

WEDNESDAY, JULY 26, 2023

ANALYSE DES OPPORTUNITÉS D'EMPLOI

AYMANE SABRI

DATA DEVELOPER

Objet du projet :

Analyse approfondie du marché du travail dans les domaines émergents.

Visualisation claire des résultats pour faciliter la compréhension des parties prenantes.

Mise en place d'un système de stockage des données pour une gestion efficace .

I. Methodologie Suivie :

1. Introduction :

Au cours de cette partie, nous allons examiner de manière approfondie les **but**s demandées dans le brief en matière d'analyse des opportunités d'emploi.

Nous allons également présenter les différentes étapes suivies, en détaillant les choix technologiques, les outils et les méthodes utilisées pour mener à une analyse du marché de travail .

L'objectif principal de ce travail se résume à la “**L'analyse Des Opportunités D'emploi**”

- Exploration des données .
- Nettoyage des données .
- Transformation des données .
- Affichage des visualisations statistiques .
- Implementation d'un script pour la création d'une base de données relationnelle .
- Intégration des données dans la base de données .
- Redaction d'un rapport technique



2. Planification du plan de réalisation du brief :

Dans l'objectif de bien mener le **Brief**, j'ai commencé par établir le **planning** à suivre durant la période de brief .

Pour ce faire, j'ai d'abords décomposé mon projet en **phases**, où chaque phase est définie par un certain nombre de tâches. Ensuite, j'ai élaboré une planification de ces phases sur la durée du projet, à l'aide d'un diagramme de Gantt.

2.1. Etapes Suivie :

Les **étapes** suivies sont :

- **Comprendre** la nature et l'étendue du travail demandé.
- **Identifier** le type de recherche d'information demandé.
- Comprendre les **objectifs** d'apprentissage visés par le projet et les relier à la matière de l'étude.
- **Adopter** la démarche logique pour exécuter le travail.
- **Prêter** attention aux consignes et aux critères d'évaluation du projet (indiqués par écrit afin d'éviter toute erreur d'interprétation).
- Connaitre les échéances et avoir l'intention de les respecter.

Suite à ces étapes j'ai identifié les besoins à satisfaire, définit l'aspect fonctionnel de projet et sa conception, réalisé le système et finalement je l'ai soumis à plusieurs **tests** pour s'assurer de son adaptation aux **besoins** exprimés précédemment.



2.2. Diagramme de Gant :

Ce diagramme représente la durée de chaque tâche effectué dans mon projet.

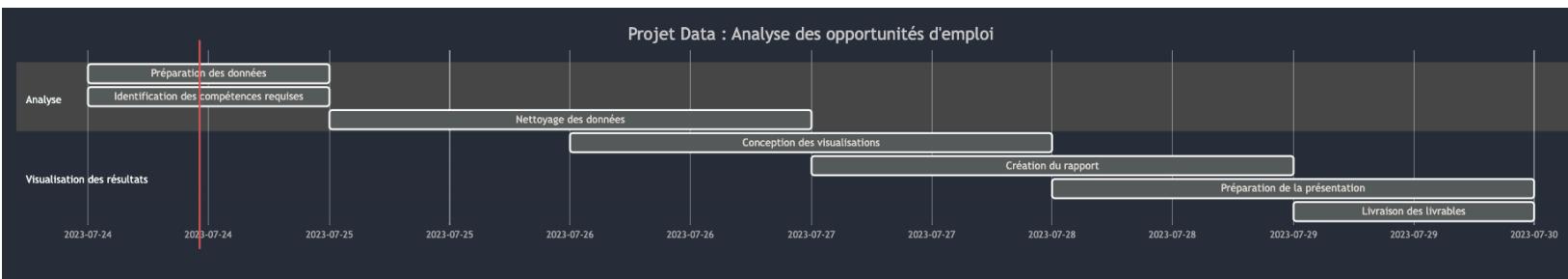
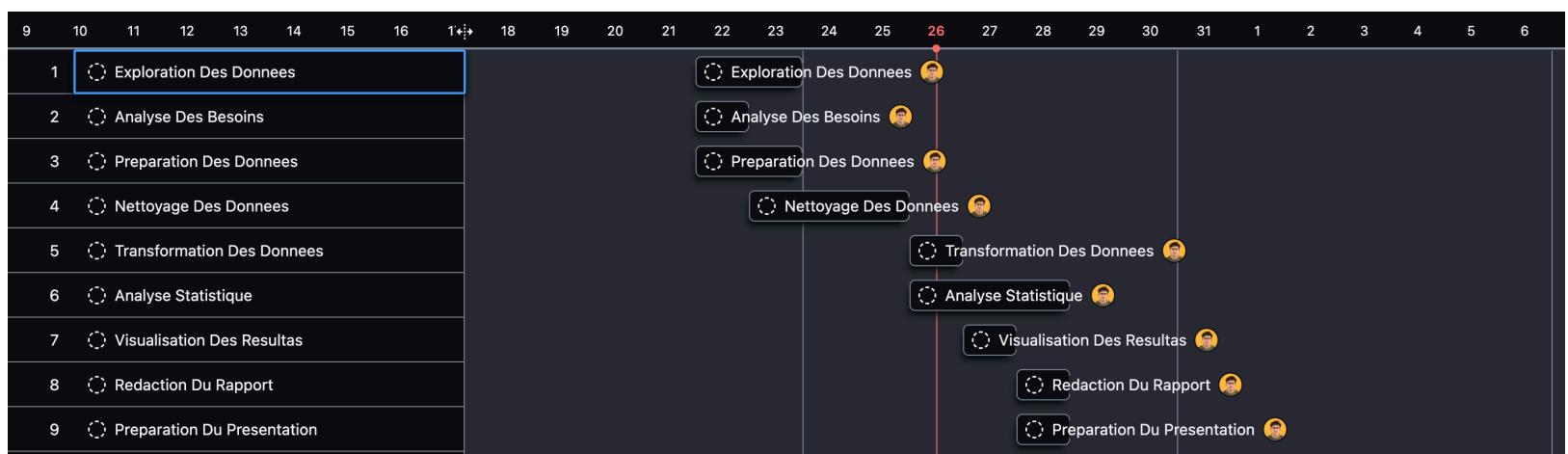


Figure : Diagramme de Gant

2.3. Planification des tâches sous GitHub :



II. Exploration des données :

1. Introduction :

a) Rappel des objectifs du projet :

Dans cette section, nous rappellerons brièvement les objectifs du projet d'analyse des opportunités d'emploi en domaine de la data. Il est essentiel de préciser clairement les questions que nous cherchons à répondre à travers cette étude, telles que l'identification des tendances de demande d'emploi, des compétences recherchées par les employeurs et des salaires moyens proposés.

b) Présentation du jeu de données utilisé :

Nous soulignerons l'importance de la phase d'exploration des données dans le cadre du projet, car elle nous permet d'obtenir une compréhension approfondie de notre jeu de données avant de procéder à des analyses plus avancées. L'exploration nous aide à identifier les caractéristiques clés du jeu de données et à repérer d'éventuelles relations entre les variables.

Get Data Frame Information

```
[658]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3198 entries, 0 to 3197
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Company          3197 non-null    object 
 1   Job Title         3197 non-null    object 
 2   Location          3197 non-null    object 
 3   Job Type          3197 non-null    object 
 4   Experience level  2962 non-null    object 
 5   Salary            3009 non-null    object 
 6   Requirment of the company 3198 non-null    object 
 7   Facilities        3198 non-null    object 
dtypes: object(8)
memory usage: 200.0+ KB
```

La DataFrame contient un total de [3198](#) entrées réparties sur 8 colonnes.

Voici une description de chaque colonne :

- Company : Le nom de l'entreprise offrant l'opportunité d'emploi.
- Job Title : Le poste proposé par l'entreprise.
- Location : L'emplacement géographique où se situe le poste proposé.
- Job Type : Le type d'emploi offert (par exemple : temps plein, temps partiel, stage, etc.).
- Experience level : Le niveau d'expérience requis pour le poste. Cette colonne contient 2962 valeurs non nulles, indiquant qu'il y a quelques données manquantes dans cette colonne.
- Salary : Le salaire associé au poste, exprimé sous forme de texte ou de nombre. Cette colonne contient 3009 valeurs non nulles, ce qui signifie qu'il y a également des données manquantes dans cette colonne.
- Requirement of the company : Les exigences spécifiques énoncées par l'entreprise pour le poste.
- Facilities : Les avantages ou les installations proposés par l'entreprise pour les employés occupant ce poste.

Company	-----> Dtype : object
Job Title	-----> Dtype : object
Location	-----> Dtype : object
Job Type	-----> Dtype : object
Experience level	-----> Dtype : object
Salary	-----> Dtype : object
Requirement of the company	-----> Dtype : object
Facilities	-----> Dtype : object

1. Analyse Observatoire :

a) Représentations distinctes des valeurs des colonnes :

▼ Values Representation ¶

```
[13]: columns = ['Company' , 'Job Title' , 'Experience level']

for i in columns:
    print(f"{i}:\n {str(df[i].unique())}\n")
```

Company:
['SGS' 'Ocorian' 'Cricut' ... 'DNSFilter' 'MUFG Investor Services'
'Galileo Financial Technologies']

Job Title:
['Clinical Data Analyst' 'AML/CFT & Data Analyst'
'Machine Learning Engineer' ...
'Application Integration Engineer, Computer Vision Program'
'Senior Software Engineer, Machine Learning – Ads Intelligence'
'Data Scientist – New College Graduate']

Experience level:
['Entry-level' nan 'Mid-level' 'Senior-level' 'Executive-level']

b) Pourcentages des valeurs null par colonne :

	Column Data Types	Column Valid Values	Column Null Values
Company	object	99.968730	0.031270
Job Title	object	99.968730	0.031270
Location	object	99.968730	0.031270
Job Type	object	99.968730	0.031270
Experience level	object	92.620388	7.379612
Salary	object	94.090056	5.909944
Requirement of the company	object	100.000000	0.000000
Facilities	object	100.000000	0.000000

c) Representation de pourcentages des valeurs null par colonne :

On remarque , que les colonnes :

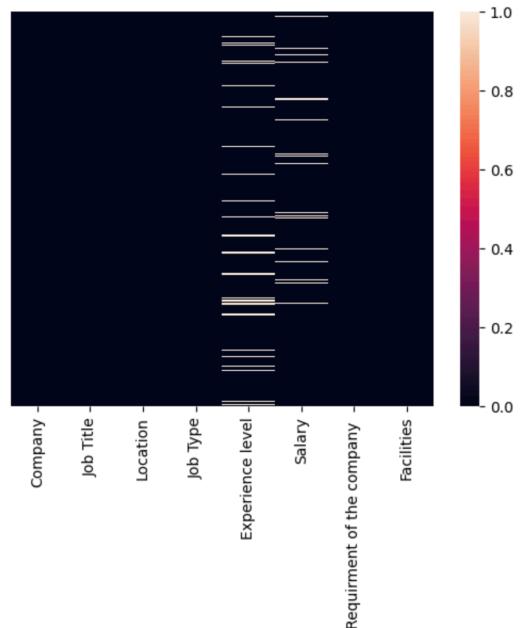
- Experience Level .
- Salary .

Contiennent des valeur nulles , ce qui nécessite une prétraitemet de ces colonne pour ne pas affecter négativement l.analyse souhaite

Visualize Null Values Distribution

```
[663]: sns.heatmap(df.isnull() , yticklabels = False)
```

```
[663]: <Axes: >
```



III. Nettoyage Du Jeu De Données :

1. Gestion des Valeurs Manquantes :

Data Cleaning

Define The Missing Values Keys

```
[664]: missing_values = ["Nan", "n/a", "null", np.nan]
```

```
[665]: df = pd.read_csv('job.csv', na_values = missing_values, encoding = 'unicode_escape')
```

Getting the duplicates rows number

```
[666]: print(f"The number of duplicated rows is : {df.duplicated().sum()} Rows")
```

```
The number of duplicated rows is : 202 Rows
```

Les valeurs manquantes ont été définies à l'aide de la liste `missing_values` comprenant les valeurs "Nan", "n/a", "null", et `np.nan`. Ensuite, le jeu de données a été lu à partir du fichier CSV en utilisant en spécifiant les valeurs manquantes à remplacer par les valeurs définies dans `na_values`. Cela permet de standardiser les valeurs manquantes dans le jeu de données pour une meilleure manipulation et analyse.

2. Suppression des Duplications :

Le nombre de lignes en double a été obtenu en utilisant la méthode `duplicated().sum()`. Ensuite, les lignes en double ont été supprimées à l'aide de `drop_duplicates()`, afin d'éliminer les redondances et de conserver des données uniques pour l'analyse.

Dropping the duplicated rows

```
[667]: df = df.drop_duplicates()
```

```
[668]: print(f" Remaining Rows : {df.shape[0]} ")
```

```
Remaining Rows : 2996
```

3. Gestion des Valeurs Manquantes pour la Colonne 'Experience level' :

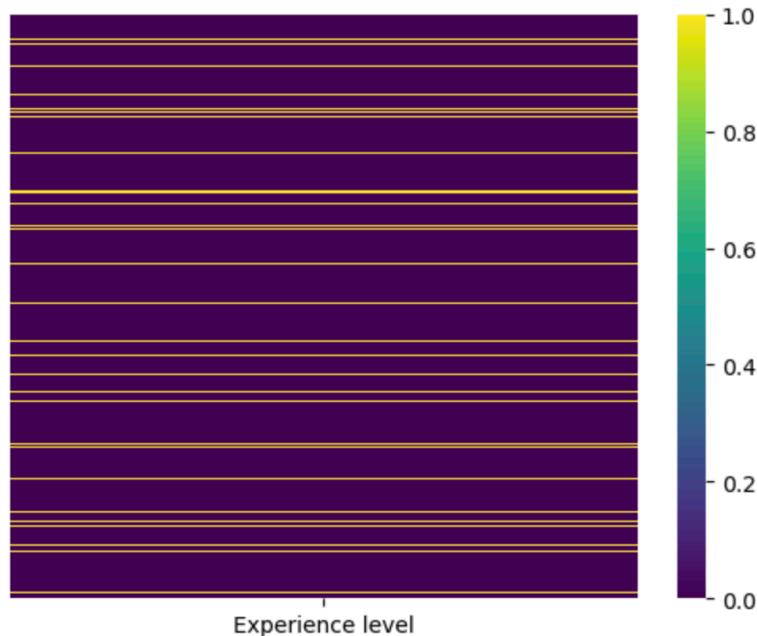
Les valeurs manquantes dans la colonne 'Experience level' ont été visualisées en utilisant un heatmap .

Ensuite, les lignes avec des valeurs manquantes dans cette colonne ont été supprimées.

Create Experience_Level Column

```
[669]: sns.heatmap(df[['Experience level']].isnull(), yticklabels = False, cmap='viridis')
```

```
[669]: <Axes: >
```



Pour compléter le processus, les valeurs de la colonne 'Experience level' ont été uniformisées en gardant uniquement la partie gauche du texte (la première partie avant le caractère '-'), et le terme "Mid" a été remplacé par "Middle" pour assurer la cohérence.

```
[671]: df.dropna(subset=['Experience level'], axis=0, inplace=True)
```

```
[672]: df["Experience level"] = df["Experience level"].str.split('-').str[0]
```

```
[673]: df['Experience level'].replace('Mid', 'Middle', inplace = True)
```

4. Conversion des Salaires en USD :

Une fonction `extract_currency_code` a été créée pour extraire le code de devise (USD, GBP ou EUR) à partir de la colonne '`Salary`'. Ensuite, une fonction `convert_to_dollars` a été mise en place pour convertir les salaires en USD à l'aide d'une API d'échange de devises. Les salaires ont été convertis en valeurs numériques et la colonne '`Salary`' a été renommée en '`Salary(USD)`' pour refléter cette transformation. Les valeurs manquantes dans cette colonne ont été supprimées.

```
: df['Salary'] = df['Salary'].apply(convert_to_dollars)
Conversion from EUR to USD succeeded.
Conversion from GBP to USD succeeded.
Conversion from EUR to USD succeeded.
Conversion from GBP to USD succeeded.
Conversion from GBP to USD succeeded.
Conversion from EUR to USD succeeded.
Conversion from GBP to USD succeeded.
Conversion from EUR to USD succeeded.
Conversion from GBP to USD succeeded.
```

```
: df.rename(columns={'Salary': 'Salary(USD)'}, inplace=True)
: df['Salary(USD)'].replace(0, np.nan, inplace=True)
```

Les valeurs manquantes dans la colonne '`Salary(USD)`' ont été imputées en utilisant la [moyenne](#) des salaires associés à chaque niveau d'expérience. Cela a été réalisé en regroupant les données par '`Experience level`' et en remplissant les valeurs manquantes à l'aide de la moyenne correspondante. Cette étape permet de remplir [les valeurs manquantes](#) avec des estimations raisonnables en fonction du niveau d'expérience.

```
mean_salaries = df.groupby('Experience level')['Salary(USD)'].mean()
df['Salary(USD)'] = df.apply(lambda row: mean_salaries[row['Experience level']] if pd.isna(row['Salary(USD)']) else row['Salary(USD)'], axis=1)

print(f"Rmaining : {df.shape[0]} Rows \ {df.shape[1]} Columns")
Rmaining : 2768 Rows \ 8 Columns
```

5. Traitement des Colonnes 'Requirement of the company' et 'Facilities' :

Les données dans les colonnes 'Requirement of the company' et 'Facilities' ont été nettoyées en divisant les chaînes de caractères séparées par des virgules en listes de mots-clés pertinents. Cela a été accompli en utilisant `str.split(',')` pour séparer les mots-clés et `apply(lambda x: [item.strip() for item in x if item.strip()])` pour supprimer les espaces inutiles autour des mots-clés.

Create Requirement Column

```
: df['Requirement of the company '] = df['Requirement of the company '].str.split(',')
df['Requirement of the company '] = df['Requirement of the company '].apply(lambda x: [item.strip() for item in x if item.strip()])
```

Create Facilities column

```
: df['Facilities'] = df['Facilities'].apply(lambda x: re.sub(r',{2,}', '', x))
: df['Facilities'] = df['Facilities'].str.split(',')
```



6. Extraction des Pays et des Villes à partir de la Colonne 'Location':

Les noms des pays et des villes ont été extraits à partir de la colonne 'Location' en utilisant la bibliothèque "geotext" et des fonctions personnalisées `extract_country` et `extract_city`. Ces fonctions identifient les noms de pays et de villes présents dans la colonne et les séparent de la chaîne de caractères d'origine.

Create The Location Column

Extracting The Countries Names

```
: def extract_country(location):
    geo_text = GeoText(location)
    countries = list(geo_text.countries)
    if countries:
        return countries[0], location.replace(countries[0], str(countries[0])).strip()
    else:
        return None, location
df[['Country', 'Location']] = df['Location'].apply(extract_country).apply(pd.Series)
```

```
for index, row in df[df['Country'].isnull() & df['City'].notnull()].iterrows():
    city = row['City']
    country = get_country_from_city(city)
    df.at[index, 'Country'] = country
```

API Response for 'Tallinn': Estonia

API Response for 'Lehi': United States

API Response for 'Los Angeles': United States

API Response for 'Indianapolis': United States

API Response for 'Cardiff': United Kingdom

API Response for 'San Francisco': United States

API Response for 'Arlington': United States

API Response for 'Dubai': United Arab Emirates

API Response for 'New York': United States

API Response for 'Toronto': Canada

Extracting The Cities Names

```
: def extract_city(location):
    geo_text = GeoText(location)
    cities = list(geo_text.cities)
    if cities:
        return cities[0], location.replace(cities[0], '').strip()
    else:
        return None, location
df[['City', 'Location']] = df['Location'].apply(extract_city).apply(pd.Series)
```

7. Détermination des Type de Travail 'face-to-face' ou 'Remote' :

Une fonction `set_job_type` a été créée pour identifier si le travail est "face-to-face" (en présentiel) ou "Remote" (à distance) en recherchant le mot "remote" dans la colonne 'Location'. Cette fonction attribue ensuite le type de travail correspondant à chaque ligne du jeu de données dans une nouvelle colonne appelée 'work setups'.

Recognize The Job Type From Location Column

```
def set_job_type(row):
    if re.search(r'\bremote\b', row['Location'], re.IGNORECASE):
        return 'Remote'
    else:
        return 'face-to-face'

df['work setups'] = df.apply(set_job_type, axis=1)
```

Fix The Remote Values

```
def fill_Remote(row):
    if re.search(r'\bremote\b', row['Job Type'], re.IGNORECASE) and pd.isnull(row['Country']) and pd.isnull(row['City']):
        return 'WorldWide', 'WorldWide'
    return row['Country'], row['City']

df['Country'], df['City'] = zip(*df.apply(fill_Remote, axis=1))
```

8. Identification des Catégories d'Emplois à partir des Titres de Poste:

Les catégories d'emplois ont été identifiées en utilisant une approche basée sur la correspondance de mots-clés. Une liste de mots-clés pertinents pour chaque catégorie d'emploi a été établie, puis chaque titre de poste a été comparé à cette liste pour déterminer sa catégorie correspondante. Les catégories ont été enregistrées dans une nouvelle colonne 'Job Post' qui remplace la colonne d'origine 'Job Title'.

```
df['Job Post'] = df['Job Title'].apply(detect_category_with_spacy)
```

```
df.drop('Job Title', axis=1, inplace=True)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2010 entries, 0 to 2009
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
 ---  -- 
 0   Company          2010 non-null    object 
 1   Job Type         2010 non-null    object 
 2   Experience level 2010 non-null    object 
 3   Salary(USD)      2010 non-null    float64
 4   Requirment of the company 2010 non-null    object 
 5   Facilities        2010 non-null    object 
 6   Country           2010 non-null    object 
 7   City              2010 non-null    object 
 8   work setups       2010 non-null    object 
 9   Job Post          2010 non-null    object 
dtypes: float64(1), object(9)
memory usage: 157.2+ KB
```

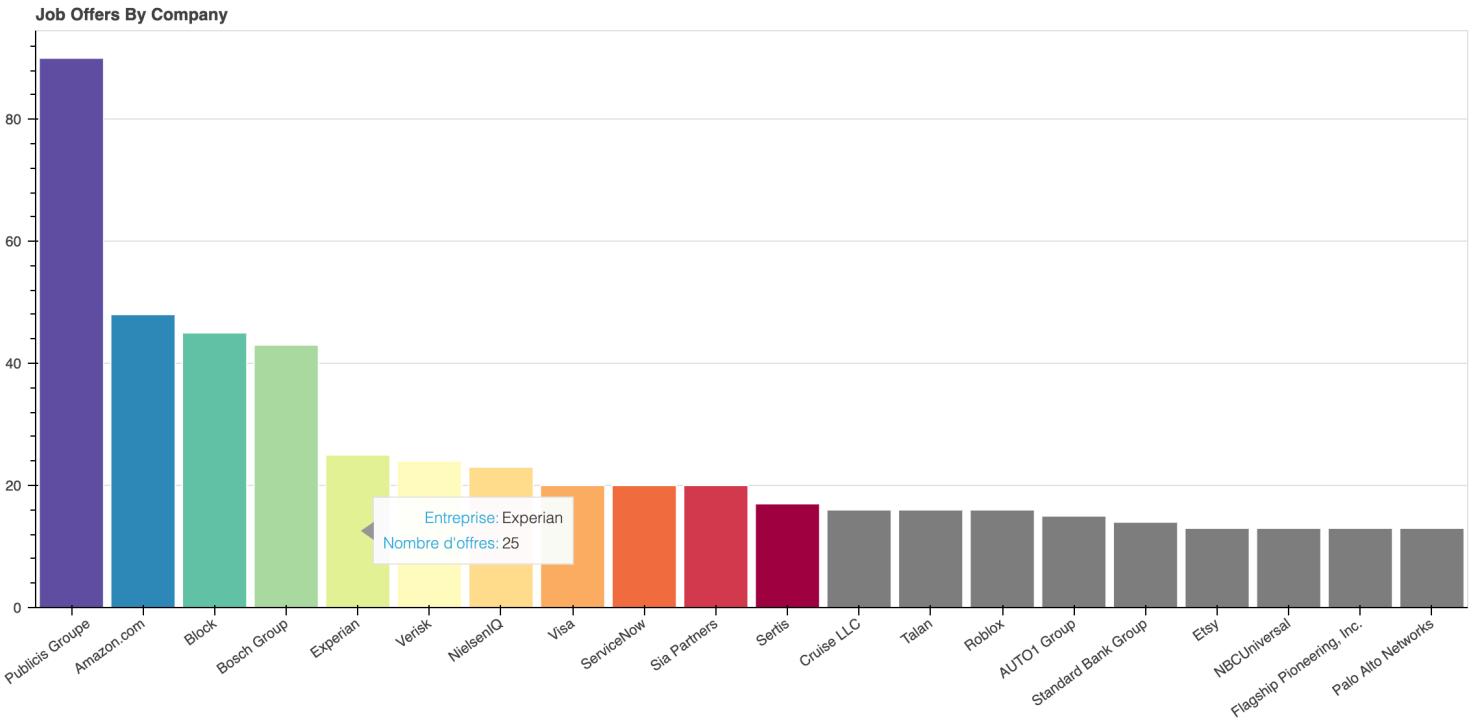
III. Analyse :

- Carte des offres d'emploi par pays :



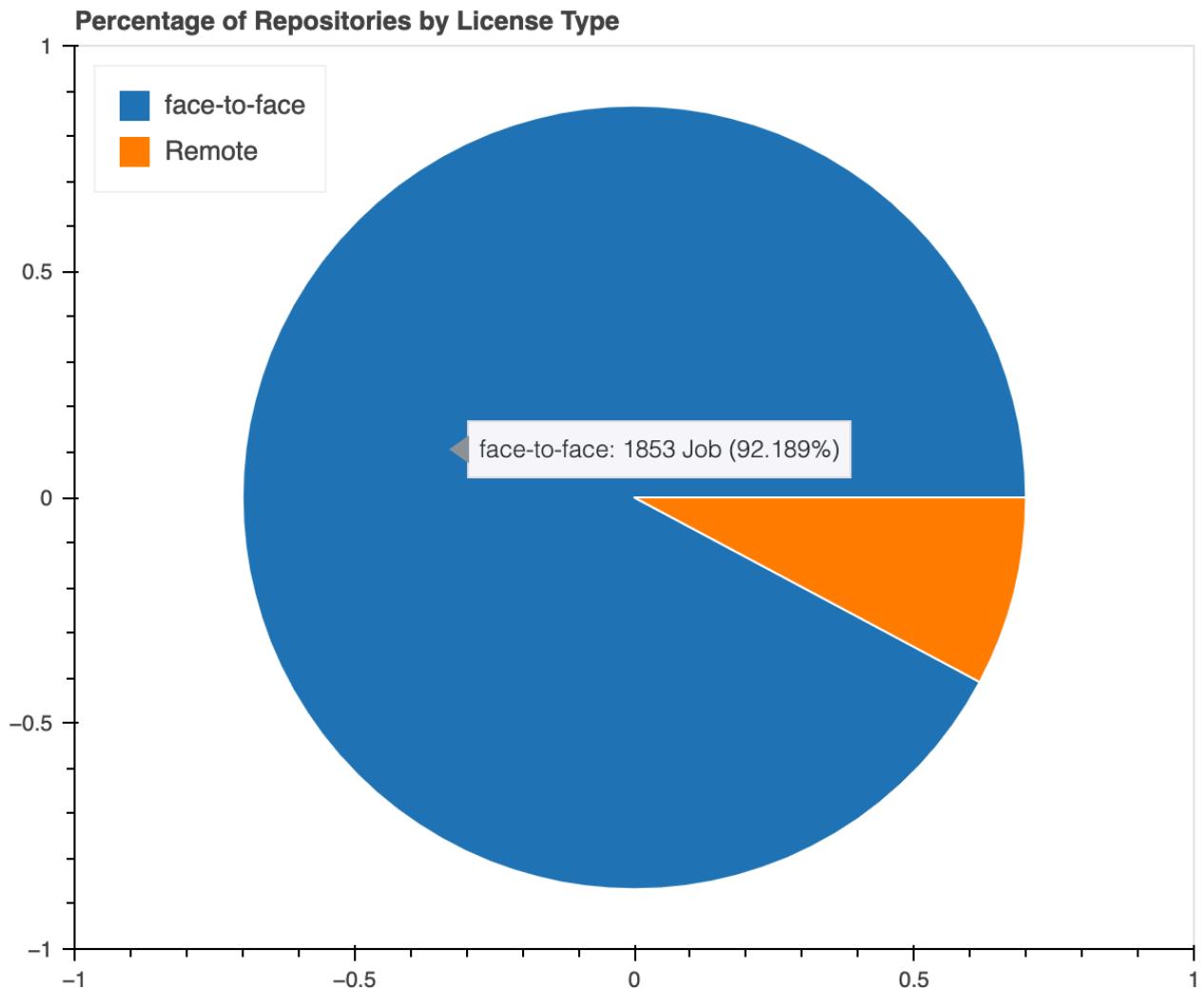
Les catégories d'emplois ont été identifiées en utilisant une approche basée sur la correspondance de mots-clés. Une liste de mots-clés pertinents pour chaque catégorie d'emploi a été établie, puis chaque titre de poste a été comparé à cette liste pour déterminer sa catégorie correspondante. Les catégories ont été enregistrées dans une nouvelle colonne 'Job Post' qui remplace la colonne d'origine 'Job Title'.

- Top 20 des entreprises offrant des emplois :



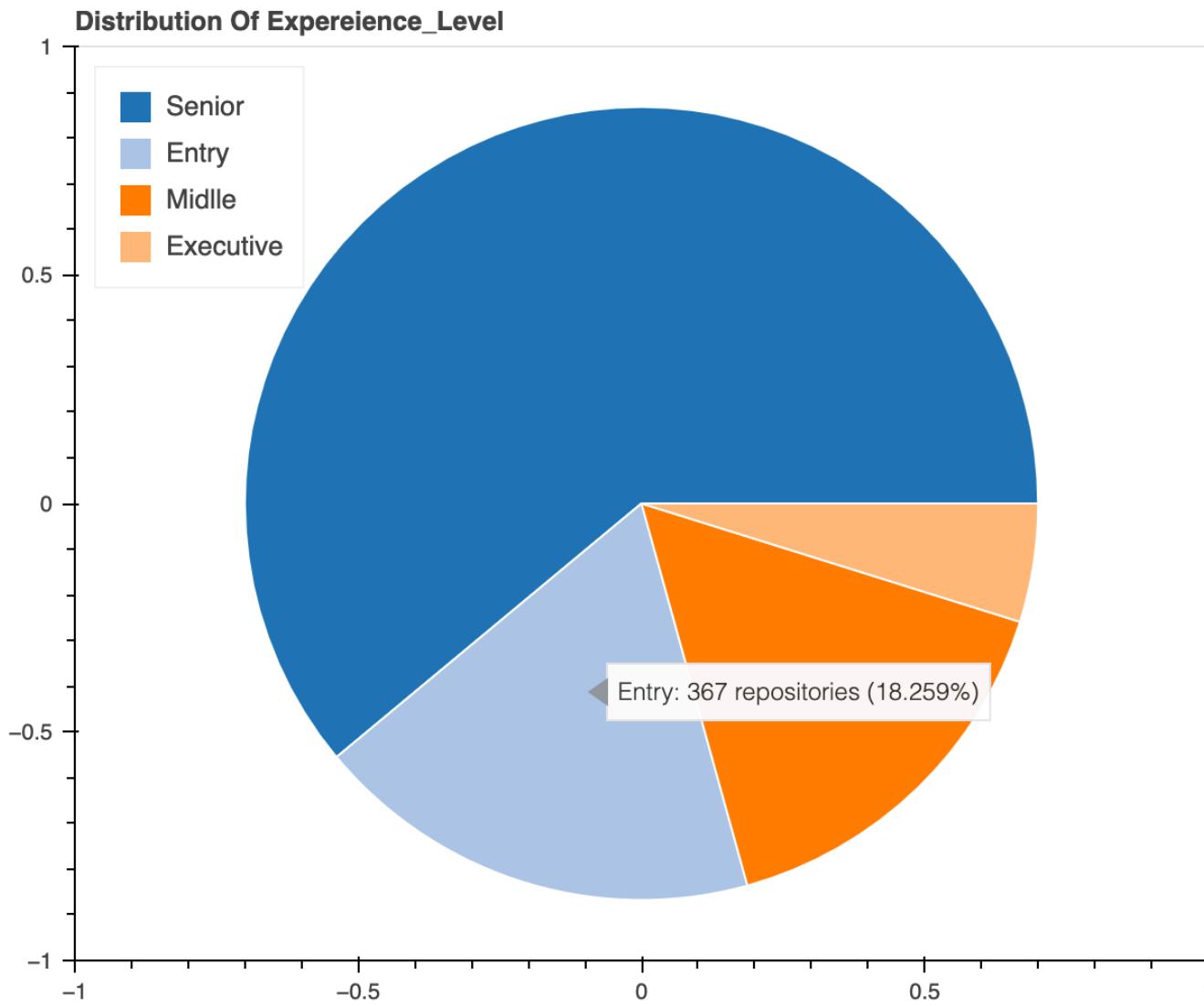
Le graphique présente les 20 meilleures entreprises qui offrent le plus grand nombre d'emplois dans le domaine de la data. Chaque entreprise est représentée par une barre verticale dont la hauteur indique le nombre d'offres d'emploi disponibles. Ce graphique permet d'identifier les entreprises les plus actives dans le recrutement de talents en data.

- Répartition des types de postes (face-to-face vs. remote):



Le graphique montre la répartition des types de postes, c'est-à-dire les emplois en face-à-face et les emplois à distance (remote),

- Répartition des niveaux d'expérience requis :



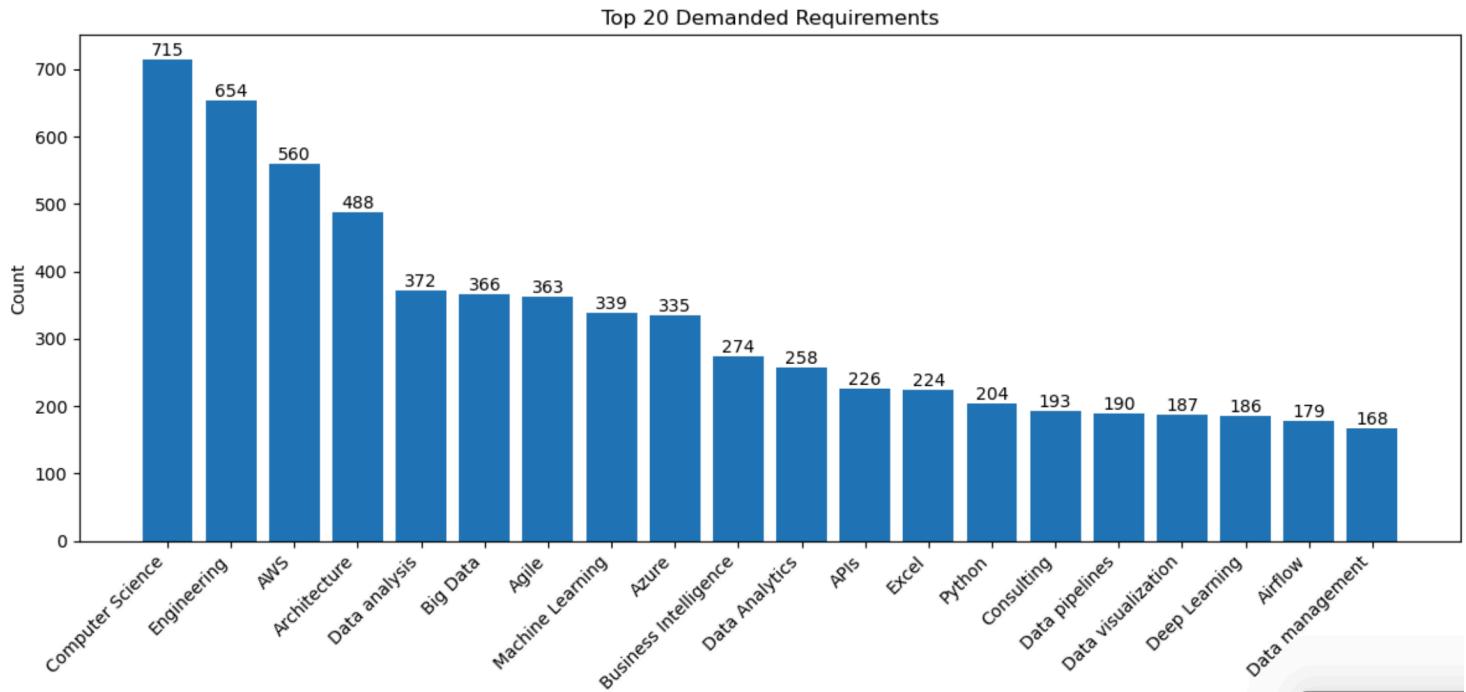
Le graphique présente la répartition des niveaux d'expérience requis dans les offres d'emploi. Chaque niveau d'expérience (débutant, intermédiaire, expert, etc.) est représenté par une part de secteur circulaire, où la taille de chaque secteur est proportionnelle au pourcentage d'offres d'emploi pour ce niveau d'expérience. Ce graphique permet de comprendre quels niveaux d'expérience sont les plus demandés par les employeurs.

- Salaire moyen par niveau d'expérience :



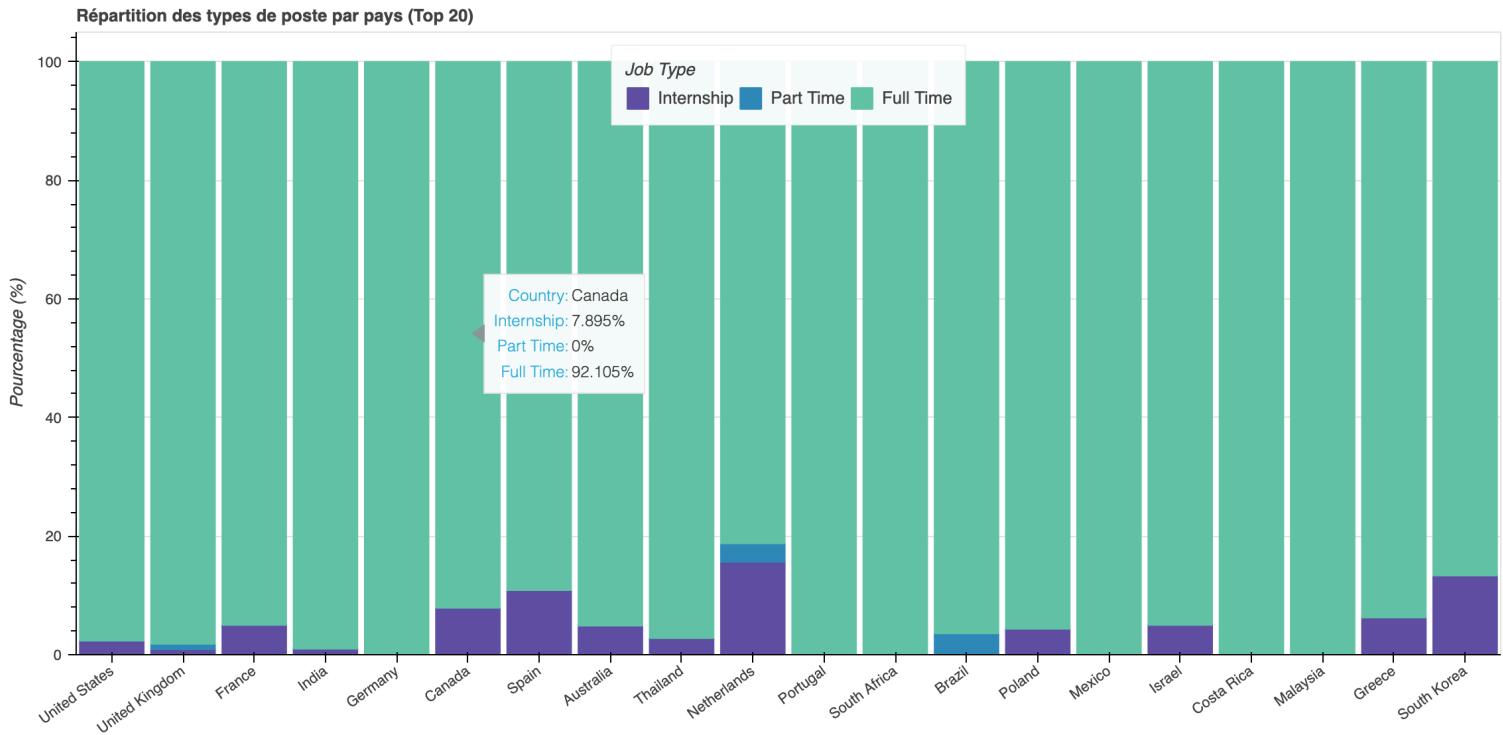
Le graphique illustre le salaire moyen offert pour chaque niveau d'expérience. Chaque niveau d'expérience est représenté par une barre verticale dont la hauteur indique le salaire moyen correspondant. Ce graphique permet de comparer les salaires moyens pour différents niveaux d'expérience et de déterminer les différences salariales entre les niveaux.

- Top 20 des compétences les plus recherchées :



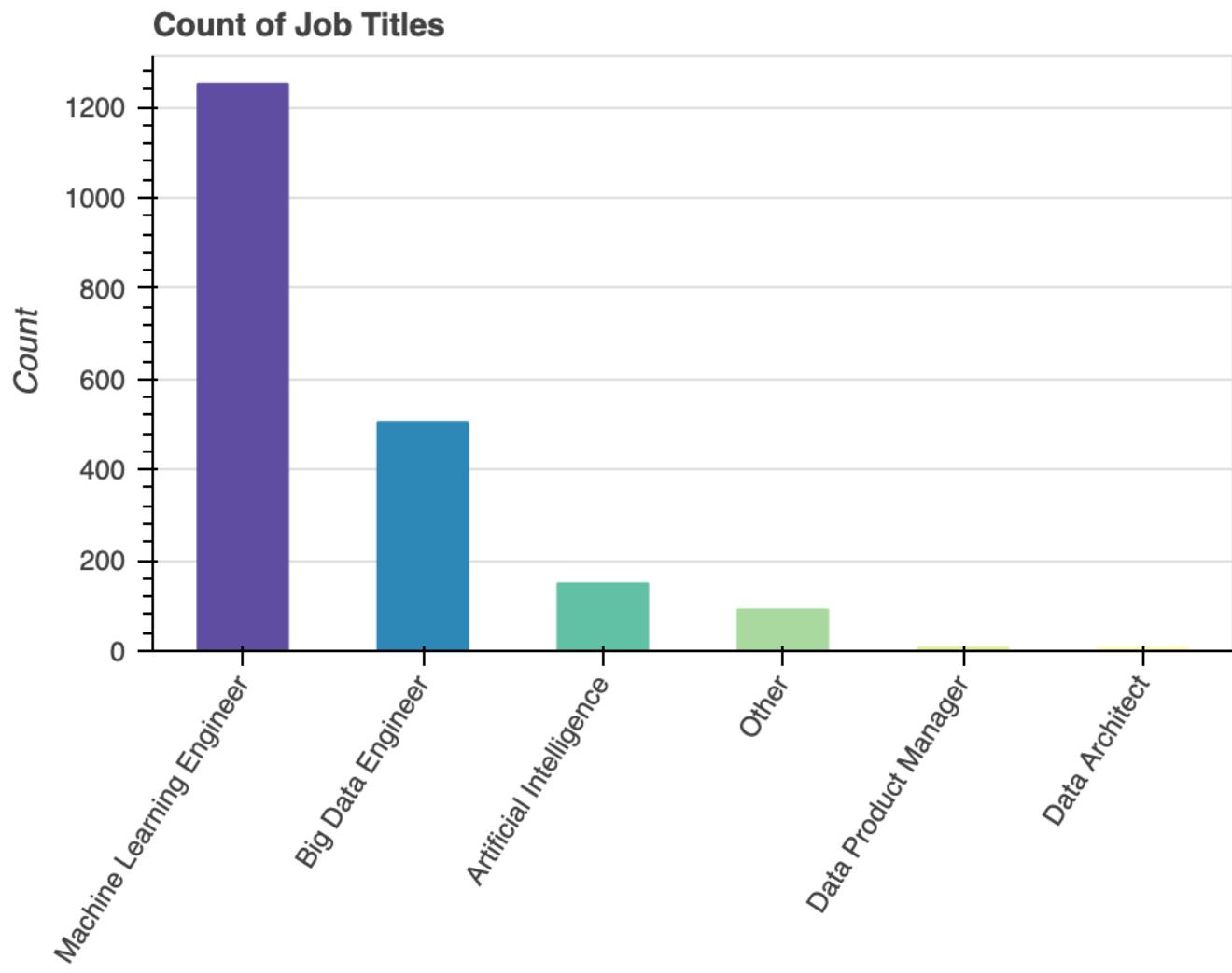
Le sixième graphique présente les 20 principales compétences les plus recherchées dans les offres d'emploi en data. Chaque compétence est représentée par une barre verticale dont la hauteur indique le nombre d'offres d'emploi mentionnant cette compétence. Ce graphique permet d'identifier les compétences les plus demandées par les employeurs.

- Répartition des types de poste par pays :



Le graphique illustre la distribution des types de postes dans les 20 pays les plus actifs en termes d'offres d'emploi dans le domaine de la data. Les pays sont représentés sur l'axe des abscisses, tandis que l'axe des ordonnées indique le pourcentage d'offres d'emploi pour chaque type de poste. Les types de postes comprennent des catégories telles que "Full Time", "Internship", "Contract", "Part Time", etc.

- Le nombre d'offres d'emploi pour chaque titre de poste dans le domaine de la data.:



Le graphique "Count of Job Titles" présente le nombre d'offres d'emploi pour chaque titre de poste dans le domaine de la data. Les titres de poste sont répertoriés sur l'axe des abscisses, tandis que l'axe des ordonnées indique le nombre total d'offres d'emploi associées à chaque titre de poste.

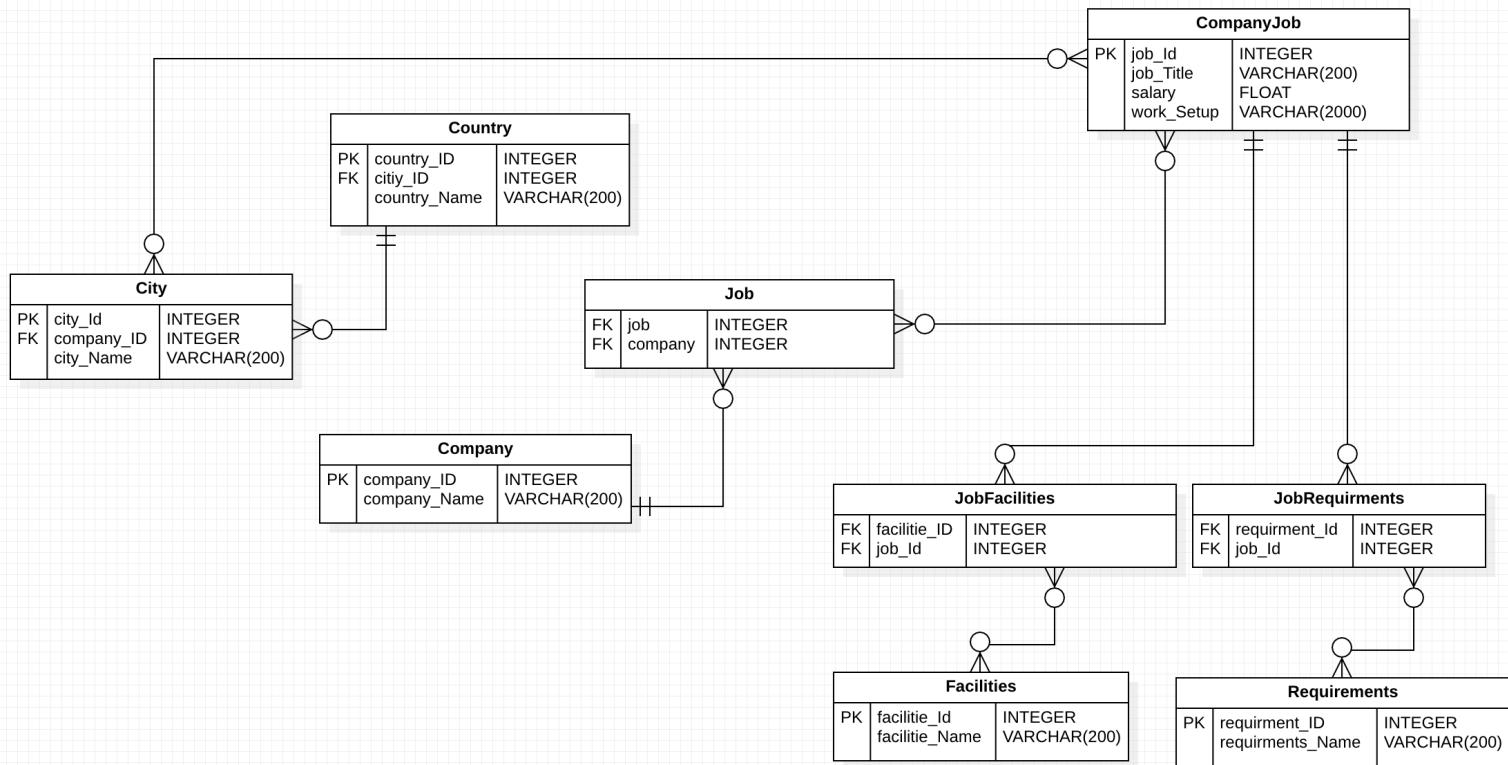
Etc ...

III. Integration Des Données :

1. Introduction :

Dans la partie d'intégration du dataframe nettoyé dans la base de données relationnelle, nous avons réalisé plusieurs opérations pour créer les tables correspondantes et insérer les données du dataframe dans ces tables.

• ER Diagramm Model :



2. Creation Des Tables :

Tout d'abord, nous avons établi une connexion avec la base de données MySQL en utilisant les informations connexion telles que l'hôte, le nom d'utilisateur, le mot passe et le nom de la base de données.

Import the necessary libraries

```
: import mysql.connector
import ast
```

```
: mydb = mysql.connector.connect(
    host = "localhost",
    username = "root",
    password = "",
    database = "Job",
)
```

Set a cursor connection

```
: cursor = mydb.cursor()
```

Ensute, nous avons créé les tables nécessaires pour stocker les données du dataframe. Nous avons créé les tables Requirement, Facilitie, Job, Company, City, Country, JobCompanyAssociation, JobFacilitiesAssociation et JobRequirementsAssociation avec les attributs appropriés tels que l'ID, le nom, le titre de l'emploi, le type d'emploi, le salaire, etc. Nous avons également établi des contraintes de clé primaire et clé étrangère pour garantir l'intégrité des données.

Create Requirement Table

```
create_tableRequirement_query = """
CREATE TABLE Requirement (
    requirement_ID INT PRIMARY KEY AUTO_INCREMENT ,
    requirement_Name VARCHAR(500)
);
"""
cursor.execute(create_tableRequirement_query)
```

Create Job Table

```
createTableJob_query = """
CREATE TABLE Job (
    job_ID INT PRIMARY KEY AUTO_INCREMENT ,
    requirement_ID INT ,
    facilitie_ID INT ,
    job_Title VARCHAR(500),
    work_Setup VARCHAR(500),
    salary Float,
    FOREIGN KEY (requirement_ID) REFERENCES Requirement(requirement_ID),
    FOREIGN KEY (facilitie_ID) REFERENCES Facilitie(facilitie_ID)
);
"""
cursor.execute(createTableJob_query)
```

Create facilities Table

```
createTableFacilitie_query = """
CREATE TABLE Facilitie (
    facilitie_ID INT PRIMARY KEY AUTO_INCREMENT ,
    facilitie_Name VARCHAR(500)
);
"""
cursor.execute(createTableFacilitie_query)
```

Create Company Table

```
createTableCompany_query = """
CREATE TABLE Company (
    company_ID INT PRIMARY KEY AUTO_INCREMENT ,
    company_Name VARCHAR(500)
);
"""
cursor.execute(createTableCompany_query)
```

3. Insertion des données :

Après avoir créé les tables, nous avons parcouru chaque ligne du dataframe et inséré les données dans les tables correspondantes. Nous avons utilisé des dictionnaires pour stocker les IDs générés pour chaque enregistrement, afin de les utiliser ultérieurement pour les associations.

Transform string representation of lists back to lists

```
df['Requirement of the company'] = df['Requirement of the company'].apply(ast.literal_eval)  
df['Facilities'] = df['Facilities'].apply(ast.literal_eval)
```

Pour la table Requirement, nous avons extrait les exigences de chaque entreprise et les avons insérées dans la table. De même, pour la table Facilitie, nous avons extrait les installations associées à chaque entreprise et les avons insérées dans la table. Pour la table Job, nous avons inséré les détails de l'emploi tels que l'ID de l'exigence, l'ID de la facilité, le titre de l'emploi, le type de travail et le salaire.

Pour les tables Company, City et Country, nous avons inséré les détails correspondants, tels que le nom de l'entreprise, la ville et le pays.

Enfin, nous avons créé des associations entre les tables pour refléter les relations entre les emplois, les entreprises, les exigences et les installations. Nous avons utilisé les IDs stockés dans les dictionnaires pour effectuer ces associations.

