Certainly! Let’s go through each table in the schema and explain its purpose, columns, and relationships.

**1. users Table**

This table stores information about each user of the system.

* **Columns:**
  + user\_id: Primary key, unique identifier for each user.
  + email: Stores the user’s email address, must be unique.
  + last\_name and first\_name: Store the user's last and first names, respectively.
  + phone: Stores the user's phone number.
  + active: Boolean indicating if the user is active (default is true).
  + creation\_date: Timestamp recording when the user was created.
  + status: Specifies the role/status of the user. It’s an ENUM with possible values 'candidate', 'recruiter', or 'admin'. The default value is 'candidate'.
  + password: Stores the user's password in an encrypted format (varchar).
* **Purpose:** This table is the main user table for the application, where all users (candidates, recruiters, admins) are stored.

**2. admin Table**

This table stores administrators. Each admin is also a user in the users table.

* **Columns:**
  + admin\_id: Primary key, also a foreign key that references user\_id in the users table.
* **Purpose:** Represents administrators specifically. It relies on user\_id to ensure each admin is also a user in the system. If an admin is deleted in the users table, they will automatically be removed from admin due to the ON DELETE CASCADE rule.

**3. organization Table**

This table holds information about organizations that can post job offers.

* **Columns:**
  + organization\_id: Primary key, unique identifier for each organization.
  + siren: Unique identifier for the organization (e.g., a business registration number).
  + name: Name of the organization.
  + type: Type of organization, with values restricted to 'association', 'eurl', 'sarl', 'sasu', or 'other'.
  + headquarters: Address of the organization's headquarters.
  + added\_by: References admin\_id in the admin table, indicating which admin added this organization.
* **Purpose:** Stores details about each organization and links each organization to an admin who added it.

**4. recruiter Table**

This table holds information about recruiters who are linked to an organization and validated by an admin.

* **Columns:**
  + recruiter\_id: Primary key, also a foreign key referencing user\_id in the users table.
  + validated\_by: References admin\_id in the admin table, indicating which admin validated this recruiter.
  + organization\_id: References organization\_id in the organization table, linking the recruiter to an organization.
* **Purpose:** Defines recruiters in the system and links each recruiter to an organization and an admin who validated them. A recruiter must also be a user, ensuring that they are listed in the users table.

**5. job Table**

This table stores information about different jobs or professions.

* **Columns:**
  + job\_id: Primary key, unique identifier for each job.
  + title: Title or name of the job (e.g., Software Engineer, Data Analyst).
* **Purpose:** Provides a standardized list of jobs or professions that can be referenced by job descriptions.

**6. job\_description Table**

This table describes the specific job positions within an organization.

* **Columns:**
  + description\_id: Primary key, unique identifier for each job description.
  + title: Title of the position (e.g., Senior Developer).
  + level: ENUM indicating the level of the job, with possible values 'executive', 'non-executive', or 'other'.
  + supervisor: Name of the person supervising the position.
  + location: Location where the job is based.
  + hours\_per\_week: Number of working hours per week.
  + remote\_work: Boolean indicating if the job allows remote work.
  + organization\_id: References organization\_id in the organization table, linking this job to an organization.
  + recruiter\_id: References recruiter\_id in the recruiter table, linking this job to a recruiter.
  + job\_id: References job\_id in the job table, indicating the general job title.
* **Purpose:** Provides detailed job descriptions for specific positions within organizations, linking to both the job and recruiter tables.

**7. job\_offer Table**

This table represents job offers that can be published by recruiters.

* **Columns:**
  + offer\_id: Primary key, unique identifier for each job offer.
  + status: Status of the job offer with values 'not published', 'published', or 'expired'.
  + validation\_date: Date when the job offer was validated, defaults to the current date.
  + notes: Additional information about the job offer.
  + document\_count: Number of documents required for the application.
  + description\_id: References description\_id in the job\_description table, linking this offer to a specific job description.
  + description: Text description of the job offer.
* **Purpose:** Stores job offers for positions that recruiters can publish, linking each offer to a specific job description.

**8. application Table**

This table stores applications from users applying for job offers.

* **Columns:**
  + application\_id: Primary key, unique identifier for each application.
  + application\_date: Date of application, defaulting to the current date.
  + user\_id: References user\_id in the users table, linking the application to a candidate.
  + offer\_id: References offer\_id in the job\_offer table, linking the application to a specific job offer.
* **Purpose:** Represents applications submitted by users (candidates) for job offers.

**9. document Table**

This table stores documents uploaded by users (e.g., resumes, cover letters).

* **Columns:**
  + document\_id: Primary key, unique identifier for each document.
  + file: Stores the binary data of the document file.
* **Purpose:** Holds files or documents uploaded by users as part of their job application.

**10. folder Table**

This table links applications to documents, representing a collection of documents submitted with each application.

* **Columns:**
  + application\_id: References application\_id in the application table.
  + document\_id: References document\_id in the document table.
  + **Primary Key:** Composite key (application\_id, document\_id) to ensure uniqueness.
* **Purpose:** Manages a many-to-many relationship between applications and documents, allowing multiple documents to be associated with each application.

**11. organization\_request Table**

This table records requests for new organizations to be added to the system.

* **Columns:**
  + request\_id: Primary key, unique identifier for each request.
  + name: Name of the requested organization.
  + type: Type of organization (same ENUM values as in the organization table).
  + headquarters: Address of the headquarters.
  + siren: Unique identifier like a business registration number.
  + recruiter\_id: References recruiter\_id in the recruiter table, linking the request to a recruiter.
* **Purpose:** Allows recruiters to submit requests to add new organizations, which can be approved or managed by admins.

**12. recruiter\_request Table**

This table records requests from users to become recruiters.

* **Columns:**
  + request\_id: Primary key, unique identifier for each request.
  + organization\_id: References organization\_id in the organization table, linking the request to an organization.
  + user\_id: References user\_id in the users table, linking the request to a user.
* **Purpose:** Allows users to request the recruiter role within a specific organization. This request can then be validated by an admin.

**Summary of Relationships:**

* The **users** table is central, as all roles (candidate, recruiter, admin) are derived from here.
* **recruiter** and **admin** are both extensions of **users**, enforcing specific roles.
* **organization** and **job** tables store information about employers and roles, respectively.
* **job\_description** and **job\_offer** add detail to job roles and availability.
* **application** links candidates with **job\_offer**, and **folder** links applications to documents.
* **organization\_request** and **recruiter\_request** are for managing access and new organization creation, handled by admins.