Expense Tracker Project

Introduction

The Expense Tracker project aims to help users manage their finances by tracking their expenses. Efficiently retrieving and manipulating data from the database is crucial for generating meaningful insights and reports. This document demonstrates the techniques used to retrieve and manipulate financial data in the Expense Tracker project, focusing on the SELECT statement, wildcards, comparison operators, the WHERE clause, logical operators, and the ORDER BY clause.

Database Setup

To begin with, we need to create our database and tables. The steps to set up the database in MySQL are as follows:

Creating the Database and Tables

```
Sql code
-- Create the ExpenseTracker database
CREATE DATABASE ExpenseTracker;

-- Use the newly created database
USE ExpenseTracker;

-- Create the expenses table
CREATE TABLE expenses (
    id INT AUTO_INCREMENT PRIMARY KEY,
    description VARCHAR(255) NOT NULL,
    amount DECIMAL(10, 2) NOT NULL,
    date DATE NOT NULL,
    category VARCHAR(100)
);

-- Create the categories table (optional)
CREATE TABLE categories (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL
);
```

Inserting Sample Data

Next, we insert some sample data into our tables to work with.

Inserting Data into Categories Table

```
Sql code
INSERT INTO categories (name) VALUES
('Food'),
('Entertainment'),
('Transport');
```

Inserting Data into Expenses Table

```
Sql code
INSERT INTO expenses (description, amount, date, category) VALUES
('Grocery store', 150.00, '2024-06-25', 'Food'),
('Movie tickets', 40.00, '2024-06-26', 'Entertainment'),
('Dinner', 75.00, '2024-06-26', 'Food'),
('Uber ride', 20.00, '2024-06-27', 'Transport'),
('Grocery store', 80.00, '2024-06-27', 'Food');
```

SQL Queries and Explanations

Here we demonstrate the use of SQL queries to retrieve and manipulate the data.

1. Select Specific Columns

```
Sql code
-- Retrieve amount, date, and category from expenses table
SELECT amount, date, category
FROM expenses;
```

Explanation: This query retrieves the `amount`, `date`, and `category` columns from the `expenses` table.

2. Using Wildcards and Comparison Operators

```
Sql code
-- Find all expenses with descriptions containing the word 'grocery'
SELECT *
FROM expenses
WHERE description LIKE '%grocery%';
```

Explanation: This query finds all expenses where the description contains the word "grocery" using the wildcard `%`.

3. Applying WHERE Clause with Logical Operators

```
Sql code
-- Retrieve records where amount > 100 and category is 'Food'
SELECT *
FROM expenses
WHERE amount > 100 AND category = 'Food';
```

Explanation: This query retrieves records where the amount is greater than 100 and the category is 'Food', using the `AND` logical operator.

4. Organizing Results with ORDER BY

```
Sql code
-- Sort expenses by date in descending order
SELECT amount, date, category
FROM expenses
ORDER BY date DESC;
```

Explanation: This query sorts the expenses by date in descending order using the `ORDER BY` clause.

Results and Explanation

1. Select Specific Columns

Query:

```
Sql code
SELECT amount, date, category
FROM expenses;
```

Expected Result:

amount	date	category
150.00	2024-06-25	Food
40.00	2024-06-26	Entertainment
75.00	2024-06-26	Food
20.00	2024-06-27	Transport
80.00	2024-06-27	Food

Explanation: This query retrieves the amount, date, and category of each expense.

2. Using Wildcards and Comparison Operators

Query:

```
Sql code
SELECT *
FROM expenses
WHERE description LIKE '%grocery%';
```

Expected Result:

id	description	amount	date	category
1	Grocery store	150.00	2024-06-25	Food
5	Grocery store	80.00	2024-06-27	Food

Explanation: This query finds all expenses where the description contains the word "grocery".

3. Applying WHERE Clause with Logical Operators

Query:

```
Sql code
SELECT *
FROM expenses
WHERE amount > 100 AND category = 'Food';
```

Expected Result:

id	description	amount	date	category
1	Grocery store	150.00	2024-06-25	Food

Explanation: This query retrieves records where the amount is greater than 100 and the category is 'Food'.

4. Organizing Results with ORDER BY

Query:

```
Sql code
SELECT amount, date, category
FROM expenses
ORDER BY date DESC;
```

Expected Result:

amount	date	category
20.00	2024-06-27	Transport
80.00	2024-06-27	Food
40.00	2024-06-26	Entertainment
75.00	2024-06-26	Food
150.00	2024-06-25	Food

Explanation: This query sorts the expenses by date in descending order.

Conclusion

Through this exercise, I learned how to use the `SELECT` statement to retrieve specific data, apply wildcards and comparison operators for targeted searches, use the `WHERE` clause with logical operators for effective filtering, and organize data using the `ORDER BY` clause. These SQL techniques are essential for managing and analyzing financial data effectively.