Week 2: Essential Data Retrieval & Filtering (Focus on Expense Tracker Data)

This week, we'll dive deeper into your Expense Tracker project by mastering essential techniques for retrieving and manipulating your financial data! We'll explore the SELECT statement to extract specific information and leverage the power of filtering with the WHERE clause. Finally, we'll learn to organize our results using ORDER BY.

Learning Objectives:

- Utilize the SELECT statement to retrieve data from the Expense Tracker database.
- Apply wildcards (%) and comparison operators for targeted data retrieval.
- Employ the WHERE clause with logical operators (AND, OR, NOT) to filter data effectively.
- Organize retrieved data using the ORDER BY clause.

Instructions
This assignment is designed to be completed in approximately 2 hours.
What you'll need:
Access to a computer with internet access A text editor (e.g. Microsoft Word Docs) Access to a sample Expense Tracker database (or instructor-provided data) that includes the table you designed in Week 1 (likely named "Expenses"). See the database script in this repo titled "Week2.txt" Scenario: Imagine you've diligently tracked your expenses using the Expense Tracker database you designed last week. Now, it's time to analyze your spending habits!

Submission:

1. Open MySQL WorkBench or any SQL Management Tool and Import the file/database
2. Copy the code in week2.txt and run it into MySQL Workbench
3. Write SQL scripts to answer all questions below.
4. Submit your document by uploading your document through onto this repo

Part 1: Retrieving Data with SELECT (30 minutes)
Based on the Expense Tracker table you designed in Week 1, which likely includes columns like "expense_id," "amount," "date," and "category," complete the following tasks:

1.1 Retrieving All Expenses:
Write an SQL query to retrieve all data points (columns) from the "Expenses" table.
Answer: SELECT * FROM Expenses;

1.2 Specific Columns:
Modify your query to select only specific columns relevant to your analysis. For example, you might choose "date," "category," and "amount" to analyze spending patterns by category and date.
Answer: SELECT date,category,amount FROM Expenses;

1.3 Filtering by Date Range: Write a query to retrieve expenses charged between a specific date range (e.g., January 1, 2021, to December 15, 2024). Remember to use the appropriate data type for the "date" column when specifying the date range in your query.
Answer: SELECT * FROM Expenses WHERE date >= '2021-01-01' AND date <= '2024-12-15';

Part 2: Filtering with WHERE Clause (45 minutes)
2.1 Filtering by Category: Write a query to find all expenses belonging to a specific category (e.g., "Entertainment").
Answer: SELECT * FROM Expenses WHERE category LIKE '%Entertainment%';

2.2 Filtering with Comparison Operators: Find expenses with an amount greater than a certain value (e.g., $50).
Answer: SELECT * FROM Expenses WHERE amount > 50;

2.3 Combining Filters (AND):
Refine your query to find expenses that meet multiple criteria. For example, you might search for expenses greater than $75 AND belonging to the "Food" category.
Answer: SELECT * FROM Expenses WHERE amount > 75 AND category LIKE '%Food%';

2.4 Combining Filters (OR): Modify your query to find expenses belonging to one category or another (e.g., "Transportation" OR "Groceries").
Answer: SELECT * FROM Expenses WHERE category LIKE '%Transportation' OR category LIKE '%Groceries%';
2.5 Filtering with NOT: Write a query to display expenses unrelated to a specific category (e.g., "Rent").
Answer: SELECT * FROM Expenses WHERE category NOT LIKE '%Rent%';

Part 3: Sorting Retrieved Data (45 minutes)
3.1 Sorting by Amount: Write a query to display all expenses sorted by amount in a specific order (e.g., descending order for highest to lowest spending).
Answer: SELECT * FROM Expenses ORDER BY amount DESC;

3.2 Sorting by Date and Category:
Modify your query to sort expenses based on multiple columns. For example, you might sort first by date (descending order) and then by category (ascending order) to see recent spending trends by category.
Answer: SELECT * FROM Expenses ORDER BY date DESC, category ASC;

Part 4: Database Upgrade
Imagine you're tasked by the CIO to expand your Expense Tracker database. Practice creating, modifying, and removing a table to manage spending habits.
4.1 Write SQL commands to achieve the following:

- We don't have a table for income yet. Create a table named "Income" with columns for:

income_id (INT) - Primary Key (auto-increment)
amount (DECIMAL(10,2)) - NOT NULL
date (DATE) - NOT NULL
source (VARCHAR(50)) - NOT NULL
Answer: CREATE TABLE IF NOT EXISTS Income(income_id INT PRIMARY KEY AUTO-INCREMENT, amount DECIMAL(10,2) NOT NULL, date DATE NOT NULL);

4.2 After creating the "Income" table, you realize you also want to track the income category "source" (e.g., "Salary," "Freelance Work").

- Use ALTER TABLE to add a new column named "category" of type VARCHAR(50).
  Answer: ALTER TABLE Income ADD source VARCHAR(50);
-

4.3 Let's say you decide tracking the income source isn't necessary for now.

- Use ALTER TABLE again to remove the "source" column from the "Income" table.
  Answer: ALTER TABLE Income DROP COLUMN source;

Imagine you no longer need the "Income" table entirely. Experiment how to Use DROP TABLE to permanently remove it from your database.
Answer: DROP TABLE Income;
Ensure to save all your queries in a document and upload onto this repo.