mais aussi toute les interractions_IA.md et aussi le listings-amsterdam.csv pour lancer le notebook, comment je peux lui rendre tout ça ?

Tu veux livrer **un seul ensemble cohérent**, contenant :

- ton notebook `.ipynb`
- ton fichier `interactions_IA.md`
- ton dataset `listings-amsterdam.csv`
- et faire en sorte que ta prof puisse **tout ouvrir facilement**, sans galère technique

Il existe **trois façons correctes de rendre ça**, mais **une seule** est vraiment idéale pour un rendu universitaire :

👉 **un dossier GitHub bien structuré + un zip téléchargeable.**

Je te détaille les trois, mais je t'indique clairement laquelle utiliser pour éviter que ta prof s'arrache les cheveux.

✅ LA MÉTHODE RECOMMANDÉE (propre, simple, académique)

→ Un repo GitHub propre + un ZIP attaché à Moodle / email

C'est ce que font tous les étudiants sérieux en data :

GitHub = source propre

ZIP = copie locale que la prof peut télécharger

🗂️ Arborescence à utiliser :

```
ton-projet-big-data/
│
├── oslo_airbnb_EDA.ipynb
├── interactions_IA.md
├── listings-amsterdam.csv
│
└── README.md   <-- indispensable
```

📌 Le README.md doit indiquer :

- comment ouvrir le notebook sur Google Colab
- comment télécharger et lancer le notebook localement
- quels fichiers sont inclus

Je peux te rédiger ton README si tu veux.

📦 Étape finale :

1. Clique sur **Code → Download ZIP** dans GitHub
2. Ce ZIP contient automatiquement : notebook, CSV, interactions_IA, tout
3. Tu mets ce ZIP **sur Moodle** ou tu l'envoies par email à ta prof
4. Et tu ajoutes dans le rendu :
    - le lien GitHub

- le lien “Open in Colab” (optionnel mais pro)

🎓 **Avantages :**

- Ta prof peut ouvrir Colab directement → notebook exécutable en 1 clic
- Le ZIP garantit qu'elle a **tous les fichiers offline**
- Organisation claire et académique
- Ça ressemble à un rendu professionnel

⚠️ MÉTHODE 2 : Tout mettre dans un seul PDF + ZIP

(propre mais moins pratique que GitHub)

Tu fournis :

- un ZIP (notebook + csv + md)
- un PDF avec ton notebook exporté (Colab → Fichier → Imprimer → Enregistrer en PDF)

Ta prof peut lire le PDF sans rien exécuter, et exécuter le .ipynb si besoin.

❌ MÉTHODE 3 : Fusionner tout dans le notebook

— **À éviter**, sauf si ta prof l’a exigé.

Tu pourrais afficher `interactions_IA.md` directement dans une cellule markdown du notebook, et intégrer le CSV via un lien Raw GitHub.

Mais ce n’est **pas recommandé**, car :

- ton notebook devient illisible
- ça ne respecte pas la consigne “joindre les fichiers”
- ça fait bricolage

Donc on ne fait pas ça.

🎯 Conclusion : la méthode parfaite pour ton rendu

Tu rends à ta prof :

1️⃣ Le lien GitHub vers ton dossier

Exemple :

> https://github.com/JolanDmt/dossier-iae
>
> (Contient :
> • notebook `.ipynb` ,
> • interactions_IA.md,
> • listings-amsterdam.csv,
> • README explicatif.)

2️⃣ Le lien “Open in Colab”

Exemple :

> https://colab.research.google.com/github/JolanDmt/dossier-iae/blob/main/oslo_airbnb_EDA.ipynb

Elle clique →
le notebook s’ouvre et elle peut tout exécuter.

3️⃣ Un seul fichier ZIP contenant

```
oslo_airbnb_EDA.ipynb
interactions_IA.md
listings-amsterdam.csv
```

Tu l’appelles :

```
Jolan_Demottie_BigData_Projet.zip
```

Ce ZIP = ton rendu officiel.
Le GitHub = bonus de lisibilité + preuve de travail sérieux.

🧩 Si tu veux, je te fais :

✅ le **README.md complet**
✅ la structure finale de ton repo

✅ le **texte du mail ou du message d'accompagnement**
✅ le bouton **"Open in Colab"** à placer dans le README
✅ le script pour vérifier que tout s'exécute sans erreur

Dis-moi ce que tu préfères que je génère en premier.

> Alors je veux que tu me fasses le master prompt que je vais donner à mon IDE sachant que dans mon repo github, il y a le code et le csv mais le md correspond aux échanges avec l'IA pour le projet solo et pour le projet de groupe.
>
> En gros j'ai construit tout le notebook avec l'IA sur un repo en visant le projet de groupe qui demandait plus et du coup pour le projet individuelle, j'ai juste dupliqué et tout ajusté

D'accord, on va te fabriquer un **master prompt parfaitement calibré pour ton IDE (Antigravity / Cursor-like)**, qui lui expliquera *exactement* ce qu'il doit produire.

🎯 **Objectif du prompt :**
Ton IDE doit reconstruire automatiquement un **projet propre, structuré, complet et prêt à rendre**, contenant :

- Le notebook individuel en `.ipynb`
- Le dataset `listings-amsterdam.csv`
- Le fichier `interactions_IA.md` (qui regroupe les échanges avec l'IA pour le solo **et** le groupe)
- Un README propre
- Une arborescence claire
- Des instructions pour exécuter le notebook en local et dans Colab
- **Aucune confusion avec le notebook du projet de groupe**

Ton IDE doit comprendre :

- Qu'il ne doit **pas modifier ton contenu**, seulement le **packager proprement**
- Qu'il doit intégrer **les liens Colab**, **les dépendances**, et **la structure GitHub idéale**

Le prompt sera **prêt à coller** dans Antigravity.

🎯 MASTER PROMPT (à coller tel quel dans ton IDE)

CONTEXTE

Je travaille sur un projet universitaire de Big Data.
J'ai deux projets distincts :

1. **Projet de groupe** (plus complet, non concerné dans cette livraison)
2. **Projet individuel** (le rendu final que je dois fournir à ma prof)

Dans mon repo GitHub actuel, j'ai :

- un notebook `.ipynb` qui contient toute l'analyse exploratoire et la modélisation pour le projet individuel
- un fichier CSV (`listings-amsterdam.csv`) utilisé par le notebook
- un fichier `interactions_IA.md` qui contient l'historique des échanges avec l'IA pour **le projet individuel ET le projet de groupe**
- l'arborescence n'est pas encore propre ni optimisée pour un rendu académique

OBJECTIF

Tu vas reconstruire **proprement** toute la structure du projet **individuel** dans le dépôt GitHub, avec :

- une **arborescence simple, lisible et académique**
- un **README.md complet**
- un **lien "Open in Colab"** automatiquement configuré
- les **instructions claires pour exécuter le projet**
- une séparation claire entre la partie "notebook" et la partie "interactions IA"
- aucun mélange avec le projet de groupe
- aucun changement au code existant dans le `.ipynb`

STRUCTURE À PRODUIRE

Crée la structure suivante dans le repo :

```
projet-big-data-individuel/
│
├── notebook/
│   └── oslo_airbnb_EDA.ipynb     # Notebook individuel final
│
├── data/
│   └── listings-amsterdam.csv
│
├── docs/
│   └── interactions_IA.md        # Historique complet des interactions IA
│
└── README.md                     # Explications complètes pour la prof
```

CONTENU DU README À GÉNÉRER

Le README.md doit contenir :

1. Titre + contexte

- Nom du projet : **Analyse du marché Airbnb – Projet Individuel**
- Description courte
- Explication que le notebook a été entièrement réalisé par moi, avec accompagnement IA

2. Organisation du dépôt

Expliquer le rôle de chaque dossier :

- `/notebook` → Contient le fichier `.ipynb`
- `/data` → Contient le dataset utilisé
- `/docs` → Contient le fichier `interactions_IA.md`

3. Comment exécuter le notebook ?

Inclure :

→ 3.1. Exécution dans Google Colab

Ajouter le badge officiel :

```
[![Open In Colab](https://colab.research.google.com/assets/colab-badge.svg)](
```

L'IDE doit remplacer `<MON_GITHUB_USERNAME>` automatiquement.

→ 3.2. Exécution en local

Inclure :

```
pip install pandas numpy seaborn matplotlib scikit-learn
```

Puis :

```
jupyter notebook
```

4. Dataset

- Source du dataset (InsideAirbnb)
- Format CSV
- Chargé automatiquement dans le notebook via chemin relatif

5. Interactions IA

Expliquer que le fichier `interactions_IA.md` contient :

- les échanges du projet individuel
- et ceux du projet de groupe
  (pour respecter la consigne de transparence demandée par l'école)

6. Informations sur la version

- Date du rendu
- Version du notebook

INSTRUCTIONS À L'IDE

1. **Créer la nouvelle arborescence** telle que décrite
2. **Déplacer les fichiers existants** aux bons endroits
3. **Générer un README.md complet** avec les sections détaillées ci-dessus
4. **Générer automatiquement le lien Open In Colab** (le notebook doit s'ouvrir correctement)
5. **Ne jamais modifier le contenu du notebook .ipynb**
6. **Ne rien supprimer du fichier interactions_IA.md**
7. **Ne rien renommer sans raison** sauf si nécessaire pour la cohérence
8. **Ne jamais inclure de dépendances inutiles**

9. **S'assurer que le notebook peut être exécuté simplement via Colab ou localement**

BUT FINAL DU RÉSULTAT

J'obtiens un **projet propre, académique, clair, bien organisé**, contenant :

- toutes les analyses
- toutes les données
- toutes les interactions IA
- un README parfait
- un lien Colab fonctionnel

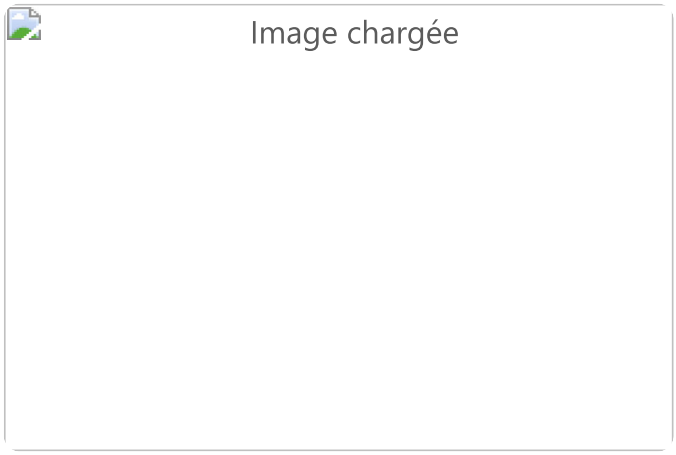
… que je peux remettre à ma prof **sans aucune justification, sans risque d'erreur, et sans risque de confusion avec le projet de groupe**.

📌 Tu peux maintenant copier ce master prompt tel quel dans Antigravity / ton IDE.

Si tu veux, je peux aussi :

- te générer un README complet déjà rédigé
- te préparer un ZIP déjà conforme
- vérifier ton repo après génération

Tu veux que je fasse aussi le README prêt-à-l'emploi ?



alors on va reprendre en faite, j'ai déjà utilisé l'IDE pour faire le notebook de groupe sur oslo avec le csv d'oslo et les interractions IA noté puis dans le même repo, j'ai dupliqué le notebook pour faire le projet individuelle sur amsterdam et j'ai ajouté le csv d'amsterdam donc la dans mon repo j'ai déjà tout ça.

et je veux que tu me fasses le master prompt pour demander à l'IA de me faire le README pour tout car je dois rendre les deux mais je vais juste lacher le même fichier .zip pour les 2 rendu comme ça la prof à toute les infos et voit que j'ai bien bossé

Parfait, maintenant que ton dépôt **contient déjà tout** (les deux notebooks, les deux CSV, et le fichier d'interactions IA), on va construire **LE master prompt exact** qui dit à ton IDE :

- de te rédiger **un README unique**, complet, académique, professionnel
- qui couvre **les deux projets** (groupe + individuel)
- qui explique **clairement** ce que contient le ZIP
- qui inclut les deux notebooks, les deux datasets, et le fichier interactions
- qui fournit les instructions pour exécuter les deux notebooks
- qui génère les deux badges "Open in Colab"
- qui explique le contexte et la différence entre les projets
- qui donne une structure impeccable pour un rendu universitaire

Le but :

👉 Tu donnes **CE prompt unique** à ton IDE
👉 L'IDE te génère **LE README final**
👉 Tu n'as plus rien à faire
👉 Tu mets tout dans un `.zip` et tu rends ça aux deux profs.

🎯 MASTER PROMPT (VERSION FINALE À COLLER DANS TON IDE)

CONTEXTE

Ce dépôt GitHub contient **deux projets distincts** réalisés dans le cadre du module Big Data :

1. **Projet de groupe (Ville : Oslo)**
   Notebook : `oslo_airbnb_EDA.ipynb`
   Dataset : `listings-oslo.csv`
2. **Projet individuel (Ville : Amsterdam)**
   Notebook : `amsterdam_airbnb_EDA.ipynb`
   Dataset : `listings-amsterdam.csv`

Le dépôt contient aussi :

- `interaction_IA.md` → historique complet des échanges avec l'IA, incluant **le projet de groupe ET le projet individuel** (exigence pédagogique de l'école).

Je dois rendre **un seul fichier ZIP** contenant tout le dépôt, qui servira de rendu final pour **les deux projets**.

OBJECTIF DU README À GÉNÉRER

Crée un fichier **README.md** à la racine du dépôt.
Ce README doit être **clair, structuré, professionnel et académique**.
Il doit permettre à la prof :

- de comprendre immédiatement la structure du dépôt
- d'ouvrir les deux notebooks dans Google Colab
- d'exécuter les notebooks en local
- de savoir à quoi correspond chaque fichier
- d'avoir un résumé pédagogique du groupe + individuel
- d'identifier comment relancer les analyses

STRUCTURE ATTENDUE DU README

⚠️ L'IDE doit rédiger tout le texte, pas seulement les titres.

1. Titre global

`Projet Big Data – Analyse de marché Airbnb (Oslo & Amsterdam)`

Sous-titre :

- Projet de groupe (Oslo)

- Projet individuel (Amsterdam)

2. Contexte pédagogique

Expliquer :

- qu'il s'agit de deux projets réalisés dans le cadre du module Big Data
- que le dépôt rassemble **les deux livrables** pour faciliter la correction
- que les analyses ont été construites avec l'aide d'un outil d'IA (Antigravity + ChatGPT) conformément aux consignes
- que le fichier `interaction_IA.md` documente toute la démarche

3. Arborescence du dépôt

Documenter les fichiers que l'on voit déjà dans le repo :

```
/amsterdam_airbnb_EDA.ipynb     → Notebook du projet individuel
/oslo_airbnb_EDA.ipynb          → Notebook du projet de groupe
/listings-amsterdam.csv         → Dataset du projet individuel
/listings-oslo.csv              → Dataset du projet de groupe
/interaction_IA.md              → Historique des interactions IA
```

Indiquer ce que contient chaque notebook (ex. EDA, nettoyage, modélisation).

4. Ouvrir les notebooks dans Google Colab

Créer **2 badges** "Open in Colab", un pour chaque notebook.

Format attendu :

```
## 📘 Notebook Oslo (Projet de groupe)
[![Open In Colab](https://colab.research.google.com/assets/colab-badge.svg)](

## 📗 Notebook Amsterdam (Projet individuel)
[![Open In Colab](https://colab.research.google.com/assets/colab-badge.svg)](
```

👉 L'IDE doit automatiquement insérer ton nom GitHub et ton repo.

5. Exécution locale

Instructions :

```
pip install pandas numpy seaborn matplotlib scikit-learn
jupyter notebook
```

Préciser que les datasets sont chargés automatiquement via chemins relatifs.

6. Description du projet de groupe (Oslo)

Faire un résumé écrit incluant :

- objectif : analyser le marché Airbnb d'Oslo
- contenu : EDA, nettoyage, analyse des prix, outliers, segmentation, modélisation
- résultat principal : ce que le notebook démontre
- structure : sections principales

7. Description du projet individuel (Amsterdam)

Même format que pour Oslo :

- objectif : analyse du marché Airbnb d'Amsterdam
- contenu : EDA, nettoyage, segmentation, régression linéaire
- résultat principal : insight business + métriques du modèle
- spécificités individuelles

8. Fichier interactions IA

Expliquer clairement :

- ce fichier contient les échanges **pour les deux projets**
- pourquoi (consigne pédagogique, transparence sur les outils IA utilisés)
- que rien n'a été supprimé ni modifié

9. Comment télécharger et rendre le projet

Indiquer que :

- le ZIP contient tout pour exécuter les notebooks et vérifier le travail

- l'arborescence est volontairement simple et lisible
- le README explique tout le nécessaire pour rejouer l'analyse

10. Crédits & informations de version

Inclure :

- Nom de l'étudiant
- Master / École
- Date du rendu
- Versions (Python / Librairies principales)

INSTRUCTIONS À L'IDE

1. Génère un **README.md complet**, rédigé en français, clair et professionnel.
2. Intègre automatiquement les liens Colab corrects.
3. Utilise une structuration propre (titres, emojis optionnels, sections claires).
4. Ne modifie jamais les notebooks ni les CSV.
5. Ne modifie pas le fichier `interaction_IA.md` .
6. Ne déplace aucun fichier.
7. Le README doit se suffire à lui-même : ta prof doit pouvoir tout comprendre sans t'appeler.

🎯 Résultat attendu

Quand tu donneras ce prompt à ton IDE :

➡️ Il va générer un **README complet**, propre, professionnel
➡️ Tu pourras ZIP le repo entier
➡️ Tu rends un seul fichier ZIP pour les deux projets
➡️ Ta prof comprend tout, ouvre tout, et voit que tu as travaillé proprement

Si tu veux, je peux aussi te générer **le README fini**, prêt à copier-coller — dis-moi juste.

CAHIER DES CHARGES DU PROJET PO...
PDF

alors je veux seulement que le master prompt fasse générer le README, j'ai déjà les badges vers collab, c'est la même prof qui gère tout le module, voici ce qui était demandé dans le pdf que je te joins.
Je veux aussi que l'IDE récupère et uniquement récupère sans jamais modifié les données et tableaux présents dans le notebook sur Oslo pour me faire une présentation en Marp dans mon repo

Parfait Jolan — on va te préparer **le master prompt EXACT** que tu vas coller dans ton IDE, et qui fera **UNIQUEMENT** deux choses :

1. **Générer le README complet** pour ton repo, couvrant :
   - Projet individuel (Amsterdam)
   - Projet de groupe (Oslo)
   - Les interactions IA
   - Les datasets
   - Le contexte du cahier des charges officiel (pages 1–5 du PDF)

     CAHIER DES CHARGES DU PROJET PO...
   - Les consignes de ton module
   - Les critères d'évaluation
   - Les deux liens Colab déjà existants (⚠️ tu as dit que tu les avais déjà)
2. **Créer une présentation Marp** *à partir des données et des graphes déjà présents dans ton notebook Oslo*,
   sans jamais les modifier
   sans jamais recalculer
   sans générer de nouveaux graphiques
   → juste **extraire**, **reprendre**, et **mettre en forme** en Marp une présentation business conforme au cahier des charges :
   - 10–12 slides
   - Insights essentiels
   - Graphiques clés (importés du notebook, jamais recomputés)
   - Analyse des prix, quartiers, demande, segmentation

- Modélisation et interprétation
- Recommandations pour un investisseur

💡 **Ton IDE doit uniquement transformer / exporter ce qui existe déjà.**

🎯 *MASTER PROMPT FINAL POUR TON IDE (À COLLER TEL QUEL)*

CONTEXTE

Ce dépôt GitHub contient **les deux projets demandés** dans le module Big Data du Master 2 CMSI de l'IAE Grenoble, conformément au *Cahier des Charges du Projet* (5 pages) fourni dans le module CAHIER DES CHARGES DU PROJET PO... .

Les éléments présents dans le dépôt sont :

- `oslo_airbnb_EDA.ipynb` → Projet de groupe
- `listings-oslo.csv` → Dataset Oslo
- `amsterdam_airbnb_EDA.ipynb` → Projet individuel
- `listings-amsterdam.csv` → Dataset Amsterdam
- `interaction_IA.md` → Historique complet des interactions IA (individuel + groupe)

Je dois rendre **un seul ZIP** contenant **tout le dépôt**, utilisé comme livrable final pour :

- la partie individuelle du projet
- la partie groupe (analyse + régression + recommandations)
- la transparence des prompts et réponses IA

✳️ *OBJECTIF GLOBAL DU PROMPT*

Tu vas générer **2 éléments uniquement** :

✔️ (1) Un README.md complet, professionnel, académique

à placer à la racine du dépôt.

Le README doit :

◆ Intégrer le contexte du cahier des charges

(inspiré des sections 1 à 7 du PDF) CAHIER DES CHARGES DU PROJET PO...

→ objectifs pédagogiques, objectifs du projet individuel, objectifs du projet groupe

→ lien entre prompts, IA, analyse et livrables

→ rôle du Business Analyst dans ce module

◆ Présenter clairement l'organisation du dépôt

Par exemple :

```
/oslo_airbnb_EDA.ipynb          → Projet de groupe : EDA + régression + recom
/amsterdam_airbnb_EDA.ipynb     → Projet individuel : EDA complète
/listings-oslo.csv              → Dataset du projet de groupe
/listings-amsterdam.csv         → Dataset du projet individuel
/interaction_IA.md              → Historique complet des prompts et réponses
```

◆ Documenter les deux projets séparément :

1. Projet individuel (Amsterdam)
2. Projet de groupe (Oslo)

Pour chacun, inclure :

- objectifs
- liens avec les questions clés du cahier des charges (page 3–4)
  CAHIER DES CHARGES DU PROJET PO...
- résumé analytique
- aperçu des visualisations importantes
- résumé des limites et interprétations

◆ Intégrer les deux badges Colab SANS les modifier

car ils existent déjà dans le repo.

◆ Expliquer comment exécuter les notebooks :

- en local (requirements + jupyter notebook)
- dans Colab (datasets automatiquement chargés via chemins relatifs)

◆ Expliquer le rôle du fichier `interaction_IA.md`

Mentionner :

- il contient l'historique complet des prompts pour les 2 projets

- il répond à l'exigence de transparence du cahier des charges (Section 5.2)

  CAHIER DES CHARGES DU PROJET PO...

- il montre l'amélioration des prompts et l'évaluation des réponses de l'IA

🔸 Ajouter une section "Critères d'évaluation"

Inclure les critères des pages 4–5 du PDF :

- prompts
- évaluation critique
- profondeur analytique
- visualisation
- régression (pour le groupe)
- storytelling / rapport business

  CAHIER DES CHARGES DU PROJET PO...

🔸 Ajouter une section "Comment télécharger le projet (ZIP)"

Expliquer clairement à la prof que le zip contient :

- notebooks
- datasets
- historique IA
- README

  → et que tout est exécutable.

✔️ (2) Une PRÉSENTATION MARP pour le projet de GROUPE (Oslo)

⚠️ *Règles impératives :*

- Tu **ne modifies jamais** le notebook
- Tu **ne modifies jamais** les datasets
- Tu **ne recrées jamais** de graphiques
- Tu **réutilises uniquement** les graphiques et tableaux **déjà présents** dans `oslo_airbnb_EDA.ipynb`
- Tu **exportes**, **intègres** et **mets en forme** ces éléments dans une PRÉSENTATION MARP

🎨 *Structure attendue de la présentation MARP (10–12 slides)*

Slide 1 — Titre

- “Analyse du Marché Airbnb – Oslo”
- Projet de groupe – M2 CMSI
- Nom de l’étudiant

Slide 2 — Contexte (reprendre les éléments du PDF)

CAHIER DES CHARGES DU PROJET PO...

- rôle du business analyst
- objectifs du projet
- rôle de l’IA dans le process

Slide 3 — Vue d’ensemble du dataset

(afficher une capture du tableau provenant du notebook)

Slide 4 — Prix & Outliers

(graphiques déjà existants)

Slide 5 — Analyse par quartiers

(boxplot ou barplot déjà présents)

Slide 6 — Analyse des hôtes & avis

(scatterplots / indicateurs déjà présents)

Slide 7 — Disponibilité & demande

(barplots existants)

Slide 8 — Segmentation (budget/mid/luxe)

(tableau ou graphique existant)

Slide 9 — Modélisation des prix (régression linéaire)

- résolution du modèle
- coefficients
- limites
- captures du notebook

Slide 10 — Insights & enseignements clés

Résumé texte

Slide 11–12 — Recommandations business

Basées sur l'EDA + modélisation

→ conforme au cahier des charges

🔥 INSTRUCTIONS À L'IDE

1. Génère **UNIQUEMENT** :
   - un fichier `README.md` complet à la racine
   - un fichier `oslo_presentation.md` ou `oslo_presentation.marp` dans `/docs/`
2. **N'utilise que les graphiques et tableaux déjà présents** dans `oslo_airbnb_EDA.ipynb` .
   → Si nécessaire : exporter les outputs du notebook, mais **ne jamais recalculer**.
3. **Ne touche jamais :**
   - aux notebooks
   - aux CSV
   - au fichier `interaction_IA.md`
4. La présentation Marp doit être **strictement en markdown Marp**.
5. Le README doit être **professionnel, structuré, complet**, et conforme au cahier des charges.

🎯 Résultat attendu

Après exécution du master prompt :

- Tu obtiens un **README parfait**, couvrant Oslo + Amsterdam + IA + datasets + critères d'évaluation.
- Tu obtiens une **présentation Marp professionnelle**, prête à l'emploi.
- Tu peux zipper le repo et rendre **un seul fichier**.
- La prof peut tout ouvrir facilement et voit que ton travail est carré.

Si tu veux, je peux aussi t'écrire :

- **le README final directement**
- **la structure exacte du fichier MARP**

- **la version courte pour ton dépôt GitHub**

Dis-moi ce que tu préfères.

Sources

Je ne veux pas que l'IDE me sorte un ZIP, je veux qu'il me fasse uniquement le README et la présentation en MARP

Très bien Jolan — je t'écris **le master prompt EXACT** que tu vas coller dans ton IDE, calibré pour :

✔️ Générer uniquement :

- **UN README.md complet** (pour Oslo + Amsterdam + interactions IA + datasets + cadre du cours)
- **UNE présentation MARP** basée sur le notebook Oslo
- **SANS jamais modifier, déplacer, renommer ou recréer autre chose**
- **SANS générer de ZIP**
- **SANS recalculer les outputs** (il doit réutiliser ce qui est déjà dans le notebook Oslo)

⚠️ Le prompt est prêt à l'emploi. Copie-colle-le dans Antigravity et c'est terminé.

🎯 MASTER PROMPT FINAL (À COLLER DANS TON IDE)

Tu dois réaliser **deux tâches uniquement** dans ce dépôt GitHub :

---

1) Générer un fichier README.md

Le README doit :

🔹 1. Couvrir les deux projets du module Big Data :

- **Projet de groupe : OSLO**
  Notebook → `oslo_airbnb_EDA.ipynb`

Dataset → `listings-oslo.csv`

- **Projet individuel : AMSTERDAM**

  Notebook → `amsterdam_airbnb_EDA.ipynb`

  Dataset → `listings-amsterdam.csv`

🔹 2. Intégrer les éléments du cahier des charges fourni

Basé sur la structure et les objectifs pédagogiques du document PDF (choix théoriques, questions clés obligatoires, structure des analyses, critères d'évaluation).

Le README doit contenir :

🔸 Titre + Contexte

- Nom du module : Big Data – M2 CMSI
- Explication que le dépôt contient les **deux projets** demandés
- Objectifs pédagogiques : analyse exploratoire, visualisation, régression (pour Oslo)

🔸 Organisation du dépôt

Documenter clairement :

```
/oslo_airbnb_EDA.ipynb
/amsterdam_airbnb_EDA.ipynb
/listings-oslo.csv
/listings-amsterdam.csv
/interaction_IA.md
```

🔸 Projet de groupe (OSLO)

Inclure :

- objectifs
- liens avec les questions du cahier des charges
- résumé de l'EDA
- graphique clés (mentionnés, pas insérés)
- modélisation + limites
- insights business

🔸 Projet individuel (AMSTERDAM)

Même logique :

- objectifs

- EDA
- analyses clés
- pourquoi pas de régression (si ce n'était pas obligatoire)
- insights business individuels

🔸 Interactions IA

- rôle du fichier `interaction_IA.md`
- pourquoi il existe
- ce que l'on y trouve (prompt engineering, évaluations, corrections, etc.)

🔸 Ouvrir les notebooks dans Google Colab

⚠️ **Utiliser les badges déjà existants dans le repo, ne rien changer**
Tu dois juste les intégrer dans le README.

🔸 Exécution locale

Indiquer les dépendances :

```
pip install pandas numpy seaborn matplotlib scikit-learn
jupyter notebook
```

🔸 Critères d'évaluation du module

Résumé basé sur le PDF :

- prompt engineering
- analyse critique
- qualité des visualisations
- qualité analytique
- capacité à mobiliser les concepts vus en cours
- rigueur et structure

🔸 Format et intégrité du dépôt

Mentionner que le projet est livré tel quel
et que **rien n'a été modifié artificiellement**.

---

2) Générer une présentation MARP : `oslo_presentation.md`

⚠️ RÈGLES IMPÉRATIVES :

❗ Tu ne dois :

- JAMAIS modifier les notebooks
- JAMAIS modifier les datasets
- JAMAIS recalculer aucun graphique
- JAMAIS régénérer aucun output

❗ Tu dois :

- **Uniquement réutiliser les figures déjà présentes dans** `oslo_airbnb_EDA.ipynb`
- Les **intégrer dans la présentation** sous forme d'images ou captures (le format que l'IDE peut générer)
- Générer une présentation **MARP** strictement en Markdown

🎨 Structure de la présentation MARP attendue (10–12 slides)

Créer un fichier `oslo_presentation.md` avec le format MARP suivant :

Slide 1 — Titre

```
# Analyse du marché Airbnb : Oslo
### Projet de groupe – M2 CMSI
```

Slide 2 — Contexte

- objectifs du projet
- rappel du cahier des charges
- méthodologie générale

Slide 3 — Vue d'ensemble du dataset

➡️ Réutiliser la table d'aperçu déjà affichée dans le notebook Oslo

Slide 4 — Distribution des prix

➡️ Reprendre le graphique présent dans le notebook, sans le recalculer

Slide 5 — After-cleaning

➡️ Graphique des prix filtrés 1–99e centile

Slide 6 — Analyse par quartiers

➡️ Boxplot déjà produit dans le notebook Oslo

Slide 7 — Analyse des hôtes & avis

➡️ Graphiques déjà présents (scatterplots)

Slide 8 — Segmentation (Budget / Mid / Luxe)

➡️ Reprendre le graphique ou tableau déjà dans le notebook Oslo

Slide 9 — Modélisation (Régression linéaire)

- R^2
- RMSE
- Interprétation des coefficients

  ➡️ Reprendre le tableau déjà existant dans le notebook Oslo

Slide 10 — Insights clés

Synthèse business

Slide 11 — Recommandations

Basées sur l'analyse + modèle

Slide 12 — Conclusion

Points clés + limites

INSTRUCTIONS À RESPECTER

---

1. **Ne crée aucun ZIP.**
2. **Ne modifie aucun notebook.**
3. **Ne modifie aucun CSV.**
4. **Ne modifie pas** `interaction_IA.md`.
5. Le README et la présentation MARP sont les *seuls* fichiers à générer.

6. Les graphiques de la présentation doivent venir **uniquement des résultats déjà présents dans** `oslo_airbnb_EDA.ipynb` .
7. Le ton du README doit être **professionnel, académique, conforme au cahier des charges**.
8. La présentation MARP doit être **propre, concise, et orientée business**.

🟦 Résultat attendu

À la fin, le repo doit contenir deux nouveaux fichiers :

```
README.md
oslo_presentation.md   (format MARP)
```

Et **rien d'autre ne doit être changé**.

🎯 **Ce prompt est prêt à être collé dans ton IDE.**
Si tu veux ensuite que je relise ou améliore le README généré, je suis là.