**Assignment – 4**

**Normalization**

**Initial Jobs Database Schema:**

The ER diagram is as shown below.

Diagram

Description automatically generated

**Fig 1: Initial Schema**

Below is the sample screenshot of how the data inserted into the above tables looks like

**Schema and table:**

conn = sqlite3.connect('JobsFinalDatabase.db')

print("Opened database successfully");

conn.execute('''DROP TABLE IF EXISTS job\_basic\_details;''')

conn.execute('''CREATE TABLE IF NOT EXISTS job\_basic\_details

(

job\_title TEXT,

company\_name TEXT,

location TEXT,

salary TEXT,

job\_type TEXT,

job\_post\_date datetime,

job\_rating Float,

job\_link TEXT,

job\_desc TEXT,

company\_website TEXT

);''')

print("job\_basic\_details Table created successfully");

conn.close()

Graphical user interface

Description automatically generatedGraphical user interface, application, Teams

Description automatically generated

**Fig 2: Initial Schema Table 1**

conn = sqlite3.connect('JobsFinalDatabase.db')

print("Opened database successfully");

conn.execute('''DROP TABLE IF EXISTS user\_basic\_details;''')

conn.execute('''CREATE TABLE IF NOT EXISTS user\_basic\_details

(

user\_name TEXT ,

dob datetime,

user\_location TEXT,

mobile\_number TEXT,

email\_id TEXT,

portfolio\_link TEXT,

skillset TEXT,

saved\_application\_id TEXT,

certifications TEXT

);''')

print("user\_basic\_details Table created successfully");

conn.close()

****

**Fig 3: Initial Schema Table 2**

conn = sqlite3.connect('JobsFinalDatabase.db')

print("Opened database successfully");

conn.execute('''DROP TABLE IF EXISTS user\_login\_basic\_details;''')

conn.execute('''CREATE TABLE IF NOT EXISTS user\_login\_basic\_details

(

user\_platform\_name TEXT ,

user\_password TEXT,

user\_role TEXT

);''')

print("user\_login\_basic\_details Table created successfully");

conn.close()

Table

Description automatically generated

**Fig 4: Initial Schema Table 3**

Here the data is not normalized and has high chances of losing data integrity and potential duplication of data. Hence, normalizing the existing schema using the First Normal Form(1NF).

**First Normal Form (1NF):**

Every column in the table must be unique. Separate tables must be created for each set of related data. Each table must be identified with a unique column or concatenated columns called the primary key.

The existing data contains the location column that can be further reduced into city, state, and zip code. The salary column contains the min and max range of estimated pay for a particular job role. This can be further reduced to min salary and max salary.

Similarly, the location column in the user\_details table can also be broken down into city, state and zip code.

user\_details table has multiple skill sets, certifications, saved applications. This can be further reduced using the foreign keys as shown below.

Diagram, table

Description automatically generated

**Fig 5: 1NF Schema**

**Schema and table:**

conn = sqlite3.connect('JobsFinalDatabase.db')

print("Opened database successfully");

conn.execute('''DROP TABLE IF EXISTS job\_details;''')

conn.execute('''CREATE TABLE IF NOT EXISTS job\_details

(

job\_id INTEGER PRIMARY KEY,

job\_title TEXT,

company\_name TEXT,

job\_type TEXT,

job\_post\_date datetime,

job\_rating Float,

job\_link TEXT,

job\_desc TEXT,

company\_website TEXT

);''')

print("job\_details Table created successfully");

conn.close()

Table

Description automatically generated with low confidence Graphical user interface, application, Teams

Description automatically generated

**Fig 6: 1NF table 1**

conn = sqlite3.connect('JobsFinalDatabase.db')

print("Opened database successfully");

conn.execute('''DROP TABLE IF EXISTS salary\_details;''')

conn.execute('''CREATE TABLE IF NOT EXISTS salary\_details

(

job\_id INTEGER PRIMARY KEY ,

min\_salary Float,

max\_salary Float,

FOREIGN KEY (job\_id) REFERENCES job\_details(job\_id)

);''')

print("salary\_details Table created successfully");

conn.close()

Table

Description automatically generated

**Fig 7: 1NF Table 2**

conn = sqlite3.connect('JobsFinalDatabase.db')

print("Opened database successfully");

conn.execute('''DROP TABLE IF EXISTS job\_location\_details;''')

conn.execute('''CREATE TABLE IF NOT EXISTS job\_location\_details

(

job\_id INTEGER PRIMARY KEY ,

city TEXT,

state TEXT,

zipcode INTEGER,

FOREIGN KEY (job\_id) REFERENCES job\_details(job\_id)

);''')

print("job\_location\_details Table created successfully");

conn.close()

Table

Description automatically generated

**Fig 8: 1NF table 3**

**Second Normal Form (2NF):**

Second Normal Form (2NF) is based on the concept of full functional dependency. Second Normal Form applies to relations with composite keys, that is, relations with a primary key composed of two or more attributes. A relation with a single-attribute primary key is automatically in at least 2NF.

As we are using single column reference in constructing the primary key, there is no partial dependency on the composition of the primary key but full dependent on the primary key. Hence, there are no changes in the schema from 1NF to 2 NF.

Diagram

Description automatically generated

**Fig 3: 2NF Schema**

**Third Normal Form (3NF):**

A given relation is said to be in its third normal form when it's in 2NF but has no transitive partial dependency. Meaning, when no transitive dependency exists for the attributes that are non-prime, then the relation can be said to be in 3NF.

From the 2NF schema, it can be observed that company website column in the job\_details table is dependent on company\_name which is non candidate key. Hence, as per 3NF, the columns in the job\_details table are reduced to be completely functionally dependent on the primary key and created a new table(company\_details) that has company relevant information.

**Diagram

Description automatically generated**

**Fig 4: 3NF Schema**

**Final Normalized Schema:**

**Diagram

Description automatically generated**

**Fig : Final Schema**

**View for all the Use-cases:**

* **Maheswara Sai Ram Palakurthy:**

1. Create view analyst\_jobs as

Select j.job\_title, j.job\_link, l.city || ', ' || l.state || ', ' || l.zipcode as location, s.min\_salary as salary, j.job\_post\_date, j.job\_desc, j.job\_rating, j.job\_type from job\_details j inner join salary\_details s on j.job\_id = s.job\_id inner join job\_location\_details l on j.job\_id = l.job\_id where j.job\_title LIKE "%Analyst%";

Select \* from analyst\_jobs;

**Use Case**: Displaying the details of the job profiles for Analyst Roles.

**Description**: User can view the job profile details for the Analyst Roles.

**Actor**: User

**Precondition**: Jobs descriptions must contain details about the job profiles.

**Steps**:

**Actor action**: View job profile details for the Analyst Roles.

**System Responses**: The list of Analyst Roles Jobs are displayed to User.

Alternate Path: There are no jobs postings.

Error: No history of jobs available.

1. Create view jobs\_rating as

select c.company\_name, avg(j.job\_rating) from company\_details c inner join job\_details j on c.company\_name = j.company\_name group by c.company\_name having avg(j.job\_rating) > 2.0 and avg(j.job\_rating) < 4.0;

Select \* from jobs\_rating;

**Use Case**: Display the company names with average rating greater than 2 and less than 4.

**Description**: User can view the company names with average rating greater than 2 and less than 4.

**Actor**: User

**Precondition**: Companies should have company rating.

**Steps**:

**Actor action**: User can view the company names with average rating greater than 2 and less than 4.

**System Responses**: the company names with average rating greater than 2 and less than 4 are displayed.

Alternate Path: No company names are displayed.

Error: Job ratings data not available.

1. Create view job\_location as

select c.company\_name, c.company\_website, l.city || ', ' || l.state || ', ' || l.zipcode as location from company\_details c inner join job\_details j on c.company\_name = j.company\_name inner join job\_location\_details l on j.job\_id = l.job\_id where l.state LIKE "%MA%";

Select \* from job\_location;

**Use Case**: List the companies that offer employment in MA state.

**Description**: User gets the list of companies that offer employment in MA state.

**Actor**: User

**Precondition**: Companies must be present in the given state.

**Steps**:

**Actor action**: View the companies that offer employment in MA state.

**System Responses**: Gives the list of companies that offer employment in MA state.

Alternate Path: No companies are present in the given state.

Error: No jobs available now.

1. Create view software\_jobs as

select j.job\_title, j.job\_link, l.city || ', ' || l.state || ', ' || l.zipcode as location, s.min\_salary as salary, j.job\_rating, j.job\_type from job\_details j inner join salary\_details s on j.job\_id = s.job\_id inner join job\_location\_details l on j.job\_id = l.job\_id where j.job\_desc LIKE "%Software%" ORDER BY j.job\_rating desc limit 5;

Select \* from software\_jobs;

**Use Case**: Display the list of top 5 highest rated job profiles in the domain of Software Engineering.

**Description**: User views the top 5 highest rated jobs in the domain - Software Engineering. **Actor**: User

**Precondition**: Database must have the data for the jobs in the domain - Software Engineering.

**Steps**:

**Actor action**: User views the top 5 highest rated jobs.

**System Responses**: The list of top 5 highest rated job profiles in the domain of Software Engineering.

Alternate Path: There are no jobs under the domain - Software Engineering.

Error: No jobs available now.

1. Create view intern\_jobs as

Select j.job\_title, j.job\_link, j.job\_type, s.min\_salary as salary from job\_details j inner join salary\_details s on j.job\_id = s.job\_id where j.job\_type LIKE "%Full-time%" and j.job\_title LIKE "%Intern%";

Select \* from intern\_jobs;

**Use Case**: List the jobs that offer Full-time opportunities for Interns.

**Description**: User views the jobs that offer Full-time opportunities for Interns.

**Actor**: User

**Precondition**: Database must have the data for the above jobs.

**Steps**:

**Actor action**: User views all the jobs that offer Full-time opportunities for Interns.

**System Responses**: The list jobs that offer Full-time opportunities for Interns.

Alternate Path: There are no jobs available.

Error: No history of jobs available.

* **Namitha J C:**

1. Create view fullstack\_jobs as

Select c.company\_name from company\_details c inner join job\_details j on c.company\_name = j.company\_name where j.job\_title LIKE "%Full Stack%" group by c.company\_name having max(j.job\_rating) > 4;

Select \* from fullstack\_jobs;

**Use Case**: Displaying the list of companies with excellent(>4.5) employee rating in the job role of Full Stack Developer.

**Description**: User can view the list of companies with excellent employee rating in the job role of Full Stack Developer.

**Actor**: User

**Precondition**: Companies must have excellent employee rating in the given job role.

**Steps**:

**Actor action**: view the list of companies with excellent employee rating in the job role of Full Stack Developer.

**System Responses**: Display the list of companies with excellent employee rating in the job role of Full Stack Developer.

Alternate Path: There are no employee rating for the given job role.

Error: No data available.

1. Create view remote\_jobs as

Select j.job\_title, c.company\_name, j.job\_link, l.city || ', ' || l.state || ', ' || l.zipcode as location, s.min\_salary as salary from job\_details j left join company\_details c on c.company\_name = j.company\_name inner join salary\_details s on j.job\_id = s.job\_id inner join job\_location\_details l on j.job\_id = l.job\_id where j.job\_title LIKE "%Remote%" and s.min\_salary >= 70000;

Select \* from remote\_jobs;

**Use Case**: Display the job profiles that offer at least $70000 for Remote jobs.

**Description**: Filter jobs based on salary - at least $70000 for the Remote jobs.

**Actor**: User

**Precondition**: Jobs must have salary of at least $70000 for the given role.

**Steps**:

**Actor action**: View the job profiles that offer at least $70000 for the Remote jobs**.**

**System Responses**: Display all the jobs with salary at least $70000 for the given role.

Alternate Path: No jobs are Displayed.

Error: Data unavailable.

1. Create view jobs\_location as

Select j.company\_name, j.job\_type, l.city || ', ' || l.state || ', ' || l.zipcode as location from job\_details j inner join job\_location\_details l where j.job\_type = "Internship" and l.city LIKE "%Boston%";

Select \* from jobs\_location;

**Use Case**: List the company urls that offer Internships in location Boston.

**Description**: Displays the company urls for the given conditions.

**Actor**: User

**Precondition**: Company urls must be present.

**Steps**:

**Actor action**: User can view the list of company urls that offer Internships in location Boston.

**System Responses**: Display the list of company urls for the given conditions.

Alternate Path: No urls are displayed.

Error: Data currently unavailable.

1. Create view developer\_jobs as

Select j.job\_title, AVG(j.job\_rating), c.company\_name from job\_details j inner join company\_details c on j.company\_name = c.company\_name where j.job\_title LIKE "%Developer%" group by j.job\_title, c.company\_name;

Select \* from developer\_jobs;

**Use Case**: List the mean(rating) of the jobs available for the role of Developer.

**Description**: User can view the mean rating of the jobs available for the role of Developer.

**Actor**: User

**Precondition**: Jobs postings must contain ratings.

**Steps**:

**Actor action**: View listed jobs.

**System Responses**: The list of mean(rating) of the jobs available for the role of Developer.

Alternate Path: There are no ratings for the given jobs.

Error: No history of jobs available.

1. Create view software\_engineer\_jobs as

Select l.city || ', ' || l.state || ', ' || l.zipcode as Location from job\_location\_details l inner join job\_details j on l.job\_id = j.job\_id where j.job\_type = "Internship" and j.job\_title LIKE "%Software Engineer%";

Select \* from software\_engineer\_jobs;

**Use Case**: List all the locations where Internship opportunities are offered in the role of Software Engineer.

**Description**: User can view the list of locations for the given conditions.

**Actor**: User

**Precondition**: Internship opportunities must be available at the given locations.

**Steps**:

**Actor action**: User can view the list of locations for the given conditions.

**System Responses**: Lists all the locations where Internship opportunities are offered in the role of Software Engineer.

Alternate Path: Internship opportunities data not available at the given locations.

Error: Data not available.

* **Sinchana Kumara:**

1. Create view finance\_jobs as

Select j.job\_link as URL from job\_details j where job\_title LIKE "%Java%" UNION Select c.company\_website as URL from company\_details c where c.company\_name LIKE "%Cigna%";

Select \* from finance\_jobs;

**Use Case**: List the Job URLs for the job role Java and the job URLs available from a company Cigna.

**Description**: User views the job urls for the given conditions.

**Actor**: User

**Precondition**: Jobs urls must be present in the database.

**Steps**:

**Actor action**: User views all the job urls for the given conditions.

**System Responses**: The list of Job URLs for the job role Java and the job URLs available from a company Cigna.

Alternate Path: There are no job urls present in the database.

Error: No data available.

1. Create view user\_skillset as

Select j.job\_title, j.job\_link, s.min\_salary as salary from job\_details j inner join salary\_details s on j.job\_id = s.job\_id where j.job\_title LIKE "%Co-op%" group by j.job\_title,s.min\_salary, j.job\_link having avg(j.job\_rating) > 3.0;

Select \* from user\_skillset;

**Use Case**: Display the list of jobs having hashtags with user\_platform\_name is “@Hilton”.

**Description**: Display data for above given condition.

**Actor**: User

**Precondition**: “@Hilton” should be present in the database.

**Steps**:

**Actor action**: User can view the list of jobs with the given conditions.

**System Responses**: Display all the jobs with the given conditions.

Alternate Path: No data is displayed.

Error: Data unavailable.

1. Create view jobs\_postings as

Select j.job\_title, j.job\_link, s.min\_salary as salary from job\_details j inner join salary\_details s on j.job\_id = s.job\_id where j.job\_post\_date >= DATE("now", "-30 day");

Select \* from jobs\_postings;

**Use Case**: Provide the job listings that was posted 7 days ago.

**Description**: User can view the Job posted 7 days ago.

**Actor**: User

**Precondition**: Jobs postings must contain posted date.

**Steps**:

**Actor action**: View the Job posted 7 days ago.

**System Responses**: Display the list of jobs posted 7 days ago.

Alternate Path: No Jobs are available.

Error: No history of jobs available.

1. Create view job\_roles as

Select j.job\_title, l.city || ', ' || l.state || ', ' || l.zipcode as location, j.job\_desc, j.job\_type from job\_details j inner join job\_location\_details l on j.job\_id = l.job\_id where j.job\_type = "Contract";

Select \* from job\_roles;

**Use Case**: Display the job description of the roles that offer Contract opportunities.

**Description**: User can view the job descriptions with the given conditions.

**Actor**: User

**Precondition**: Job descriptions must be present in the database.

**Steps**:

**Actor action**: User can view the job descriptions with the given conditions.

**System Responses**: Display the job description of the roles that offer Contract opportunities.

Alternate Path: No data available.

Error: Job description data not available.

1. Create view software\_jobs as

Select j.job\_title, j.job\_link, c.company\_website, j.job\_type, s.min\_salary as salary, l.city || ', ' || l.state || ', ' || l.zipcode as location from job\_details j inner join company\_details c on j.company\_name = c.company\_name inner join salary\_details s on j.job\_id = s.job\_id inner join job\_location\_details l on j.job\_id = l.job\_id where l.state LIKE "%MA%" and j.job\_title LIKE "%Software%";

Select \* from software\_jobs;

**Use Case**: List the job urls, company urls, job\_type from a role Software and location is MA.

**Description**: User will be able to view the job urls, company urls, job\_type for the given role and location.

**Actor**: User

**Precondition**: Urls must be present in the given database.

**Steps**:

**Actor action**: User views all the job urls, company urls, job\_type for the given role and location.

**System Responses**: Lists the job urls, company urls, job\_type from a role Software and location is MA.

Alternate Path: There are no job urls present.

Error: No history of data available.