## ABSTRACT:

This project aims to develop a comprehensive cyber security data management system for an organization, focusing on the efficient and secure handling of sensitive data. The project involves the analysis of the organization's existing data management practices and identification of potential vulnerabilities and risks. The system design includes the implementation of access controls, encryption techniques, and monitoring mechanisms to prevent unauthorized access, data breaches, and data loss. The project also includes the development of a disaster recovery plan and regular data backup procedures to ensure business continuity. The outcome of the project will be a robust and scalable data management system that complies with regulatory requirements and provides a secure environment for the organization's data.

## TABLES:

1.*User table* - This table will store information about the users of the system, including their username, password, email address, role, and access privileges.

2. *Data table* - This table will store information about the organization's sensitive data, including data type, classification level, owner, location, and access controls.

3. *Audit table* - This table will store information about system activities, including user login/logout, data access, data modifications, and security events.

4. *Backup table* - This table will store information about the organization's data backups, including backup schedule, location, and status.

5. *Disaster recovery table* - This table will store information about the organization's disaster recovery plan, including recovery procedures, contact information, and test results.

## CARDINALITIES AND CONSTRAINTS:

*User table:*

- Mapping cardinality: One-to-many with the Data table, One-to-many with the Audit table.

- Participation constraints: The user table must have at least one record for an organization to use the system.

- Key constraints: The username column is unique and serves as the primary key.

*Data table:*

- Mapping cardinality: Many-to-one with the User table.

- Participation constraints: None.

- Key constraints: The combination of the data type and location columns is unique and serves as the primary key.

*Audit table:*

- Mapping cardinality: Many-to-one with the User table, Many-to-one with the Data table.

- Participation constraints: None.

- Key constraints: A composite key consisting of the timestamp, user, and activity columns serves as the primary key.

*Backup table:*

- Mapping cardinality: Many-to-one with the User table, Many-to-one with the Data table.

- Participation constraints: None.

- Key constraints: The combination of the timestamp and location columns is unique and serves as the primary key.

*Disaster recovery table:*

- Mapping cardinality: Many-to-one with the User table, Many-to-one with the Data table.

- Participation constraints: None.

- Key constraints: The procedure column is unique and serves as the primary key.

ATTRIBUTES:

*User table:*

- username (string, primary key)

- password (encrypted string)

- email (string)

- role (enum: admin, user)

- access privileges (enum: read, write, delete)

*Data table:*

- data type (string)

- classification level (enum: confidential, secret, top secret)

- owner (string)

- location (string)

- access controls (enum: role-based, attribute-based)

- data_id (integer, primary key)

*Audit table:*

- timestamp (datetime)
- user (string, foreign key referencing User table)
- activity (enum: login, logout, access, modify, security event)
- data (integer, foreign key referencing Data table)
- result (enum: success, failure)
- audit_id (integer, primary key)

*Backup table:*

- timestamp (datetime)
- location (string)
- status (enum: success, failure)
- data (integer, foreign key referencing Data table)
- user (string, foreign key referencing User table)
- backup_id (integer, primary key)

*Disaster recovery table:*

- procedure (string, primary key)
- contact information (string)
- test results (enum: pass, fail)
- data (integer, foreign key referencing Data table)
- user (string, foreign key referencing User table)
- recovery_id (integer, primary key)

DDL OPERATIONS:

*User Table:*

```
CREATE TABLE user_table (
  username VARCHAR (50) PRIMARY KEY,
  password VARCHAR (50),
  email VARCHAR (100),
  role ENUM ('admin', 'user'),
  access_privileges ENUM ('read', 'write', 'delete'));
```

*Data Table:*

```
CREATE TABLE data_table (
  data_id INT PRIMARY KEY AUTO_INCREMENT,
  data_type VARCHAR (50),
  classification_level ENUM ('confidential', 'secret', 'top secret'),
  owner VARCHAR (50),
  location VARCHAR (50),
  access_controls ENUM ('role-based', 'attribute-based')
);
```

*Audit Table:*

```
CREATE TABLE audit_table (
  audit_id INT PRIMARY KEY AUTO_INCREMENT,
  timestamp DATETIME,
  user VARCHAR (50),
  activity ENUM ('login', 'logout', 'access', 'modify', 'security event'),
  data INT,
  result ENUM ('success', 'failure'),
  FOREIGN KEY (user) REFERENCES user_table(username),
  FOREIGN KEY (data) REFERENCES data_table(data_id)
);
```

*Backup Table:*

```
CREATE TABLE backup_table (
  backup_id INT PRIMARY KEY AUTO_INCREMENT,
  timestamp DATETIME,
  location VARCHAR (50),
  status ENUM ('success', 'failure'),
  data INT,
  user VARCHAR (50),
  FOREIGN KEY (data) REFERENCES data_table(data_id),
  FOREIGN KEY (user) REFERENCES user_table(username));
```

*Disaster Recovery Table:*

```
CREATE TABLE recovery_table (
  recovery_id INT PRIMARY KEY AUTO_INCREMENT,
  procedure VARCHAR (50),
  contact_information VARCHAR (100),
  test_results ENUM ('pass', 'fail'),
  data INT,
  user VARCHAR (50),
  FOREIGN KEY (data) REFERENCES data_table(data_id),
  FOREIGN KEY (user) REFERENCES user_table(username)
);
```

DML OPERATIONS:

Inserting data into the User table:

```
INSERT INTO user_table (username, password, email, role, access_privileges)
VALUES ('user1', 'password1', 'user1@example.com', 'user', 'read');
```

Output: "Query OK, 1 row affected"

Inserting data into the Data table:

```
INSERT INTO data_table (data_type, classification_level, owner, location, access_controls)
VALUES ('financial', 'secret', 'user1', 'server1', 'role-based');
```

Output: "Query OK, 1 row affected"

Inserting data into the Audit table:

**INSERT INTO audit_table (timestamp, user, activity, data, result)**

**VALUES ('2023-04-30 13:00:00', 'user1', 'login', NULL, 'success');**

Output: "Query OK, 1 row affected"

Updating data in the User table:

**UPDATE user_table**

**SET access_privileges = 'write'**

**WHERE username = 'user1';**

Output: "Query OK, 1 row affected"

Deleting data from the Backup table:

**DELETE FROM backup_table**

**WHERE timestamp < '2023-04-28 00:00:00';**

Output: "Query OK, 2 rows affected"

Selecting data from the Disaster recovery table:

**SELECT procedure, contact_information, test_results**

**FROM recovery_table**

**WHERE user = 'user1';**

Output:

```
+-----------+-----------------------+-------------+
| procedure | contact_information    | test_results|
+-----------+-----------------------+-------------+
| procedure1| user1@example.com      | pass        |
+-----------+-----------------------+-------------+
1 row in set (0.00 sec)
```

These DML operations demonstrate how to insert, update, delete, and select data from the tables. The outputs show the number of rows affected by each operation or the result set of the select operation. Note that the actual data output may vary depending on the data stored in the tables.