**DATABASE WEEK 2**

**Part 1: Retrieving Data with SELECT**

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.

To retrieve all data points (columns) from the "Expenses" table, we use the query

SELECT \* FROM Expenses;

The results are as shown in the figure below

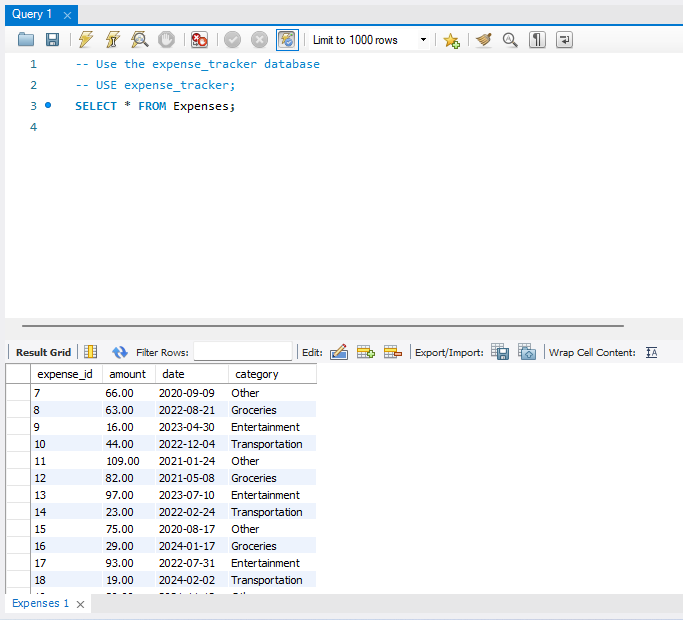


Figure 1.1

**1.2 Specific Columns:**

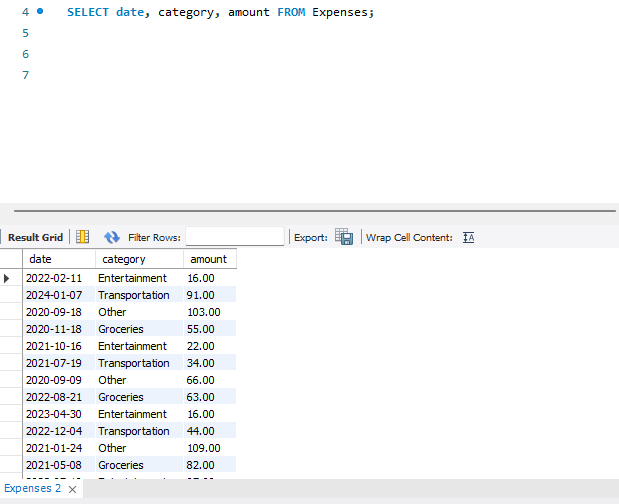
Modify your query to select only specific columns relevant to your analysis. For example, you might choose "date," "category," and "amount" to analyze spending patterns by category and date.

To modify the query to select only specific we use the query

SELECT date, category, amount FROM Expenses;

This query selects only the date, category, and amount columns from the Expenses table.

This is shown in the figure below.



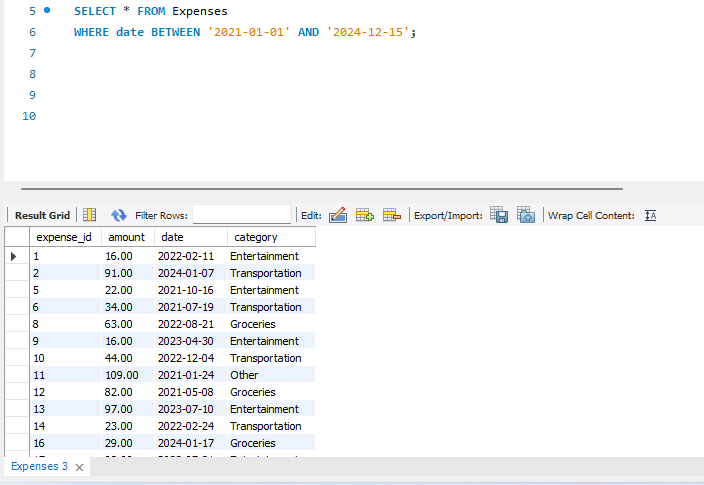
**1.3 Filtering by Date Range:** Write a query to retrieve expenses charged between a specific date range (e.g., January 1, 2021, to December 15, 2024). Remember to use the appropriate data type for the "date" column when specifying the date range in your query.

Write a query to retrieve expenses charged between a specific date range.

SELECT \* FROM Expenses

WHERE date BETWEEN '2021-01-01' AND '2024-12-15';

This query retrieves all columns from the Expenses table where the date is between January 1, 2021, and December 15, 2024.



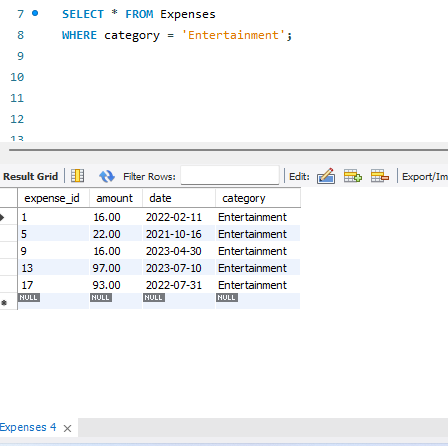
**Part 2: Filtering with WHERE Clause**

**2.1 Filtering by Category:** Write a query to find all expenses belonging to a specific category (e.g., "Entertainment").We use;

SELECT \* FROM Expenses

WHERE category = 'Entertainment';

This query selects all columns from the Expenses table where the category is "Entertainment"



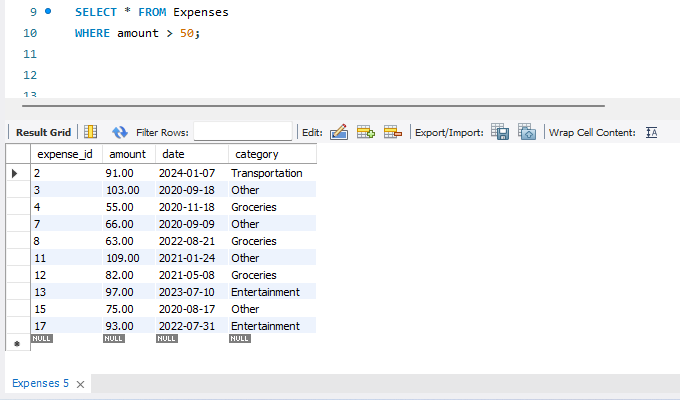
**2.2 Filtering with Comparison Operators:** Find expenses with an amount greater than a certain value (e.g., $50).

To find expenses with an amount greater than a certain value (e.g., $50), we use the query

SELECT \* FROM Expenses

WHERE amount > 50;

This query selects all columns from the Expenses table where the amount is greater than $50.

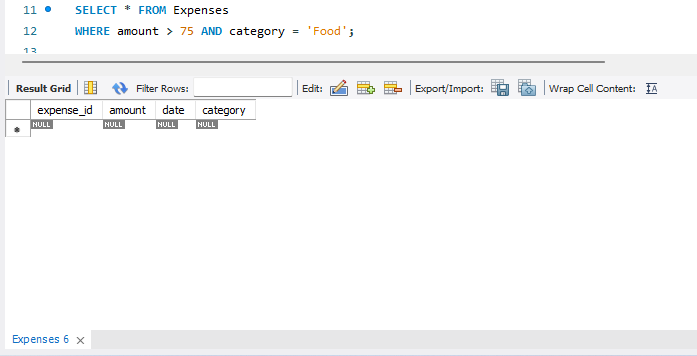


**2.3 Combining Filters (AND):**  
Refine your query to find expenses that meet multiple criteria. For example, you might search for expenses greater than $75 AND belonging to the "Food" category. We use the query;

SELECT \* FROM Expenses

WHERE amount > 75 AND category = 'Food';

This query selects all columns from the Expenses table where the amount is greater than $75 and the category is "Food".

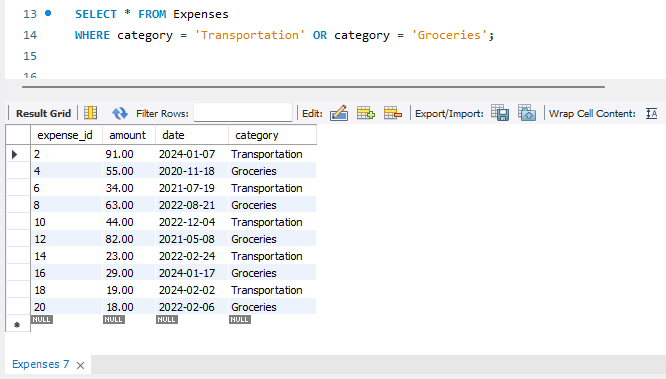


**2.4 Combining Filters (OR):** Modify your query to find expenses belonging to one category or another (e.g., "Transportation" OR "Groceries").We will use the following query

SELECT \* FROM Expenses

WHERE category = 'Transportation' OR category = 'Groceries';

This query selects all columns from the Expenses table where the category is either "Transportation" or "Groceries".

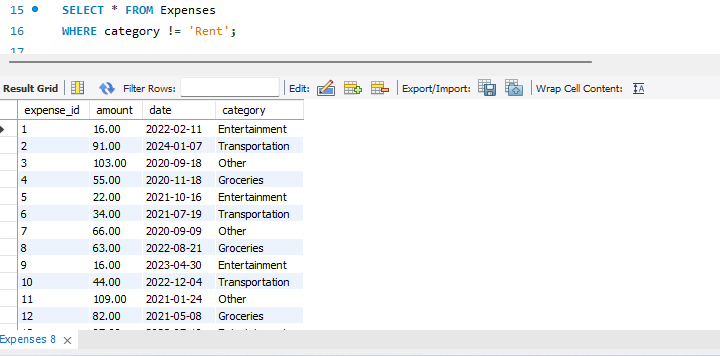


**2.5 Filtering with NOT:** Write a query to display expenses unrelated to a specific category (e.g., "Rent").We will use this query;

SELECT \* FROM Expenses

WHERE category != 'Rent';

This query selects all columns from the Expenses table where the category is not "Rent".



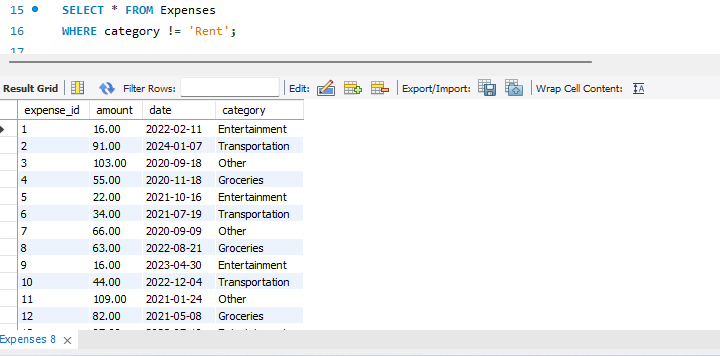
**Part 3: Sorting Retrieved Data**

**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).We use

SELECT \* FROM Expenses

ORDER BY amount DESC;

This query selects all columns from the Expenses table and sorts the results by amount in descending order.

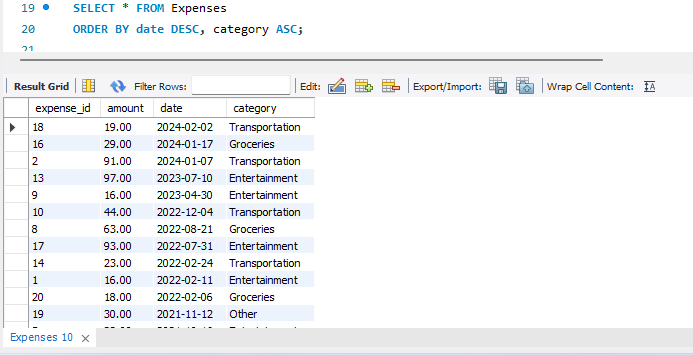


**3.2 Sorting by Date and Category:**  
Modify your query to sort expenses based on multiple columns. For example, you might sort first by date (descending order) and then by category (ascending order) to see recent spending trends by category.We will use the following query

SELECT \* FROM Expenses

ORDER BY date DESC, category ASC;

This query selects all columns from the Expenses table and sorts the results first by date in descending order and then by category in ascending order.



**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:

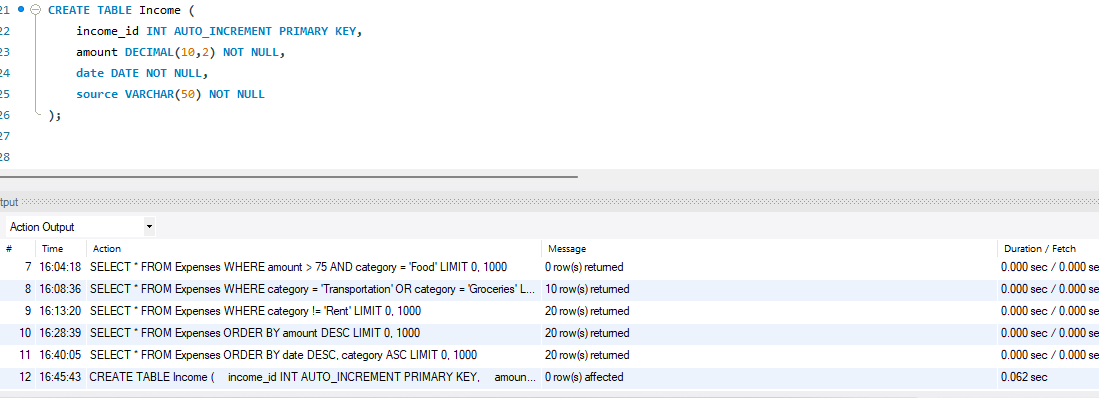
income\_id (INT) - Primary Key (auto-increment)

amount (DECIMAL(10,2)) - NOT NULL

date (DATE) - NOT NULL

source (VARCHAR(50)) - NOT NULL

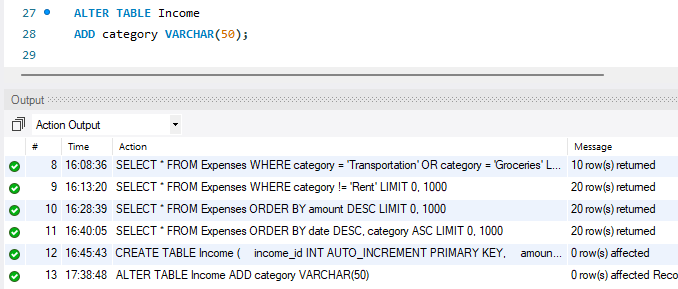
This command creates a new table named Income with columns for income\_id, amount, date, and source.



**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).

This command adds a new column named category to the Income table.



**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.

Imagine you no longer need the "Income" table entirely. Experiment how to Use DROP TABLE to permanently remove it from your database.

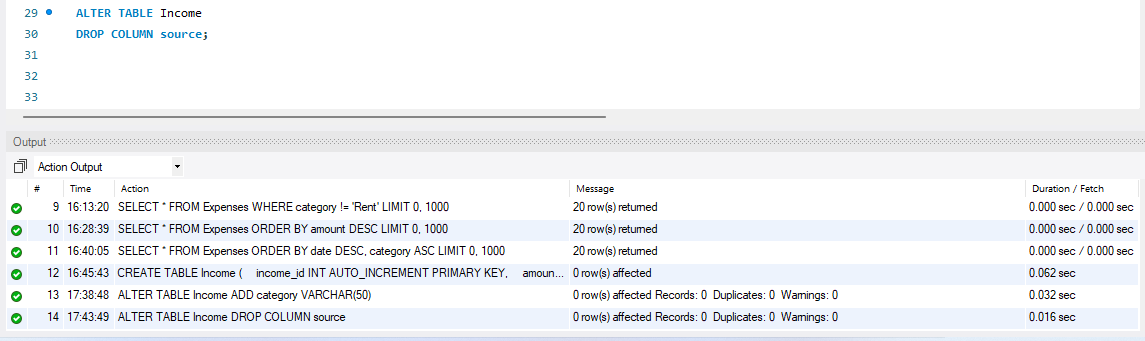
**4.3.1 Remove the "Source" Column**

**Use the query;**

ALTER TABLE Income

DROP COLUMN source;

This command removes the source column from the Income table.



**4.3.2 Drop the "Income" Table**

**Use the query;**

DROP TABLE Income;

This command permanently removes the Income table from the database.

