#### Password Manager

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

#### Bachelor of Technology/Master of Technology

In

#### Computer Science and Engineering School of Engineering and Sciences

Submitted by

#### Akhilesh Vallabhaneni (AP22110011504)



Under the Guidance of

#### Mrs. Karnena Kavitha Rani

**SRM University–AP Neerukonda, Mangalagiri, Andhra Pradesh – 522502**

**[April - 2024]**

# **Certificate**

Date: 24-April-2024

This is to certify that the work present in this Project entitled “**Password Manager**” has been carried out by **Akhilesh Vallabhaneni** under my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/Master of Technology in **School of Engineering and Sciences**.

#### Supervisor

(Signature)

Prof: Mrs. Karnena Kavitha Rani

Designation: Coding Trainer

Affiliation: CSC

#### Acknowledgements

Technology is moving fast, and hackers are keeping up. Surprisingly, though, human customs aren't changing quickly. The same tricks that have fooled people for ages are still wired into our brains. Those same methods are still being used to hack individuals and businesses today. But the good news is, it doesn't have to be that way!

Our project revolves around password manager and the technicalities of how data is created, stored, retrieved and deleted in a database management system.

I would like to thank my faculty mentor, my team members and everyone who was involved in their contributions in this project.

Special thanks to Apna College for teaching us how ER Model works in DBMS.

Thanks to various online articles and forms which are available on the internet to know in depth related topics of database management and their use cases.

In this project report you will be able to understand how changes are made to a password manager database through ER-Model.

**Table of Contents**

[**Certificate** 2](#_gjdgxs)

[**Project Background** 5](#_1fob9te)

[**Project Description** 5](#_2et92p0)

[**Statement of Contributions** 7](#_tyjcwt)

[**Description of ER Diagram**](#_4d34og8) 8

**Schemas** [**Visual Diagram**](#_3dy6vkm)10

**Entity Relationship Visual Diagram** 11

[**Conversion of ER Diagrams into Tables** 1](#_2s8eyo1)2

[**Description of Tables** 1](#_17dp8vu)3

[**Normalization of tables in the database** 1](#_3rdcrjn)3

[**Creation of Data in the tables** 1](#_26in1rg)6

[**SQL Queries** 1](#_lnxbz9)7

[**Creation of 5 view using the tables**](#_35nkun2) 21

# **Project Background**

In the fast-paced digital era, where our lives are revolving with multiple online activities, the need to prioritize our cybersecurity has never been more critical, especially for young engineering students aged 17-20. This report addresses the common oversight of online security among this demographic, drawing attention to the pervasive threats posed by cybercriminals exploiting our hectic online lifestyles. It tells us the busy digital landscape we navigate daily, highlighting how hackers can manipulate our sense of urgency, pushing us into quick decisions and making us fall to their scams.

The heart of the report explains the importance of generating unique passwords and the transformative role of password managers. By resonating with the students' daily experiences, it shows the risks associated with password reuse and the potential threat across personal and professional domains. Introducing LastPass as an example, the report explains how password managers simplify the creation and management of secure passwords, freeing from the burden of memorization.

# **Project Description**

The Password Manager System is a software solution designed to address the growing need for secure password management in today's digital world. With the proliferation of online accounts across various platforms, users often struggle to manage their passwords securely and efficiently. The Password Manager System aims to alleviate this challenge by providing users with a centralized platform to store, organize, and retrieve their passwords for different online accounts.

**Key Features:**

1. **User Registration and Authentication:** Users can create an account by providing a username, password, and email address. They can then log in securely using their credentials.
2. **Secure Storage of Passwords:** The system securely stores passwords for various online accounts, encrypting them to ensure confidentiality.
3. **Multi-Platform Access:** Users can access their stored passwords from any device with internet access, providing convenience and flexibility.
4. **Password Generation:** The system offers a password generation feature to create strong, random passwords for new accounts, enhancing security.
5. **Category-based Organization:** Users can categorize their accounts based on type (e.g., email, social media, banking) for easier management and retrieval.
6. **Password Expiry Management:** The system notifies users when passwords are due for renewal or have expired, helping them maintain account security.
7. **Security Questions:** Users can set up security questions and answers to further enhance account security and facilitate password recovery if needed.
8. **User-Friendly Interface:** The system provides an intuitive and user-friendly interface for easy navigation and usage.

**Project Goals:**

1. Develop a robust and secure password management system that prioritizes data integrity and user privacy.
2. Ensure seamless user experience through an intuitive and responsive user interface.
3. Implement encryption and other security measures to protect sensitive user information.
4. Facilitate efficient organization and retrieval of passwords through categorization and search functionalities.
5. Provide timely notifications and reminders to users to maintain strong password hygiene.

**Target Audience:**

The Password Manager System targets individuals and organizations seeking a reliable solution to manage their passwords securely. It caters to users of all technical backgrounds, offering a simple yet powerful tool to safeguard their online accounts and sensitive information. This System also includes features such as enhanced security, authentication, auto filling passwords, Multi factor authentication support, etc.

**Project Implementation:**

The Password Manager System will be implemented using a database management system (DBMS) to store and manage user data securely. The system will be developed using modern web development technologies, ensuring compatibility across various devices and platforms. Strong encryption algorithms will be employed to protect user passwords and sensitive information.

# **Statement of Contributions**

Below are the responsibilities and contributions of each candidate in this paper.

Dattatreya Nammina (AP22110010025)

Narasimham Rallabandi (AP22110010028)

Yuvaraj Prudhvi Majeti (AP22110010037)

Akhilesh Vallabhaneni (AP22110011504)

# Description of ER Diagram

In an Entity-Relationship (ER) model, symbols are graphical representations used to represent various components of the model. These symbols help visualize the structure of the data model and the relationships between different entities. Here are some common symbols used in ER diagrams:

1. **Entity:** An entity is represented by a rectangle. It represents a real-world object or concept, such as a person, place, thing, or event. The entity's name is written inside the rectangle.

2. **Attribute:** An attribute is represented by an oval or ellipse connected to its respective entity. It represents a property or characteristic of an entity. The attribute's name is written inside the oval.

3. **Primary Key**: The primary key is an attribute (or combination of attributes) that uniquely identifies each record in an entity. It is underlined within the attribute oval.

4. **Relationship**: A relationship is represented by a diamond shape connecting two or more entities. It indicates how entities are related to each other. The type of relationship (such as one-to-one, one-to-many, or many-to-many) is often labeled near the diamond.

5. **Cardinality**: Cardinality describes the numerical relationship between entities in a relationship. It is represented using symbols near the relationship lines, such as "1" (for one), "M" (for many), or a range (e.g., "1..N").

6. **Weak Entity**: A weak entity is an entity that depends on another entity (called the owner entity) for its existence. It is represented by a double-bordered rectangle.

7. **Derived Attribute**: A derived attribute is an attribute whose value can be calculated from other attributes. It is represented by a dashed oval connected to its derived-from attributes.

8. **Recursive Relationship:** A recursive relationship occurs when an entity is related to itself. It is represented similarly to other relationships but with the same entity on both ends of the relationship line.

These symbols provide a standardized way to represent the structure and relationships within an ER model, making it easier for stakeholders to understand the data model's design and constraints.

The ER Diagram illustrates the relationships between entities such as Users, Accounts, Passwords, Categories, and Security Questions. It shows how these entities are connected through primary and foreign keys.

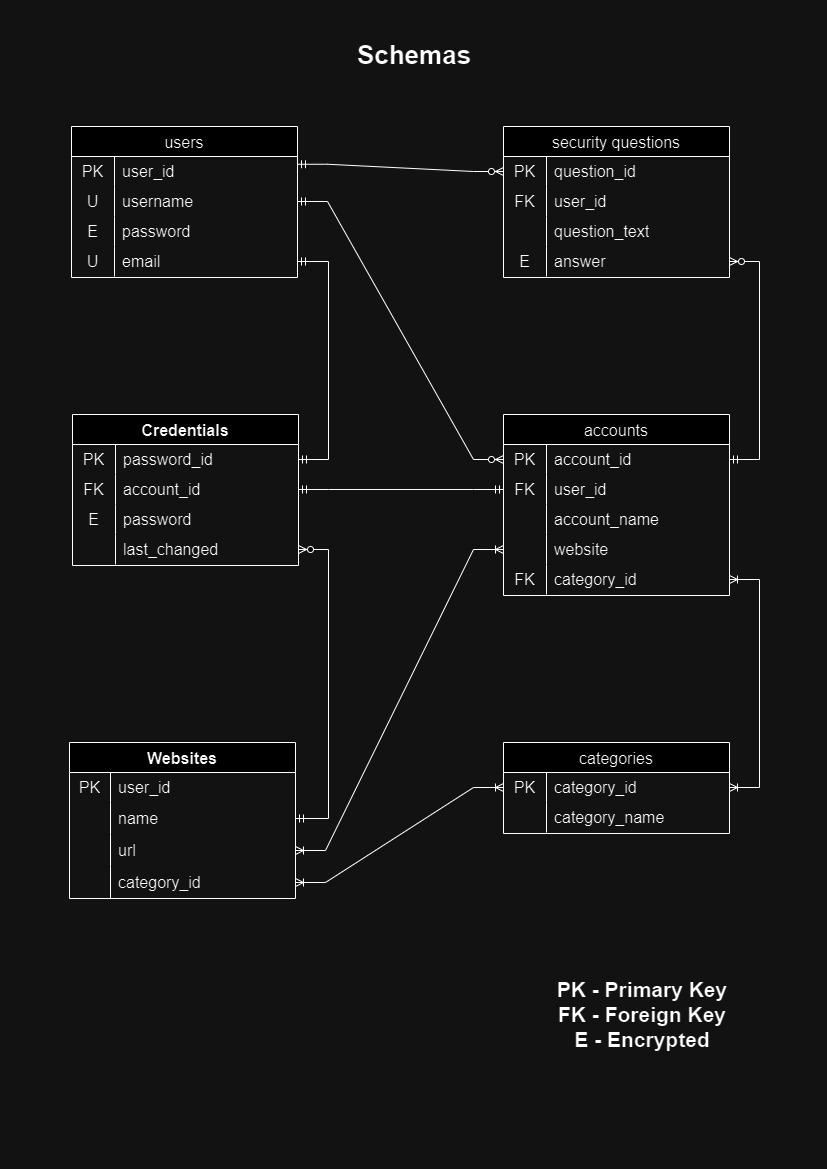
For instance, a User can have multiple Accounts, where each Account is associated with a unique Password. Categories classify Accounts and each User is linked to Security Questions for password recovery.

Furthermore, the diagram indicates that Accounts are sorted into Categories, helping to organize and manage them more effectively. For instance, one might have categories such as personal, work, or social media accounts.

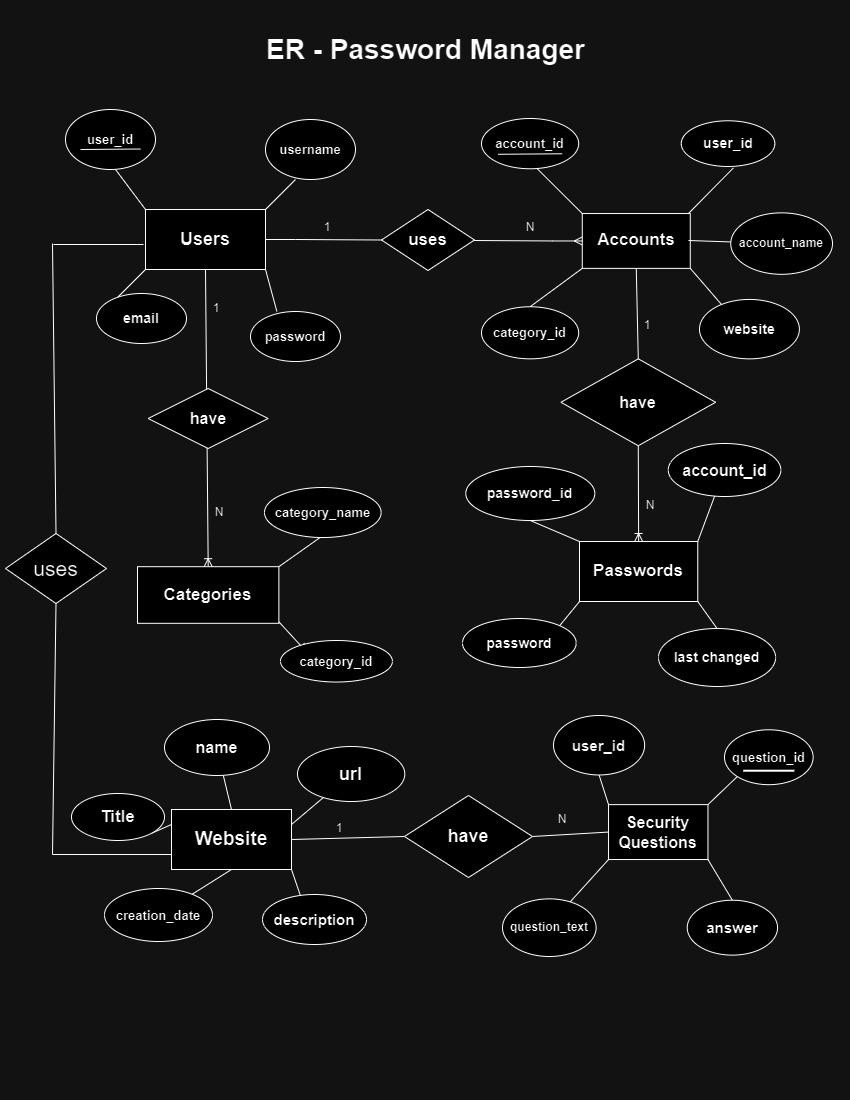
Also, the Security Questions play a crucial role in maintaining the security of the user's account. They serve as an additional layer of authentication when recovering a lost or forgotten password. The user is required to answer these questions correctly to verify their identity and gain access to their account.

The ER diagram essentially serves as a roadmap for the design and structure of the database, ensuring that data is stored in an organized and efficient manner.

# **Schemas Visual Diagram**



# Entity Relationship Visual Diagram



# **Conversion of ER Diagrams into Tables**

**User Table:**

| Column Name | Data Type | Description |
| --- | --- | --- |
| user\_id | INT | Primary Key |
| username | VARCHAR 50) | Unique |
| password | VARCHAR(50) | Encrypted |
| email | VARCHAR(100) | Unique |

**Password Table:**

| Column Name | Data Type | Description |
| --- | --- | --- |
| password\_id | INT | Primary Key |
| account\_id | INT | Foreign Key (Accounts) |
| password | VARCHAR(50) | Encrypted |
| last\_changed | DATE |  |

**Categories Table:**

| Column Name | Data Type | Description |
| --- | --- | --- |
| category\_id | INT | Primary Key |
| category\_name | VARCHAR(50) |  |

**Security Questions Table:**

| Column Name | Data Type | Description |
| --- | --- | --- |
| question\_id | INT | Primary Key |
| user\_id | INT | Foreign Key (Accounts) |
| question\_text | VARCHAR(255) |  |
| answer | VARCHAR(255) | Encrypted |

# 

# **Description of Tables**

Each table will have specific columns representing attributes related to the entity it represents. For example:

The Users table will have columns like user\_id, username, password, and email.

The Accounts table will have columns like account\_id, user\_id, account\_name, website, and category\_id.

The Passwords table will have columns like password\_id, account\_id, password, and last\_changed.

# **Normalization of tables in** the Database

The tables are normalized to minimize data redundancy and maintain data integrity. This may involve breaking down tables further and establishing relationships between them.

**First Normal Form (1NF)**

A relation will be 1NF if it contains an atomic value.

It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attributes.

First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

**Second Normal Form (2NF)**

In the 2NF, relation must be in 1NF.

In the second normal form, all non-key attributes are fully functional dependent on the primary key

**Third Normal Form (3NF)**

A relation will be in 3NF if it is in 2NF and does not contain any transitive partial dependency.

3NF is used to reduce data duplication. It is also used to achieve data integrity.

If there is no transitive dependency for non-prime attributes, then the relation must be in the third normal form.

**Boyce Codd normal form (BCNF)**

BCNF is the advanced version of 3NF. It is stricter than 3NF.

A table is in BCNF if every functional dependency X → Y, X is the super key of the table.

For BCNF, the table should be in 3NF, and for every FD, LHS is a super key.

**Fourth Normal Form (4NF)**

A relation will be in 4NF if it is in Boyce Codd normal form and has no multivalued dependency.

For a dependency A → B, if for a single value of A, multiple values of B exists, then the relation will be a multivalued dependency.

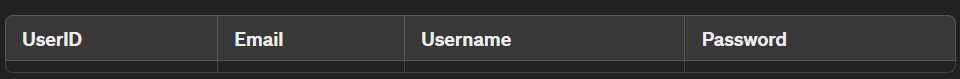
**Fifth Normal Form (5NF)**

A relation is in 5NF if it is in 4NF and does not contain any join dependency and joining should be lossless.

5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.

5NF is also known as Project-join normal form (PJ/NF).

* **USERS**



1NF: The table satisfies 1NF as each cell contains atomic values.

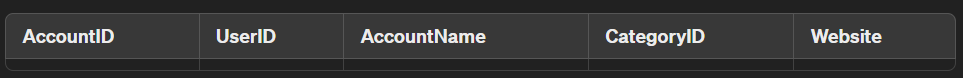
2NF: Since there are no partial dependencies, the table automatically satisfies 2NF.

3NF: There are no transitive dependencies, so it also satisfies 3NF.

4NF: There are no multivalued dependencies, so it satisfies 4NF.

5NF: There are no join dependencies, so it satisfies 5NF.

* **Accounts**



1NF: The table satisfies 1NF.

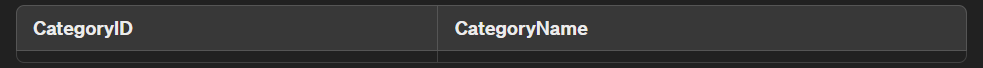
2NF: AccountName depends only on AccountID, so it satisfies 2NF.

3NF: Website depends on UserID, CategoryID, and AccountName. It needs to be decomposed into a separate table to satisfy 3NF.

4NF: There are no multivalued dependencies after normalization, so it satisfies 4NF.

5NF: There are no join dependencies, so it satisfies 5NF.

* **Categories**



1NF: The table satisfies 1NF.

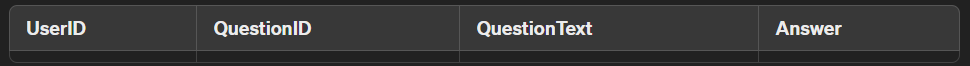
2NF: It has only one attribute, so it trivially satisfies 2NF.

3NF: It has no transitive dependencies, so it satisfies 3NF.

4NF: There are no multivalued dependencies, so it satisfies 4NF.

5NF: There are no join dependencies, so it satisfies 5NF.

* **Security Questions**



1NF: The table satisfies 1NF.

2NF: It has only one attribute, so it trivially satisfies 2NF.

3NF: There are no transitive dependencies, so it satisfies 3NF.

4NF: There are no multivalued dependencies, so it satisfies 4NF.

5NF: There are no join dependencies, so it satisfies 5NF.

* **Websites**



1NF: The table satisfies 1NF.

2NF: It has only one attribute, so it trivially satisfies 2NF.

3NF: There are no transitive dependencies, so it satisfies 3NF.

4NF: There are no multivalued dependencies, so it satisfies 4NF.

5NF: There are no join dependencies, so it satisfies 5NF.

# **Creation of Data in the tables**

We will populate the table with sample data to demonstrate how the system works. This will include at least 5 records for each table, representing different users, accounts, passwords, categories, and security questions .

**Users Table:**

| user\_id | username | password | email |
| --- | --- | --- | --- |
| 1 | User 1 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | user1@gmail.com |
| 2 | User 2 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | user2@gmail.com |

**Accounts Table:**

| account\_id | user\_id | account\_name | website | category\_id |
| --- | --- | --- | --- | --- |
| 1 | 1 | Gmail | www.gmail.com | 1 |
| 2 | 1 | Facebook | www.facebook.com | 2 |
| 3 | 2 | Linkedin | www.linkedln.com | 2 |

**Passwords Table:**

| password\_id | account\_id | password | last\_changed |
| --- | --- | --- | --- |
| 1 | 1 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 01-04-2024 |
| 2 | 2 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 28-03-2024 |
| 3 | 3 | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | 25-03-2024 |

**Categories Table:**

| category\_id | category\_name |
| --- | --- |
| 1 | Email |
| 2 | Social Media |

**Security Questions Table:**

| question\_id | user\_id | question\_text | answer |
| --- | --- | --- | --- |
| 1 | 1 | What is your pet’s name? | 01-04-2024 |
| 2 | 1 | What is your mother's maiden name? | 28-03-2024 |
| 3 | 2 | What is the name of your first school? | 25-03-2024 |

# Websites:

| name | url | description | Title | Creation Date |
| --- | --- | --- | --- | --- |
| google | www.google.com | google browser | GOOGLE | 01-04-2024 |
| facebook | www.facebook.com | social media platform | FACEBOOK | 28-03-2024 |
| naukri | www.naukri.com | job search | NAUKRI | 25-03-2024 |

# SQL Queries

* **Creating relations using SQL:**

**Users:**

CREATE TABLE Users (

UserID INT AUTO\_INCREMENT PRIMARY KEY,

Email VARCHAR(255) UNIQUE NOT NULL,

Username VARCHAR(100) UNIQUE NOT NULL,

Password VARCHAR(255) NOT NULL

);

**Accounts:**

CREATE TABLE Accounts (

AccountID INT PRIMARY KEY AUTO\_INCREMENT,

UserID INT,

AccountName VARCHAR(255),

CategoryID INT,

Website VARCHAR(255),

FOREIGN KEY (UserID) REFERENCES Users(UserID),

FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)

);

**Categories:**

CREATE TABLE Categories (

category\_id INT AUTO\_INCREMENT PRIMARY KEY,

category\_name VARCHAR(255) NOT NULL

);

**Passwords:**

CREATE TABLE Passwords (

PasswordID INT PRIMARY KEY AUTO\_INCREMENT,

AccountID INT,

Password VARCHAR(255),

LastChanges TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

**Security Questions:**

CREATE TABLE SecurityQuestions (

UserID INT,

QuestionID INT PRIMARY KEY,

QuestionText VARCHAR(255),

Answer VARCHAR(255)

);

**Websites:**

CREATE TABLE Websites (

WebsiteID INT PRIMARY KEY AUTO\_INCREMENT,

Title VARCHAR(255),

Name VARCHAR(255),

URL VARCHAR(255),

CreationDate DATE,

Description TEXT

);

* **Storing data in relations using SQL:**

**Users:**

INSERT INTO Users (UserID, Email, Username, Password)

VALUES (1, 'example@example.com', 'example\_username', 'example\_password');

**Accounts:**

INSERT INTO Accounts (AccountID, UserID, AccountName, CategoryID, Website)

VALUES (1, 101, 'Savings', 1, 'www.example.com');

**Categories:**

INSERT INTO Categories (CategoryName, CategoryID)

VALUES ('CategoryNameValue', CategoryIDValue);

**Passwords:**

INSERT INTO passwords (password\_id, account\_id, password, last\_changes)

VALUES (1, 1, 'example\_password', '2024-05-01 12:00:00');

**Security Questions:**

INSERT INTO SecurityQuestions (UserID, QuestionID, QuestionText, Answer)

VALUES

(1, 1, 'What is your mother''s maiden name?', 'Smith'),

(1, 2, 'What is the name of your first pet?', 'Fluffy'),

(2, 1, 'What city were you born in?', 'New York');

**Websites:**

INSERT INTO Websites (Title, Name, URL, CreationDate, Description)

VALUES ('Example Website', 'Example', 'http://www.example.com', '2024-05-01', 'This is an example website.');

1. Retrieve all accounts of a specific user

SELECT \* FROM Accounts WHERE user\_id = 1;

1. Find the account with strongest password

SELECT \* FROM Passwords ORDER BY LENGTH(password) DESC LIMIT 1;

1. Update the password for a particular account

UPDATE Passwords SET password = 'new\_password' WHERE account\_id = 1;

1. Identify accounts with passwords expiring soon

SELECT \* FROM Passwords WHERE DATEDIFF(CURDATE(), last\_changed) > 90;

1. Counts the number of accounts in each category

SELECT c.category\_name, COUNT(\*) AS account\_count

FROM Accounts a

JOIN Categories c ON a.category\_id = c.category\_id

GROUP BY c.category\_name;

# **Creation of 5 view using the tables**

1. Display all accounts and their passwords

CREATE VIEW AccountPasswords AS

SELECT a.account\_name, p.password

FROM Accounts a

JOIN Passwords p ON a.account\_id = p.account\_id;

1. Show users and their security questions

CREATE VIEW UserSecurityQuestions AS

SELECT u.username, sq.question\_text

FROM Users u

JOIN SecurityQuestions sq ON u.user\_id = sq.user\_id;

1. List accounts sorted by password strength

CREATE VIEW StrongPasswords AS

SELECT a.account\_name, p.password

FROM Accounts a

JOIN Passwords p ON a.account\_id = p.account\_id

ORDER BY LENGTH(p.password) DESC;

1. Display accounts expiring with a specified period

CREATE VIEW ExpiringPasswords AS

SELECT a.account\_name, p.last\_changed

FROM Accounts a

JOIN Passwords p ON a.account\_id = p.account\_id

WHERE DATEDIFF(CURDATE(), p.last\_changed) > 90;

1. Show accounts grouped by category

CREATE VIEW AccountsByCategory AS

SELECT c.category\_name, a.account\_name

FROM Accounts a

JOIN Categories c ON a.category\_id = c.category\_id;