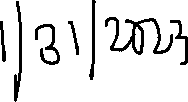
Project Report

|  |  |
| --- | --- |
| Product Name | Advanced Certificate in Web Development |
| Qualification Name (NICF) | NICF-Advanced Certificate in Infocomm Technology (Software & Applications) |
| Product Name | Database Design and Implementation |
| Module Name (NICF) | ITSF-Database Design and Implementation |

|  |  |  |  |
| --- | --- | --- | --- |
| Student name | | Assessor name | |
| Francis Roel L. Abarca | |  | |
| Date issued | Completion date | | Submitted on |
|  | January 27, 2023 | | January 31, 2023 |
|  | |  | |
| Project title | Design, Implement, Test & Document Community Portal Database. | | |



|  |
| --- |
| Learner declaration |
| I certify that the work submitted for this assignment is my own and research sources are fully acknowledged.  Student signature: Date: |

Content

1. Project background
2. Project Objectives
3. Database Requirement Specifications
4. Task 1
5. Task 2
6. Task 3
7. Task 4
8. Task 5
9. Task 6

Project Background

a. Project Purpose  
- The company, ABC Jobs Pte Ltd, has approached us to create a community portal for software engineers. The portal itself has 2 general users whom are the Software Engineers and Administrators. The Administrators must be able to log-in and register easily to the portal and be able to reset their password just in case. The community portal must allow the administrators to manage bulk mail, search user profiles, view user profiles, update user profiles and delete user profiles. For the Software Engineers, they must be a able to view their own profile, update it, search for other users, view their profiles, create threads, send and reply to messages, post for jobs and be notified for any updates. The developers have already made a graphical user interface for the Community Portal and their next goal is implementing its back-end which starts with the Database creation.

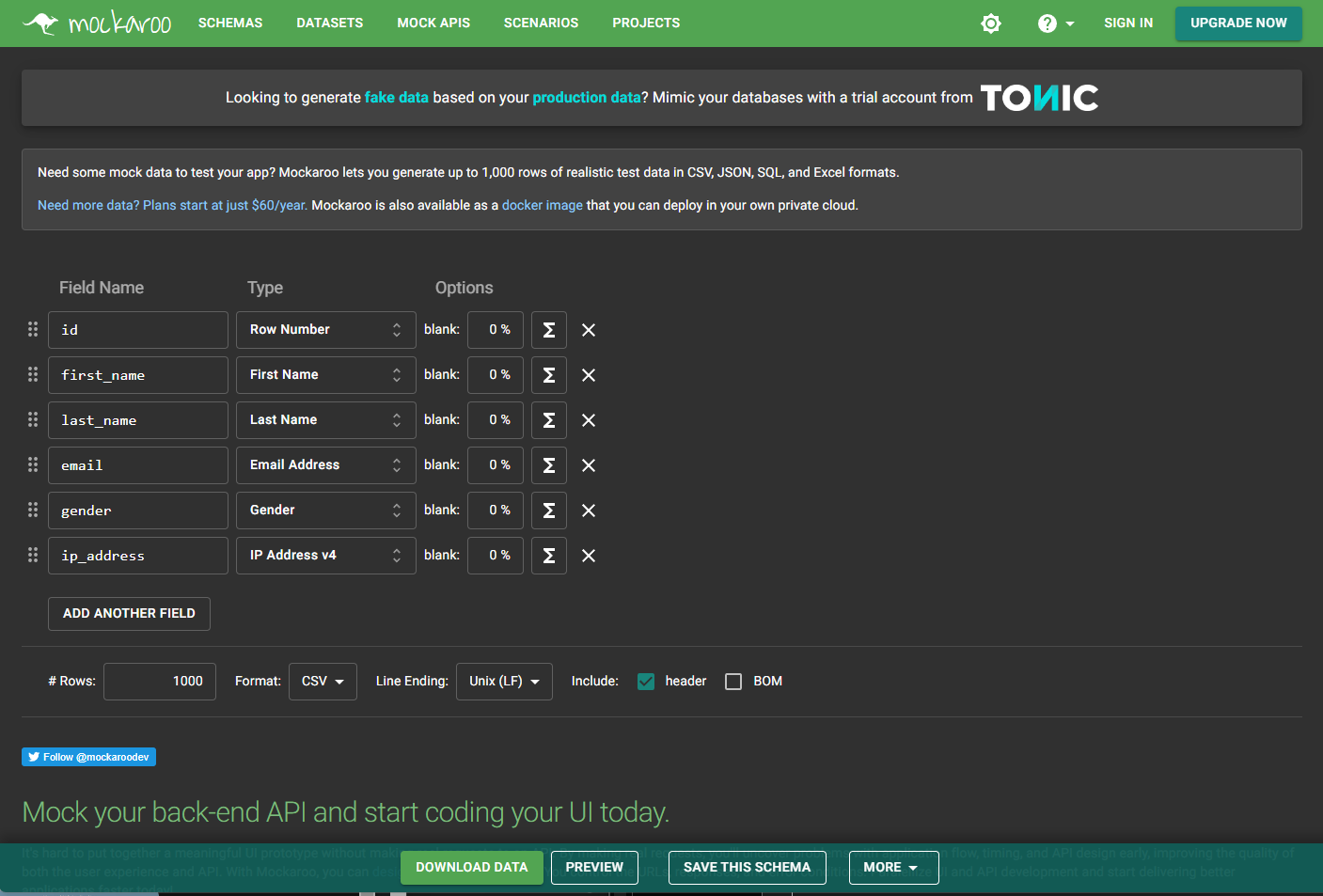
b. Existing System  
- The previous module of our course tasked us to create an interface for the Community Portal and after that, we are ready to create our own database to power the said community portal.

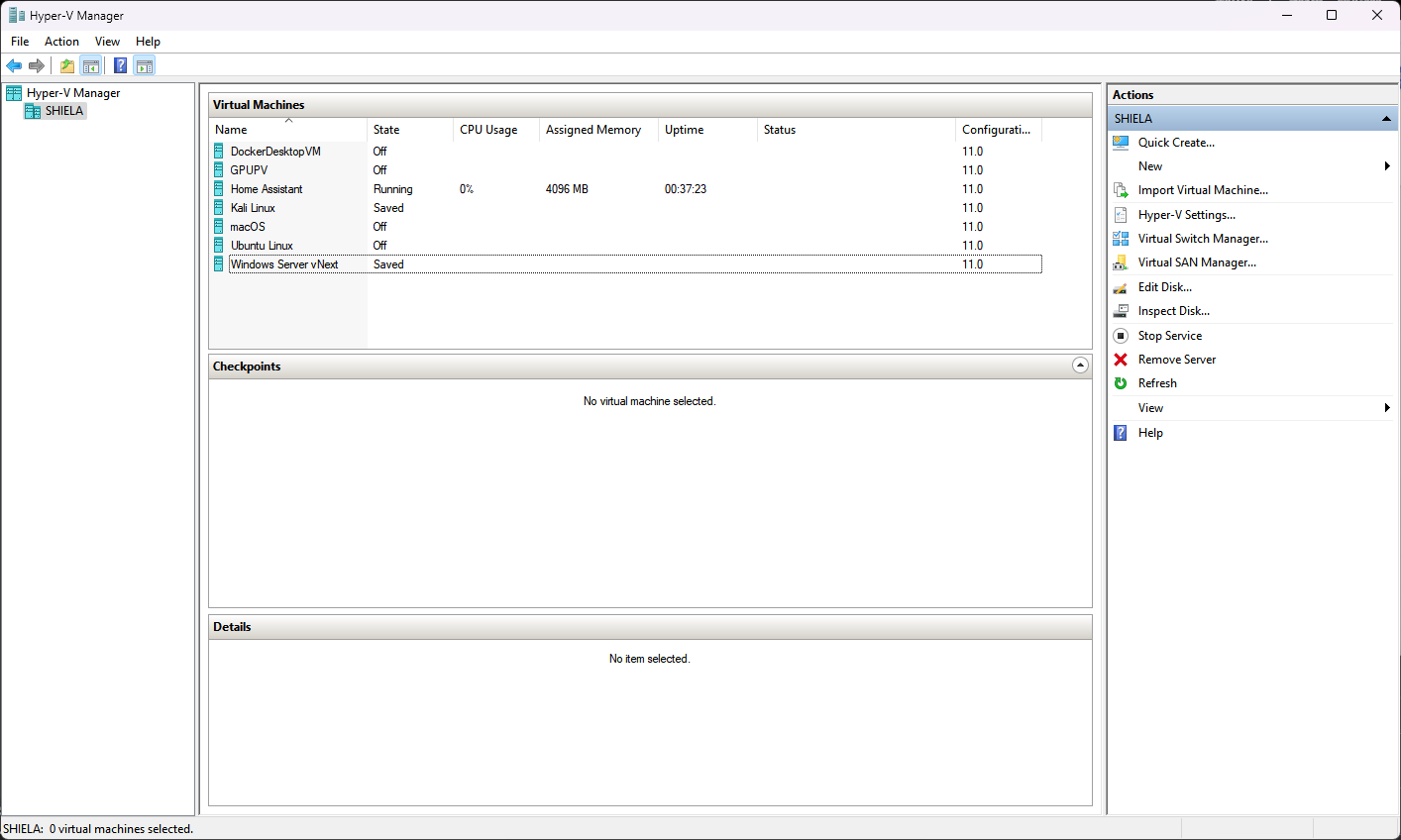
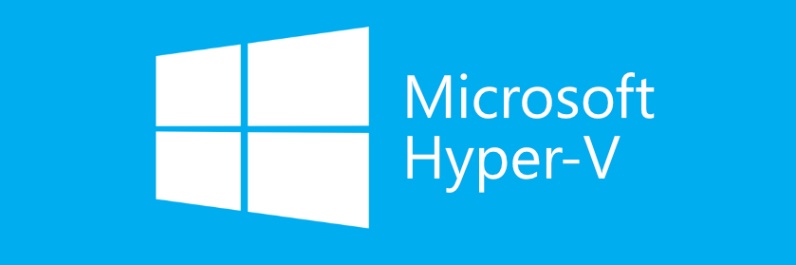
c. Need for a Database System  
- To make the database for this Community Portal, we will use the Relational Database Management System (RDBMS) to let the users perform CRUD onto the database itself. In terms of reliability and safety of the database, we also have a Backup and Restoration tool set up and available for the database itself.

Project Objective

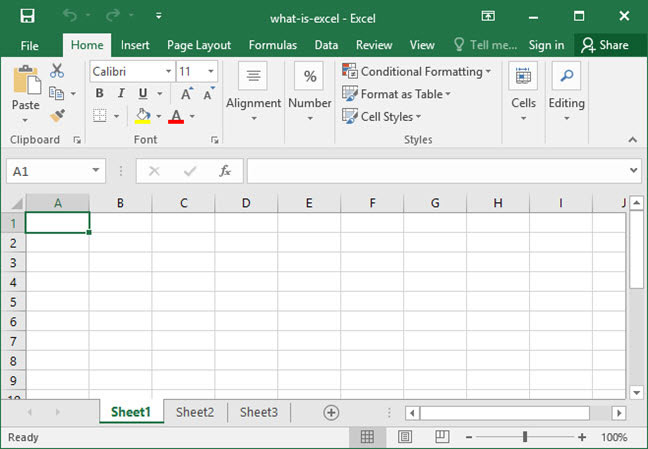
a. Project Goal

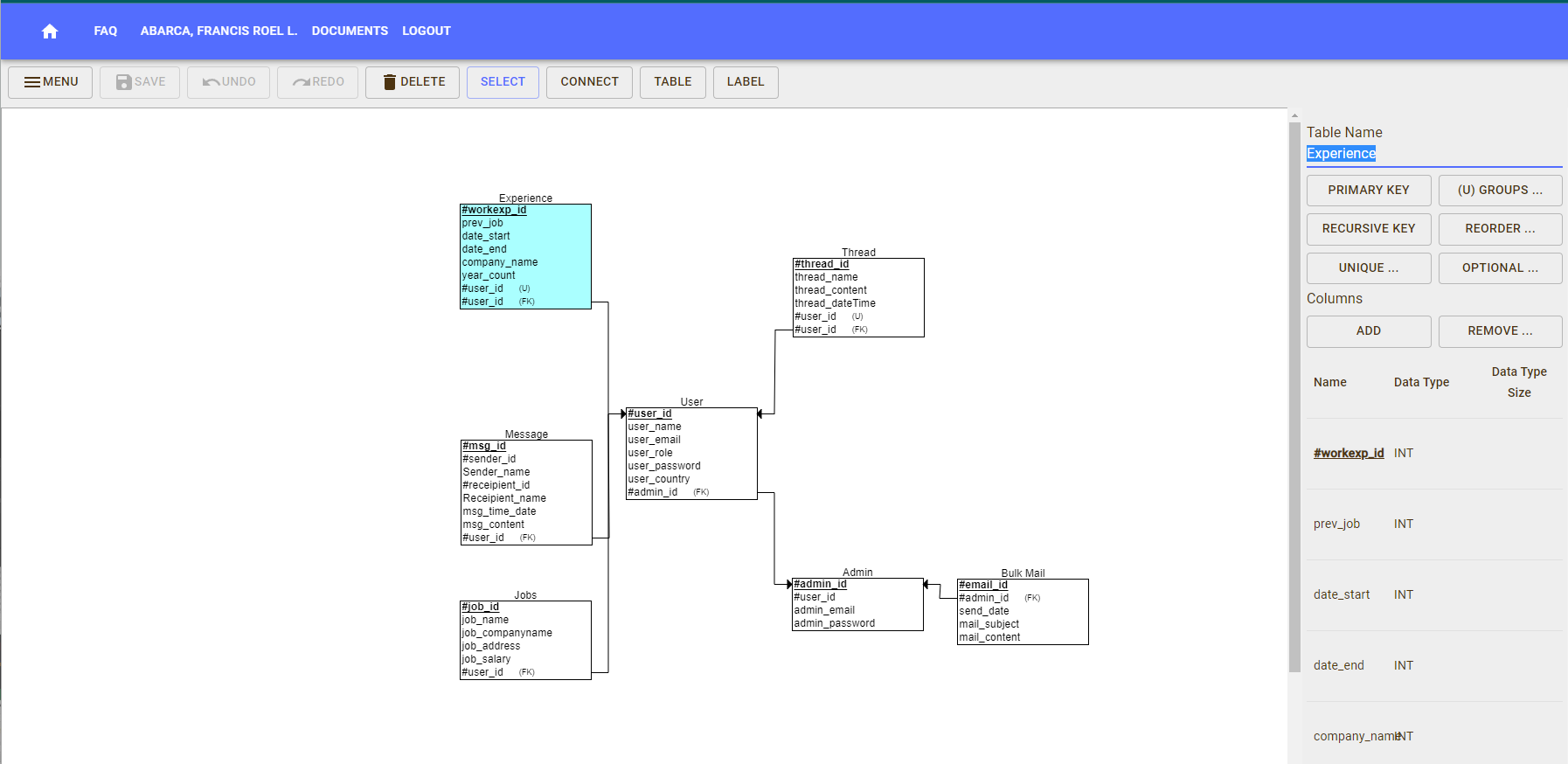
Our goal for this project is to create a Relational Database Management System for ABC Jobs Pte Ltd’s Community Portal website for Software Engineers.

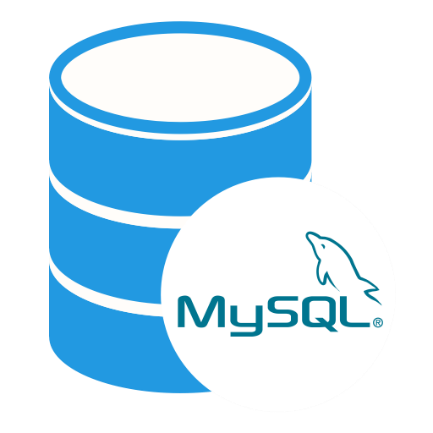
b. Tools & Platform Used:  
1. Mockaroo.com – Create Sample Data quickly  
  


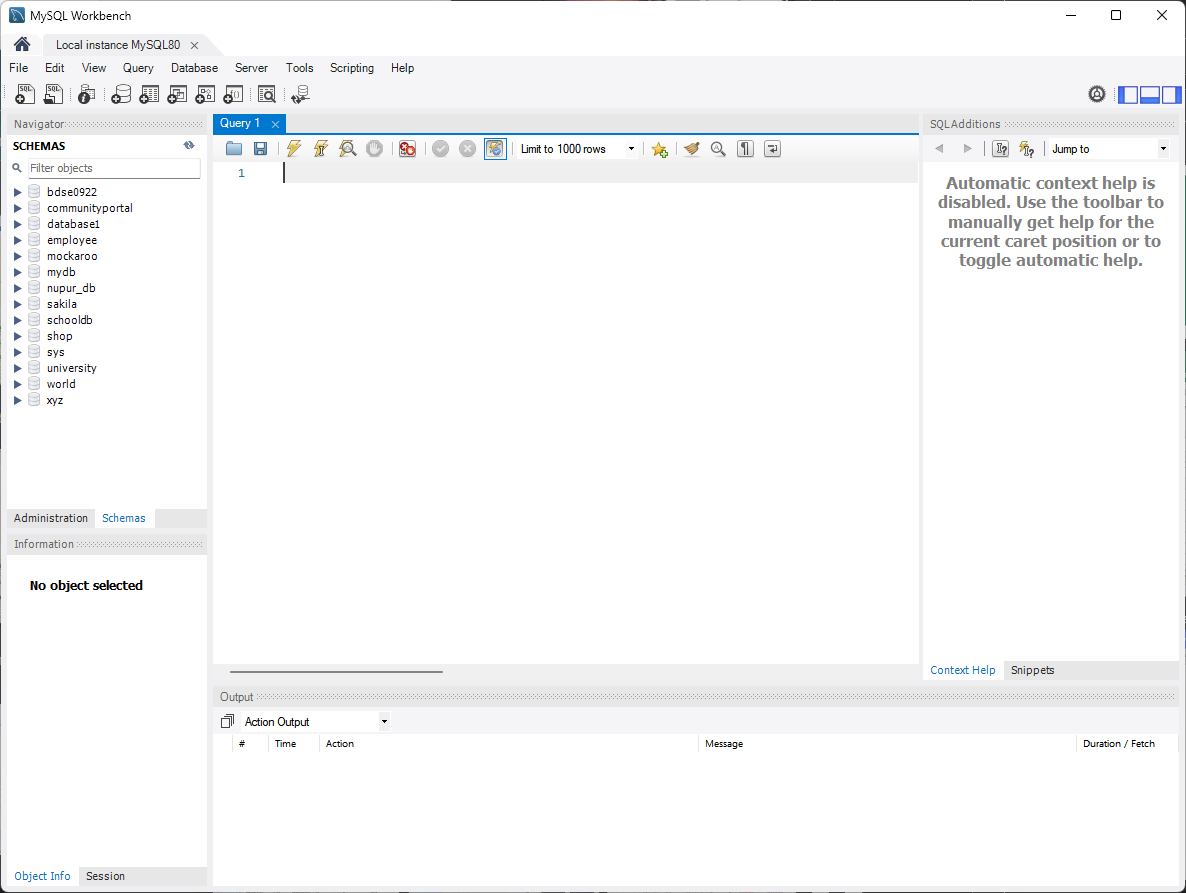
2. Hyper-V – To run the Database Server  


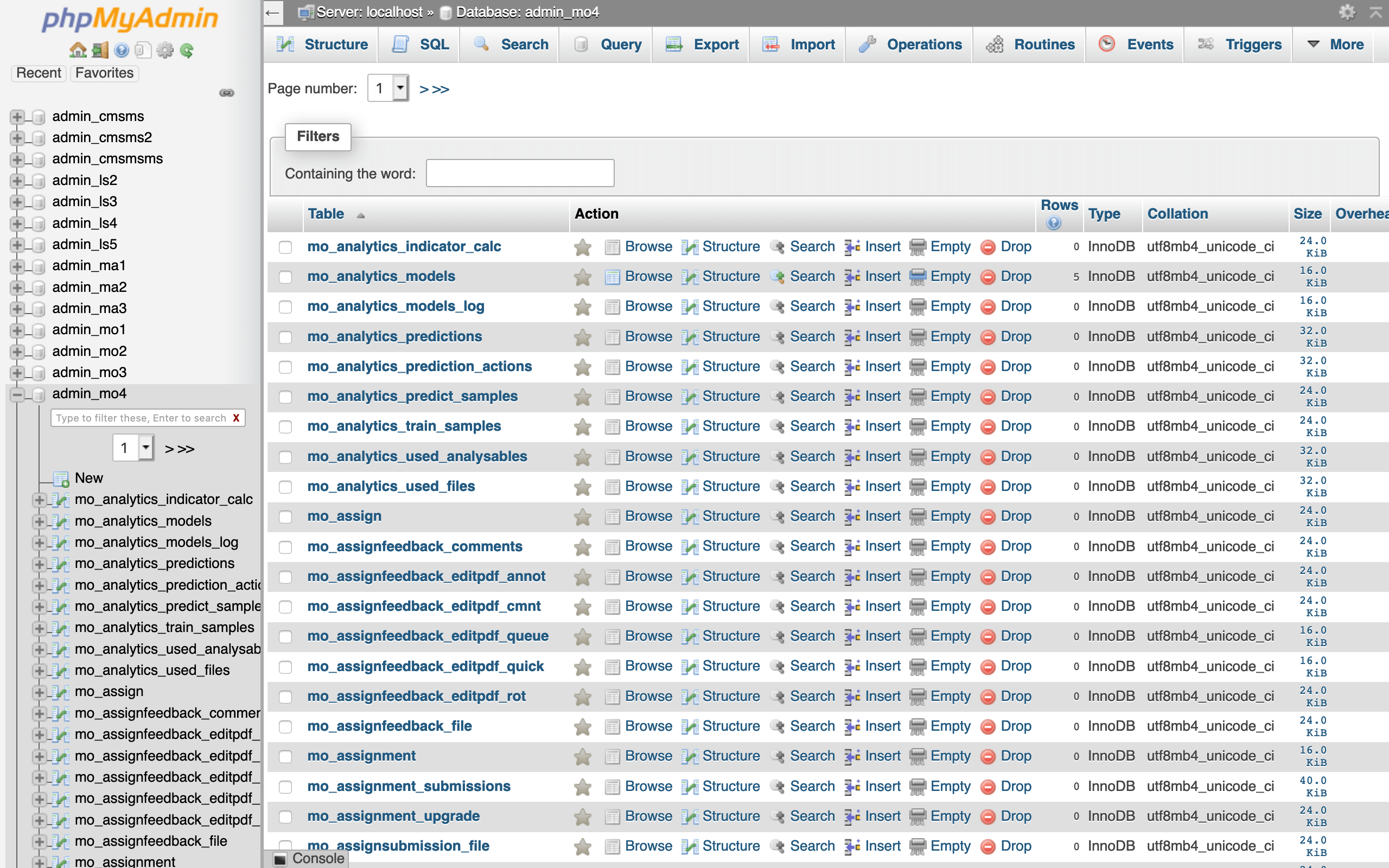
3. Microsoft Excel – Normalize the Data   

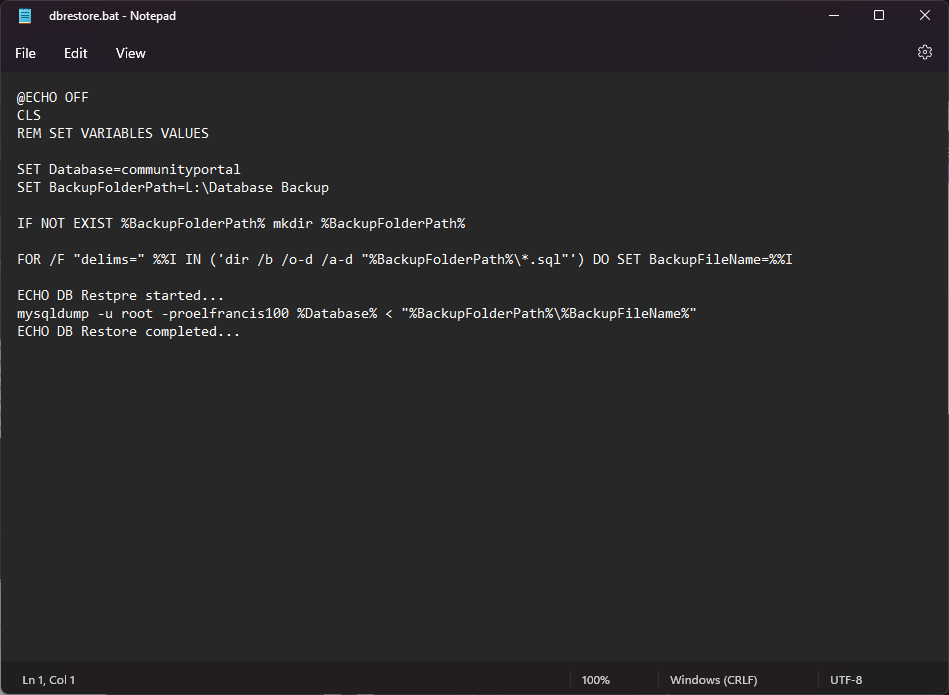



4. ERDPlus – Create the Relational and ER Diagram  


5. MySQL Server 8.0 – Host the Database  


6. MySQL Workbench – Perform CRUD operations to the Database  


7. Docker + phpMyAdmin – Host the phpMyAdmin container for Data Dictionary  
  


8. Microsoft Notepad – Creation of the Restore and Backup Batch Files  


Database Requirements Specification

Task 1

Task Statement:

Create a Database Requirements Specification Document

Solution:

a. Hardware Requirements

- Multi-core CPU (6 cores is preferred)

- 16GB of RAM or higher

- 250GB SSD or higher

- 500GB HDD or higher

b. Software Requirements

- Windows 10 Pro/Windows 11 Pro

- Microsoft Hyper-V

- MySQL Server (Database Server)

- MySQL Workbench (Database Client)

- Docker

- phpMyAdmin

- Google Chrome

- Command Prompt & SSH

- Task Scheduler

- Notepad

c. Database Requirements

|  |  |  |
| --- | --- | --- |
| * **Entity** | * **Description** | * **Page/Function** |
| * Users | * Contains information relevant to the user during log-in sessions and profile viewing. | * Log-in, Registration |
| * Jobs | * Contains information about the Jobs available currently. | * Search, Profile Page |
| * Work Experience | * Contains information about the user’s experience on specific jobs. | * Profile Page |
| * Admin | * Contains the list of people who have administrative privileges to the website. | * Profile Page, Mail Manager |
| * Message | * Contains the messages and conversations they have with the users and companies in the Community Portal. | * Messages |
| * Thread | * Provides users a space to discuss and talk about work related stuff inside the Community Portal. | * Thread Page |
| * Bulk Mail | * Contains the list of bulk emails that are sent to users | * Mail Manager |

d. Entity Relationship

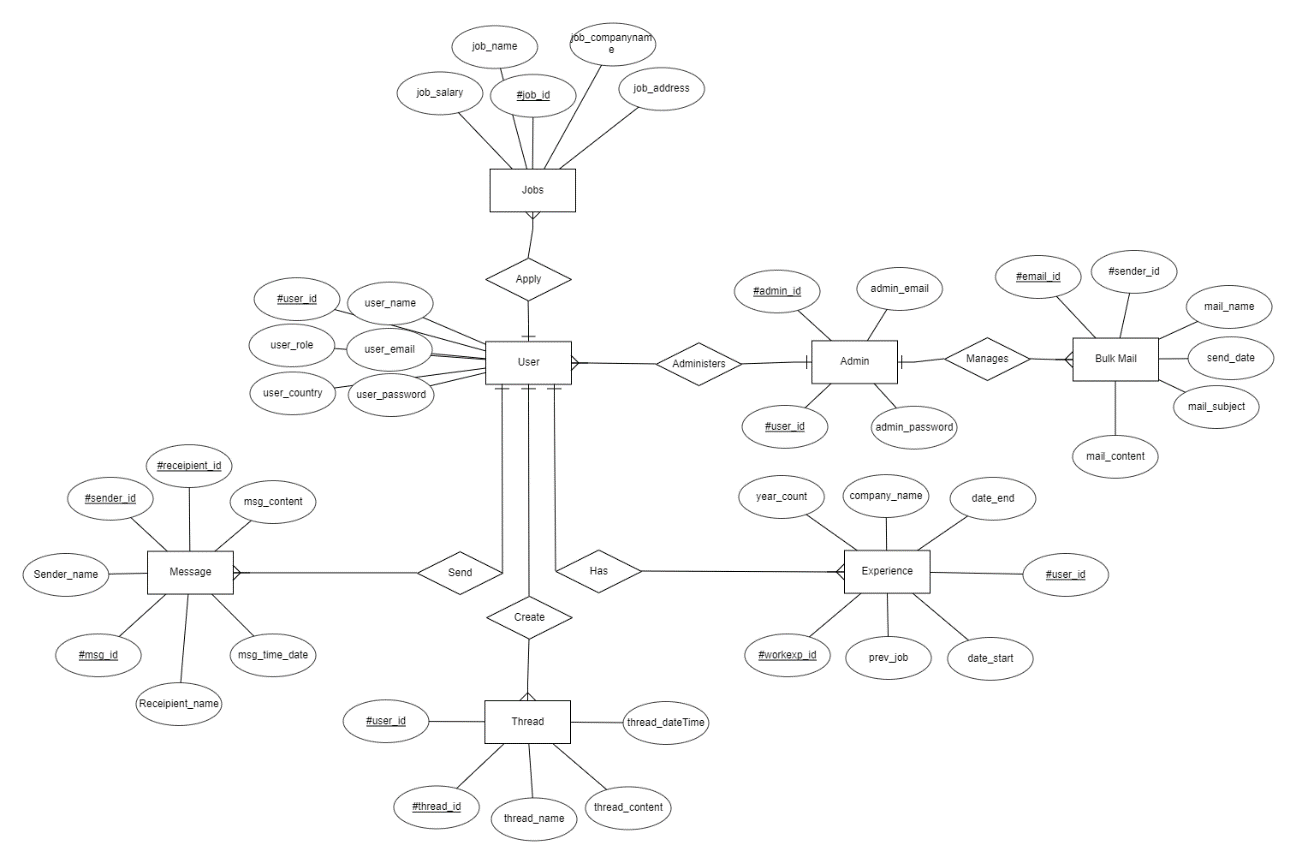
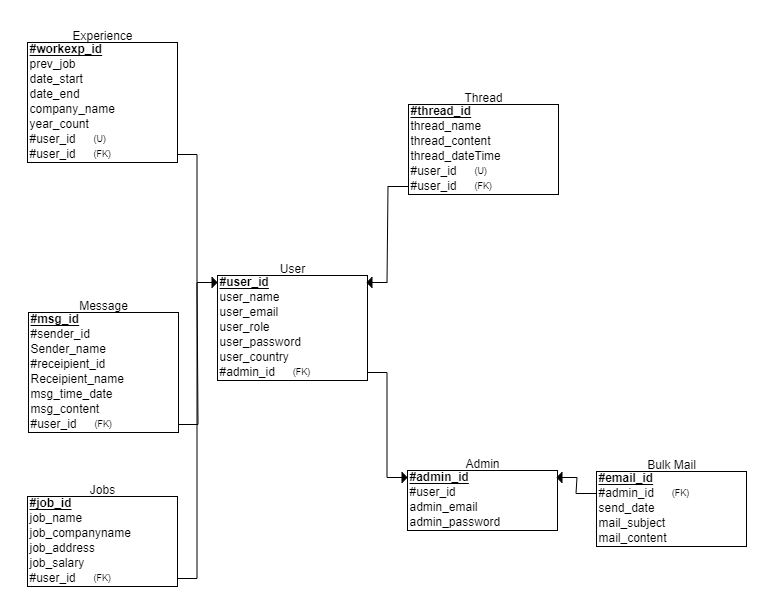
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Emtity One | Entity Two | Relationship | Note/Description |
| 1 | User | Admin | Many to One | Many users can be an admin |
| 2 | User | Jobs | One to Many | One user can apply multiple jobs. |
| 3 | User | Work Experience | One to Many | One user can have many work experiences. |
| 4 | User | Message | One to Many | One User can send and receive messages. |
| 5 | User | Threads | One to Many | One user can create and interact with many threads. |
| 6 | Admin | Bulk Email | Many to Many | Many admins can create multiple bulk mails. |

Task 2

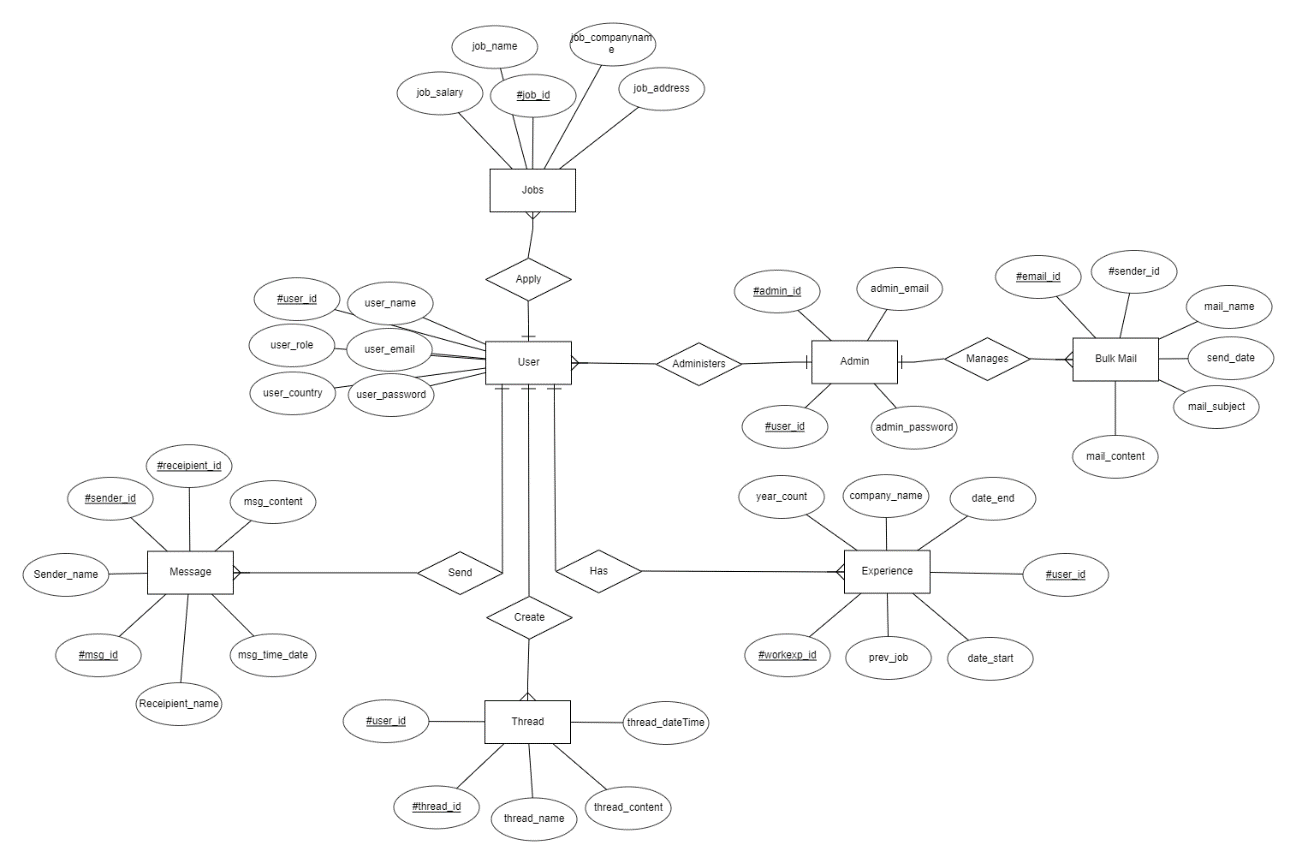
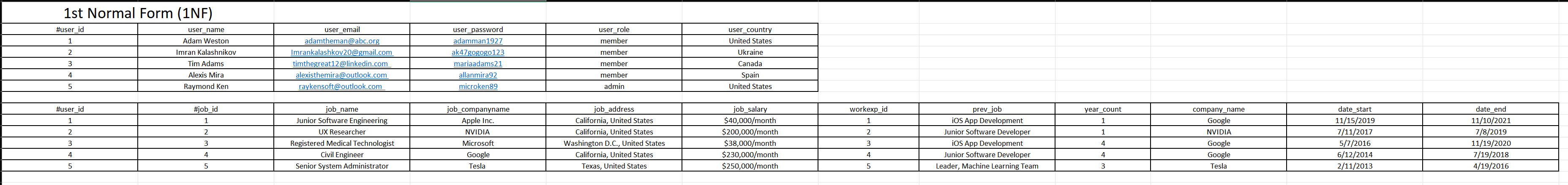
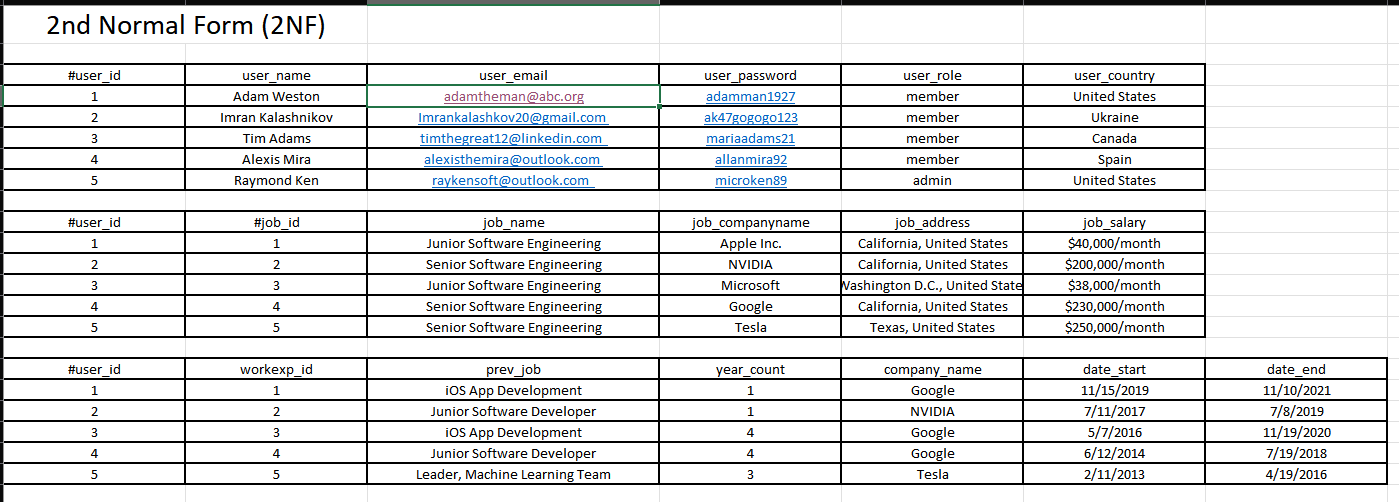
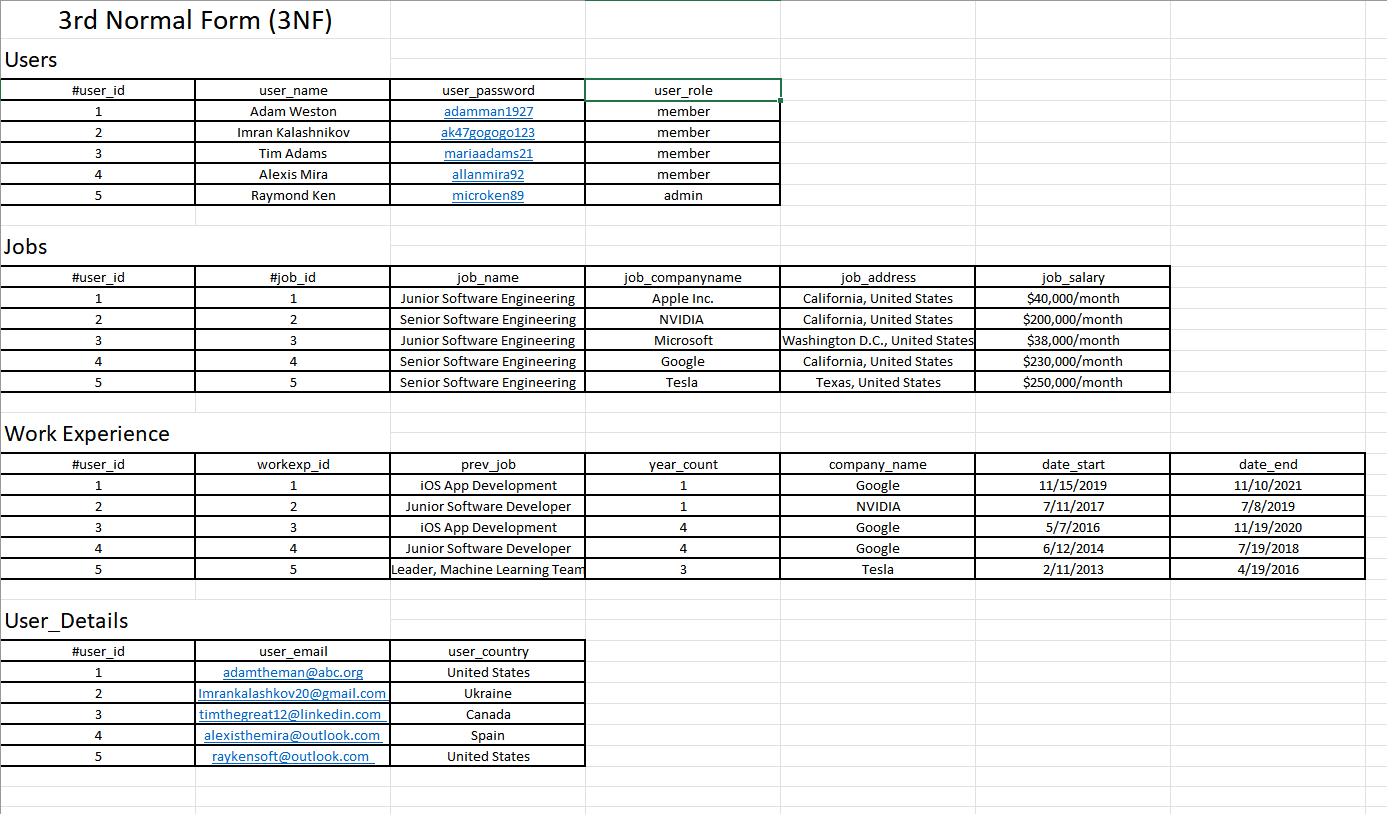
Task Statement:

Create a Database Design Document, Create a Conceptual Design for the proposed database, Create logical design for the proposed database, Create the ER diagram for the project Scenario, Include the above as part of the Project Presentation.

Solution:

1. Explain what is RDBMS  
   A Relational Database Management System (RDBMS) is a type of Database Management system that uses a relational method of managing multiple databases together.
2. Conceptual Design  
   The idea of creating a conceptual design for this project is to create a model based on the criteria provided by ABC Jobs for their Community Portal project. This shows the relationship of the entities, the entities themselves and their attributes.
3. Logical Design  
   The idea of creating a Logical Design is to create an information model based on the data models set for the database.  
   
4. Physical Design  
   The idea of a Physical Design is to show the exact design and label of the data inside the entities themselves and show its arrangement.   
     
   Entities and Attributes for Community Portal

|  |  |  |
| --- | --- | --- |
| * **Entity** | * **Description** | * **Attributes** |
| * Users | * Contains information relevant to the user during log-in sessions and profile viewing. | * **#user\_id** int pk * User\_name varchar(50) * User\_email varchar(20) * User\_role varchar(20) * User\_password varchar(15) * User\_country varchar(20) * #admin\_id int |
| * Jobs | * Contains information about the Jobs available currently. | * **#job\_id** int pk * #user\_id int * Job\_name varchar(50) * Job\_companyname varchar(50) * Job\_address varchar(50) * Job\_salary DOUBLE |
| * Work Experience | * Contains information about the user’s experience on specific jobs. | * **#workexp\_id** int pk * prev\_job varchar(50) * company\_name varchar(50) * date\_start varchar(25) * date\_end varchar(25) * year\_count int * #user\_id int |
| * Admin | * Contains the list of people who have administrative privileges to the website. | * **#admin\_id** int pk * #user\_id int * Admin\_password varchar(15) * Admin\_email varchar(20) |
| * Message | * Contains the messages and conversations they have with the users and companies in the Community Portal. | * **#msg\_id** int pk * #sender\_id int * #receipient\_id int * msg\_time\_date datetime * msg\_content varchar(100) |
| * Thread | * Provides users a space to discuss and talk about work related stuff inside the Community Portal. | * **#thread\_id** int pk * #user\_id int * Thread\_name varchar(50) * Thread\_content varchar(200) * Thread\_dateTime varchar(25) |
| * Bulk Mail | * Contains the list of bulk emails that are sent to users | * **#email\_id** int pk * #admin\_id int * Send\_date varchar(25) * Mail\_subject varchar(25) * Mail\_content varchar(200) |

1. ER Diagram  
     
     
     
     
   Normalization  
   1. 1NF  
     
   2. 2NF  
     
   3. 3NF  
     
   In 3NF, my database has been separated into multiple entities where the user details table is created and most of the tables that rely on user have a foreign key in them which connects to the #user\_id column.

Task 3

Task Statement:

Create the Database Dictionary with tables, fields and datatypes  
Include the above part as part of Project Report.

Solution:

Admin

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column | Type | Null | Default | Comments |
| #admin\_id (Primary) | Int | No |  |  |
| #user\_id | Int | No |  |  |
| Admin\_password | Varchar(15) | No |  |  |
| Admin\_email | Varchar(50) | Yes | NULL |  |

Indexes

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Keyname | Type | Unique | Packed | Column | Cardinality | Collation | Null | Comment |
| PRIMARy | BTREE | Yes | No | #admin\_id | 1 | A | No |  |

Bulk Mail

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column | Type | Null | Default | Comments |
| #email\_id (Primary) | Int | No |  |  |
| #admin\_id | Int | No |  |  |
| Send\_date | Varchar(25) | No |  |  |
| Mail\_subject | Varchar(25) | No |  |  |
| Mail\_content | Varchar(200) | No |  |  |

Indexes

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Keyname | Type | Unique | Packed | Column | Cardinality | Collation | Null | Comment |
| PRIMARy | BTREE | Yes | No | #emai\_id | 5 | A | No |  |
| #admin\_id | BTREE | No | No | #admin\_id | 1 | A | No |  |

Experience

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column | Type | Null | Default | Comments |
| #workexp\_id (Primary) | Int | No |  |  |
| Prev\_job | Varchar(50) | No |  |  |
| Company\_name | Varchar(50) | No |  |  |
| Date\_start | Varchar(25) | No |  |  |
| Date\_end | Varchar(25) | No |  |  |
| Year\_count | int | No |  |  |
| #user\_id | Int | No |  |  |

Indexes

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Keyname | Type | Unique | Packed | Column | Cardinality | Collation | Null | Comment |
| PRIMARy | BTREE | Yes | No | #workexp\_id | 5 | A | No |  |
| #user\_id | BTREE | No | No | #user\_id | 5 | A | No |  |

Jobs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column | Type | Null | Default | Comments |
| #job\_id (Primary) | Int | No |  |  |
| #user\_id | Int | No |  |  |
| job\_name | Varchar(50) | No |  |  |
| Job\_companyname | Varchar(50) | No |  |  |
| Job\_address | Varchar(50) | No |  |  |
| Job\_salary | double | No |  |  |

Indexes

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Keyname | Type | Unique | Packed | Column | Cardinality | Collation | Null | Comment |
| PRIMARy | BTREE | Yes | No | #job\_id | 5 | A | No |  |
| #user\_id | BTREE | No | No | #user\_id | 5 | A | No |  |

Message

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column | Type | Null | Default | Comments |
| #msg\_id (Primary) | Int | No |  |  |
| #sender\_id | Int | No |  |  |
| #receipient\_id | Int | No |  |  |
| Msg\_time\_date | datetime | No |  |  |
| Msg\_content | Varchar(500) | No |  |  |

Indexes

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Keyname | Type | Unique | Packed | Column | Cardinality | Collation | Null | Comment |
| PRIMARy | BTREE | Yes | No | #job\_id | 5 | A | No |  |
| #sender\_id | BTREE | No | No | #sender\_id | 4 | A | No |  |
| #receipient\_id | BTREE | No | No | #receipient\_id | 2 | A | No |  |

Thread

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column | Type | Null | Default | Comments |
| #thread\_id (Primary) | Int | No |  |  |
| #user\_id | Int | No |  |  |
| Thread\_name | Varchar(50) | No |  |  |
| Thread\_content | Varchar(200) | No |  |  |
| Thread\_dateTime | datetime | No |  |  |

Indexes

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Keyname | Type | Unique | Packed | Column | Cardinality | Collation | Null | Comment |
| PRIMARy | BTREE | Yes | No | #job\_id | 5 | A | No |  |
| #thread\_id | BTREE | No | No | #thread\_id | 6 | A | No |  |
| #user\_id | BTREE | No | No | #user\_id | 4 | A | No |  |

User

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column | Type | Null | Default | Comments |
| #user\_id (Primary) | Int | No |  |  |
| User\_name | Varchar(50) | No |  |  |
| User\_email | Varchar(50) | Yes | NULL |  |
| User\_role | Varchar(20) | No |  |  |
| User\_password | Varchar(15) | No |  |  |
| User\_country | Varchar(20) | No |  |  |
| #admin\_id | Int | Yes | NULL |  |

Indexes

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Keyname | Type | Unique | Packed | Column | Cardinality | Collation | Null | Comment |
| PRIMARY | BTREE | Yes | No | #job\_id | 5 | A | No |  |
| #admin\_id | BTREE | No | No | #admin\_id | 2 | A | Yes |  |



1. Task 4

Task Statement

Create a MySQL database & Implement the database design in that database  
Implement Primary Key, Foreign Key & Constraints  
Use SQL Scripts & MySQL or phpMyAdmin to create the database  
Produce the Screen Capture of created tables in phpMyAdmin or MySQL Command Prompt  
Include it as part of Project Presentation

Solution:  
  
Admin Table

CREATE TABLE Admin

(

`#admin\_id` INT NOT NULL,

`#user\_id` INT NOT NULL,

admin\_password VARCHAR(15) NOT NULL,

admin\_email VARCHAR(20) NOT NULL,

PRIMARY KEY (`#admin\_id`)

);  


Bulk Mail Table

CREATE TABLE Bulk\_Mail

(

`#email\_id` INT NOT NULL,

`#admin\_id` INT NOT NULL,

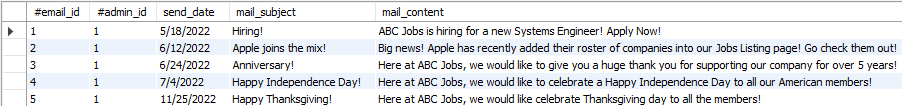
send\_date VARCHAR(25) NOT NULL,

mail\_subject VARCHAR(25) NOT NULL,

mail\_content VARCHAR(200) NOT NULL,

PRIMARY KEY (`#email\_id`),

FOREIGN KEY (`#admin\_id`) REFERENCES Admin(`#admin\_id`)

);  


User Table

CREATE TABLE User

(

`#user\_id` INT NOT NULL,

user\_name VARCHAR(50) NOT NULL,

user\_email VARCHAR(20) NOT NULL,

user\_role VARCHAR(20) NOT NULL,

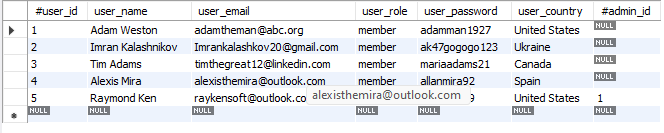
user\_password VARCHAR(15) NOT NULL,

user\_country VARCHAR(20) NOT NULL,

`#admin\_id` INT NOT NULL,

PRIMARY KEY (`#user\_id`),

FOREIGN KEY (`#admin\_id`) REFERENCES Admin(`#admin\_id`)

);  


Message Table

CREATE TABLE Message

(

`#msg\_id` INT NOT NULL,

`#sender\_id` INT NOT NULL,

`#receipient\_id` INT NOT NULL,

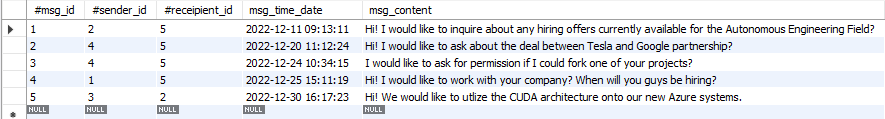
msg\_time\_date DATETIME NOT NULL,

msg\_content VARCHAR(100) NOT NULL,

PRIMARY KEY (`#msg\_id`),

FOREIGN KEY (`#sender\_id`) REFERENCES User(`#user\_id`),

FOREIGN KEY (`#receipient\_id`) REFERENCES User(`#user\_id`)

);  


Jobs Table

CREATE TABLE Jobs

(

`#job\_id` INT NOT NULL,

`#user\_id` INT NOT NULL,

job\_name VARCHAR(50) NOT NULL,

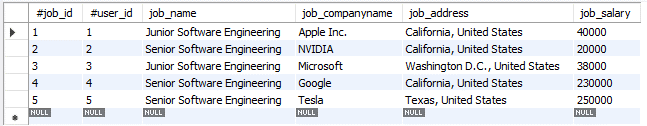
job\_companyname VARCHAR(50) NOT NULL,

job\_address VARCHAR(50) NOT NULL,

job\_salary DOUBLE) NOT NULL,

PRIMARY KEY (`#job\_id`),

FOREIGN KEY (`#user\_id`) REFERENCES User(`#user\_id`)

);  


Work Experience Table

CREATE TABLE Experience

(

`#workexp\_id` INT NOT NULL,

prev\_job VARCHAR(50) NOT NULL,

company\_name VARCHAR(50) NOT NULL,

date\_start VARCHAR(25) NOT NULL,

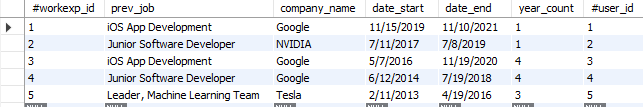
date\_end VARCHAR(25) NOT NULL,

year\_count INT NOT NULL,

`#user\_id` INT NOT NULL,

PRIMARY KEY (`#workexp\_id`),

FOREIGN KEY (`#user\_id`) REFERENCES User(`#user\_id`)

);  


CREATE TABLE Thread

(

`#thread\_id` INT NOT NULL,

`#user\_id` INT NOT NULL,

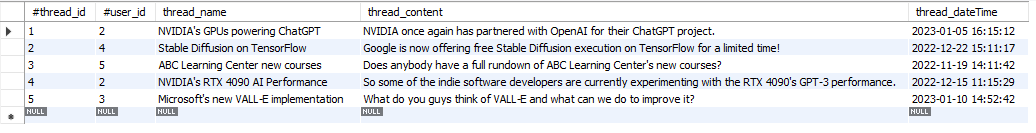
thread\_name VARCHAR(50) NOT NULL,

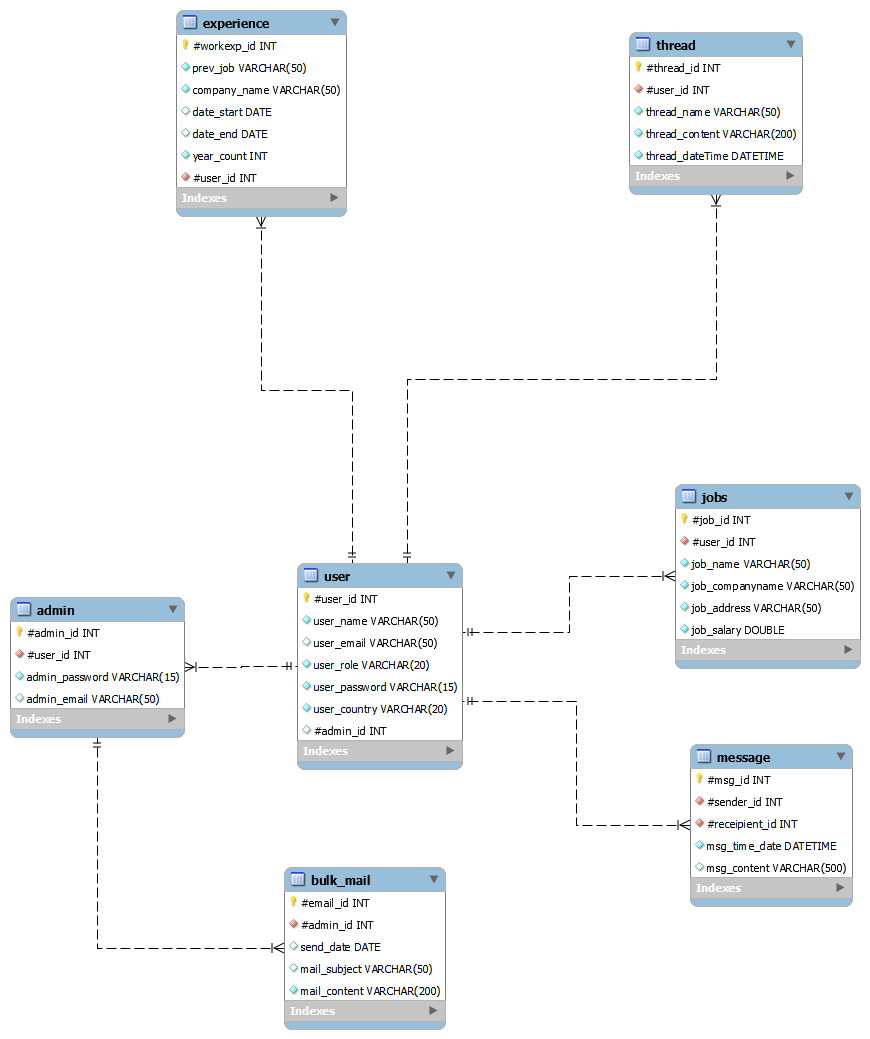
thread\_content VARCHAR(200) NOT NULL,

thread\_dateTime VARCHAR(25) NOT NULL,

PRIMARY KEY (`#thread\_id`),

FOREIGN KEY (`#user\_id`) REFERENCES User(`#user\_id`)

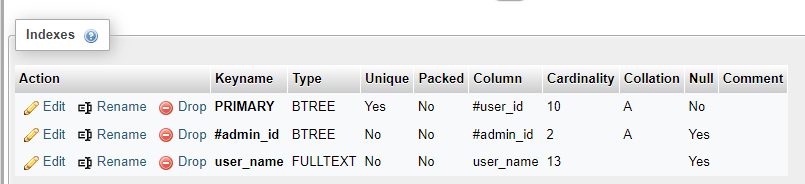
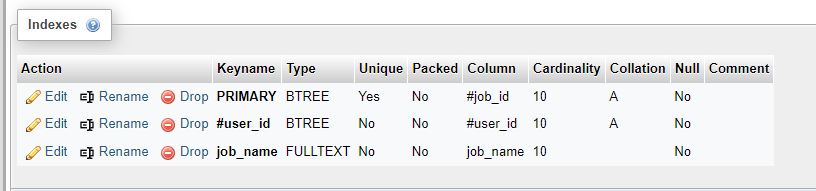
);  


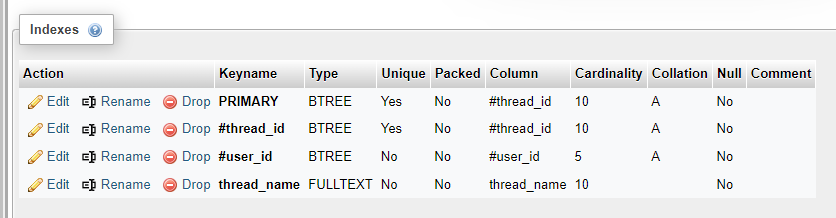
**EER Diagram**

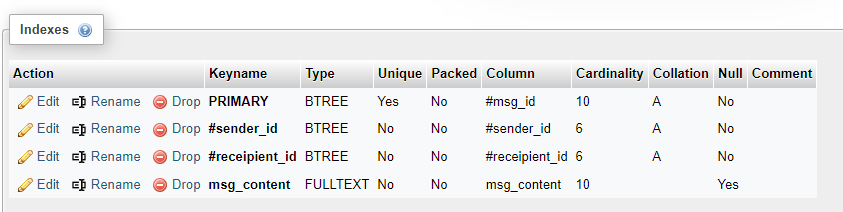
1. Task 5

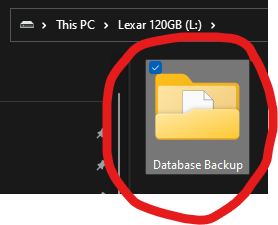
Task Statement

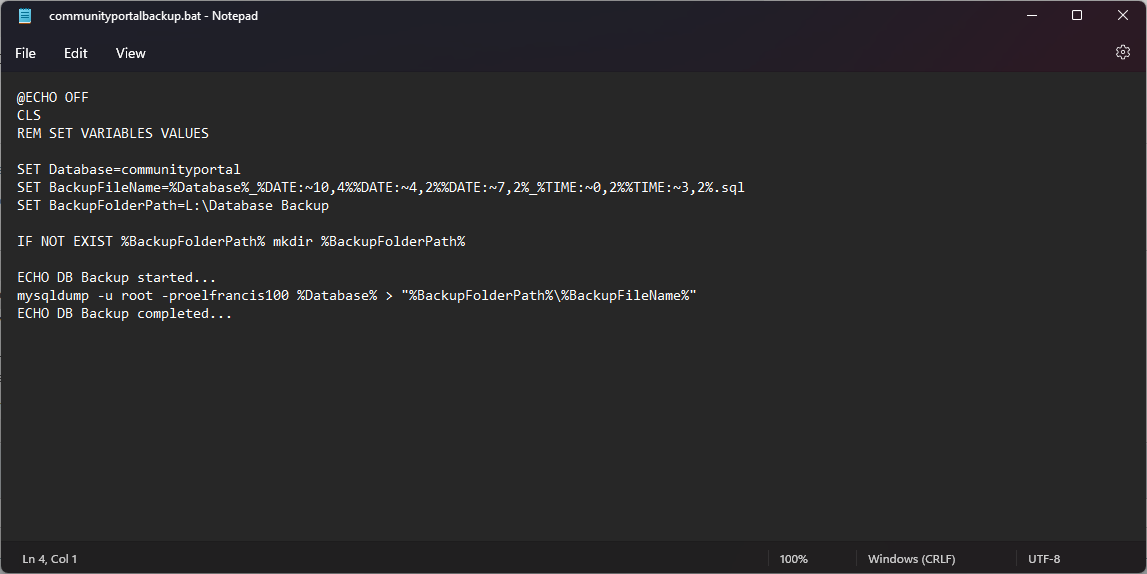
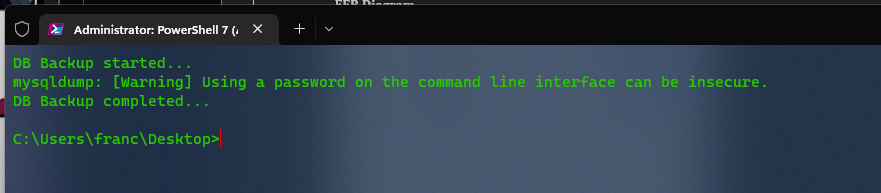
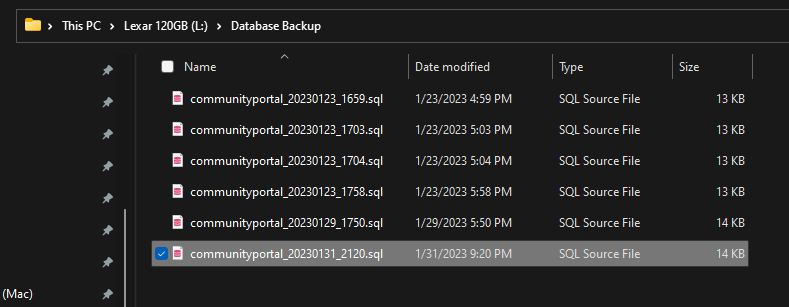
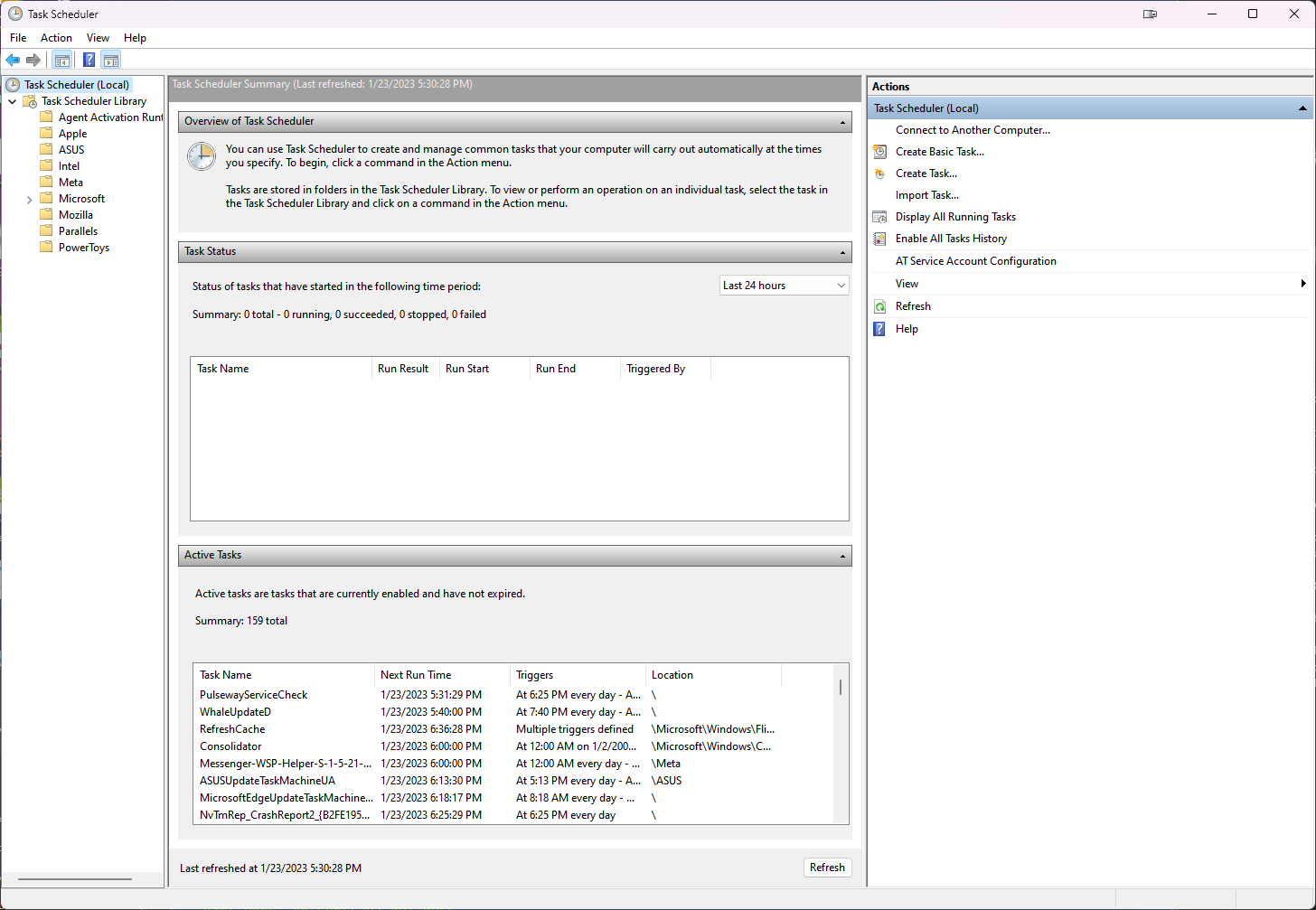
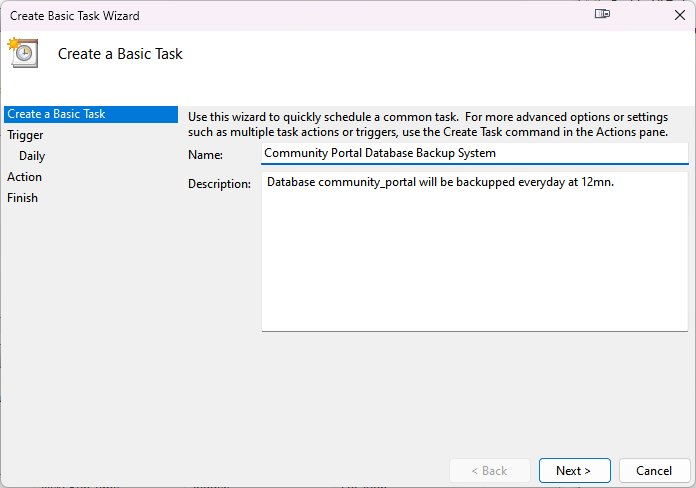
Create indexes in table, provide script for creating the index  
Create a Backup script to back up the database every 6 hours  
Include the mechanism and sources as part of Project Report

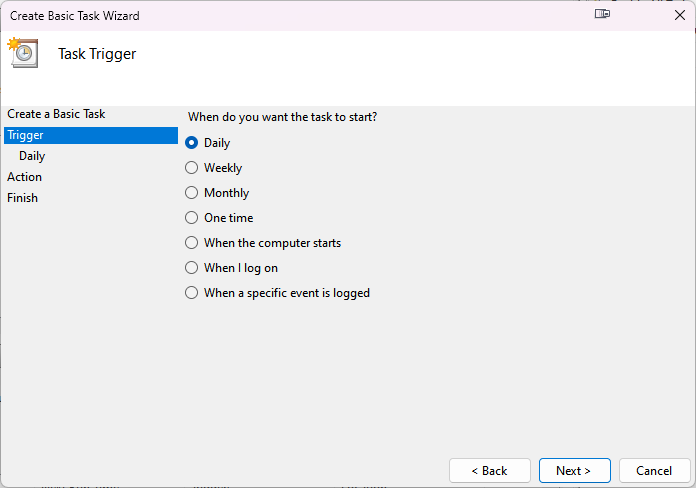
Solution  
  
Part 1. Create Indexes for any 4 tables  
  
CREATE FULLTEXT INDEX user\_name ON user (user\_name)  
  
  
CREATE FULLTEXT INDEX job\_name ON jobs (job\_name)  


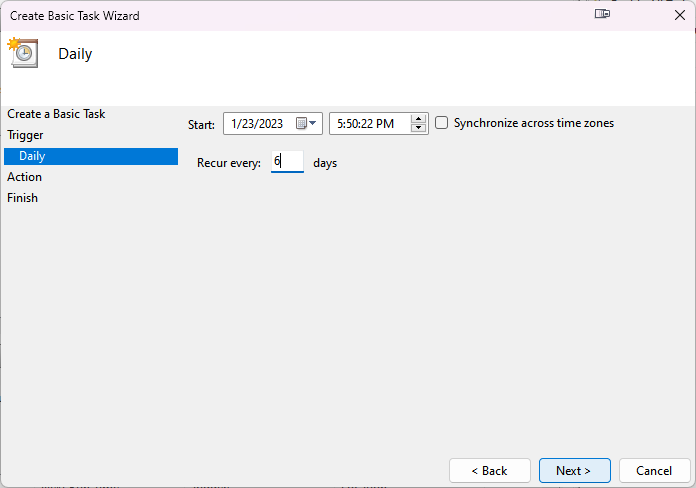
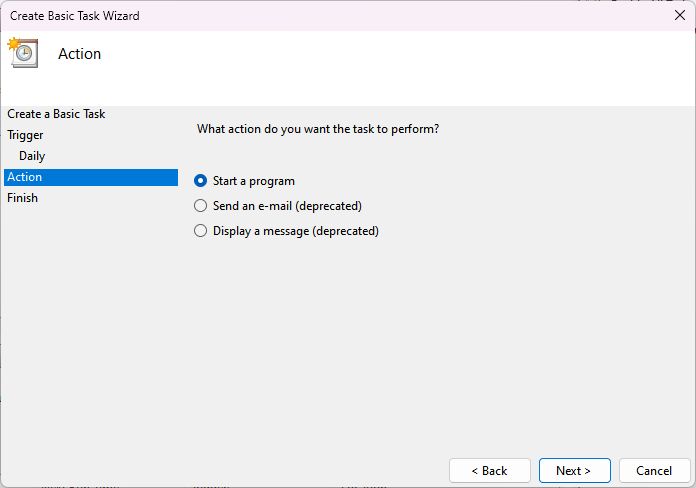
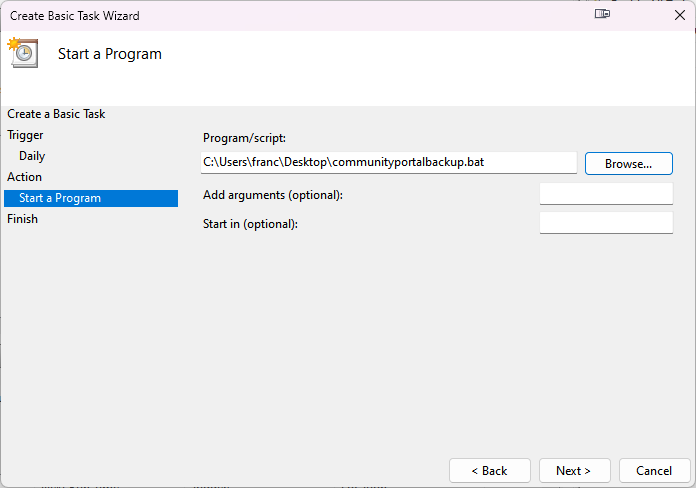
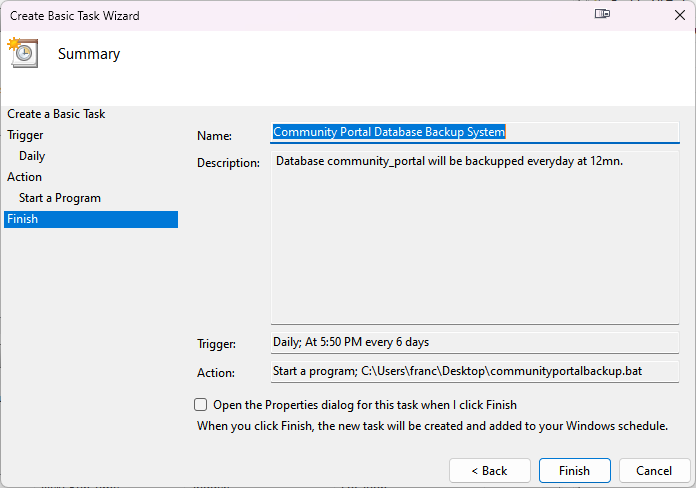
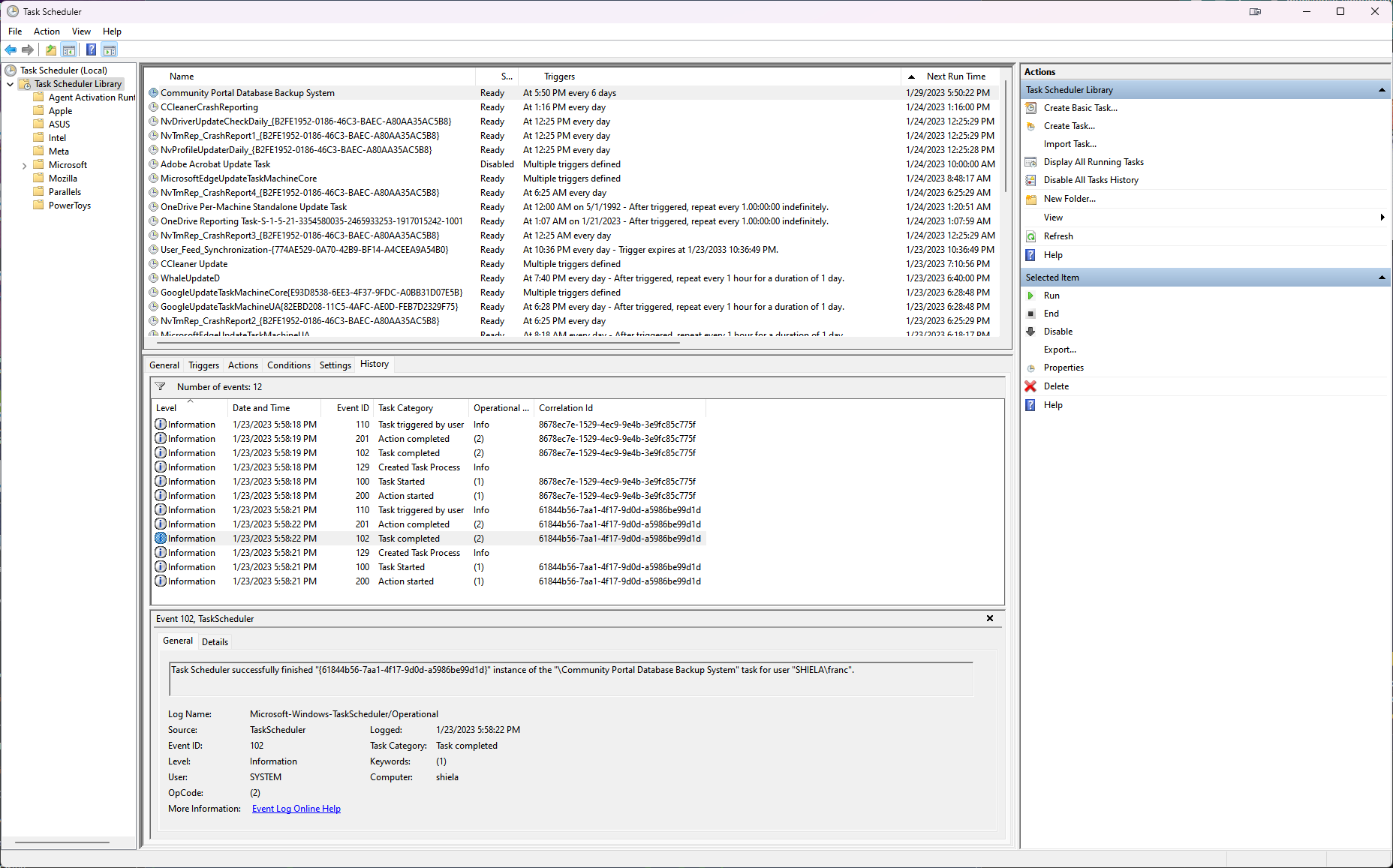
CREATE FULLTEXT INDEX thread\_name ON thread (thread\_name)  


CREATE FULLTEXT message\_content ON message (msg\_content)  


Part 2. Backup your database  
  
Create the folder where you want to back-up the database to.  


Create a script that consists of the name of your database and where you want to export the backup itself.  
  
Test the Batch File.  
  
  
After that, Open Task Scheduler  
  
Create a Basic Task  


Set start trigger for the task  


Set the start date of the task and its interval.  
  
Set the task to start a program  
  
Select your backup script  
  
Review your Configuration then click Finish  
  
Run the task inside Task Scheduler  


1. Task 6

Task Statement:

Create 8 SQL Queries which will be used by the Community Portal  
Import Sample data from CSV file in to the database.  
Provide evidence of import as part of Project Presentation.

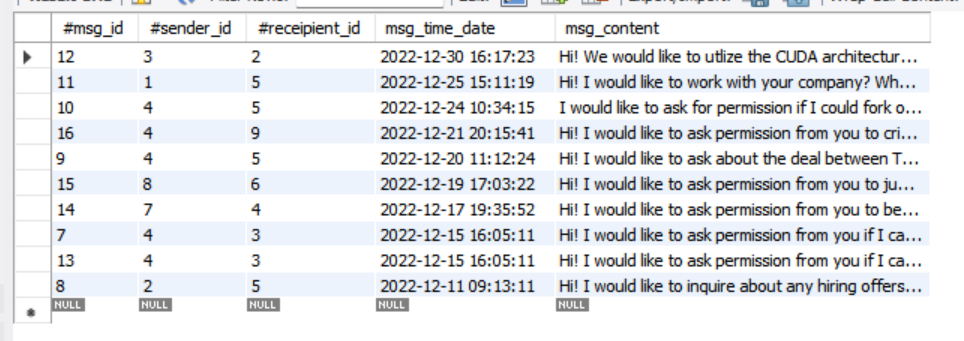
Solution

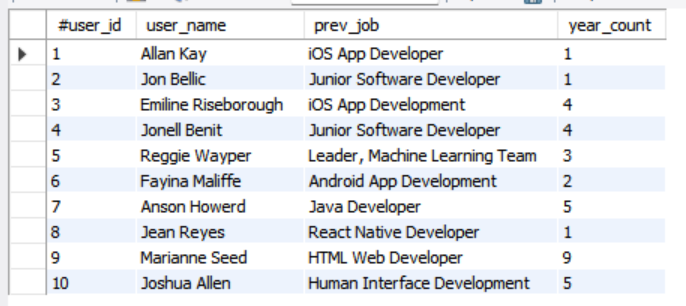
1. 8 Queries  
     
   8 Useful Queries to develop the application

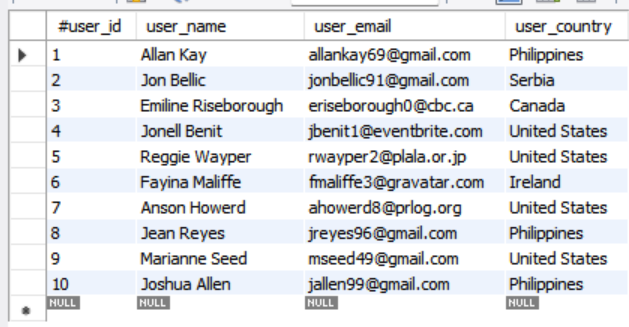
|  |  |  |  |
| --- | --- | --- | --- |
| Page | Tables | Function | Query |
| 1. Log-in | User | Validate Credentials | SELECT user\_email, user\_password FROM `user`  WHERE user\_email = 'adamtheman@abc.org' AND user\_password = 'adamman1927'; |
| 1. Registration | User | Add Credentials | INSERT INTO user (user\_name, user\_email, user\_password, user\_country) VALUES (“Allan Kay”, [allankay69@gmail.com](mailto:allankay69@gmail.com), “ieu4-81ka-1a6d”); |
| 1. View Profile | User | View Credentials | SELECT user\_name, user\_email, user\_password FROM user  WHERE Email = “registered email” AND user\_name= “registered name”; |
| 1. List Search Results | Jobs | View Jobs available | SELECT \* FROM communityportal.jobs  WHERE (job\_name LIKE '%Software%' OR 'Software' = '')  AND (job\_address LIKE '%california%' OR 'california' = '')  AND (job\_salary >= 100000 OR 100000 = 0); |
| 1. Messages | Messages | Read and Send Messages | INSERT INTO messages (sender\_id, recipient\_id, msg\_content) VALUES ({sender\_id}), {receiver\_id}, ‘{msg\_content}’, ‘{sent\_at}’); |
| 1. Threads | Thread | Create Threads | INSERT INTO thread (`#user\_id`, thread\_name, thread\_content)  VALUES (3, 'Service', 'ServiceTesting'); |
| 1. Job Opportunities | Job Experience, Jobs | Show and recommend jobs based on experience. | SELECT j.\* FROM job j  JOIN experience e ON j.id = e.job\_id WHERE e.user\_id = {user\_id} GROUP BY j.id HAVING COUNT(e.job\_id) >= {year\_count}; |
| 1. Admin | Admin | Manage the Community Portal | SELECT \* FROM admin  WHERE admin\_email = {admin\_email} AND admin\_password = ‘{admin\_password}’; |

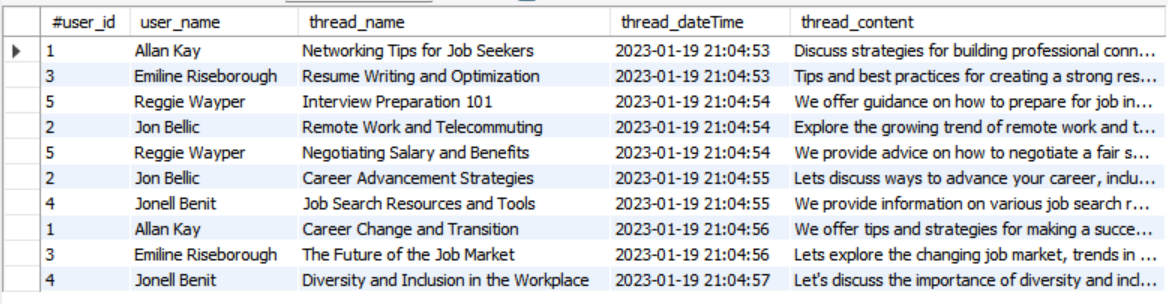
4 userful queries to meet the management requirements using Jobs

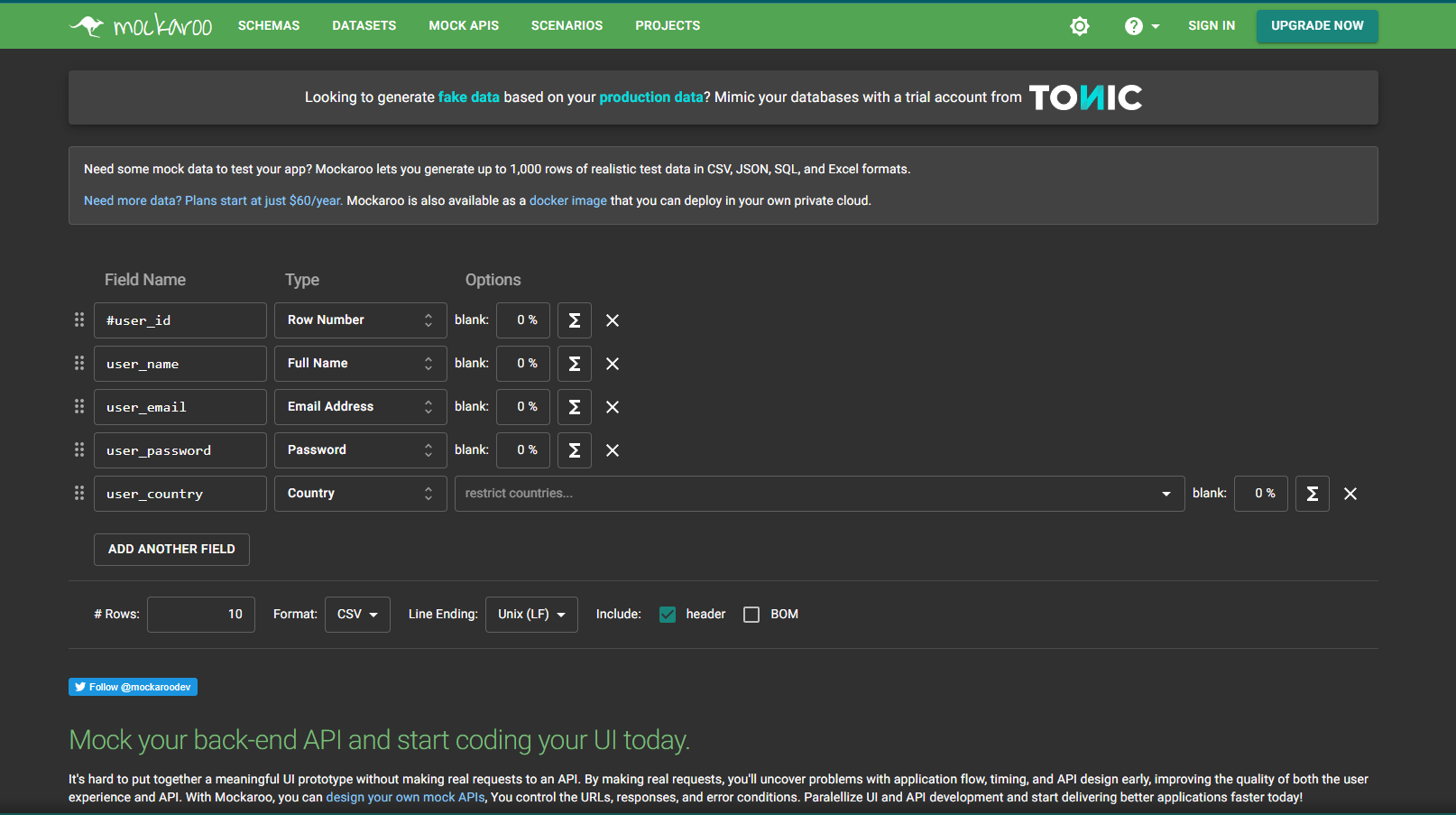
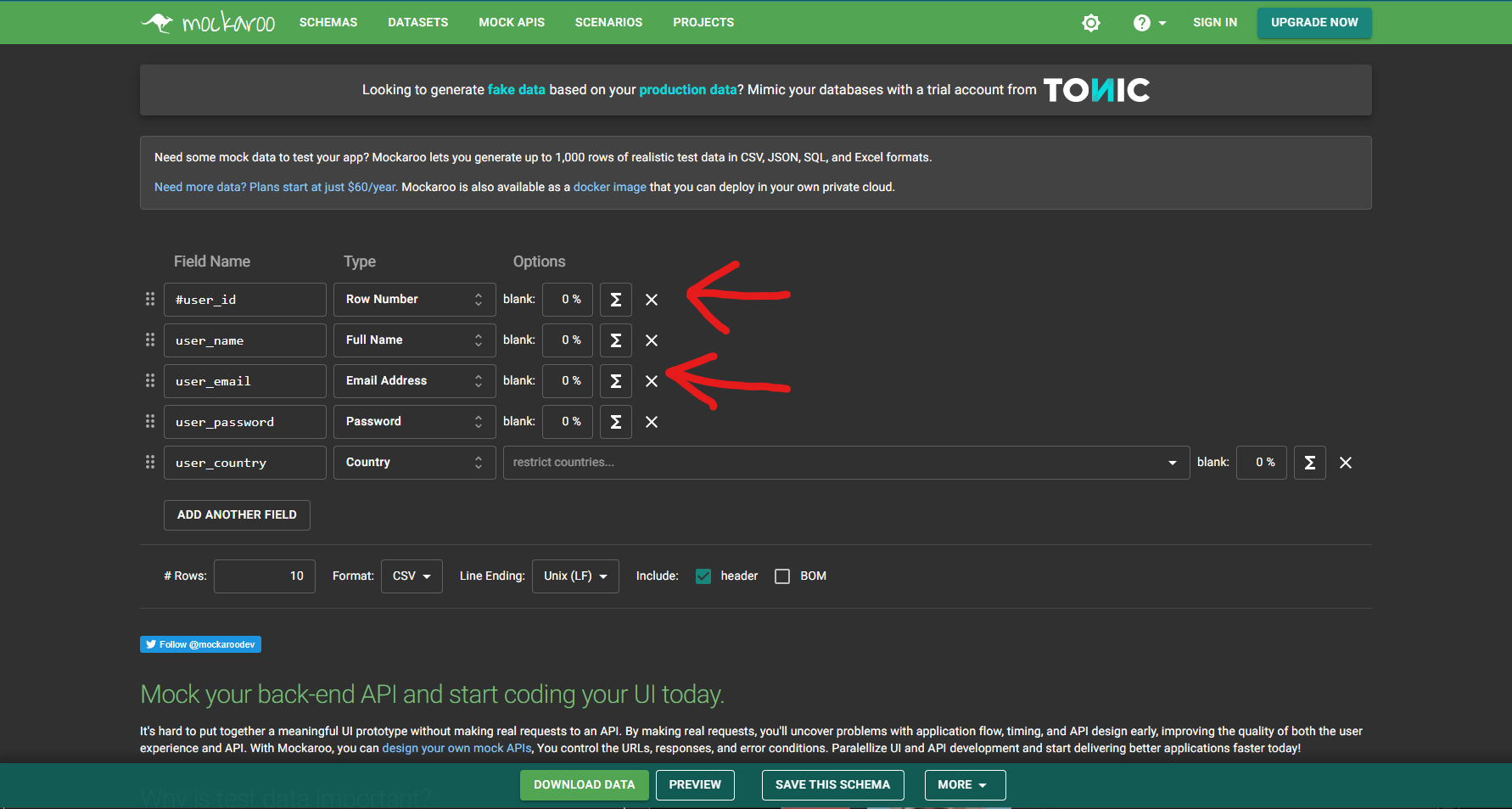
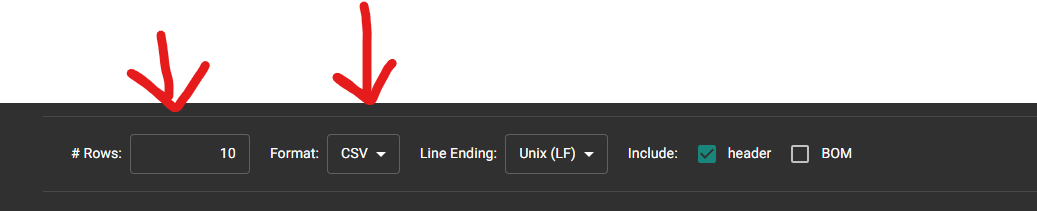
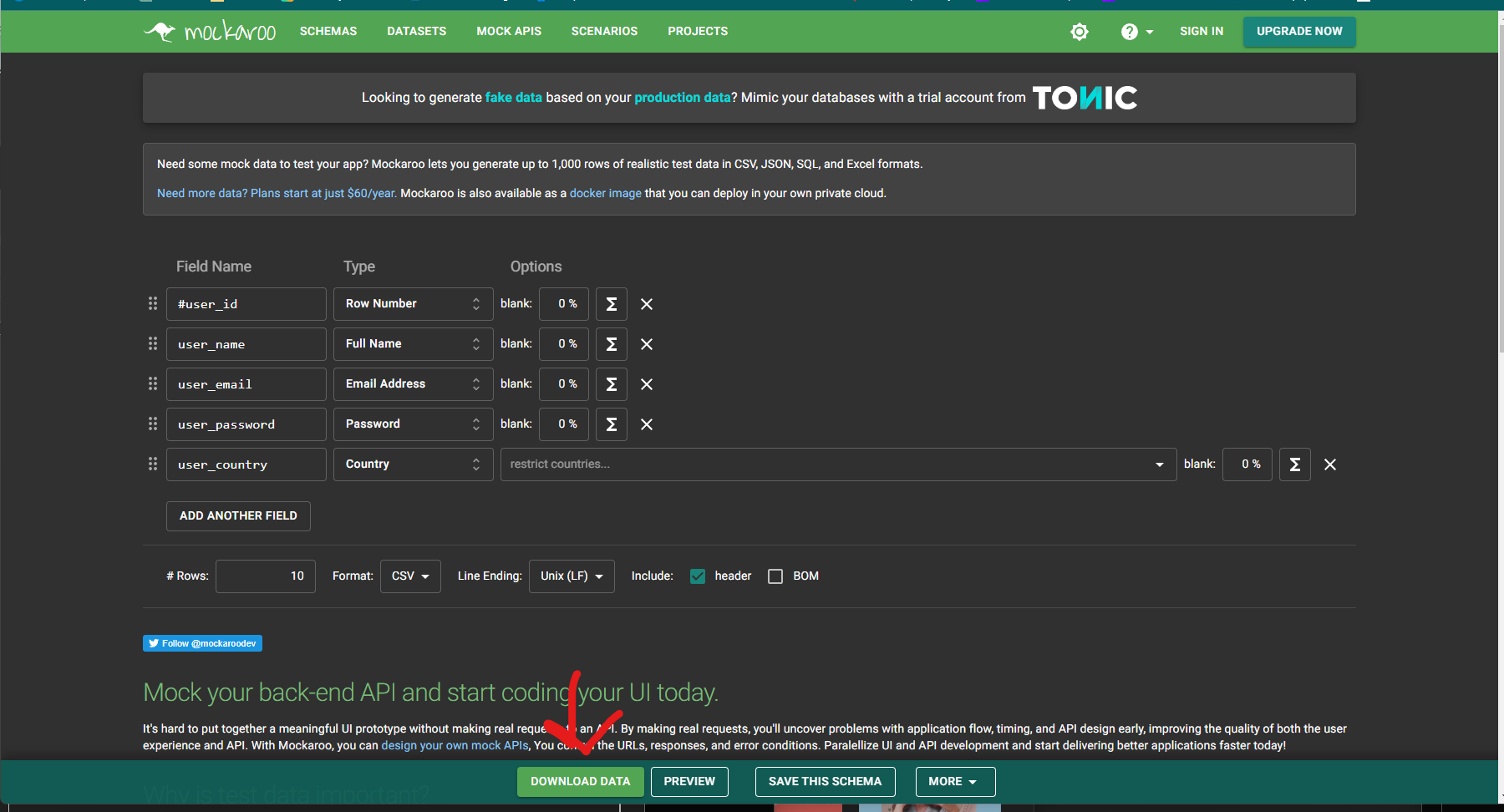
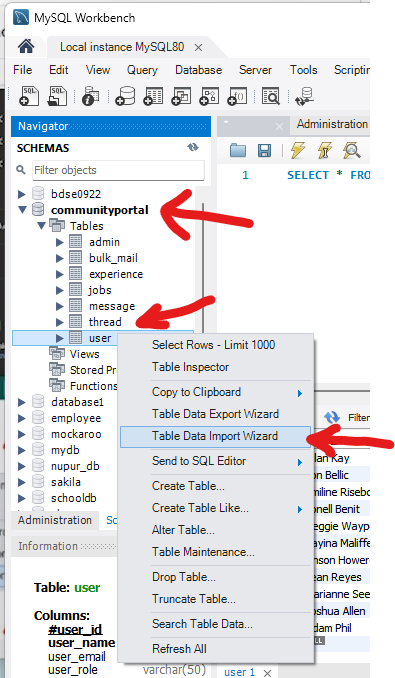
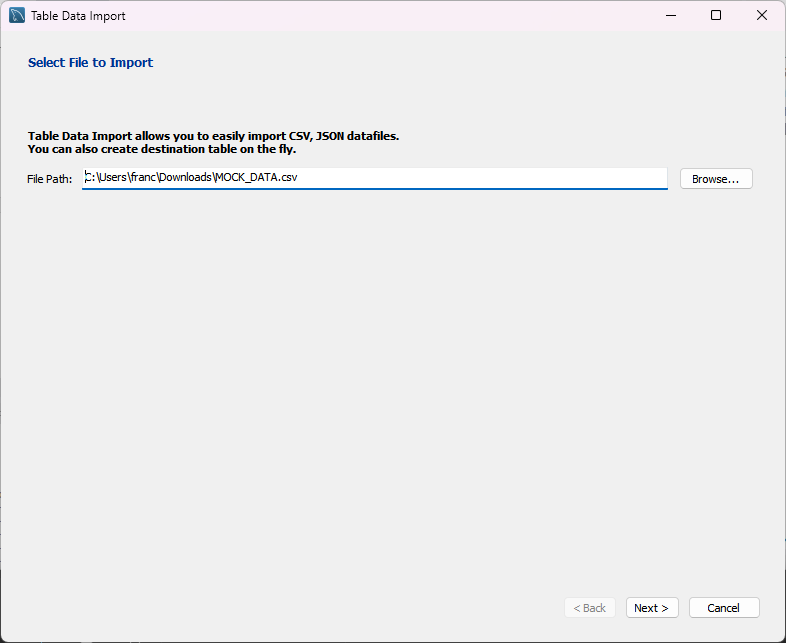
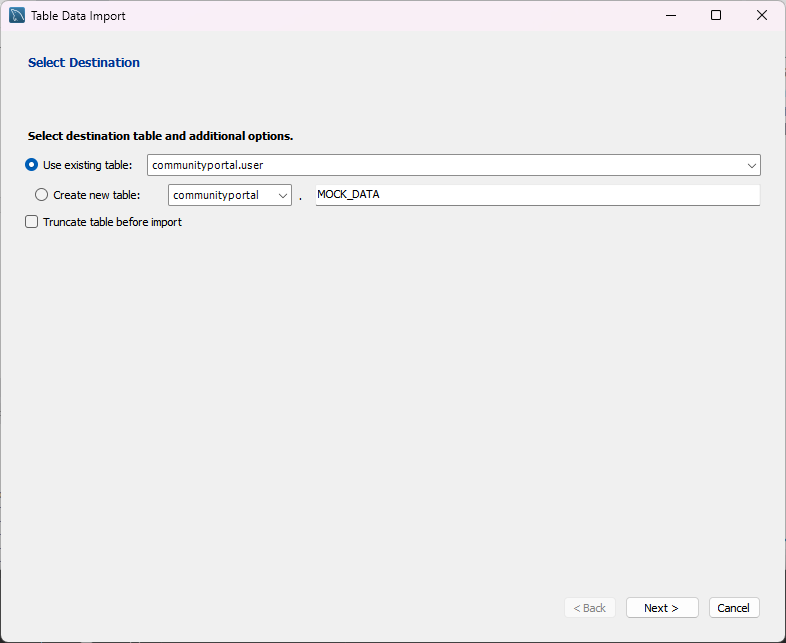
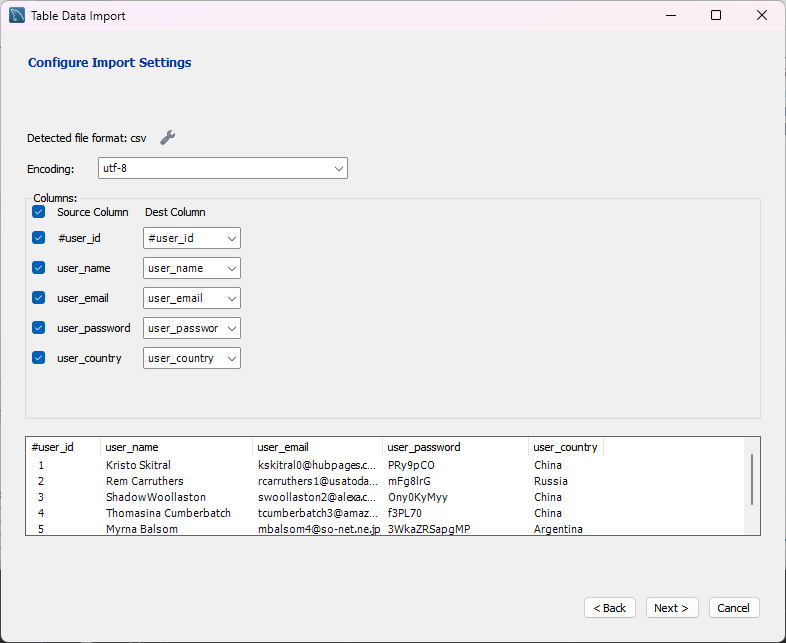
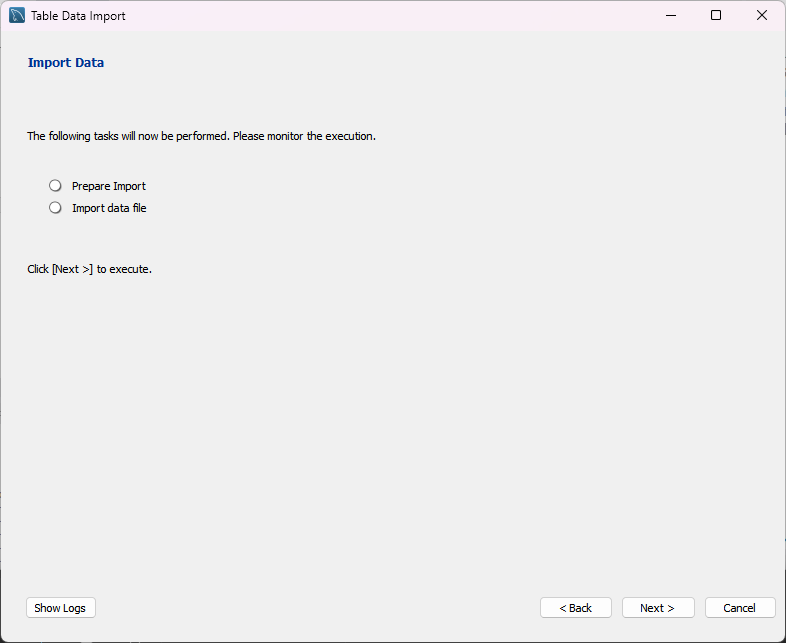
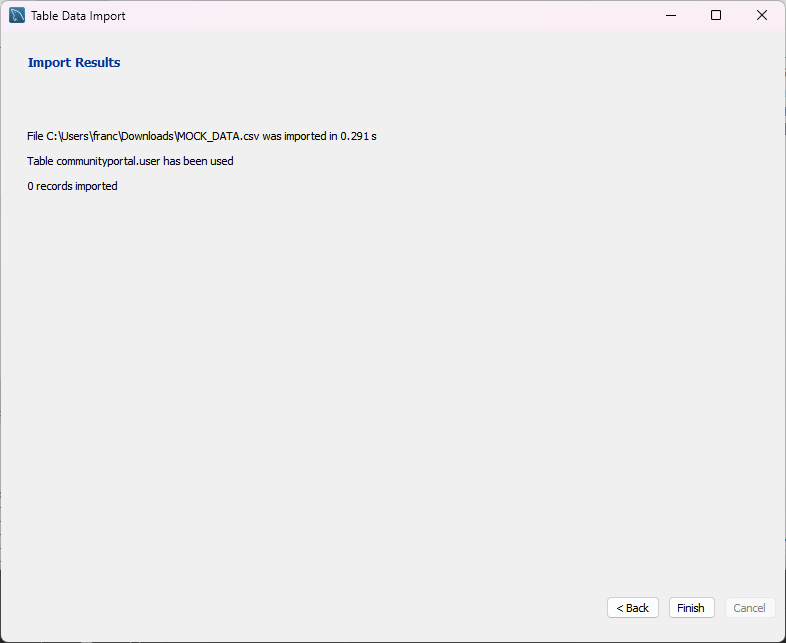
|  |  |  |  |
| --- | --- | --- | --- |
| No | Note | Query | Evidence |
| 1 | Recent Messages | SELECT \* FROM communityportal.message ORDER BY msg\_time\_date DESC LIMIT 10; | Show 10 recent messages in the database. |
| 2 | Fetch all user’s experience. | SELECT communityportal.user.`#user\_id`, communityportal.user.user\_name, communityportal.experience.prev\_job, communityportal.experience.year\_count FROM communityportal.user JOIN communityportal.experience ON communityportal.user.`#user\_id` = communityportal.experience.`#user\_id`; | Shows all of the user’s job experience in the database. |
| 3 | Fetch all user’s email and country info. | SELECT communityportal.user.`#user\_id`, communityportal.user.user\_name, communityportal.user.user\_email, communityportal.user.user\_country FROM communityportal.user; | Shows all the email and country info from each user in the database. |
| 4 | Fetch user’s threads. | SELECT communityportal.user.`#user\_id`, communityportal.user.user\_name, communityportal.thread.thread\_name, communityportal.thread.thread\_dateTime, communityportal.thread.thread\_content FROM communityportal.user JOIN communityportal.thread ON communityportal.user.`#user\_id` = communityportal.thread.`#user\_id`; | Shows all the threads created by each user. |

Report Recent Messages Log  


Report All User’s Experience Log  


Report All Users Email and Country info  


Report All User’s Threads  


1. Steps to Import CSV files  
     
   Open mockaroo.com  
     
   Configure your field name, data type and options you want to use.  
     
   Configure the number of rows and the format of your random data.  
     
   Then, download your data  
     
     
   Once downloaded, go to MySQL Workbench then right click on the table you want to add your random data then select Table Data Import Wizard  
     
   Inside the Table Data Import Wizard, copy the file path of your .csv file then paste it  
     
     
     
     
     
   Click Next then, select an existing table (which in this case is user  
     
     
   Click next then, verify your data if they are all correct  
     
     
   Then click next to start the import.  
     
   Click next then finish.  
     
   And there you have it.  
   