



ANALYSE DES OPPORTUNITÉS D'EMPLOI

Réalisé par :

Aymane Sabri

Encadre Par :

M. Tajmouati

Big DATA

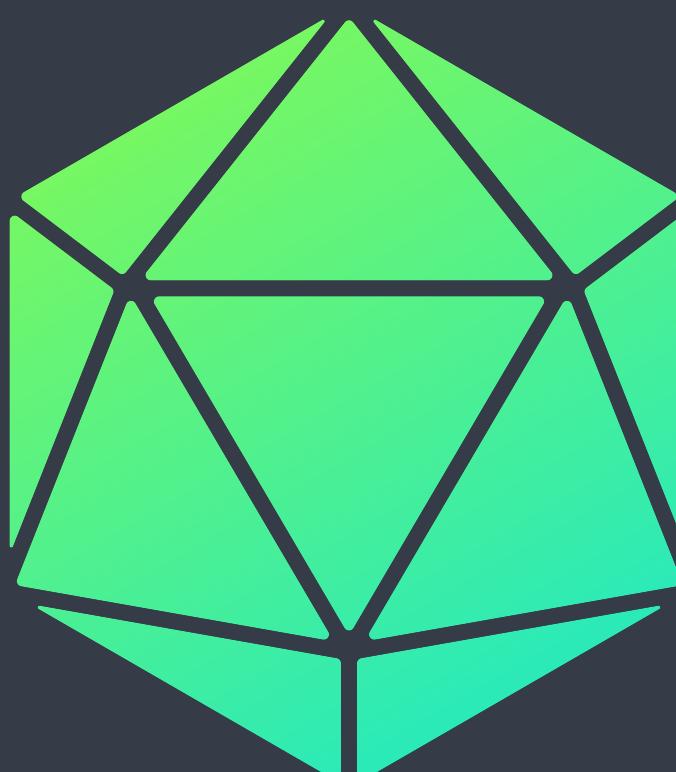
Presentation





Objectifs Du Projet

- Analyse approfondie du marché du travail dans les domaines émergents (Data ...)
- Visualisation claire des résultats pour faciliter la compréhension des parties prenantes.
- Implementation d'un système de stockage des données pour une gestion efficace .



Plan





Introduction

Nous allons 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 en domaine **intelligence artificielle**.

Voici les différents étapes suivies lors de la réalisation de projet :

- Exploration des données
- Nettoyage des données
- Transformation des données
- Visualisations statistiques
- Intégration des données dans la base de données
- Redaction d'un rapport technique



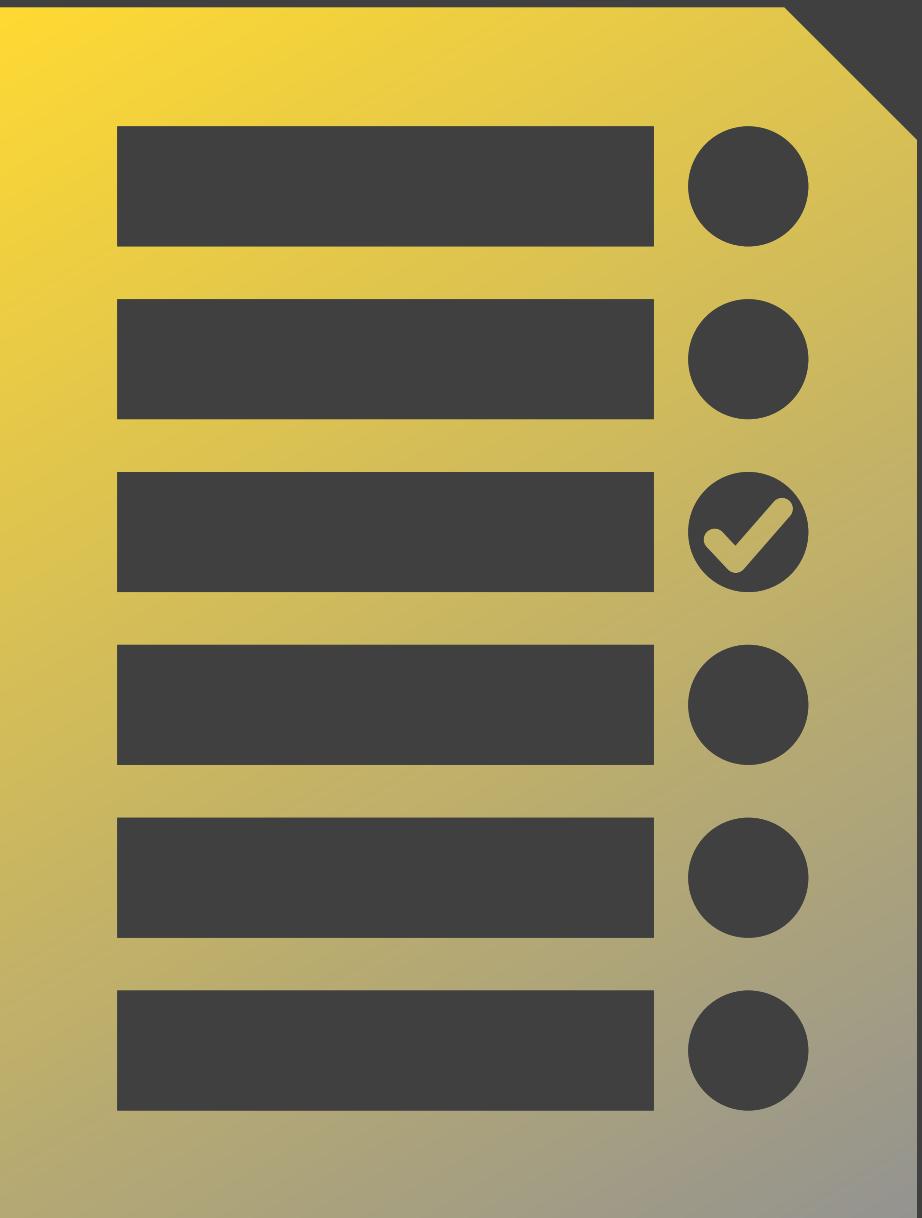


Planification du plan de réalisation du projet

J'ai commencé par établir le **planning** à suivre durant la période de brief . Pour ce faire, j'ai décomposé mon projet en **phases**, où chaque phase est définie par un certain nombre de tâches.

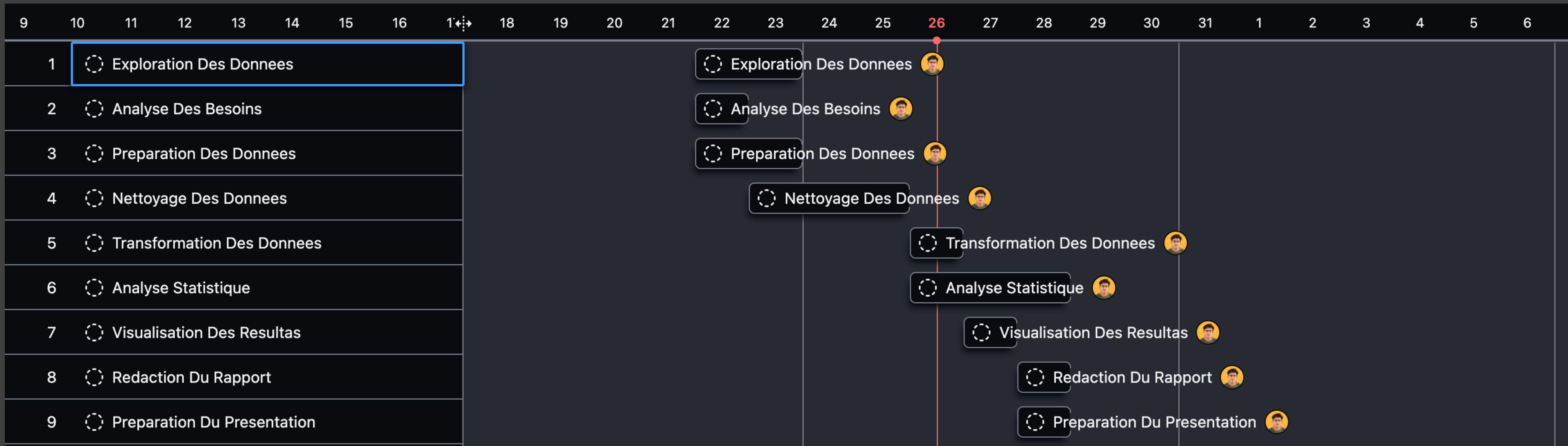
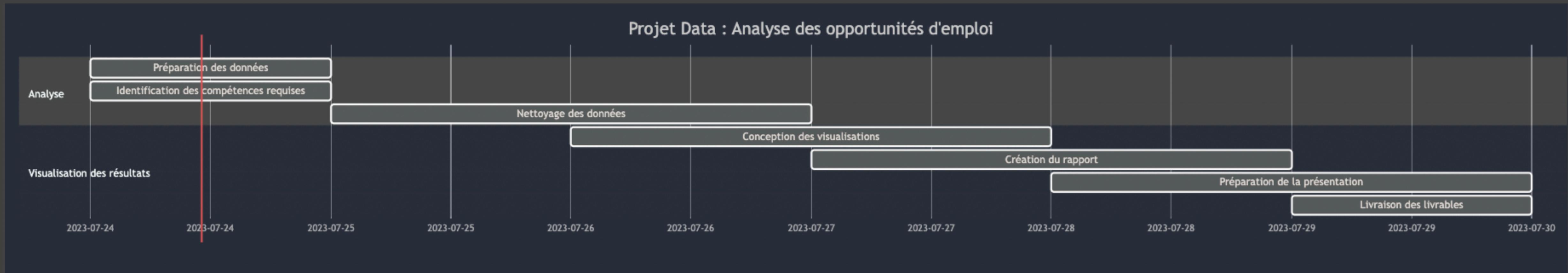
Ensuite, j'ai élaboré une planification de ces phases sur la durée du projet, à l'aide d'un diagramme de **Gantt**.

- Comprendre la nature et l'étendue du travail demandé.
- Identifier le type de recherche d'information demandé.
- Comprendre les objectifs d'apprentissage visés
- Adopter la démarche logique pour exécuter le travail
- Prêter attention aux consignes et aux critères d'évaluation du projet
- Connaitre les **échéances** et avoir l'intention de les respecter.

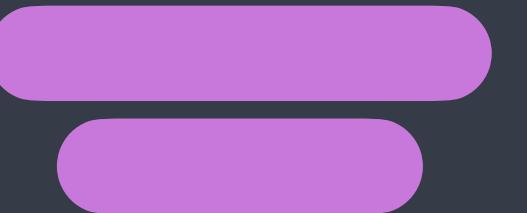




Planification du plan de réalisation du projet



Exploration Des Données





Introduction

L'exploration nous aide à identifier les caractéristiques clés du jeu de données et à repérer d'éventuelles relations entre les variables.

- Présentation du jeu de données utilisé :

La DataFrame contient un total de 3198 entrées réparties sur 8 colonnes.

Voici une description de chaque colonne :

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	Requirement of the company	3198 non-null	object
7	Facilities	3198 non-null	object

dtypes: object(8)
memory usage: 200.0+ KB

- Présentation du jeu de données utilisé :
 - **Company** : Le nom de l'entreprise offrant l'opportunité d'emploi.
 - **Job Title** : Le poste proposé par l'entreprise .
 - **Job Type** : Le type d'emploi offert (par exemple : temps **plein**, **partiel**, 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.
 - **Requirement of the company** : Les exigences spécifiques énoncées par l'entreprise pour le poste.

- Pourcentages des valeurs null par colonne :

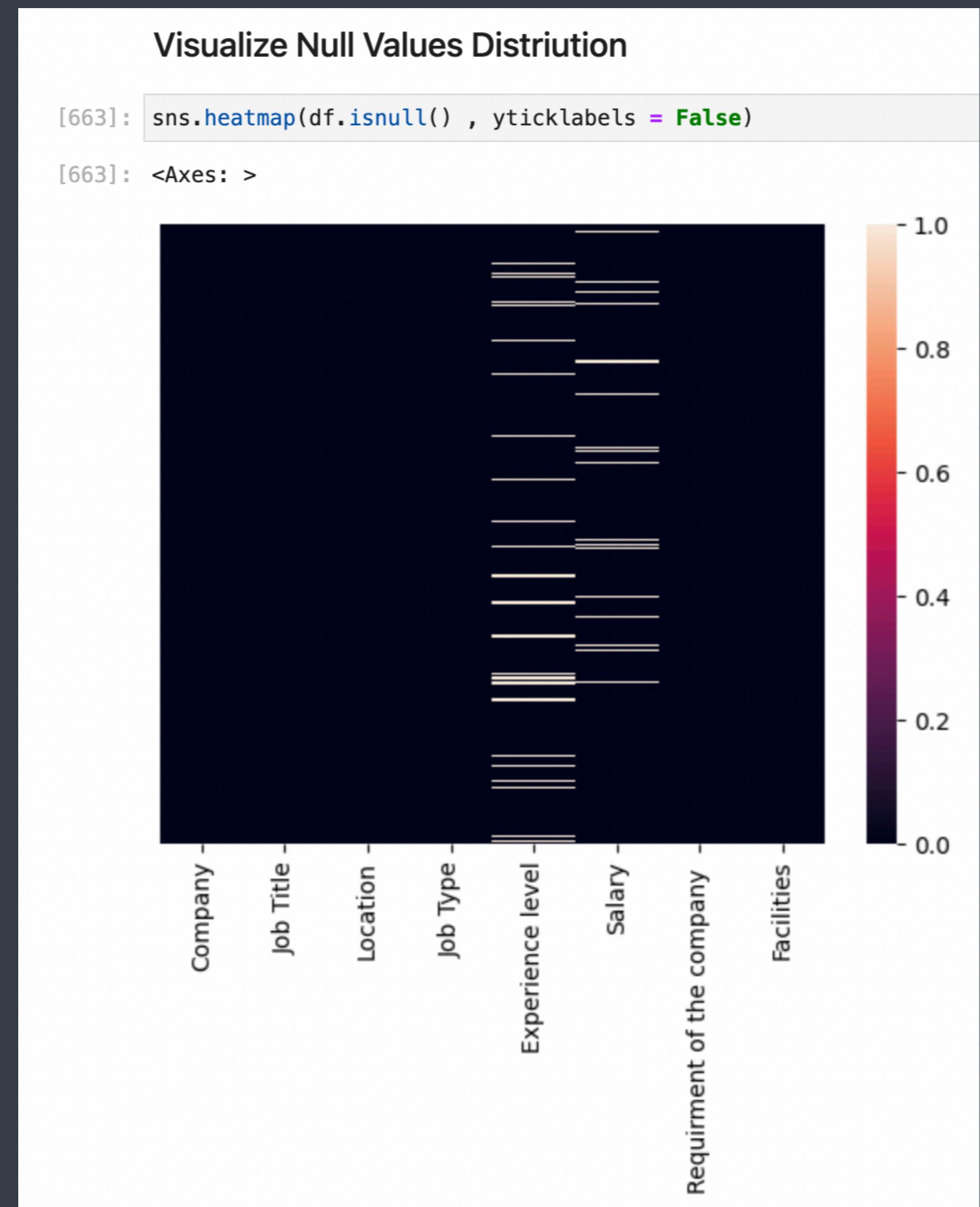
[661]:	Column Data Types	Column Valid Values	Column Null Values
Company	object	99.968730	0.031270
Job Title	object	99.968730	0.031270
Location	object	99.968730	0.031270
Job Type	object	99.968730	0.031270
Experience level	object	92.620388	7.379612
Salary	object	94.090056	5.909944
Requirement of the company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

- Representation de pourcentages des valeurs null par colonne :

On remarque , que les colonnes :

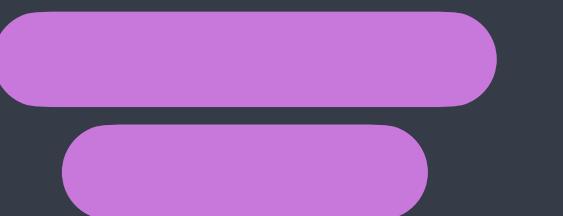
- Experience Level .
- Salary

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





Nettoyage Des
données





Nettoyage Des données

- Gestion des Valeurs Manquantes :

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 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.

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
```



Nettoyage Des données

- 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

```
df = df.drop_duplicates()
```

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

Remaining Rows : 2996



Nettoyage Des données

- Transformation réalisées pour la Colonne 'Experience level' :

Pour compléter le processus, les valeurs de la colonne 'Experience level' ont été uniformisées en gardant uniquement la partie pertinentes), 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)|
```



Nettoyage Des données

- Conversion des Salaires en USD :

Une fonction `extract_currency_code` a été créée pour extraire le code de devise (USD, GBP ou EUR) .

Ensuite, une fonction `convert_to_dollars` a été mise en place pour convertir les salaires en USD à l'aide d'une API. Les salaires ont été convertis en valeurs numériques et la colonne 'Salary' a été renommée en 'Salary(USD)' pour refléter cette transformations .

```
: 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 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 GBP to USD succeeded.
```

```
df.rename(columns={'Salary': 'Salary(USD)'}, inplace=True)
```

```
df['Salary(USD)'].replace(0, np.nan, inplace=True)
```



Nettoyage Des données

- Traitement des Colonnes 'Requirement of the company' et 'Facilities' :

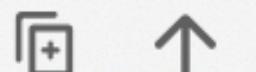
Les données dans les colonnes 'Requirement of the company' et 'Facilities' ont été nettoyées en divisant les chaînes de caractères séparées par des virgules en listes de mots-clés pertinents.

Create Requirement Column

```
df['Requirment of the company '] = df['Requirment of the company '].str.split(',')
df['Requirment of the company '] = df['Requirment 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(',')
```





Nettoyage Des données

- 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`.

```
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
```

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)
```

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)
```



Nettoyage Des données

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

Une fonction `set_job_type` a été créée pour identifier si le travail est en présentiel 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'

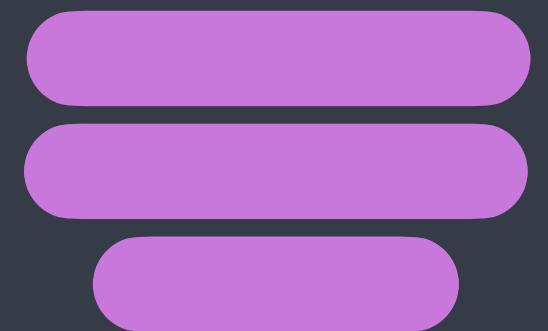
df['work setups'] = df.apply(set_job_type, axis=1)
```

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))
```

Visualizations





Analyse

- Carte des offres d'emploi par pays :



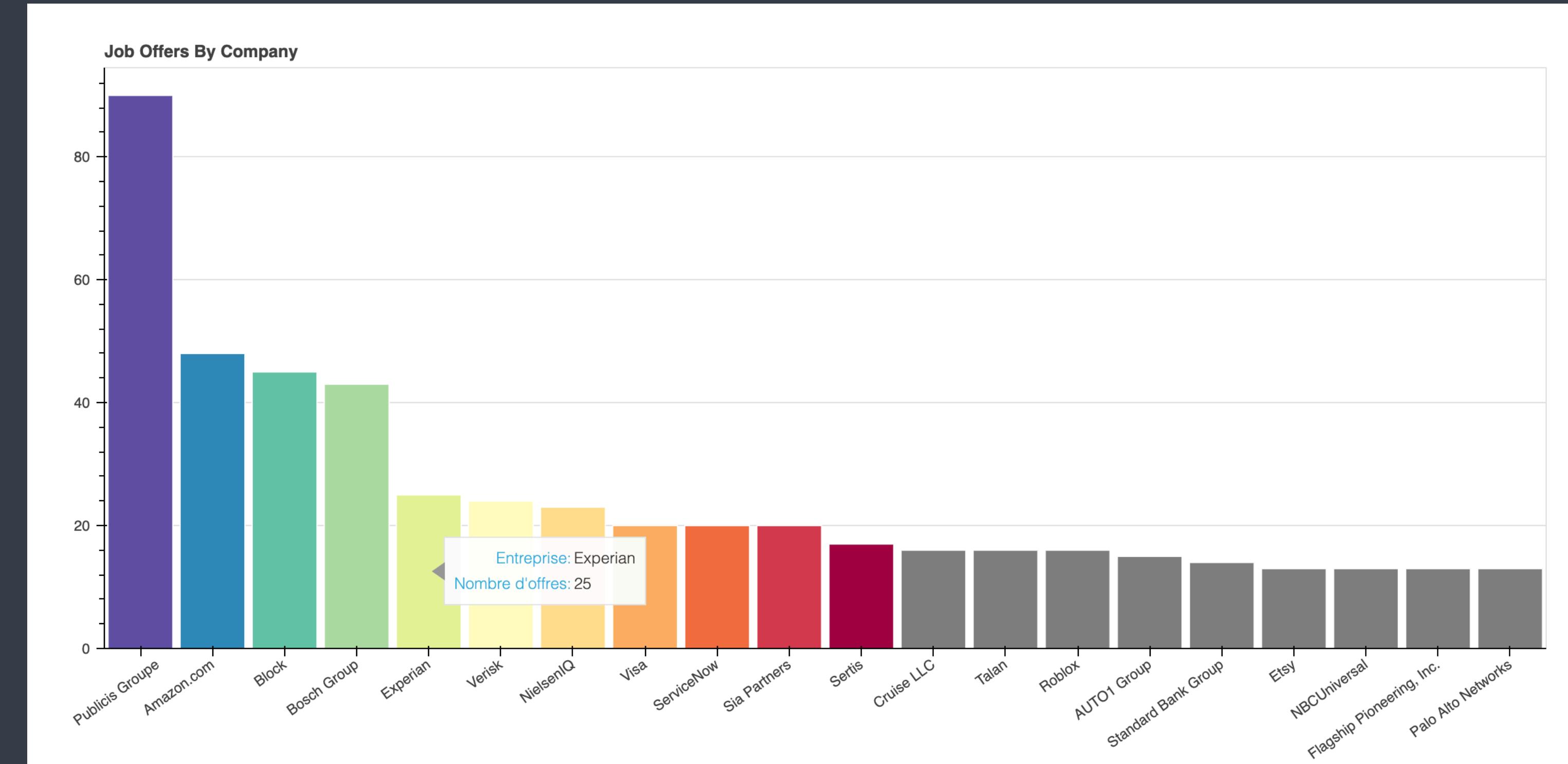


Analyse

- Top 20 des meilleurs employeurs :

Le graphique présente les 20 meilleures entreprises qui offrent le plus grand nombre d'emplois dans le domaine de la data.

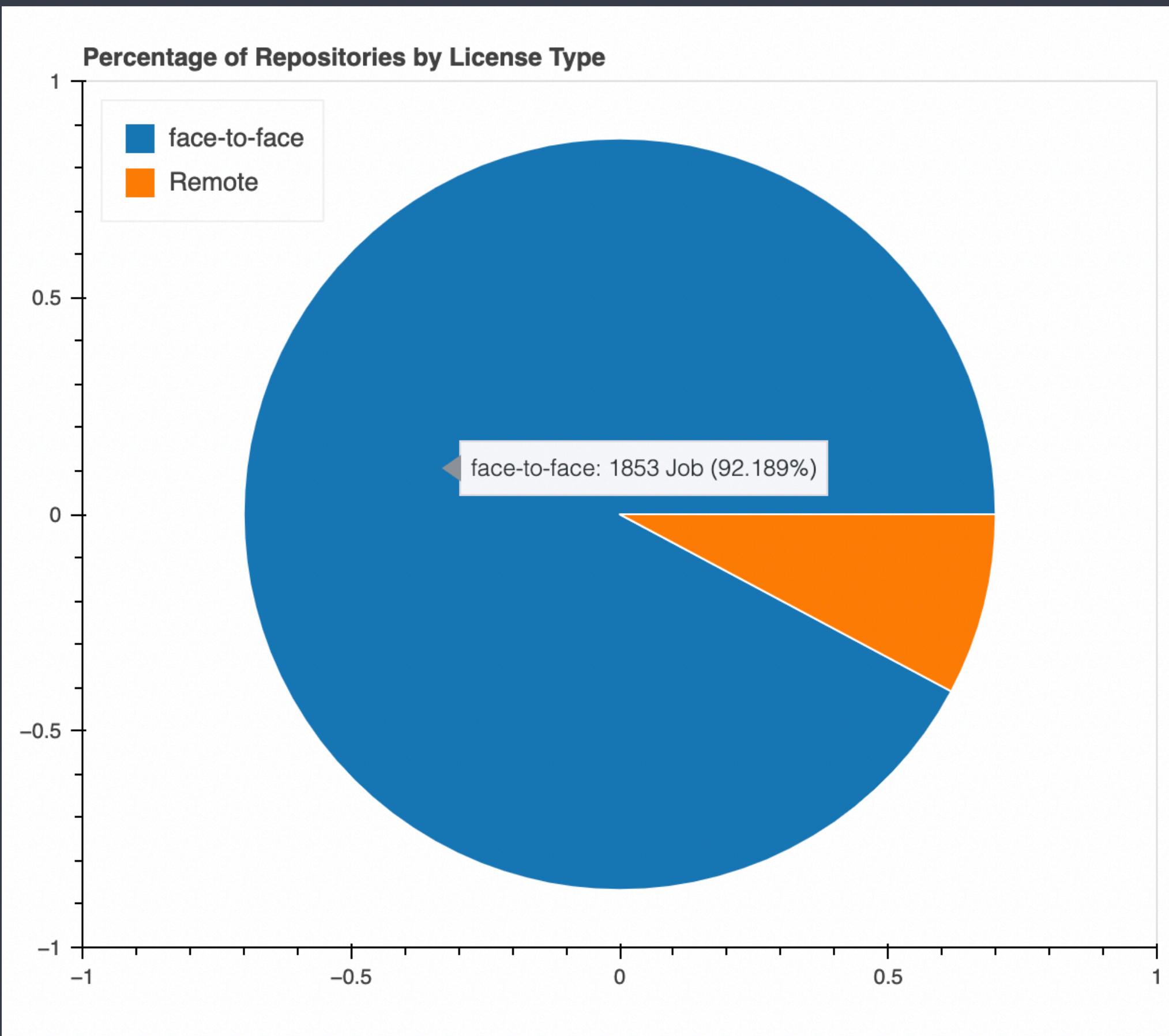
Ce graphique permet d'identifier les entreprises les plus actives dans le recrutement de talents en data.





Analyse

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

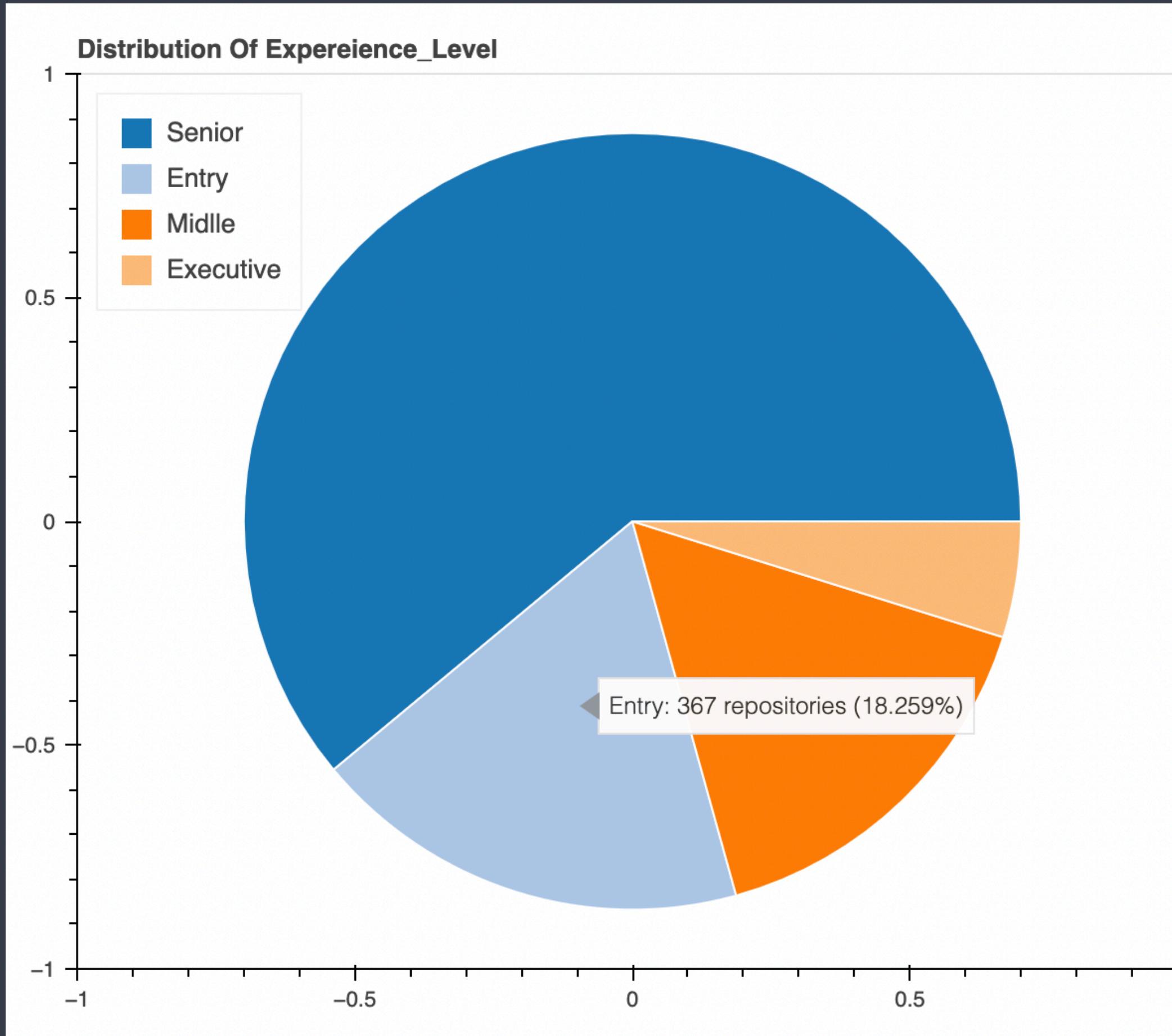


Ce graphique montre la répartition des types de postes .



Analyse

- Répartition des niveaux d'expérience requis :



Ce graphique montre la répartition des expériences demandées.

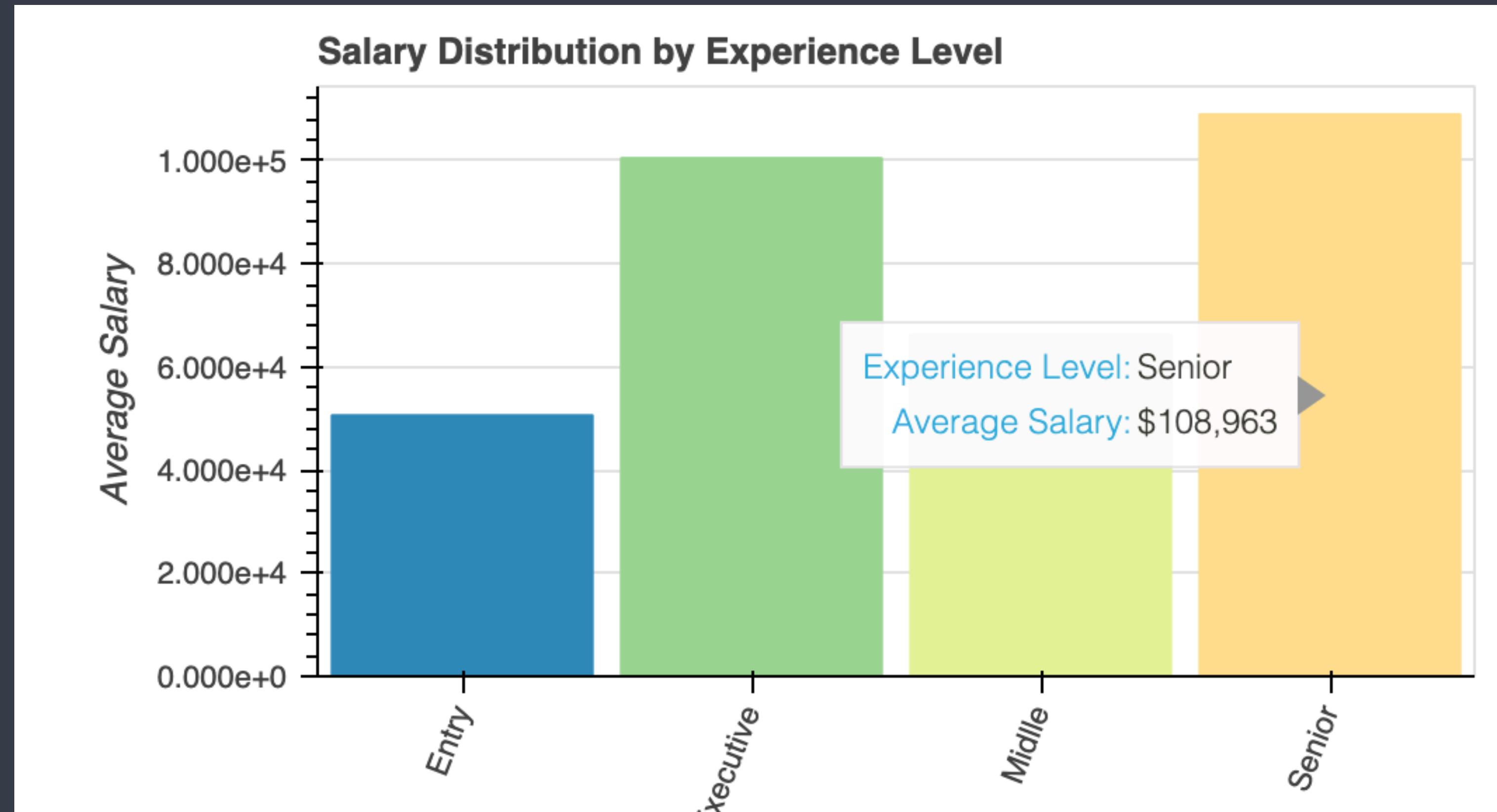


Analyse

- Répartition des niveaux d'expérience requis :

Ce graphique illustre le **salaire moyen** offert pour chaque niveau d'expérience.

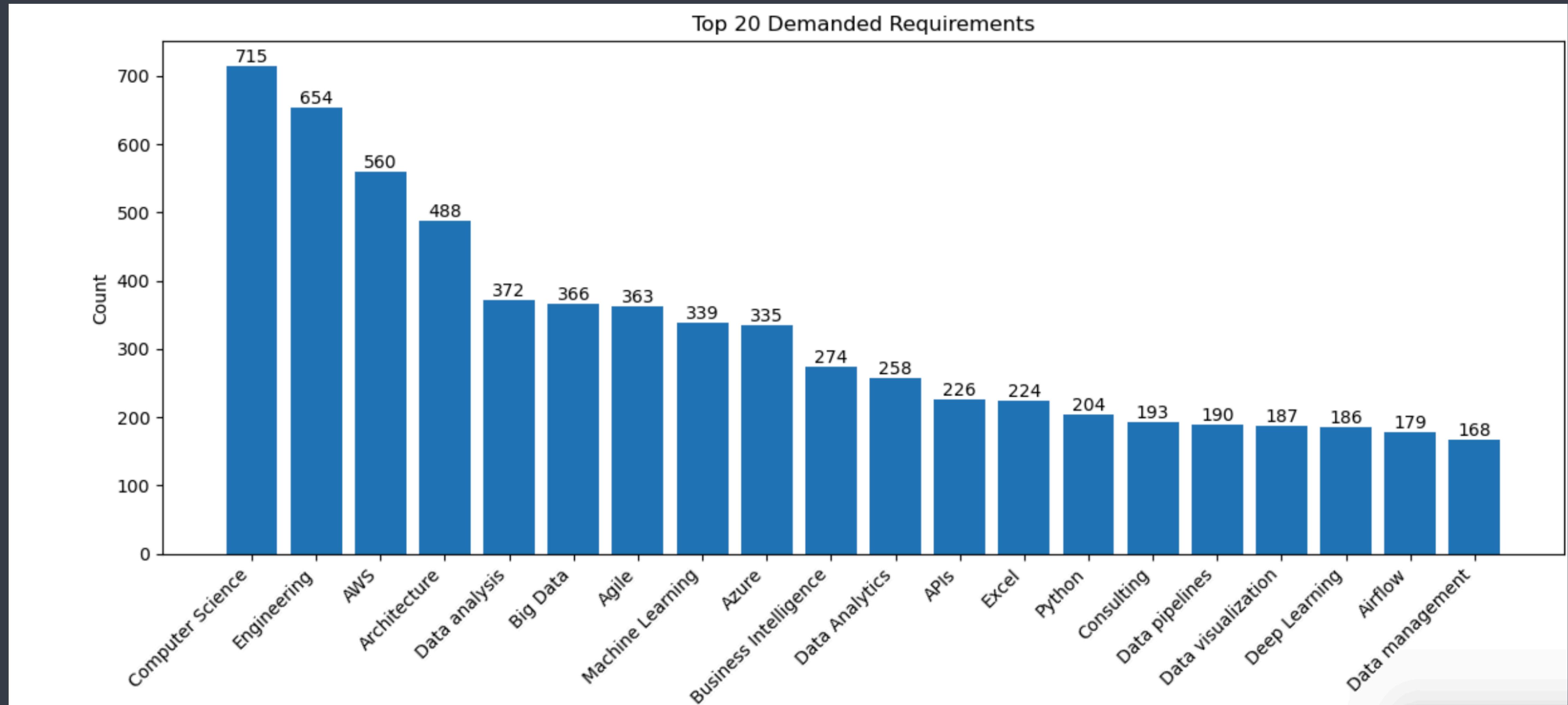
Ce graphique permet de comparer les salaires moyens pour différents niveaux d'expérience et de déterminer les **différences salariales** entre les niveaux.





Analyse

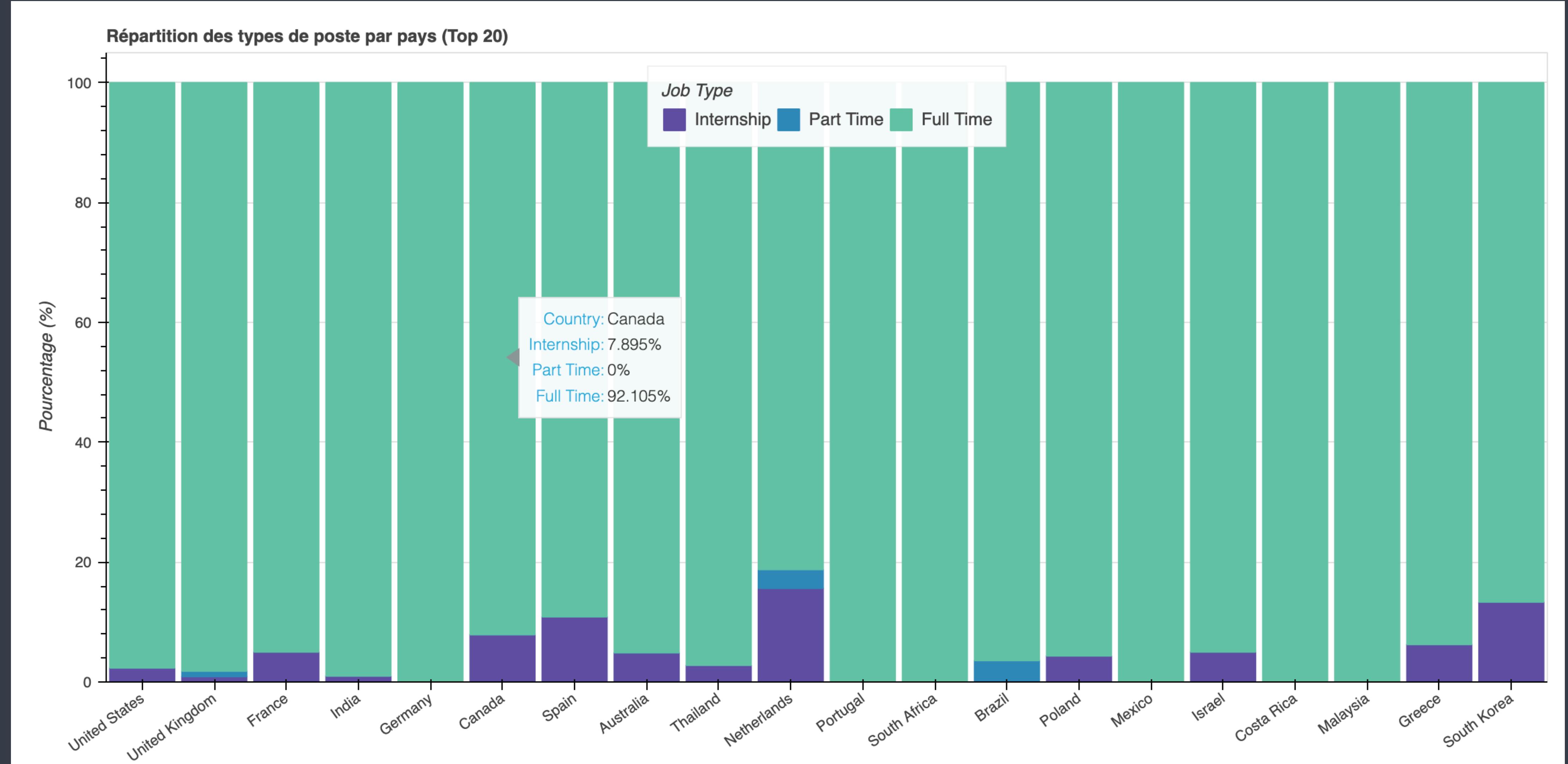
- Top 20 des compétences les plus recherchées :





Analyse

- Répartition des types de poste par pays :



Integration Données





Integration Données

- **Introduction :**

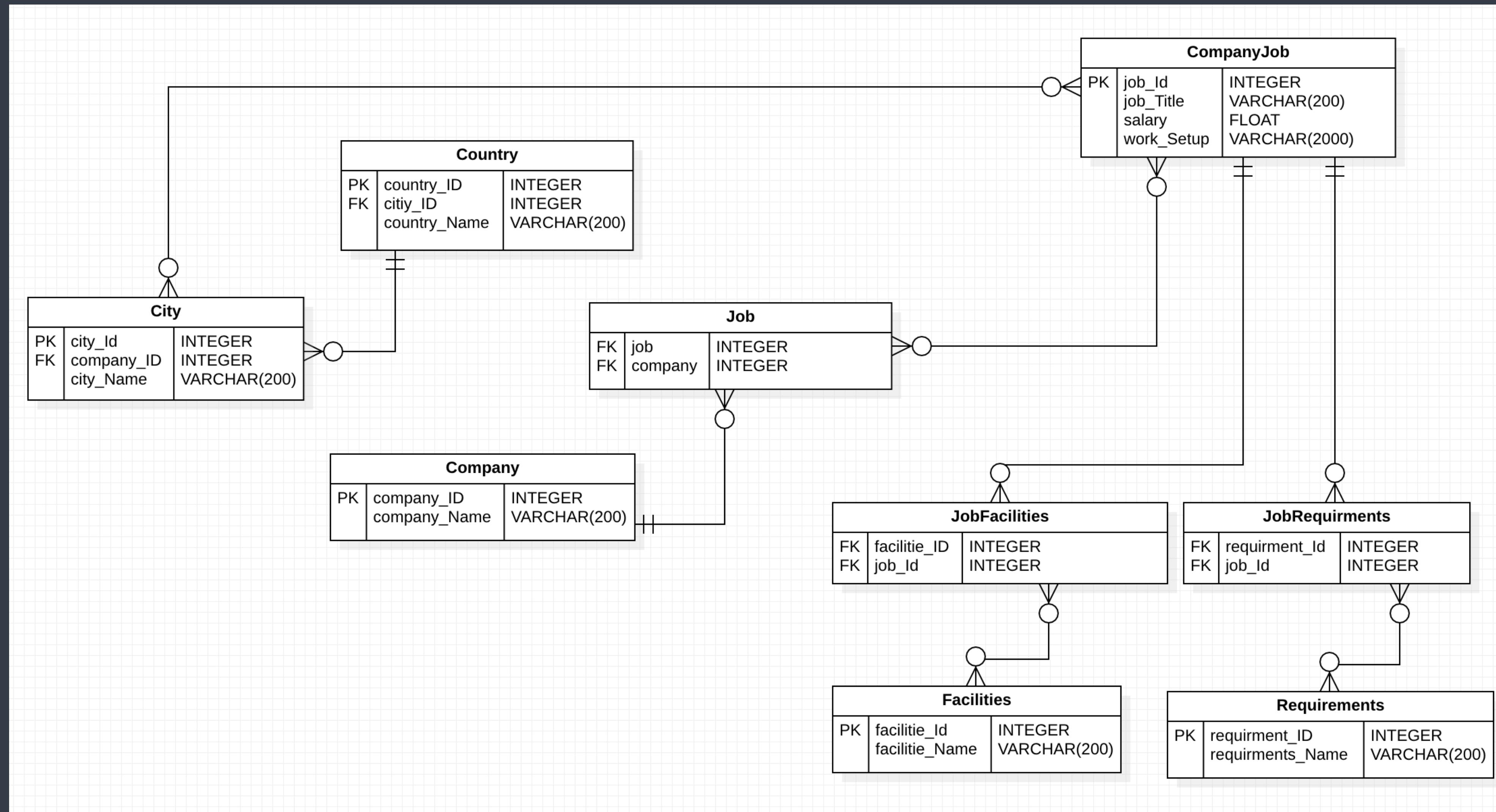
Dans la partie d'intégration du dataframe nettoyé dans la base de données relationnelle, nous avons réalisé plusieurs opérations pour créer les tables correspondantes et insérer les données du dataframe dans ces tables.





Integration Données

- ER Diagramm Model :





Integration Données

- Creation Des Tables :

Nous avons créé les tables nécessaires pour stocker les données du dataframe.

Nous avons créé les tables :

Requirement, Facilities, 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 Job Table

```
create_table_job_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(create_table_job_query)
```

Create Company Table

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



Integration Données

- Transformation des données :

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)
```

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.



Integration Données

- Obtenez le nombre total d'emplois pour chaque entreprise

Get the total number of jobs for each company

```
query = """
SELECT c.company_Name, COUNT(jca.job_ID) AS total_jobs
FROM Company c
LEFT JOIN JobCompanyAssociation jca ON c.company_ID = jca.company_ID
GROUP BY c.company_Name
ORDER BY total_jobs DESC
LIMIT 10;
"""
cursor.execute(query)
result_2 = cursor.fetchall()
```

	Jobs	Count
0	Publicis Groupe	90
1	Amazon.com	48
2	Block	45
3	Bosch Group	43
4	Experian	25
5	Verisk	24
6	NielsenIQ	23
7	Sia Partners	20
8	Visa	20
9	ServiceNow	20



Integration Données

- Trouver le salaire moyen pour chaque intitulé de poste

Find the average salary for each job title ↴

```
query = """  
SELECT job_Title, AVG(salary) AS avg_salary  
FROM Job  
GROUP BY job_Title;  
"""  
  
cursor.execute(query)  
result_3 = cursor.fetchall()
```

	Jobs	Salary
0	Artificial Intelligence	73629.929275
1	Big Data Engineer	76581.011017
2	Data Architect	85285.714286
3	Data Product Manager	58142.857143
4	Machine Learning Engineer	101931.994335
5	Other	69827.256338



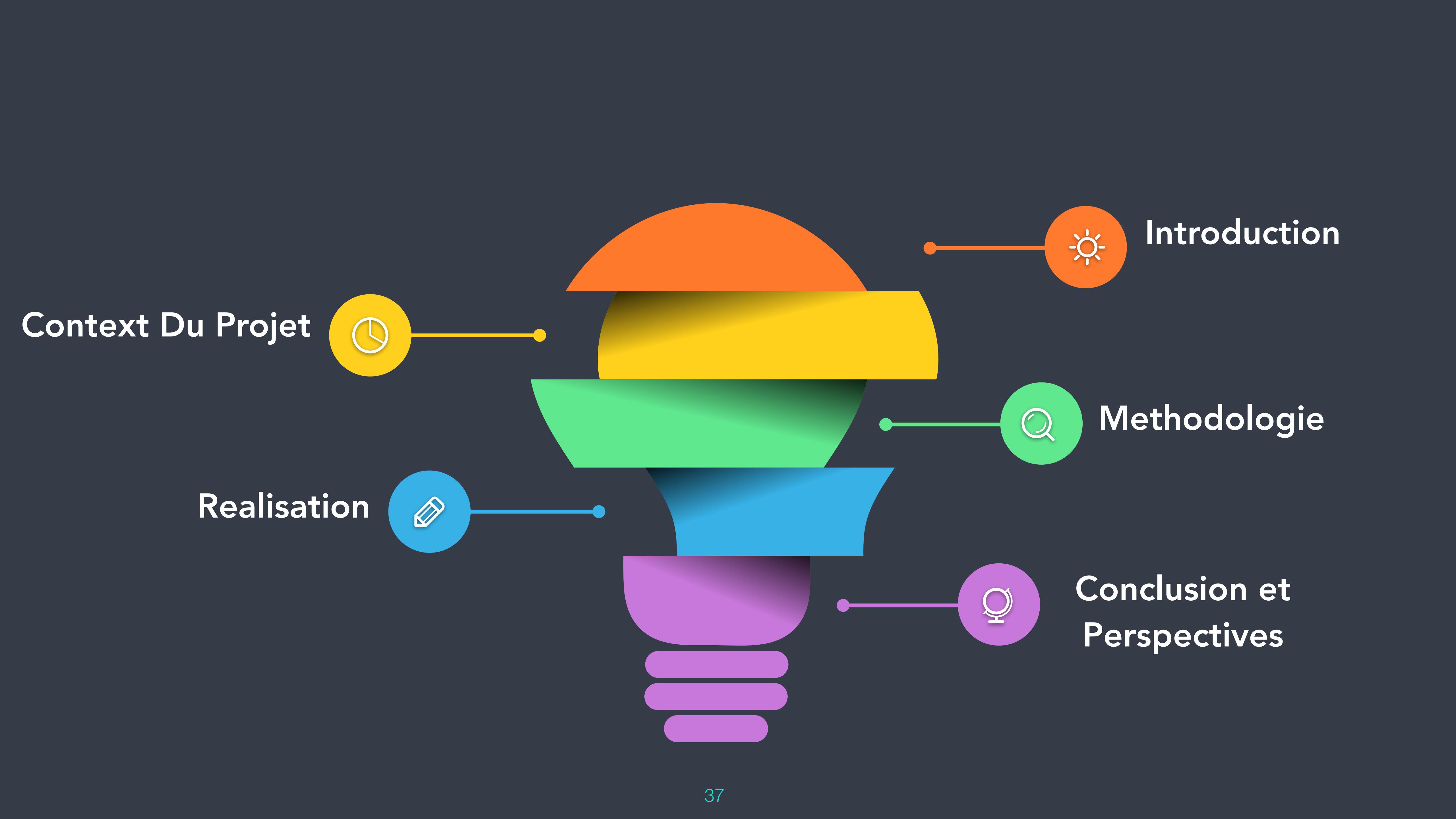
Integration Données

- Énumérez tous les titres d'emploi qui nécessitent l'exigence "Informatique"

```
query = """
SELECT DISTINCT j.job_Title
FROM Job j
JOIN JobRequirementsAssociation jra ON j.job_ID = jra.job_ID
JOIN Requirement r ON jra.requirement_ID = r.requirement_ID
WHERE r.requirement_Name = 'Computer Science';
"""

cursor.execute(query)
result_4 = cursor.fetchall()
```

Jobs	
0	Big Data Engineer
1	Machine Learning Engineer
2	Artificial Intelligence
3	Other
4	Data Architect
5	Data Product Manager





Merci Pour Votre Attention