# Workshop 001 - Documentation

*Valentina Bueno Collazos - 2230556*

## Project objective

This project analyzes data from candidates who participated in job selection processes. It includes an ETL (Extract, Transform, Load) pipeline to create the following visualizations for a dashboard:

- **Hires by technology** (pie chart)
- **Hires by year** (horizontal bar chart)
- **Hires by seniority** (bar chart)
- **Hires by country over years** (multiline chart, focusing on the USA, Brazil, Colombia, and Ecuador only)

## Project Structure

```
recruitment_analysis/
│
├── dashboard/
│   └── hires_visualization.pdf       # PDF export of the dashboard
│
├── data/
│   └── candidates.csv                # Initial dataset of candidates
│
├── docs/
│   └── documentation.pdf             # Detailed project documentation
│
├── images/
│   ├── plot1.png                     # Graphs generated during EDA
│   ├── plot2.png
│   └── plot3.png
│
├── notebooks/
│   ├── 001_initial_data_load.ipynb   # Initial data loading
│   ├── 002_eda.ipynb                 # Exploratory Data Analysis (EDA)
│   └── 003_clean_transform_load.ipynb # Data cleaning, transformation, and loading
│
├── scripts/
│   ├── credentials.json              # Credentials for database connection
│   └── db_connector.py               # Script to connect and load data to the DB
│
├── .gitignore                        # File to ignore files in Git
└── README.md                         # General project description
```

## Project ETL Process

## Extraction

Data is loaded from the `candidates.csv` file located in the `data/` folder.

### Initial Data Load Process (001_initial_data_load.ipynb)

This notebook performs the initial raw data loading from a CSV file into a PostgreSQL database without any transformations. It serves as the first step in the ETL pipeline.

- **System Path Configuration**

  Adds the project root directory to the system path to enable importing custom scripts

  ```
  import sys
  import os
  sys.path.append(os.path.abspath(".."))
  ```

- **Database Connection**

  Establishes connection to PostgreSQL using credentials from `db_connector.py`

  ```
  from scripts.db_connector import get_db_engine
  conn = get_db_engine()
  ```

- **Table Creation**

  Creates `candidates_initial_data` table with schema:

  ```
  CREATE TABLE candidates_initial_data (
      first_name TEXT,
      last_name TEXT,
      email TEXT,
      application_date DATE,
      country TEXT,
      yoe INT,
      seniority TEXT,
      technology TEXT,
      code_challenge_score INT,
      technical_interview_score INT
  )
  ```

- **Data Loading**

  Loads raw CSV data using PostgreSQL `COPY` command:

  ```
  csv_file_path = os.path.join(os.getcwd(), "data", "candidates.csv")
  copy_sql = """COPY candidates_initial_data
          FROM stdin
          DELIMITER ';'
          CSV HEADER;"""
  ```

- **Data Verification**

  Validates successful load with sample query:

```
query = "SELECT * FROM candidates_initial_data LIMIT 10;"
df = pd.read_sql(query, conn)
```

### Sample Raw Data Structure

| | first_name | last_name | email | application_date | country | yoe | seniority | technology | code_challenge_score | technical_interview_score |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Bernadette | Langworth | leonard91@yahoo.com | 2021-02-26 | Norway | 2 | Intern | Data Engineer | 3 | 3 |
| 1 | Camryn | Reynolds | zelda56@hotmail.com | 2021-09-09 | Panama | 10 | Intern | Data Engineer | 2 | 10 |
| 2 | Larue | Spinka | okey_schultz41@gmail.com | 2020-04-14 | Belarus | 4 | Mid-Level | Client Success | 10 | 9 |
| 3 | Arch | Spinka | elvera_kulas@yahoo.com | 2020-10-01 | Eritrea | 25 | Trainee | QA Manual | 7 | 1 |
| 4 | Larue | Altenwerth | minnie.gislason@gmail.com | 2020-05-20 | Myanmar | 13 | Mid-Level | Social Media Community Management | 9 | 7 |
| 5 | Alec | Abbott | juanita_hansen@gmail.com | 2019-08-17 | Zimbabwe | 8 | Junior | Adobe Experience Manager | 2 | 9 |

# Transformation

## EDA (002_EDA.ipynb)

The goal of this notebook is to understand data quality, detect anomalies, and guide subsequent cleaning/transformation steps.

- **Data Loading**

  Data was loaded from the `candidates_initial_data` table in a PostgreSQL database using `pandas.read_sql`.

- **Initial Validations**
  - **Data Shape and Structure**

```
df.shape
```

```
(50000, 10)
```

  - **Data Types**:

    After validating the DataFrame information, two key aspects were identified:

    1. Absence of Null Values: No null values were found in any columns of the DataFrame.
    2. Incorrect Data Type in application_date: Although the application_date column contains dates, its current data type is object. To facilitate analysis, it needs to be converted to datetime.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   first_name                50000 non-null  object
 1   last_name                 50000 non-null  object
 2   email                     50000 non-null  object
 3   application_date          50000 non-null  object
 4   country                   50000 non-null  object
 5   yoe                       50000 non-null  int64
 6   seniority                 50000 non-null  object
 7   technology                50000 non-null  object
 8   code_challenge_score      50000 non-null  int64
 9   technical_interview_score 50000 non-null  int64
dtypes: int64(3), object(7)
memory usage: 3.8+ MB
```

- **Descriptive Statistics**
  - **Numerical Columns**

| | yoe | code_challenge_score | technical_interview_score |
|---|---|---|---|
| count | 50000.000000 | 50000.000000 | 50000.000000 |
| mean | 15.286980 | 4.996400 | 5.003880 |
| std | 8.830652 | 3.166896 | 3.165082 |
| min | 0.000000 | 0.000000 | 0.000000 |
| 25% | 8.000000 | 2.000000 | 2.000000 |
| 50% | 15.000000 | 5.000000 | 5.000000 |
| 75% | 23.000000 | 8.000000 | 8.000000 |
| max | 30.000000 | 10.000000 | 10.000000 |

**Key Observations**:

→ `yoe` ranges from **0–30 years**, with a median of 15.

→ Score columns (`code_challenge_score`, `technical_interview_score`) follow uniform distributions between **0–10**.

- **Categorical Columns**

| | first_name | last_name | email | application_date | country | seniority | technology |
|---|---|---|---|---|---|---|---|
| count | 50000 | 50000 | 50000 | 50000 | 50000 | 50000 | 50000 |
| unique | 3007 | 474 | 49833 | 1646 | 244 | 7 | 24 |
| top | Sarai | Murazik | fern70@gmail.com | 2020-07-07 | Malawi | Intern | Game Development |
| freq | 33 | 138 | 3 | 50 | 242 | 7255 | 3818 |

- **Duplicate Analysis**
  - **Full Duplicates**

    No duplicate rows were found (`df.duplicated().sum() = 0`).

```
print(df.duplicated().sum())
```

```
0
```

  - **Email Duplicates**

    **167 duplicate emails** were identified.

```
print(df["email"].duplicated().sum())
```

```
167
```

I decided to retain duplicates as they may represent valid re-applications (confirmed by checking uniqueness across `first_name`, `last_name`, and `application_date`).

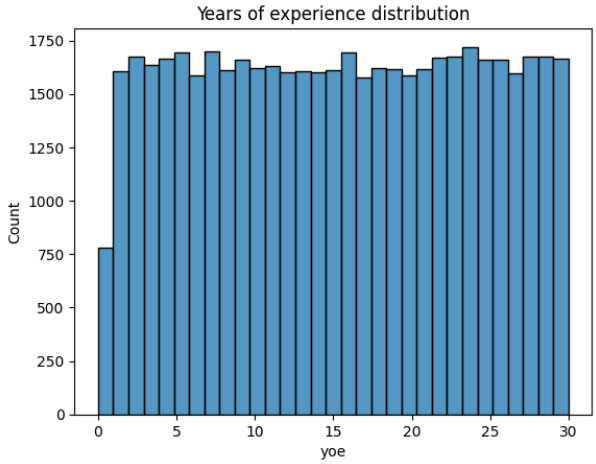
```
num_duplicates = df.duplicated(subset=["first_name", "last_name", "email", "application_date"]).sum()
print(f"Total duplicated records: {num_duplicates}")
```

```
Total de registros duplicados: 0
```

- **Outlier Detection**
  - **Numerical Columns**

    **Years of Experience (`yoe`)**

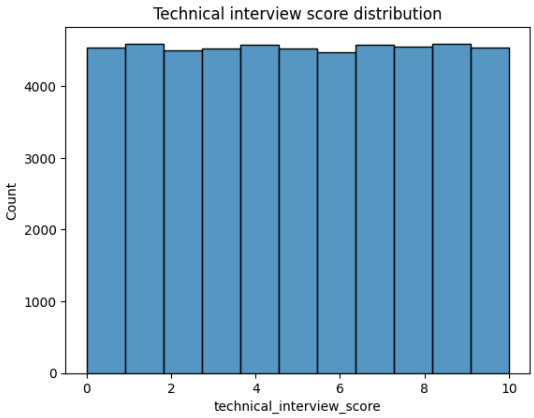
    - **Distribution**:




    - **IQR Analysis**:
      - Valid range: **[-14.5, 45.5]**.
      - All values fall within this range. **No outliers removed**.

    ### Scores (`code_challenge_score`, `technical_interview_score`)

    - **Distribution**:





    - **All scores** are within the expected **0–10 range**.
  - **Categorical Columns**
    - `seniority`: No unexpected values.

```
print(df['seniority'].unique())
```

```
['Intern' 'Mid-Level' 'Trainee' 'Junior' 'Lead' 'Architect' 'Senior']
```

- `technology`: Categories align with job specializations.

```
print(df['technology'].unique())
```

```
['Data Engineer' 'Client Success' 'QA Manual'
 'Social Media Community Management' 'Adobe Experience Manager' 'Sales'
 'Mulesoft' 'DevOps' 'Development - CMS Backend' 'Salesforce'
 'System Administration' 'Security' 'Game Development'
 'Development - CMS Frontend' 'Security Compliance'
 'Development - Backend' 'Design'
 'Business Analytics / Project Management' 'Development - Frontend'
 'Development - FullStack' 'Business Intelligence'
 'Database Administration' 'QA Automation' 'Technical Writing']
```

---

## EDA Conclusions to perform transformations

- The data type of application_date should be changed from object to datetime.
- The values of the categorical columns will be standardized by changing them to lowercase letters.
- I´ll ad 2 columns: application_status and application_year. This to be able to do the visualizations easily.
- The columns first_name, last_name and email will be deleted because they are not relevant for the visualizations.

---

## Cleaning and transformation (003_clean_transform_load.ipynb)

- **Data Cleaning**

  The goal is to prepare raw data for analysis by ensuring consistency and correct data types.

  1. **Convert `application_date` to datetime**:

     Changed the data type of `application_date` from `object` to `datetime64[ns]` to enable time-based operations.

     ```
     df["application_date"] = df["application_date"].astype("datetime64[ns]")
     ```

  2. **Standardize Categorical Columns**:

     Converted all values of `email`, `country`, `seniority`, `technology` to lowercase to ensure uniformity.

     ```
     cols = ["email", "country", "seniority", "technology"]
     df[cols] = df[cols].apply(lambda x: x.str.lower())
     ```

- **Data Transformations**

  The goal is to enhance dataset for dashboard visualizations by deriving new metrics and removing non-essential fields.

  1. **Calculate `application_status`**:

     A candidate is marked as "hired" (`1`) if both `code_challenge_score` and `technical_interview_score` are ≥7. Otherwise, "not hired" (`0`). Then a new column is created and the values calculated by the function are imputed to it.

```
def assign_application_status(df):
    if (df["code_challenge_score"] >= 7) and (df["technical_interview_score"] >= 7):
        return 1
    else:
        return 0
df["application_status"] = df.apply(assign_application_status, axis=1)
```

2. **Extract** `application_year`:

   Extract `application_year` from `application_date`. This to facilitate time-series analysis (e.g., hires per year).

   ```
   df["application_year"] = df["application_date"].dt.year
   ```

3. **Remove Non-Essential Columns**:

   - **Columns Dropped**: `first_name`, `last_name`, `email`.
   - **Reason**: Reduce noise and focus on analytical fields.

   ```
   df.drop(columns=["first_name", "last_name", "email"], inplace=True)
   ```

---

### Key Considerations

- **Data Integrity**: No imputation or deletion was needed per EDA conclusions.
- **Performance**: Used vectorized operations (e.g., `.dt.year`) for efficiency.
- **Privacy**: Explicit removal of PII (`email`, `first_name`, `last_name`) aligns with GDPR compliance.
- **Dashboard Readiness**: Final schema includes only fields relevant to analytics (e.g., `technology`, `seniority`, `application_status`).

---

This documentation reflects alignment with the notebook logic, prior discussions on data privacy, and optimization for analytical use cases.

## Loading

### Load (003_clean_transform_load.ipynb)

The goal of this part of the notebook is to persist cleaned and transformed data into the database for dashboard consumption.

- **Database connection**

  SQLAlchemy engine with credentials loaded from `credentials.json`.

  ```
  with open("scripts/credentials.json", "r", encoding="utf-8") as file:
      credentials = json.load(file)
  pg_engine = create_engine(
      f"postgresql://{credentials['db_user']}:{credentials['db_password']}@{credentials['db_host']}:5432/{crede
  ntials['db_name']}"
  )
  ```

- **Load Data to Database**:

  The processed data is loaded into a database

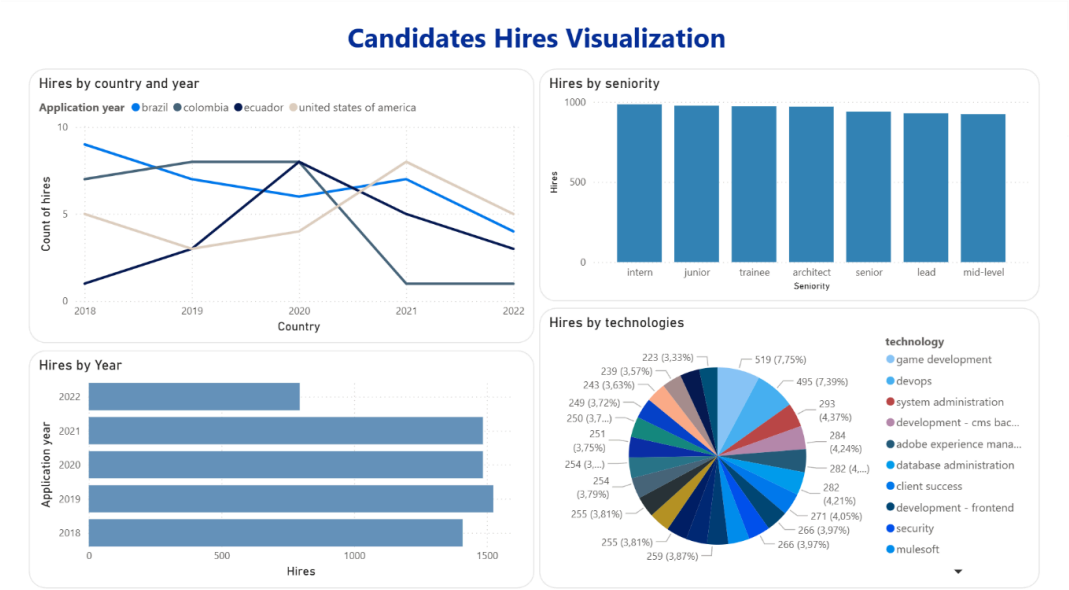
```
df.to_sql("candidates_final_data", pg_engine, index=False)
```

- **Verification**:

  A sample query confirms successful load.

```
SELECT * FROM candidates_final_data LIMIT 10;
```

# Visualizations in Power BI




## Charts

1. **Hires by Technology**: Pie chart showing hires by technology.
2. **Hires by Year**: Horizontal bar chart showing hires by year.
3. **Hires by Seniority**: Bar chart showing hires by seniority level.
4. **Hires by Country Over Years**: Multiline chart showing the evolution of hires in the USA, Brazil, Colombia, and Ecuador.

## Analysis

1. **Hiring trends by country and year:**
   - There is variability in hiring in different countries over time.
   - Brazil and Colombia have maintained a steady flow of hires, while Ecuador had an increase in 2020 and then a drop.
   - The U.S. shows a decrease in hires from 2018 to 2021, with a slight recovery in 2022.
2. **Hirings by seniority leve**l:
   - The distribution of hires by experience level is fairly balanced.

- Interns, juniors, trainees and architects have similar volumes of hires, indicating a diversified hiring strategy in terms of experience.

3. **Hirings by year:**
   - There was a high volume of hires between 2018 and 2021, with a notable decrease in 2022.
   - This could indicate a reduction in demand for talent or changes in hiring strategy.
4. **Hiring by technology:**.
   - There is a high diversity in the technologies hired.
   - Some of the most represented include **Game Development, DevOps and System Administration**.
   - This analysis allows us to identify which technology areas are most in demand and where future hiring strategies could be focused.

# Usage Instructions

## Requirements

The technologies used are:

- Python
  - *Libraries: Pandas , Psycopg2, SQLAlchemy, Matplotlib, Pandas, Seaborn*
- Jupyter Notebook
- PostgreSQL
- PowerBI Desktop

## Steps to Run the Project

1. Clone the repository.
2. Install dependencies: `pip install -r requirements.txt`.
3. Run the notebooks in order:
   - `001_initial_data_load.ipynb`: Initial data loading.
   - `002_eda.ipynb`: Exploratory Data Analysis.
   - `003_clean_transform_load.ipynb`: Data cleaning, transformation, and loading.
4. Open the `hires_visualization.pbix` file in Power BI to view the visualizations.

# Conclusion

This project provides a clear and detailed view of recruitment processes, enabling the identification of trends and data-driven decision-making. The documentation and code are designed to be easy to understand, reproduce, and maintain.