BUDGET TRACKER

(A Personal budget tracking system)

DDL CREATION:

1.Users Table

Purpose: This table stores information about the users of the expense tracker application. It includes basic personal information, authentication details.

Columns:

- **userID:** A unique identifier for each user. This is usually an auto-incremented integer.
- **userName:** The user's chosen username. This must be unique across the system.
- **email:** The user's email address. Also unique and used for login and notifications.
- **password:** the password user set when the user account was created.
- **created_at:** Timestamp when the user account was created.
- **updated_at:** Timestamp of the last update to the user's profile.

DDL Query:

```
CREATE TABLE Users (
userID INT IDENTITY(1,1) PRIMARY KEY,
userName VARCHAR(100) NOT NULL,
email VARCHAR(100) NOT NULL,
[password] VARCHAR(50) NOT NULL,
created_at DATE NOT NULL,
updated_at DATE NOT NULL
)
```

Inserting data into table:

```
INSERT INTO Users (userName, email, [password], created_at, updated_at)
VALUES
    ('Sownthari', 'sownthari.p@payoda.com', 'password123', GETDATE(), GETDATE()),
    ('Siddarth', 'siddarth.s@payoda.com', 's@123', GETDATE(), GETDATE()),
    ('Hemasri', 'hemasri.m@payoda.com', 'pass23!', GETDATE(), GETDATE()),
    ('Prathik', 'prathik.b@payoda.com', 'password$123', GETDATE(), GETDATE()),
    ('Dharshini', 'dharshini.j@payoda.com', 'd@8967', GETDATE(), GETDATE());
```

2. Categories Table

Purpose: This table organizes expenses into categories. Users can define their own categories to classify transactions, helping in budgeting and reporting.

Columns:

- **categoryID:** A unique identifier for each category.
- **userID:** A foreign key linking to the Users table, indicating which user owns the category.
- **categoryName:** The name of the category (e.g., "Groceries", "Utilities").
- **description:** A brief description of the category's purpose or use.

DDL Query:

```
CREATE TABLE Categories (
categoryID INT IDENTITY(1,1) PRIMARY KEY,
userID INT NOT NULL,
categoryName VARCHAR(100) NOT NULL,
[description] VARCHAR(max),
FOREIGN KEY (userID) REFERENCES Users(userID)
)
```

Inserting data into table:

```
INSERT INTO Categories (userID, categoryName, [description])
VALUES
    (1, 'Groceries', 'Expenses for food and household items'),
    (1, 'Rent', 'Monthly apartment rent payment'),
    (2, 'Utilities', 'Electricity, water, and other utilities'),
    (2, 'Entertainment', 'Movies, concerts, and other leisure activities'),
    (3, 'Transportation', 'Expenses for public transit, fuel, etc.'),
    (3, 'Health', 'Medical bills and health insurance payments');
```

3. Transactions Table

Purpose: This table logs all financial transactions entered by the users. It tracks both income and expenses, categorized under the user-defined categories.

Columns:

- **transactionID:** A unique identifier for each transaction.
- **userID:** A foreign key linking to the Users table, indicating which user made the transaction.
- **categoryID:** A foreign key linking to the Categories table, categorizing the transaction.
- **amount:** The monetary value of the transaction. Negative values can indicate expenses, while positive values can indicate income.
- **transactionDate:** The date on which the transaction occurred.
- **description:** A description or note about the transaction.
- **type:** An indicator of whether the transaction is an "Income" or an "Expense".

DDL Query:

```
CREATE TABLE Transcation (
transcationID INT IDENTITY(1,1) PRIMARY KEY,
userID INT NOT NULL,
categoryID INT NOT NULL,
amount DECIMAL(10, 2) NOT NULL,
transcationDate DATE NOT NULL,
transcationType VARCHAR(10) NOT NULL,
[description] VARCHAR(max),
FOREIGN KEY (userID) REFERENCES Users(userID),
FOREIGN KEY (categoryID) REFERENCES Categories(categoryID)
)
```

Inserting data into table:

```
INSERT INTO Transcation (userID, categoryID, amount, transcationDate, transcationType, [description])
VALUES
    (1, 1, 150.00, '2024-07-20', 'Expense', 'Grocery shopping at local store'),
    (2, 2, 800.00, '2024-07-21', 'Expense', 'Monthly rent payment'),
    (3, 3, 120.00, '2024-07-22', 'Expense', 'Utility bills for electricity and water'),
    (4, 4, 250.00, '2024-07-23', 'Expense', 'New kitchen appliances for home improvement'),
    (5, 5, 500.00, '2024-07-24', 'Income', 'Freelance project payment'),
    (1, 6, 75.00, '2024-07-25', 'Expense', 'Dining out with friends'),
    (2, 7, 200.00, '2024-07-26', 'Expense', 'Clothing shopping for summer season'),
    (3, 8, 300.00, '2024-07-27', 'Income', 'Part-time job salary'),
    (4, 9, 45.00, '2024-07-28', 'Expense', 'Gifts for a friends birthday'),
    (5, 10, 100.00, '2024-07-29', 'Expense', 'Pet supplies and vet visit');
```

4. Budgets Table

Purpose: This table helps users set and manage budgets for different categories. It tracks the planned and actual spending or income over a specified period.

Columns:

- **budgetID:** A unique identifier for each budget entry.
- **userID:** A foreign key linking to the Users table, indicating the owner of the budget.
- **categoryID:** A foreign key linking to the Categories table, specifying which category the budget applies to.
- **amount:** The total amount allocated for the budget period.

- **recurrence:** Specifies the period of the budget, such as "Monthly", or "Yearly".
- **description:** Additional information or notes about the budget.
- **expense**: The total amount spent in the budget category during the budget period.
- **isNotified**: A boolean or integer field indicating whether the user has been notified about reaching or exceeding the budget limit.

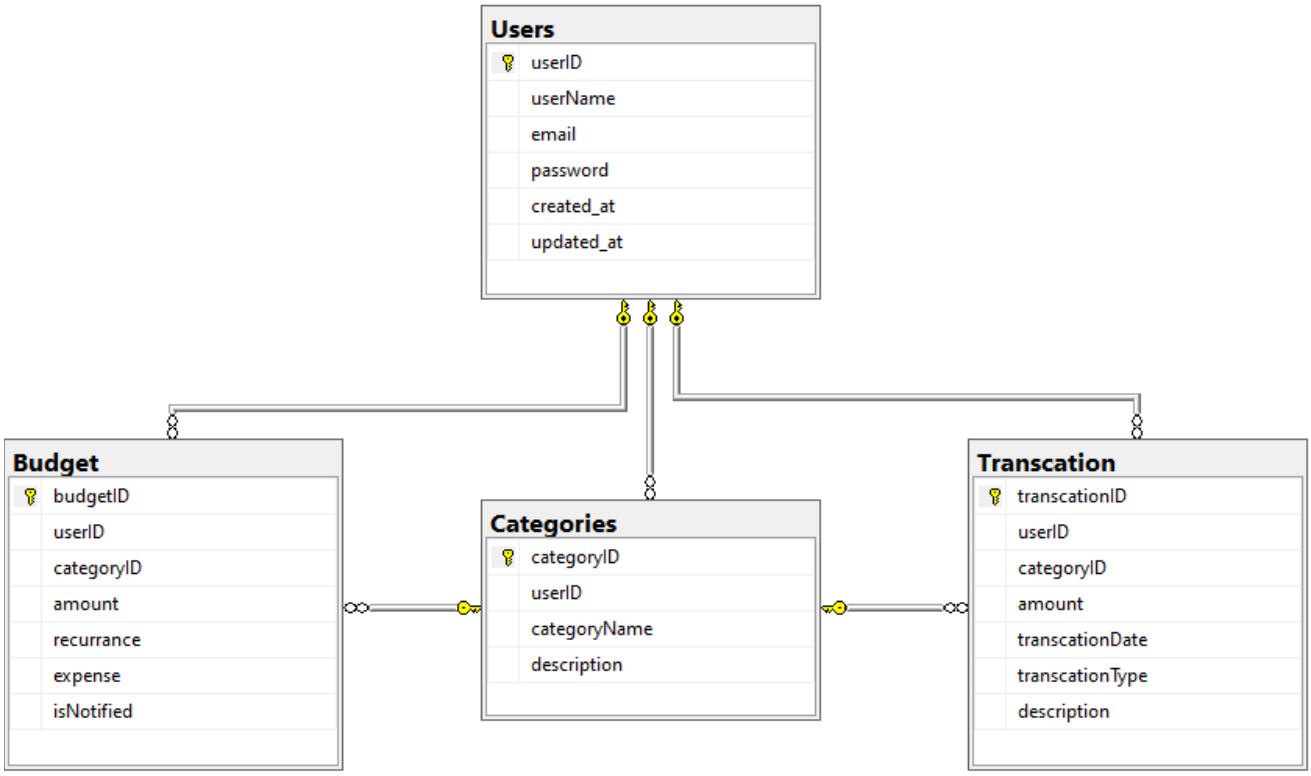
DDL Query:

```
CREATE TABLE Budget (
budgetID INT IDENTITY(1,1) PRIMARY KEY,
userID INT NOT NULL,
categoryID INT NOT NULL,
amount DECIMAL(10, 2) NOT NULL,
recurrance VARCHAR(50) NOT NULL,
expense INT NOT NULL,
isNotified BIT NOT NULL,
FOREIGN KEY (userID) REFERENCES Users(userID),
FOREIGN KEY (categoryID) REFERENCES Categories(categoryID)
)
```

Inserting data into table:

```
INSERT INTO Budget (userID, categoryID, amount, recurrance, expense, isNotified)
VALUES
    (1, 1, 500.00, 'Monthly', 450.00, 0),
    (2, 2, 1000.00, 'Monthly', 800.00, 1),
    (3, 3, 200.00, 'Monthly', 180.00, 0),
    (4, 4, 300.00, 'Monthly', 250.00, 1),
    (5, 5, 1500.00, 'Yearly', 1200.00, 1),
    (1, 6, 200.00, 'Monthly', 180.00, 0),
    (2, 7, 400.00, 'Monthly', 350.00, 1),
    (3, 8, 600.00, 'Yearly', 550.00, 1),
    (4, 9, 100.00, 'Monthly', 90.00, 0),
    (5, 10, 250.00, 'Monthly', 230.00, 0);
```

ER DIAGRAM:



DATA EXTRACTION:

1.Users Table:

Query: SELECT * FROM Users;

Result:

	userID	userName	email	password	created_at	updated_at
1	1	Sownthari	sownthari.p@payoda.com	password123	2024-07-31	2024-07-31
2	2	Siddarth	siddarth.s@payoda.com	s@123	2024-07-31	2024-07-31
3	3	Hemasri	hemasri.m@payoda.com	pass23!	2024-07-31	2024-07-31
4	4	Prathik	prathik.b@payoda.com	password$123	2024-07-31	2024-07-31
5	5	Dharshini	dharshini.j@payoda.com	d@8967	2024-07-31	2024-07-31

2.Transactions after July 29, 2023 and userID is 2

Query: SELECT * FROM Transcation WHERE transcationDate > '2024-07-26' and userID = 2;

Result:

Results | Messages

	transcationID	userID	categoryID	amount	transcationDate	transcationType	description
1	22	2	9	50.00	2024-07-30	Expense	Gift for a friend wedding
2	27	2	3	60.00	2024-08-02	Expense	Utility bill for electricity

3.Retrieve user names and their transaction IDs

Query:

```
SELECT
    Users.userName,
    STRING_AGG(CAST(Transcation.transcationID AS VARCHAR), ', ') AS transcationIDs
FROM Users
INNER JOIN Transcation ON Users.userID = Transcation.userID
GROUP BY Users.userName;
```

Result:

Results | Messages

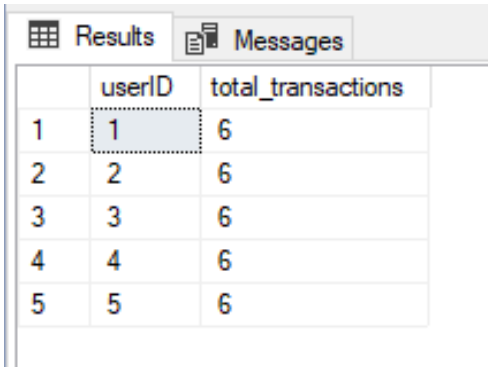
	userName	transcationIDs
1	Dharshini	5, 10, 18, 19, 25, 30
2	Hemasri	3, 8, 14, 15, 23, 28
3	Prathik	4, 9, 16, 17, 24, 29
4	Siddarth	2, 7, 12, 13, 22, 27
5	Sownthari	1, 6, 11, 20, 21, 26

4.Total number of transactions for each user

Query:

```
SELECT userID, COUNT(transcationID) AS total_transactions
FROM Transcation
GROUP BY userID;
```

Result:



Results | Messages

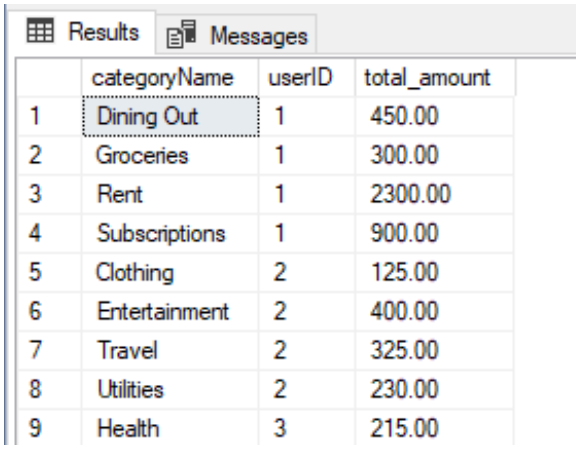
	userID	total_transactions
1	1	6
2	2	6
3	3	6
4	4	6
5	5	6

5.Total transactions for each category

Query:

```
SELECT
    Categories.categoryName,
    Categories.userID,
    SUM(Transcation.amount) AS total_amount
FROM Categories
INNER JOIN Transcation ON Categories.categoryID = Transcation.categoryID
GROUP BY Categories.categoryName, Categories.userID;
```

Result:



Results | Messages

	categoryName	userID	total_amount
1	Dining Out	1	450.00
2	Groceries	1	300.00
3	Rent	1	2300.00
4	Subscriptions	1	900.00
5	Clothing	2	125.00
6	Entertainment	2	400.00
7	Travel	2	325.00
8	Utilities	2	230.00
9	Health	3	215.00