**Job Recruitment Application**

**1. Topic Details:**

A job recruitment agency wants to expand its business by creating an application where people can apply to jobs assigned by employers.

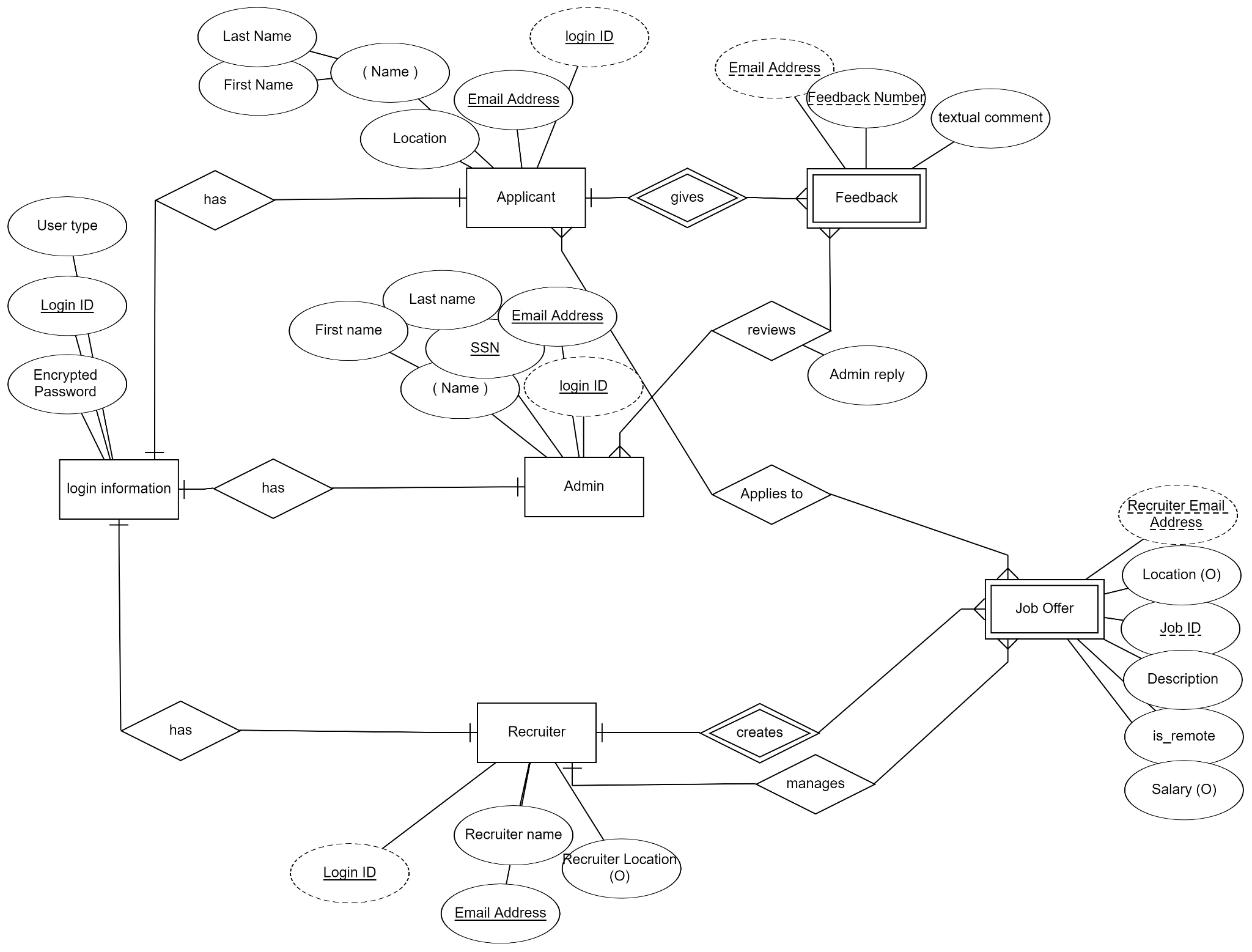
To lay the foundation they designed and implemented a database containing the following information:

* A Login Information entity to keep sensitive user-information in a dedicated and secure place, it contains: **login ID, Encrypted password** and **the type of the user***(for example: Administrator).*
* An Applicant entity. It represents the default user of the application. It contains : **Email address, location** and **Name.**

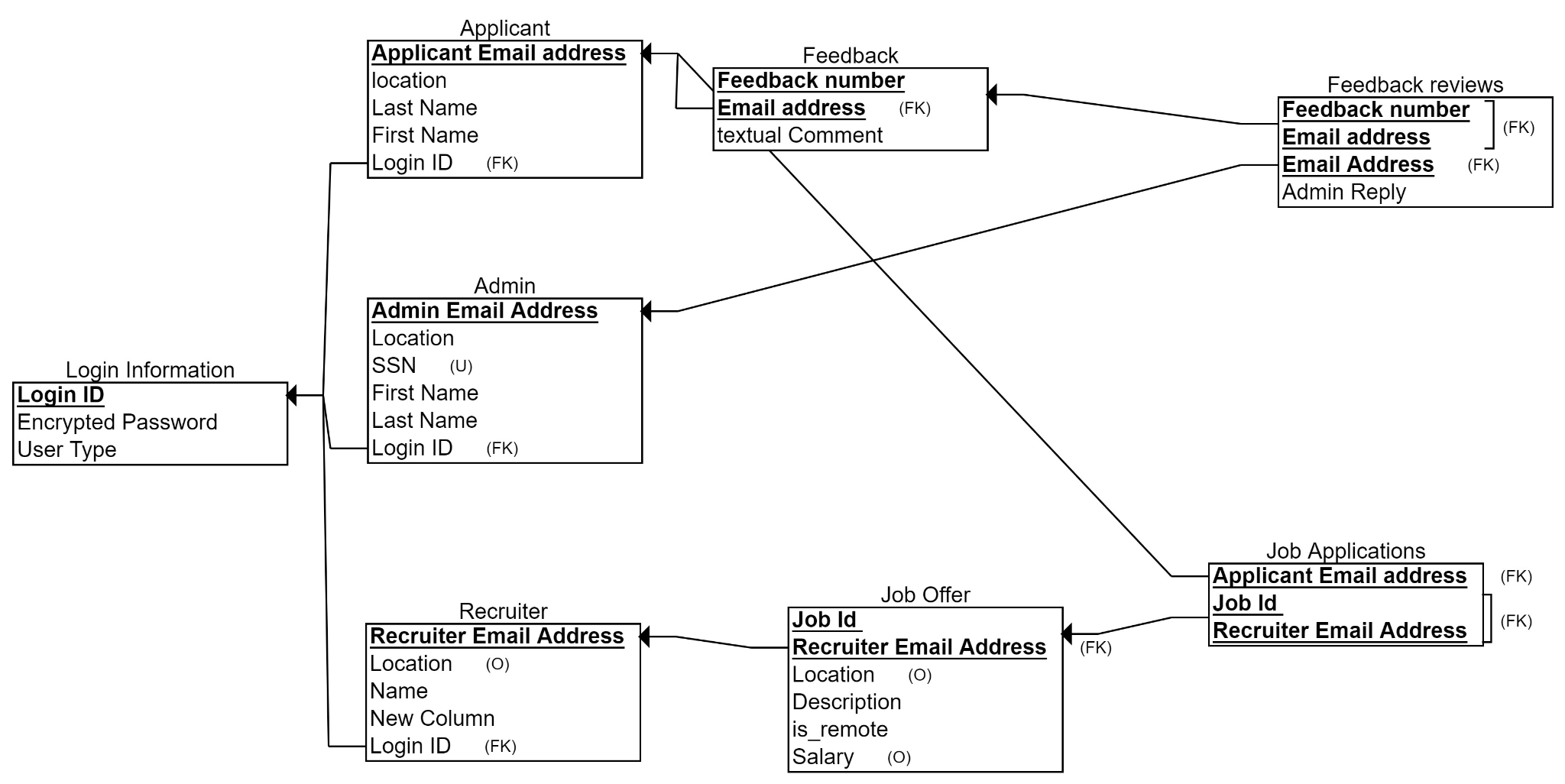
* An Admin entity. It contains: **Email Address, Location, SSN** and **Name.**
* A Recruiter entity. It represents the employers. It contains: **Email Address,Location** and **Name.**
* A Feedback Entity. It makes the user able to report any problem that faced him during his experience. It contains: **feedback number and textual Comment.**
* A Job Offer entity.It contains: **Job ID, Description, location**(optional), **Salary**(optional) and **is\_remote***(can be true or false)****.***
* The Applicant, Admin and Recruiter all **have** login credentials stored in the login information entity.
* The Entity can **give** feedback.
* The admins **Review** the feedbacks so they can solve the problem or report it to the developers to ensure a user-friendly experience.
* The recruiter **creates** and  **manages** job offers *( A job offer can be created and managed by one recruiter . but the recruiter can create and manage many jobs).*
* The applicant **Applies** to as many job offers as he likes.

**2. ER Diagram:**

After forming a clear understanding of the topic ,the next step was to design an ERD:

**3. Relational schema** 

Then the diagram was translated into a relational schema:

****

**4.SQL**

After designing the schema, SQL queries were made in order to create the database and fill it with data. Then Queries were conducted to address the following questions :

Q1: Retrieve the email addresses of applicants who have not received any feedback.

Q2: Find the recruiters who have made job offers with salaries greater than the average salary offered by all recruiters.

Q3: Count the number of applications for each job offer.

Q4: Find the top 3 locations with the highest number of applications.

Q5: Retrieve the applicants who have applied for positions in locations different from their current location.

Q6: Retrieve the top 4 most common job locations.

Q7: List the job offers along with their descriptions and salaries, ordered by salary in descending order.

Q8: Find the administrators who have reviewed feedback for the same applicant more than once.

Q9: Retrieve recruiters along with the total number of job offers they have made and the average salary of those offers. Display only recruiters who have made more than 3 job offers.

Q10: Retrieve the email addresses and names of administrators in alphabetical order.

**5.Normalization**

Lastly we need to normalize the database to make it more efficient, reduce redundancy and to prevent various kinds of errors.

* 5.1 First normal form(**1NF**):

After checking the following guidelines:

– Each column must contain only a single value

– Repeating groups of records (redundancy) must

be eliminated.

– There must not be a composite and a multi-valued

attributes.

It turned out that the database was already 1NF.

* 5.2 Second normal form(**2NF**):

Revisions were made to the table **job offer** to become like this :

| Job id |
| --- |
| Recruiter email address   | Job id | | --- | | Location | | description | | is\_remote | | Salary | |

The attributes only depend only job id so the table was paraed into two, one containing the job id and email to be able to identify the job correctly and another containing job id with other dependant attributes

* 5.3 Third normal form(**3NF**):

All attributes depended on the primary key meaning there was no transitive dependencies and the database was 3NF.

However a special case of 3nf was found in the database called bcnf. In the login table login id depended on User type so the table was seperated to the following:

| Login ID |
| --- |
| Encrypted password |

| Login id |
| --- |
| User type |