Table of Contents

I. Storyline

Mental health is a crucial aspect of overall well-being, encompassing emotional, psychological, and social well-being. It influences how individuals think, feel, and act, impacting their ability to cope with stress, interact with others, and make decisions. Good mental health is essential for functioning effectively in daily life, maintaining relationships, and achieving personal goals. Conversely, mental health disorders can significantly impair these abilities, leading to distress and dysfunction. Understanding mental health is vital for promoting resilience, seeking appropriate support, and fostering a supportive environment for individuals to thrive.

Mental health score calculation plays a role in assessing and monitoring mental well-being. It involves evaluating various factors such as emotional state, stress levels, coping mechanisms, and social support. By assigning numerical values to these factors and analyzing the overall score, professionals can gauge an individual's mental health status and tailor interventions accordingly. This approach enhances early detection of issues, facilitates targeted interventions, and promotes holistic mental health care. Integrating mental health score calculation into broader discussions on mental health promotes awareness, encourages proactive self-care, and contributes to improved mental health outcomes.

II. Components of Database Design

User:
UserID (Primary Key)
Username
Email
Password
DateOfBirth
Gender

Assessment:
AssessmentID (Primary Key)
UserID (Foreign Key)
Assessment_date

Answers:
AnswerID (Primary Key)
UserID (Foreign Key)
QuestionID (Foreign Key)
Answer_text

Score:
ScoreID (Primary Key)
UserID (Foreign Key)
AssessmentID (Foreign Key)
ScoreValue

Date

AssessmentResponse:
AssessmentResponseID (Primary Key)
AssessmentID (Foreign Key)
QuestionID (Foreign Key)
AnswerID (Foreign Key)

Threshold:
ThresholdID (Primary Key)
Score_id (Foreign key)
Description_text

ThresholdRange:
RangeID (Primary Key)
ThresholdID (Foreign Key)
MinValue
MaxValue

UserGroup:
GroupID (Primary Key)
GroupName

GroupMembership:
MembershipID (Primary Key)
UserID (Foreign Key)
GroupID (Foreign Key)

Form:
Form_ID(Primary key)
Form_title
Form_Des

Questions:
Questions_ID(primary key)
form_ID (Foreign Key)
Question_text

ActivityLog:
LogID (Primary Key)
UserID (Foreign Key)
ActivityType
Timestamp
Duration
Notes

Reminder:
ReminderID (Primary Key)
UserID (Foreign Key)
ReminderText

ReminderDate
IsCompleted
Notes

Contact:
ContactID (Primary Key)
UserID (Foreign Key)
Name
Email
Phone
Relationship
Notes

Expense:
ExpenseID (Primary Key)
UserID (Foreign Key)
Date
Amount
Category
Description
Notes

User:
Relationships:
One-to-One with Assessment
One-to-Many with GroupMembership
One-to-Many with ActivityLog
One-to-Many with Reminder
One-to-Many with Contact
One-to-Many with Expense
Diamond Shape Label: "Has"

Assessment:
Relationships:
One-to-One with User
One-to-Many with Score
One-to-Many with AssessmentResponse
Diamond Shape Label: "Belongs To"

Answer:
Relationships:
One-to-Many with User
One-to-Many with Questions
Diamond Shape Label: "Chosen By"

Score:
Relationships:
One-to-Many with User
One-to-One with Assessment
Diamond Shape Label: "Assigned For"

AssessmentResponse:
Relationships:
One-to-Many with Assessment
One-to-Many with Questions
One-to-Many with Answer
Diamond Shape Label: "Contains"

Threshold:
Relationships:
One-to-Many with ThresholdRange
Diamond Shape Label: "Has"

ThresholdRange:
Relationships:
Many-to-One with Threshold
Diamond Shape Label: "Part Of"

UserGroup:
Relationships:
Many-to-One with User (through GroupMembership)
Diamond Shape Label: "Includes"

GroupMembership:
Relationships:
One-to-Many with User
One-to-Many with UserGroup
Diamond Shape Label: "Belongs To"

Form:
Relationships:
One-to-Many with Questions
Diamond Shape Label: "Contains"

Questions:
Relationships:
One-to-Many with Form
One-to-Many with AssessmentResponse
One-to-Many with Answer
Diamond Shape Label: "Part Of"

ActivityLog:
Relationships:
One-to-Many with User
Diamond Shape Label: "Records"

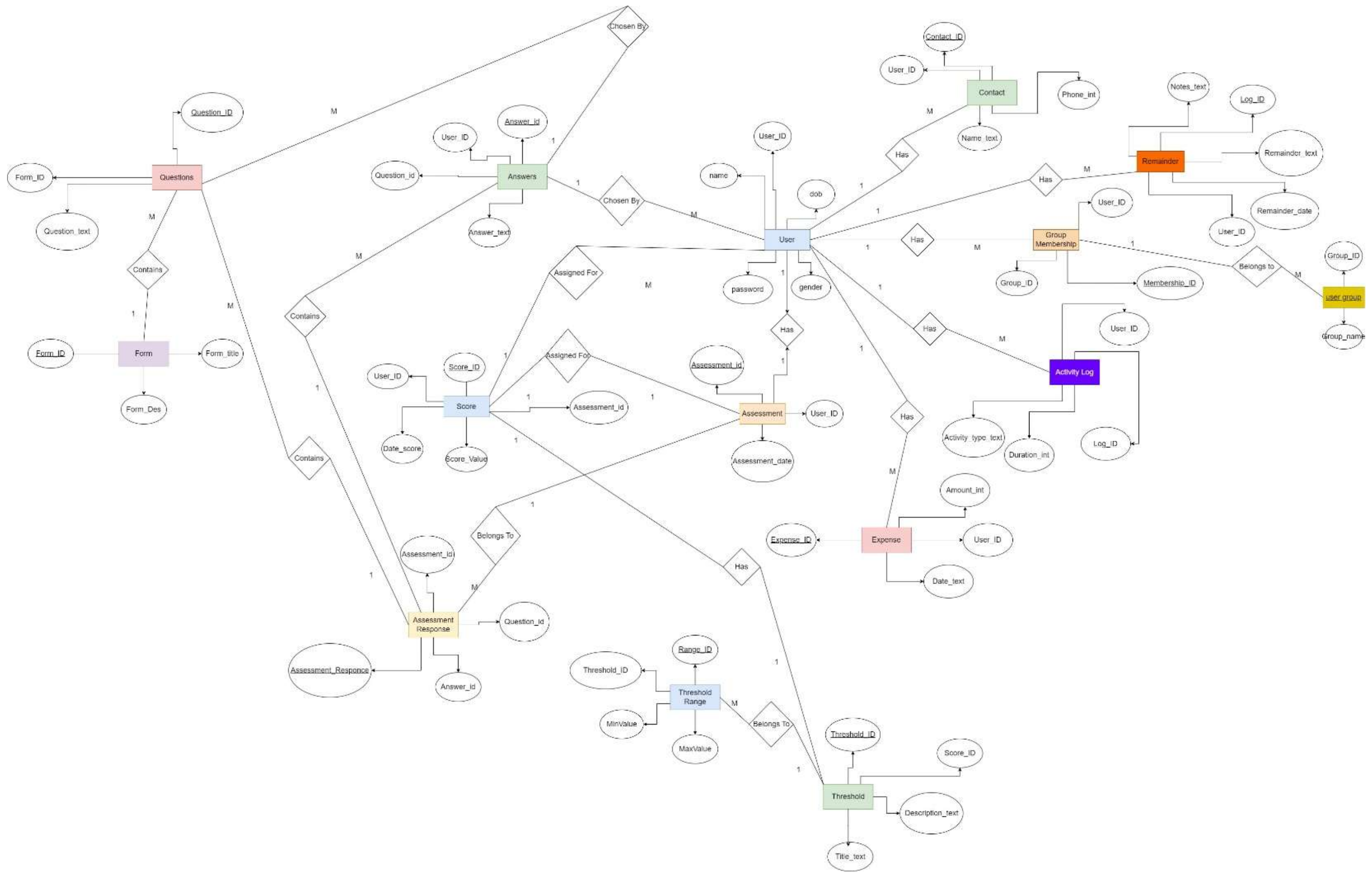
Reminder:
Relationships:
One-to-Many with User
Diamond Shape Label: "Is For"

Contact:

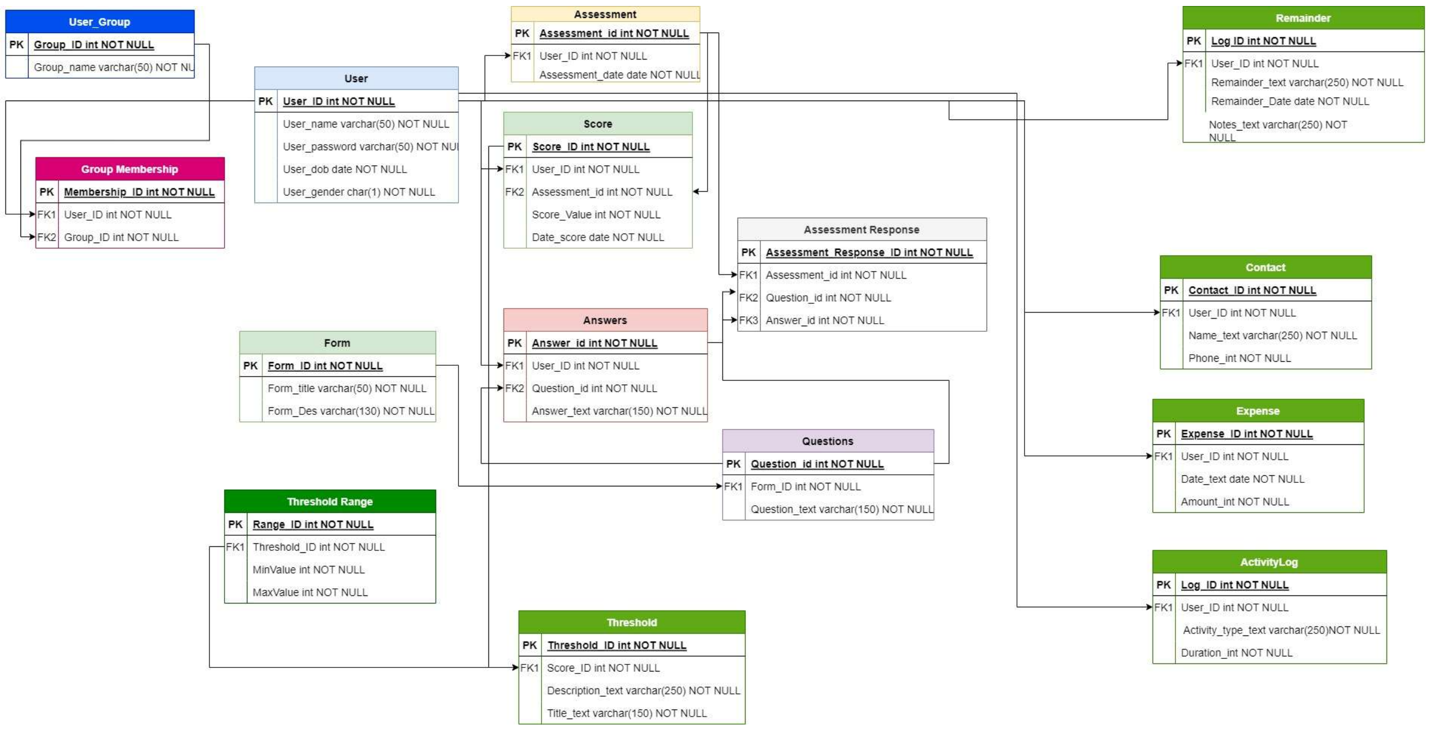
Relationships:
One-to-Many with User
Diamond Shape Label: "Related To"

Expense:
Relationships:
One-to-Many with User
Diamond Shape Label: "Incurred By"

III. Entity Relationship Diagram



IV. Relational Model



User_Group
PK Group_ID int NOT NULL
Group_name varchar(50) NOT NU
User
PK User_ID int NOT NULL
User_name varchar(50) NOT NULL
User_password varchar(50) NOT NU
User_dob date NOT NULL
User_gender char(1) NOT NULL
Group Membership
PK Membership_ID int NOT NULL
FK1 User_ID int NOT NULL
FK2 Group_ID int NOT NULL
Assessment
PK Assessment_id int NOT NULL
FK1 User_ID int NOT NULL
Assessment_date date NOT NULL
Score
PK Score_ID int NOT NULL
FK1 User_ID int NOT NULL
FK2 Assessment_id int NOT NULL
Score_Value int NOT NULL
Date_score date NOT NULL
Remainder
PK Log ID int NOT NULL
FK1 User_ID int NOT NULL
Remainder_text varchar(250) NOT NULL
Remainder_Date date NOT NULL
Notes_text varchar(250) NOT NULL
Assessment Response
PK Assessment_Response_ID int NOT NULL
FK1 Assessment_id int NOT NULL
FK2 Question_id int NOT NULL
FK3 Answer_id int NOT NULL
Contact
PK Contact_ID int NOT NULL
FK1 User_ID int NOT NULL
Name_text varchar(250) NOT NULL
Phone_int NOT NULL
Form
PK Form_ID int NOT NULL
Form_title varchar(50) NOT NULL
Form_Des varchar(130) NOT NULL
Answers
PK Answer_id int NOT NULL
FK1 User_ID int NOT NULL
FK2 Question_id int NOT NULL
Answer_text varchar(150) NOT NULL
Questions
PK Question_id int NOT NULL
FK1 Form_ID int NOT NULL
Question_text varchar(150) NOT NULL
Expense
PK Expense_ID int NOT NULL
FK1 User_ID int NOT NULL
Date_text date NOT NULL
Amount_int NOT NULL
Threshold Range
PK Range_ID int NOT NULL
FK1 Threshold_ID int NOT NULL
MinValue int NOT NULL
MaxValue int NOT NULL
ActivityLog
PK Log_ID int NOT NULL
FK1 User_ID int NOT NULL
Activity_type_text varchar(250)NOT NULL
Duration_int NOT NULL
Threshold
PK Threshold_ID int NOT NULL
FK1 Score_ID int NOT NULL
Description_text varchar(250) NOT NULL
Title_text varchar(150) NOT NULL

V. Normalization

First Normal Form (1NF):

In 1NF, each table should have a primary key, and each column should contain atomic (indivisible) values. There should be no repeating groups or arrays within a column.
Example: In the User table, each attribute (User_ID, User_name, User_email, User_password, User_dob, User_gender) contains atomic values, and there are no repeating groups.
Second Normal Form (2NF):

For a table to be in 2NF, it must first be in 1NF. Additionally, every non-prime attribute should be fully functionally dependent on the primary key.
Example: In the Score table, Score_Value and Date_Score depend on both User_ID and Assessment_ID, which form the composite primary key. There are no partial dependencies.
Third Normal Form (3NF):

A table is in 3NF if it is in 2NF and there are no transitive dependencies. In other words, non-prime attributes should not depend on other non-prime attributes.
Example: In the Threshold table, Description_text and Title_text depend only on the Assessment_ID, which is the primary key. There are no transitive dependencies.
Boyce-Codd Normal Form (BCNF):

This in BCNF.

VI. SQL Queries

```
CREATE DATABASE aq;
USE aq;

CREATE TABLE User (
   User_ID INT NOT NULL PRIMARY KEY,
   User_name VARCHAR(50) NOT NULL,
   User_password VARCHAR(50) NOT NULL,
   User_dob DATE NOT NULL,
   User_gender CHAR(1) NOT NULL
);

INSERT INTO User (User_ID, User_name, User_password, User_dob, User_gender)
VALUES
(1, 'John Doe', 'password123', '1990-05-15', 'M'),
(2, 'Jane Smith', 'pass@123', '1985-08-25', 'F'),
(3, 'Alice Johnson', 'securepwd', '1995-03-10', 'F'),
(4, 'Bob Williams', 'bob@123', '1988-11-30', 'M'),
(5, 'Emily Brown', 'emilypass', '1992-09-20', 'F'),
(6, 'Michael Clark', 'mike@123', '1987-06-18', 'M'),
(7, 'Sarah Wilson', 'sarahpass', '1998-04-05', 'F'),
(8, 'David Martinez', 'david@123', '1993-12-12', 'M'),
(9, 'Jennifer Anderson', 'jenpass', '1996-10-30', 'F'),
(10, 'William Taylor', 'will@123', '1989-07-22', 'M');

CREATE TABLE User_Group (
   Group_ID INT NOT NULL PRIMARY KEY,
   Group_name VARCHAR(50) NOT NULL
);

INSERT INTO User_Group (Group_ID, Group_name)
VALUES
(1, 'Admins'),
(2, 'Users'),
(3, 'Managers'),
(4, 'Developers'),
(5, 'Testers'),
(6, 'Support'),
(7, 'Sales'),
(8, 'Marketing'),
(9, 'Finance'),
(10, 'HR');

CREATE TABLE Group_Membership (
   Membership_ID INT NOT NULL PRIMARY KEY,
   User_ID INT NOT NULL,
   Group_ID INT NOT NULL,
    FOREIGN KEY (User_ID) REFERENCES User(User_ID),
    FOREIGN KEY (Group_ID) REFERENCES User_Group(Group_ID)
);

INSERT INTO Group_Membership (Membership_ID, User_ID, Group_ID)
VALUES
(1, 1, 1),
(2, 2, 2),
(3, 3, 3),
(4, 4, 4),
(5, 5, 5),
(6, 6, 6),
(7, 7, 7),
(8, 8, 8),
(9, 9, 9),
(10, 10, 10);

CREATE TABLE Assessment (
    Assessment_id INT NOT NULL PRIMARY KEY,
    User_ID INT NOT NULL,
    Assessment_date DATE NOT NULL,
    FOREIGN KEY (User_ID) REFERENCES User(User_ID)
);

INSERT INTO Assessment (Assessment_id, User_ID, Assessment_date)
VALUES
(1, 1, '2024-01-15'),
(2, 2, '2024-02-20'),
(3, 3, '2024-03-10'),
(4, 4, '2024-04-05'),
(5, 5, '2024-05-12'),
(6, 6, '2024-06-25'),
(7, 7, '2024-07-08'),
(8, 8, '2024-08-14'),
(9, 9, '2024-09-30'),
(10, 10, '2024-10-22');

CREATE TABLE Score (
    Score_ID INT NOT NULL PRIMARY KEY,
    User_ID INT NOT NULL,
    Assessment_ID INT NOT NULL,
    Score_Value INT NOT NULL,
    Date_Score DATE NOT NULL,
    FOREIGN KEY (User_ID) REFERENCES User(User_ID),
    FOREIGN KEY (Assessment_ID) REFERENCES Assessment(Assessment_id)
);

INSERT INTO Score (Score_ID, User_ID, Assessment_ID, Score_Value, Date_Score)
VALUES
(1, 1, 1, 80, '2024-01-15'),
(2, 2, 2, 75, '2024-02-20'),
(3, 3, 3, 90, '2024-03-10'),
(4, 4, 4, 85, '2024-04-05'),
(5, 5, 5, 70, '2024-05-12'),
(6, 6, 6, 78, '2024-06-25'),
(7, 7, 7, 82, '2024-07-08'),
(8, 8, 8, 88, '2024-08-14'),
(9, 9, 9, 92, '2024-09-30'),
(10, 10, 10, 95, '2024-10-22');

CREATE TABLE Answers (
    Answer_id INT NOT NULL PRIMARY KEY,
    User_ID INT NOT NULL,
    Question_id INT NOT NULL,
    Answer_text VARCHAR(150) NOT NULL,
    FOREIGN KEY (User_ID) REFERENCES User(User_ID)
);

INSERT INTO Answers (Answer_id, User_ID, Question_id, Answer_text)
VALUES
(1, 1, 1, 'Yes'),
(2, 2, 1, 'No'),
(3, 3, 1, 'Sometimes'),
(4, 4, 1, 'Yes'),
(5, 5, 1, 'No'),
(6, 6, 1, 'No'),
(7, 7, 1, 'Yes'),
(8, 8, 1, 'Sometimes'),
(9, 9, 1, 'No'),
(10, 10, 1, 'Yes');

CREATE TABLE Threshold (
    ID INT NOT NULL PRIMARY KEY,
    Assessment_ID INT NOT NULL,
    Description_text VARCHAR(250) NOT NULL,
    Title_text VARCHAR(150) NOT NULL,
    FOREIGN KEY (Assessment_ID) REFERENCES Assessment(Assessment_id)
);

INSERT INTO Threshold (ID, Assessment_ID, Description_text, Title_text)
VALUES
(1, 1, 'Mild Concerns', 'Mental Health Score'),
(2, 2, 'Moderate Concerns', 'Mental Health Score'),
(3, 3, 'Severe Concerns', 'Mental Health Score'),
(4, 4, 'Mild Concerns', 'Mental Health Score'),
(5, 5, 'Moderate Concerns', 'Mental Health Score'),
(6, 6, 'Mild Concerns', 'Mental Health Score'),
(7, 7, 'Moderate Concerns', 'Mental Health Score'),
(8, 8, 'Severe Concerns', 'Mental Health Score'),
(9, 9, 'Mild Concerns', 'Mental Health Score'),
(10, 10, 'Moderate Concerns', 'Mental Health Score');

CREATE TABLE Form (
    Form_ID INT NOT NULL PRIMARY KEY,
    Form_title VARCHAR(50) NOT NULL,
    Form_Des VARCHAR(130) NOT NULL
);

INSERT INTO Form (Form_ID, Form_title, Form_Des)
VALUES
(1, 'Mental Health Assessment Form', 'Assessment form for mental health evaluation'),
(2, 'Feedback Form', 'Form for collecting user feedback'),
(3, 'Employee Satisfaction Survey', 'Survey to gauge employee satisfaction'),
(4, 'Training Feedback Form', 'Form to gather feedback on training programs'),
(5, 'Customer Satisfaction Survey', 'Survey to measure customer satisfaction');

CREATE TABLE Reminder (
    Log_ID INT NOT NULL PRIMARY KEY,
    User_ID INT NOT NULL,
    Reminder_Text VARCHAR(250) NOT NULL,
    Reminder_Date DATE NOT NULL,
    Notes_Text VARCHAR(250) NOT NULL,
    FOREIGN KEY (User_ID) REFERENCES User(User_ID)
);

INSERT INTO Reminder (Log_ID, User_ID, Reminder_Text, Reminder_Date, Notes_Text)
VALUES
(1, 1, 'Appointment with therapist', '2024-04-10', 'Discuss recent stress triggers'),
(2, 2, 'Complete feedback form', '2024-03-25', 'Provide detailed feedback on services'),
(3, 3, 'Follow up on assessment results', '2024-03-15', 'Review suggestions for improvement'),
(4, 4, 'Attend team meeting', '2024-04-01', 'Discuss project progress'),
(5, 5, 'Submit expense report', '2024-05-20', 'Include all receipts and details'),
(6, 6, 'Review performance goals', '2024-06-30', 'Prepare for quarterly review'),
(7, 7, 'Submit sales report', '2024-07-15', 'Include sales figures and analysis'),
(8, 8, 'Marketing campaign brainstorming', '2024-08-05', 'Discuss ideas for upcoming campaign'),
(9, 9, 'Financial audit preparation', '2024-09-25', 'Gather financial documents for audit'),
(10, 10, 'HR policy review', '2024-10-15', 'Review and update HR policies');

CREATE TABLE Contact (
    Contact_ID INT NOT NULL PRIMARY KEY,
    User_ID INT NOT NULL,
    Contact_Name TEXT NOT NULL,
    Phone INT NOT NULL,
    FOREIGN KEY (User_ID) REFERENCES User(User_ID)
);

INSERT INTO Contact (Contact_ID, User_ID, Contact_Name, Phone)
VALUES
(1, 1, 'Therapist Office', 0000000001),
(2, 2, 'Customer Support', 0000000002),
(3, 3, 'Manager', 0000000003),
(4, 4, 'Team Lead', 0000000004),
(5, 5, 'Finance Department', 0000000005),
(6, 6, 'Support Desk', 000000006),
(7, 7, 'Sales Manager', 0000000007),
(8, 8, 'Marketing Team', 0000000008),
(9, 9, 'Finance Director', 0000000009),
(10, 10, 'HR Manager', 0000000010);


CREATE TABLE Expense (
   Expense_ID INT NOT NULL PRIMARY KEY,
   User_ID INT NOT NULL,
   Date_text DATE NOT NULL,
   Amount_Int INT NOT NULL,
   FOREIGN KEY (User_ID) REFERENCES User(User_ID)
);

INSERT INTO Expense (Expense_ID, User_ID, Date_text, Amount_Int)
VALUES
(1, 1, '2024-03-10', 100),
(2, 2, '2024-02-15', 75),
(3, 3, '2024-01-20', 150),
(4, 4, '2024-04-05', 200),
(5, 5, '2024-05-12', 50),
(6, 6, '2024-06-18', 120),
(7, 7, '2024-07-30', 180),
(8, 8, '2024-08-10', 90),
(9, 9, '2024-09-05', 250),
(10, 10, '2024-10-01', 150);

CREATE TABLE ActivityLog (
   Log_ID INT NOT NULL PRIMARY KEY,
   User_ID INT NOT NULL,
   Activity_type_text VARCHAR(250) NOT NULL,
   Duration_int INT NOT NULL,
   FOREIGN KEY (User_ID) REFERENCES User(User_ID)
);

INSERT INTO ActivityLog (Log_ID, User_ID, Activity_type_text, Duration_int)
VALUES
(1, 1, 'Exercise', 60),
(2, 2, 'Reading', 30),
(3, 3, 'Meeting', 120),
(4, 4, 'Coding', 180),
(5, 5, 'Walking', 45),
(6, 6, 'Training', 90),
(7, 7, 'Sales Call', 45),
(8, 8, 'Marketing Campaign', 240),
(9, 9, 'Financial Analysis', 150),
(10, 10, 'HR Training', 120);
```

Questions:

1) Retrieve the names of users who do not have any associated contacts.

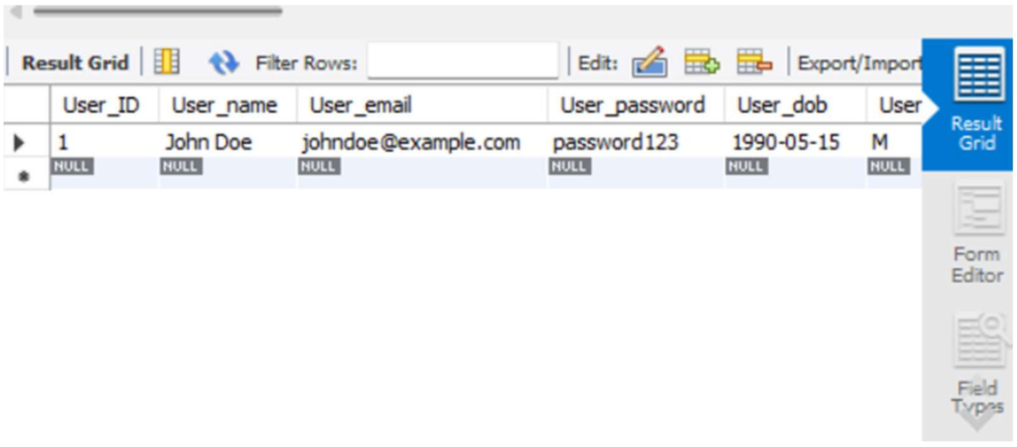
```sql
SELECT * FROM User WHERE User_name = 'John Doe';

SELECT * FROM User WHERE User_name != 'John Doe';

SELECT * FROM Score WHERE Score_Value > 80;

SELECT * FROM Score WHERE Score_Value < 80;

SELECT * FROM Score WHERE Score_Value >= 80;

SELECT * FROM Score WHERE Score_Value <= 80;

SELECT * FROM User WHERE User_gender IN ('M', 'F');

SELECT * FROM User WHERE User_gender = 'M' AND User_dob < '1990-01-01';

SELECT * FROM User WHERE User_gender = 'F' OR User_dob >= '1990-01-01';

SELECT * FROM User;

INSERT INTO User (User_ID, User_name, User_password, User_dob, User_gender)
VALUES (11, 'Jessica Johnson', 'jess@123', '1991-04-20', 'F');

SELECT * FROM Score ORDER BY Score_Value DESC;

SELECT u.User_name, g.Group_name
FROM User u
JOIN Group_Membership gm ON u.User_ID = gm.User_ID
JOIN User_Group g ON gm.Group_ID = g.Group_ID;

SELECT u.User_name, MIN(s.Score_Value) AS Min_Score, MAX(s.Score_Value) AS Max_Score
FROM User u
JOIN Score s ON u.User_ID = s.User_ID
GROUP BY u.User_name;


SELECT User_name FROM User
UNION
SELECT Contact_Name FROM Contact;

SELECT User_name FROM User
UNION ALL
SELECT Contact_Name FROM Contact;


SELECT u.User_name, c.Contact_Name
FROM User u
INNER JOIN Contact c ON u.User_ID = c.User_ID;

SELECT u.User_name, c.Contact_Name
FROM User u
INNER JOIN Contact c ON u.User_ID = c.User_ID;
```

```
SELECT u.User_name, c.Contact_Name
FROM User u
INNER JOIN Contact c ON u.User_ID = c.User_ID;
```

```
SELECT u.User_name, c.Contact_Name
FROM User u
INNER JOIN Contact c ON u.User_ID = c.User_ID;
```

Retrieve the user details for the user with the username 'John Doe'.

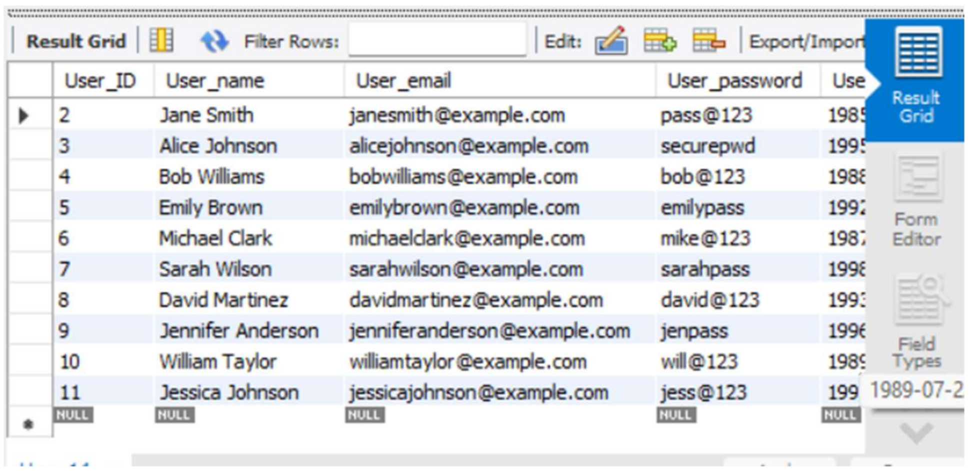
```
SELECT * FROM User WHERE User_name = 'John Doe';
```



	User_ID	User_name	User_email	User_password	User_dob	User
▶	1	John Doe	johndoe@example.com	password123	1990-05-15	M
*	NULL	NULL	NULL	NULL	NULL	NULL

Fetch all user details except for the user with the username 'John Doe'.

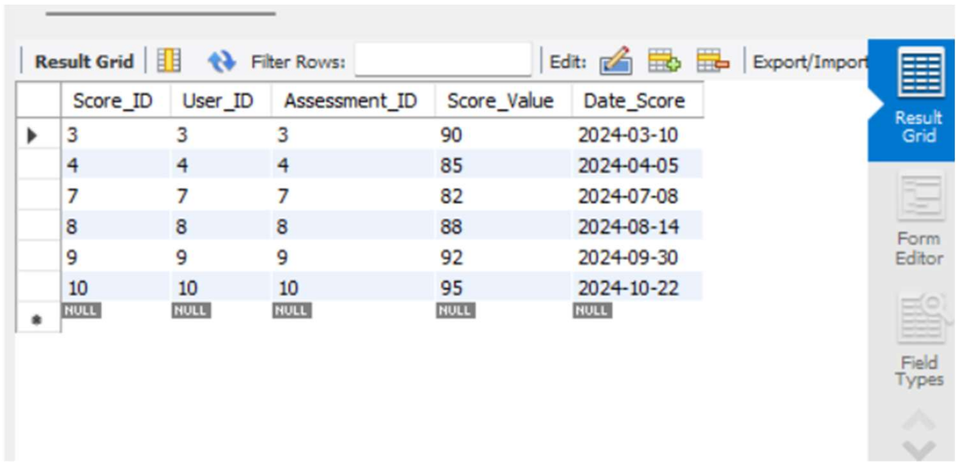
```
SELECT * FROM User WHERE User_name != 'John Doe';
```



	User_ID	User_name	User_email	User_password	Use
▶	2	Jane Smith	janesmith@example.com	pass@123	1985
	3	Alice Johnson	alicejohnson@example.com	securepwd	1995
	4	Bob Williams	bobwilliams@example.com	bob@123	1988
	5	Emily Brown	emilybrown@example.com	emilypass	1992
	6	Michael Clark	michaelclark@example.com	mike@123	1987
	7	Sarah Wilson	sarahwilson@example.com	sarahpass	1998
	8	David Martinez	davidmartinez@example.com	david@123	1993
	9	Jennifer Anderson	jenniferanderson@example.com	jenpass	1996
	10	William Taylor	williamtaylor@example.com	will@123	1989
	11	Jessica Johnson	jessicajohnson@example.com	jess@123	199 1989-07-2
*	NULL	NULL	NULL	NULL	NULL

Retrieve all scores where the score value is greater than 80.

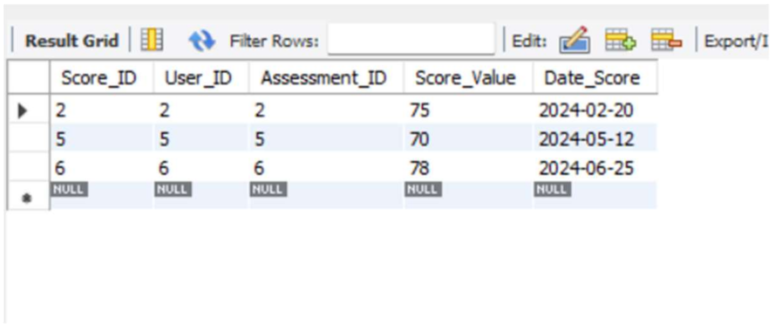
SELECT * FROM Score WHERE Score_Value > 80;



Score_ID	User_ID	Assessment_ID	Score_Value	Date_Score
3	3	3	90	2024-03-10
4	4	4	85	2024-04-05
7	7	7	82	2024-07-08
8	8	8	88	2024-08-14
9	9	9	92	2024-09-30
10	10	10	95	2024-10-22
NULL	NULL	NULL	NULL	NULL

Fetch all scores where the score value is less than 80.

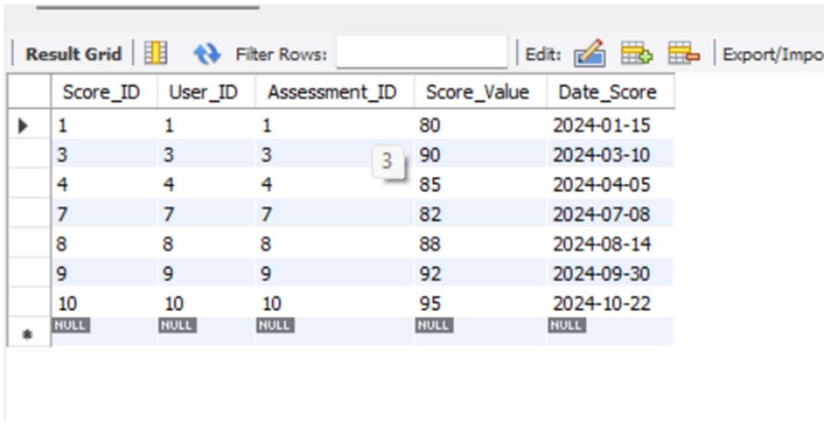
SELECT * FROM Score WHERE Score_Value < 80;



Score_ID	User_ID	Assessment_ID	Score_Value	Date_Score
2	2	2	75	2024-02-20
5	5	5	70	2024-05-12
6	6	6	78	2024-06-25
NULL	NULL	NULL	NULL	NULL

Get all scores where the score value is greater than or equal to 80.

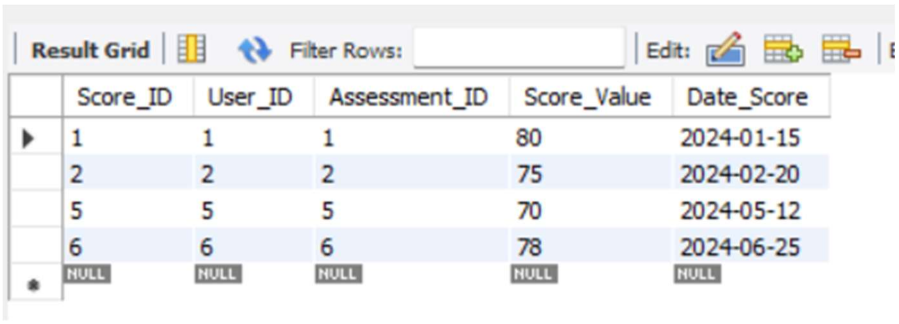
SELECT * FROM Score WHERE Score_Value >= 80;



Score_ID	User_ID	Assessment_ID	Score_Value	Date_Score
1	1	1	80	2024-01-15
3	3	3	90	2024-03-10
4	4	4	85	2024-04-05
7	7	7	82	2024-07-08
8	8	8	88	2024-08-14
9	9	9	92	2024-09-30
10	10	10	95	2024-10-22
NULL	NULL	NULL	NULL	NULL

Retrieve scores where the score value is less than or equal to 80.

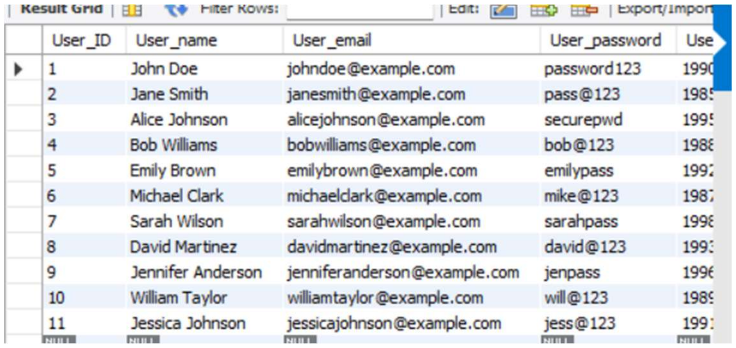
SELECT * FROM Score WHERE Score_Value <= 80;



	Score_ID	User_ID	Assessment_ID	Score_Value	Date_Score
▶	1	1	1	80	2024-01-15
	2	2	2	75	2024-02-20
	5	5	5	70	2024-05-12
	6	6	6	78	2024-06-25
*	NULL	NULL	NULL	NULL	NULL

Retrieve user details for users with genders either Male or Female.

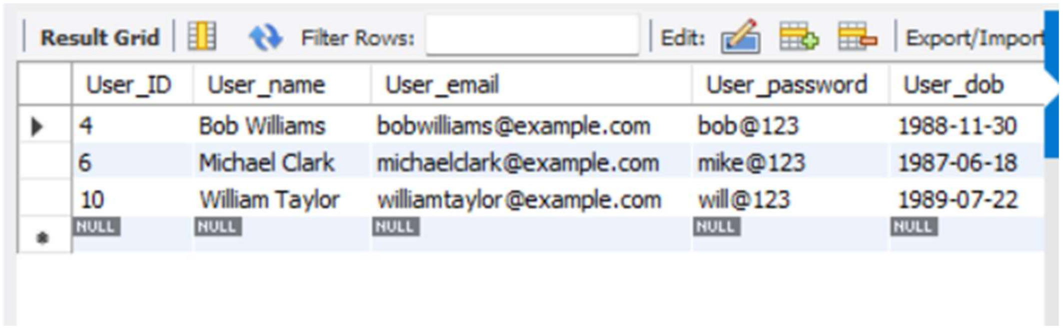
SELECT * FROM User WHERE User_gender IN ('M', 'F');



	User_ID	User_name	User_email	User_password	Use
▶	1	John Doe	johndoe@example.com	password123	1990
	2	Jane Smith	janesmith@example.com	pass@123	198
	3	Alice Johnson	alicejohnson@example.com	securepwd	199
	4	Bob Williams	bobwilliams@example.com	bob@123	198
	5	Emily Brown	emilybrown@example.com	emilypass	199
	6	Michael Clark	michaelclark@example.com	mike@123	198
	7	Sarah Wilson	sarahwilson@example.com	sarahpass	199
	8	David Martinez	davidmartinez@example.com	david@123	199
	9	Jennifer Anderson	jenniferanderson@example.com	jenpass	199
	10	William Taylor	williamtaylor@example.com	will@123	198
	11	Jessica Johnson	jessicajohnson@example.com	jess@123	199

Fetch user details for male users born before January 1, 1990.

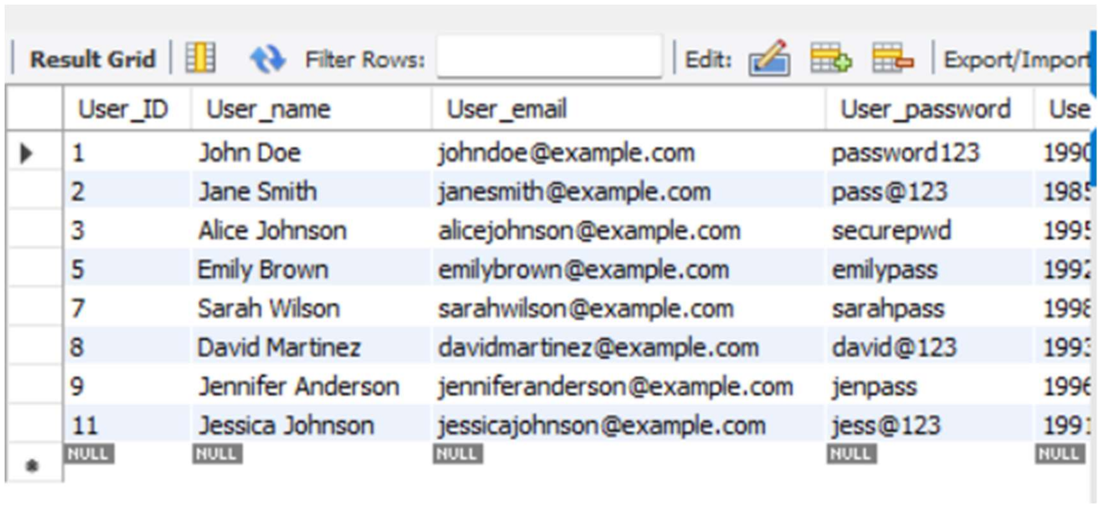
SELECT * FROM User WHERE User_gender = 'M' AND User_dob < '1990-01-01';



	User_ID	User_name	User_email	User_password	User_dob
▶	4	Bob Williams	bobwilliams@example.com	bob@123	1988-11-30
	6	Michael Clark	michaelclark@example.com	mike@123	1987-06-18
	10	William Taylor	williamtaylor@example.com	will@123	1989-07-22
*	NULL	NULL	NULL	NULL	NULL

Retrieve user details for female users or users born on or after January 1, 1990.

SELECT * FROM User WHERE User_gender = 'F' OR User_dob >= '1990-01-01';

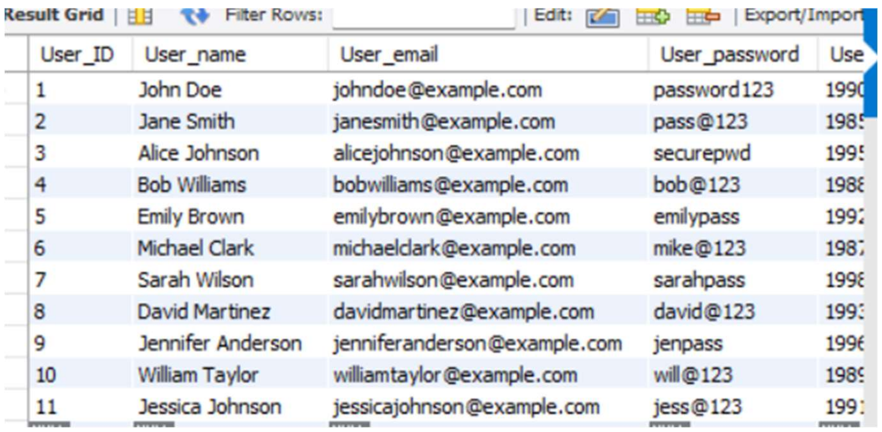


Result Grid | Filter Rows: | Edit: | Export/Import

	User_ID	User_name	User_email	User_password	Use
▶	1	John Doe	johndoe@example.com	password123	1990
	2	Jane Smith	janesmith@example.com	pass@123	1985
	3	Alice Johnson	alicejohnson@example.com	securepwd	1995
	5	Emily Brown	emilybrown@example.com	emilypass	1992
	7	Sarah Wilson	sarahwilson@example.com	sarahpass	1998
	8	David Martinez	davidmartinez@example.com	david@123	1993
	9	Jennifer Anderson	jenniferanderson@example.com	jenpass	1996
	11	Jessica Johnson	jessicajohnson@example.com	jess@123	1991
*	NULL	NULL	NULL	NULL	NULL

Fetch all user details from the database.

SELECT * FROM User;



Result Grid | Filter Rows: | Edit: | Export/Import

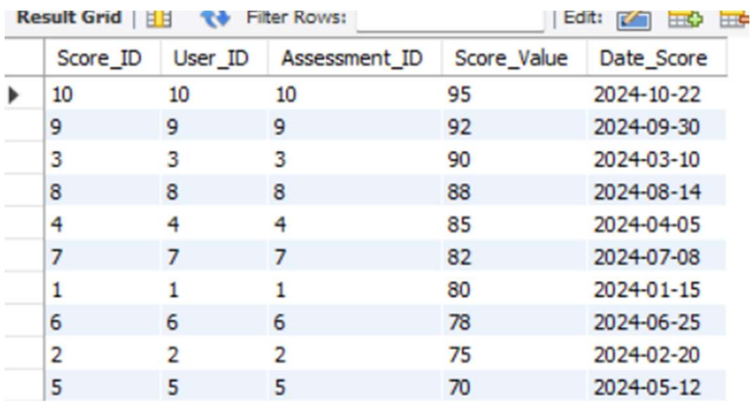
User_ID	User_name	User_email	User_password	Use
1	John Doe	johndoe@example.com	password123	1990
2	Jane Smith	janesmith@example.com	pass@123	1985
3	Alice Johnson	alicejohnson@example.com	securepwd	1995
4	Bob Williams	bobwilliams@example.com	bob@123	1988
5	Emily Brown	emilybrown@example.com	emilypass	1992
6	Michael Clark	michaelclark@example.com	mike@123	1987
7	Sarah Wilson	sarahwilson@example.com	sarahpass	1998
8	David Martinez	davidmartinez@example.com	david@123	1993
9	Jennifer Anderson	jenniferanderson@example.com	jenpass	1996
10	William Taylor	williamtaylor@example.com	will@123	1989
11	Jessica Johnson	jessicajohnson@example.com	jess@123	1991

Insert a new user record for Jessica Johnson with her email, password, date of birth, and gender.

INSERT INTO User (User_ID, User_name, User_email, User_password, User_dob, User_gender)
VALUES (11, 'Jessica Johnson', 'jessicajohnson@example.com', 'jess@123', '1991-04-20', 'F');

Retrieve all scores from the database and sort them in descending order based on the score value.

SELECT * FROM Score ORDER BY Score_Value DESC;

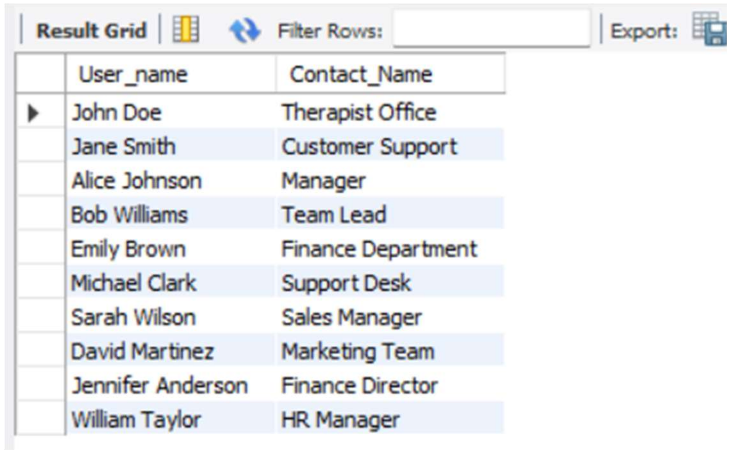


Result Grid | Filter Rows: | Edit:

Score_ID	User_ID	Assessment_ID	Score_Value	Date_Score
10	10	10	95	2024-10-22
9	9	9	92	2024-09-30
3	3	3	90	2024-03-10
8	8	8	88	2024-08-14
4	4	4	85	2024-04-05
7	7	7	82	2024-07-08
1	1	1	80	2024-01-15
6	6	6	78	2024-06-25
2	2	2	75	2024-02-20
5	5	5	70	2024-05-12

Retrieve the names of users and their corresponding contact names.

```
SELECT u.User_name, c.Contact_Name
FROM User u
INNER JOIN Contact c ON u.User_ID = c.User_ID;
```

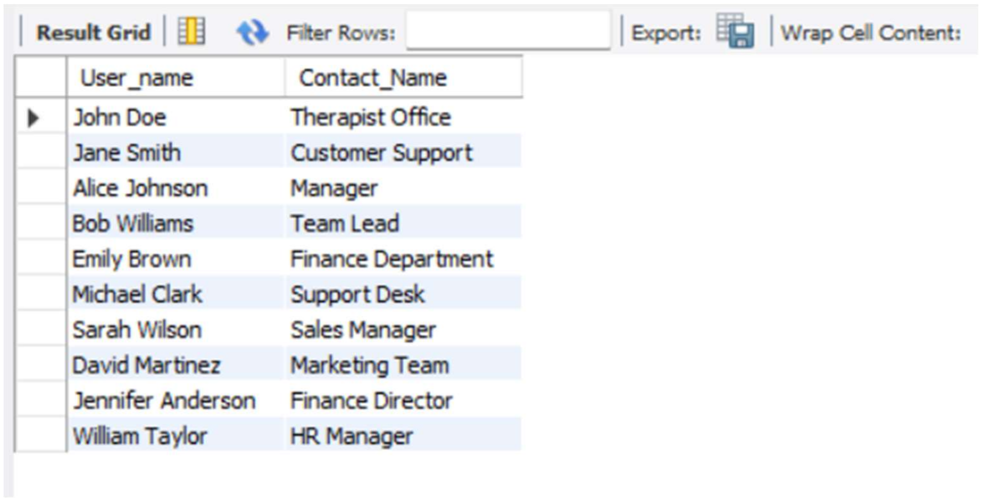


Result Grid | Filter Rows: | Export:

User_name	Contact_Name
John Doe	Therapist Office
Jane Smith	Customer Support
Alice Johnson	Manager
Bob Williams	Team Lead
Emily Brown	Finance Department
Michael Clark	Support Desk
Sarah Wilson	Sales Manager
David Martinez	Marketing Team
Jennifer Anderson	Finance Director
William Taylor	HR Manager

Fetch the user names and their associated contact names through an inner join between the User and Contact tables.

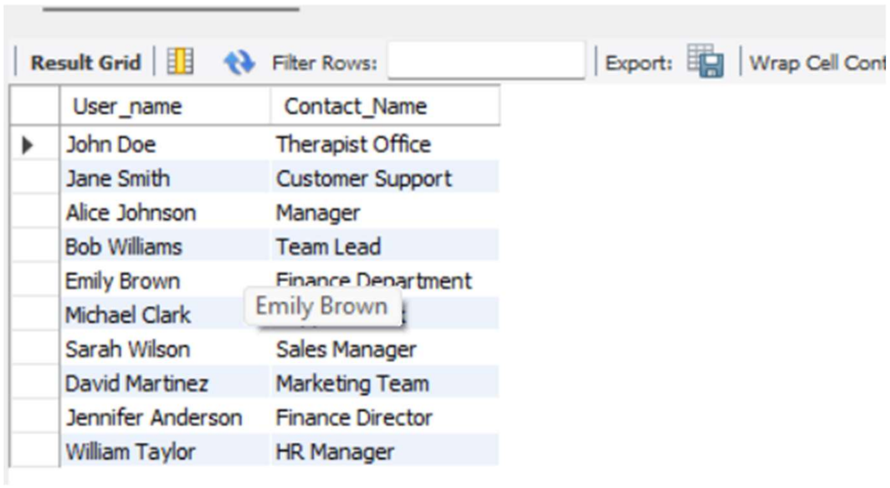
```
SELECT u.User_name, c.Contact_Name
FROM User u
INNER JOIN Contact c ON u.User_ID = c.User_ID;
```


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

User_name	Contact_Name
John Doe	Therapist Office
Jane Smith	Customer Support
Alice Johnson	Manager
Bob Williams	Team Lead
Emily Brown	Finance Department
Michael Clark	Support Desk
Sarah Wilson	Sales Manager
David Martinez	Marketing Team
Jennifer Anderson	Finance Director
William Taylor	HR Manager

Retrieve the user names and their corresponding contact names by performing an inner join operation between the User and Contact tables.

```
SELECT u.User_name, c.Contact_Name
FROM User u
INNER JOIN Contact c ON u.User_ID = c.User_ID;
```

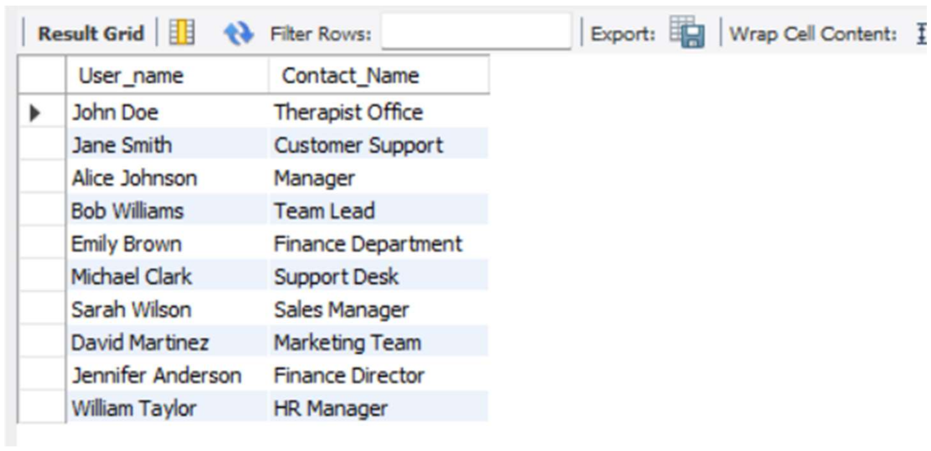

Result Grid | Filter Rows: | Export: | Wrap Cell Cont

User_name	Contact_Name
John Doe	Therapist Office
Jane Smith	Customer Support
Alice Johnson	Manager
Bob Williams	Team Lead
Emily Brown	Finance Department
Michael Clark	[illegible]
Sarah Wilson	Sales Manager
David Martinez	Marketing Team
Jennifer Anderson	Finance Director
William Taylor	HR Manager

Emily Brown

Fetch the names of users and their associated contact names using an inner join between the User and Contact tables.

```
SELECT u.User_name, c.Contact_Name
FROM User u
INNER JOIN Contact c ON u.User_ID = c.User_ID;
```



User_name	Contact_Name
John Doe	Therapist Office
Jane Smith	Customer Support
Alice Johnson	Manager
Bob Williams	Team Lead
Emily Brown	Finance Department
Michael Clark	Support Desk
Sarah Wilson	Sales Manager
David Martinez	Marketing Team
Jennifer Anderson	Finance Director
William Taylor	HR Manager

VI. Project demonstration

For our project demonstration on mental health score calculation, we utilized a combination of MySQL for writing queries and Draw.io application for creating an E-R (Entity-Relationship) Model and Relational Model. Here's an expanded overview of how we employed these tools:

1. **MySQL for Querying:** We leveraged MySQL, a widely-used relational database management system, to handle the storage and querying of data related to mental health assessments. MySQL provided us with a robust platform to store structured data efficiently and to execute complex queries to retrieve, manipulate, and analyze this data.
    - We designed a database schema tailored to our specific requirements, including tables for storing user information, assessment scores, responses, and other relevant data points.
    - Through MySQL, we were able to write SQL queries to calculate mental health scores based on various parameters and factors. These queries allowed us to aggregate, analyze, and derive meaningful insights from the collected data.
2. **Draw.io for E-R Model and Relational Model:** Draw.io served as a valuable tool for visualizing the structure and relationships within our database through the creation of both E-R and Relational Models.
    - **Entity-Relationship (E-R) Model:** Using Draw.io, we developed an E-R diagram to represent the entities involved in our mental health assessment system, along with their attributes and relationships. This provided a clear visualization of the data model and helped in understanding the conceptual framework of our database design.
    - **Relational Model:** In addition to the E-R diagram, Draw.io allowed us to construct a Relational Model that outlined the tables, attributes, and relationships in a more structured format. This model served as a blueprint for implementing the database schema in MySQL, ensuring consistency and coherence between the conceptual design and its practical implementation.

By combining the capabilities of MySQL for data management and querying with Draw.io for visual modelling, we were able to develop a comprehensive solution for mental health score calculation. This integrated approach facilitated efficient database design, data manipulation, and visualization, ultimately contributing to the success of our project demonstration.

VII. Self -Learning beyond classroom

Engaging with the mental health database this provided a learning experience in managing query and entries in the database. This activity has given me insights into principles of database design, such as organizing tables to accurately represent entities and their relationships. This provided an insight in how the database of mental health score is calculated and how the improvement steps are decided. This also provided an insight in understanding various concepts of mental health which will be taken into consideration in questions for score calculation.

VIII. Learning from the Project

This project has helped me gain the following concepts: -
Database Design and Normalization: Through this project, you likely gained a deeper understanding of database design principles, including entity-relationship modeling, normalization (1NF, 2NF, 3NF, and BCNF), primary keys, foreign keys, and ensuring data integrity and efficiency.

SQL Queries: By implementing SQL queries for creating tables, inserting data, querying information, and performing joins, you likely improved your skills in structuring and retrieving data from databases.

Entity Relationships: Designing and managing entity relationships such as users, groups, assessments, scores, answers, forms, reminders, contacts, expenses, and activity logs would have provided you with hands-on experience in defining complex relationships in a database system.

Project Management: Coordinating various aspects of the project, including storyline development, character roles, database schema design, SQL query implementation, and normalization, would have honed your project management and organizational skills.

Domain Knowledge: Working on a project related to mental health services likely enhanced your understanding of mental health assessment processes, support services, data analysis in healthcare settings, and the importance of data accuracy and security in sensitive domains.

IX. Challenges Faced

Here are some challenges faced:-

1. **Data Privacy and Security:** Ensuring the confidentiality and security of sensitive client information posed a significant challenge. Implementing robust data encryption, access control measures, and compliance with data protection regulations required meticulous planning and execution.
2. **Database Complexity:** Managing a complex database structure with interrelated tables and dependencies presented challenges in terms of data integrity, normalization, and query optimization. Developing efficient database schemas and query optimization strategies were key areas of focus.
3. **Redundancy:** Managing redundancies if any .

X. Conclusion

The project on Mental Health Services (MHS) organization has been a comprehensive learning journey that has significantly enhanced my skills and knowledge in various domains. From designing a robust database schema to implementing SQL queries, managing entity relationships, and understanding the intricacies of mental health services, this project has provided invaluable insights and practical experience.

Through this project, I have gained proficiency in database design principles, normalization techniques, and SQL querying, which are foundational skills in the field of data management. Moreover, working with a domain-specific context like mental health services has deepened my understanding of healthcare processes, data privacy considerations, and the importance of data accuracy and integrity in sensitive domains.

Furthermore, the project has honed my project management skills by requiring careful planning, coordination, and execution of tasks such as defining the storyline, assigning character roles, and ensuring the functionality and security of the database system.

Overall, this project has not only equipped me with technical skills but also provided a valuable opportunity to apply these skills in a real-world scenario, thereby preparing me for future endeavors in database management, software development, and healthcare-related projects.