- *Chargement de données (CSV, Excel, JSON)*

```
import pandas as pd

df_csv = pd.read_csv('data.csv')
df_excel = pd.read_excel('data.xlsx')
df_json = pd.read_json('data.json')
```

- *Aperçu général du DataFrame*

```
df.head()          # 5 premières lignes
df.tail()          # 5 dernières lignes
df.sample(3)       # 3 lignes aléatoires
df.shape           # dimensions (lignes, colonnes)
df.columns         # noms des colonnes
df.dtypes          # types de chaque colonne
df.info()          # résumé complet
```

- *Statistiques de base*

```
df.describe()                  # résumé statistique numérique
df['colonne'].value_counts()   # fréquence des valeurs
df['colonne'].nunique()        # nombre de valeurs uniques
```

- *Configuration d'affichage*

```
pd.set_option('display.max_columns', None)      # afficher toutes les colonnes
pd.set_option('display.precision', 2)           # nombre de décimales
```

- *Sélection de colonnes / lignes*

```
df['marque']                   # une seule colonne
df[['marque', 'prix']]         # plusieurs colonnes
df.iloc[0]                     # première ligne
df.iloc[0:5]                   # lignes 0 à 4
df.loc[df['prix'] > 10000]     # filtrage conditionnel
```

- *Création et suppression de colonnes*

```
df['prix_par_kg'] = df['prix'] / df['poids']
df.drop('ancien_colonne', axis=1, inplace=True)
```

- *Tri des valeurs*

```
df.sort_values(by='prix', ascending=False)
```

- **Atelier :**
  - Charger un fichier CSV contenant des informations sur des véhicules
  - Afficher les colonnes, types, et dimensions
  - Compter les valeurs uniques dans une colonne (ex. carburant)
  - Créer une colonne calculée (`prix_par_kg`)
  - Trier les voitures par prix décroissant et extraire les 10 premières

Prétraitement et nettoyage des données :

- *Détection des valeurs manquantes*
  - Localisation des `NaN` par colonne ou ligne

```
df.isnull().sum()
df[df.isnull().any(axis=1)]
```

- *Traitement des valeurs manquantes*

- Remplissage (imputation) par moyenne, médiane, constante, ou méthode avancée :

```
df['age'] = df['age'].fillna(df['age'].mean())
df['sexe'] = df['sexe'].fillna('Inconnu')
df = df.ffill()                                  # forward fill
df = df.bfill()                                  # backward fill
```

- Imputation avec SimpleImputer (Scikit-learn)

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='median')
df[['revenu']] = imputer.fit_transform(df[['revenu']])
```

- *Détection des valeurs aberrantes (outliers)*

- Méthode de l'écart interquartile (IQR)

```
Q1 = df['prix'].quantile(0.25)
Q3 = df['prix'].quantile(0.75)
IQR = Q3 - Q1
outliers = df[(df['prix'] < Q1 - 1.5 * IQR) | (df['prix'] > Q3 + 1.5 * IQR)]
```

- Filtrage par z-score

```
from scipy.stats import zscore
df['zscore'] = zscore(df['prix'])
df[df['zscore'].abs() > 3]
```

Suppression des doublons

- Recherche et élimination des lignes répétées

```
df.duplicated().sum()
df.drop_duplicates(inplace=True)
```

Nettoyage des chaînes de caractères

```
df['ville'] = df['ville'].str.strip().str.lower().str.replace('-', ' ')
df['ville'] = df['ville'].str.normalize('NFKD')
```

Uniformisation des types de données

```
df['date'] = pd.to_datetime(df['date'])
df['prix'] = df['prix'].astype(float)
```

Création de colonnes dérivées

- Générer de nouvelles variables utiles à partir d'existantes :

```
df['prix_par_kg'] = df['prix'] / df['poids']
df['anciennete'] = 2024 - df['annee']
```

Encodage des variables catégorielles

- One-hot encoding et label encoding

```
pd.get_dummies(df, columns=['carburant'])
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['sexe_code'] = le.fit_transform(df['sexe'])
```

Normalisation et standardisation

- Mise à l'échelle pour les modèles sensibles (KNN, régression linéaire, etc.)

```
from sklearn.preprocessing import MinMaxScaler, StandardScaler
MinMaxScaler().fit_transform(df[['revenu']])
StandardScaler().fit_transform(df[['revenu']])
```

Détection des incohérences métier

- Exemple : un poids inférieur à 300 kg ou un âge supérieur à 120 ans

```
df[df['poids'] < 300]
df[df['age'] > 120]
```

- **Atelier pratique :**
  - Identifier et traiter les valeurs manquantes, aberrantes et incohérentes
  - Nettoyer des colonnes texte, unifier les formats
  - Créer des colonnes utiles pour la modélisation
  - Appliquer une normalisation et encoder les catégories

Groupement, jointure et agrégations :

- *Moyennes, sommes, comptages par catégorie*

```
df.groupby('marque')[ df].mean()
df.groupby('carburant')['puissance'].sum()
df.groupby('type').size()                                    # Nombre d'éléments par type
```

- *Agrégations multiples avec agg()*

```
df.groupby('marque').agg({
   'prix': ['mean', 'max', 'std'],
   'poids': 'median'
})
```

- *Agrégation sur plusieurs colonnes simultanées*

```
gb = df.groupby(['marque', 'carburant'])
gb['prix'].mean()
```

- *Tableaux croisés dynamiques (Pivot Tables)*

```
pd.pivot_table(df, values='prix', index='marque', columns='carburant', aggfunc='mean')
```

- *Tri de groupes agrégés*

```
df.groupby('marque')['prix'].mean().sort_values(ascending=False)
```

- *Fusions entre DataFrames (jointures relationnelles)*

```
pd.merge(df_clients, df_commandes, on='client_id', how='inner')
pd.merge(df1, df2, left_on='id', right_on='produit_id', how='outer')
```

- *Concaténation (empiler plusieurs tableaux)*

```
pd.concat([df1, df2], axis=0)                    # ligne par ligne
pd.concat([df1, df2], axis=1)                    # colonne par colonne
```

- Atelier :

  - Créer un tableau croisé des prix moyens par marque et type de carburant
  - Fusionner des données clients et commandes pour calculer la dépense totale par client

Analyse exploratoire et statistiques :

- *Statistiques de base*

- Moyenne, médiane, écart-type, quartiles

```
df.describe()
df['prix'].mean(), df['prix'].std(), df['prix'].quantile([0.25, 0.5, 0.75])
```

- *Variance et covariance*

```
df.var()
df.cov()
df.corr()                          # corrélation de Pearson
```

- *Visualisation des distributions*

- Histogramme et boîte à moustaches

```
import seaborn as sns
sns.histplot(df['prix'], bins=20, kde=True)
sns.boxplot(x=df['prix'])
```

- *Analyse de la variance entre groupes (ANOVA simplifiée)*

```
from scipy.stats import f_oneway
f_oneway(df[df['marque'] == 'Peugeot']['prix'], df[df['marque'] == 'Renault']['prix'])
```

- *Régression linéaire simple avec Statsmodels*

```
import statsmodels.api as sm
X = df[['poids']]
y = df['prix']
X = sm.add_constant(X)                    # Ajout de l'intercept
model = sm.OLS(y, X).fit()
print(model.summary())
```

- 🩹 Atelier :
  - Comparer visuellement les distributions de prix entre marques
  - Afficher les valeurs statistiques d'une variable numérique
  - Vérifier les écarts-types et quartiles entre catégories

Visualisation des données :

- *Histogramme :*

```
import seaborn as sns
sns.histplot(df['prix'], bins=30)
```

- *Nuage de points :*

```
sns.scatterplot(x='poids', y='prix', hue='marque', data=df)
```

- *Heatmap de corrélations :*

```
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
```

- *Visualisations interactives avec Plotly :*

```
import plotly.express as px
px.scatter(df, x='poids', y='prix', color='carburant')
```

- 🩹 Atelier :
  - Comparaison visuelle de modèles de voitures selon leur performance et prix.