

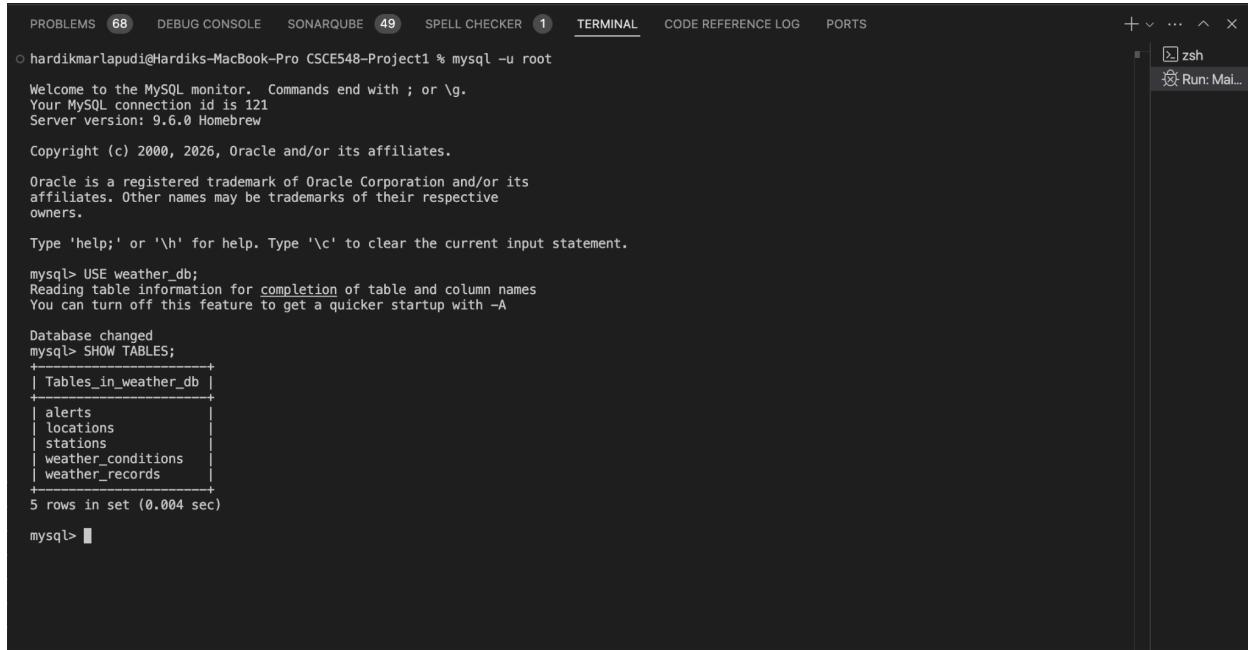
Weather Management System

CSCE 548

Hardik Marlapudi

Spring 2026

Database Schema Diagram



The screenshot shows a terminal window within a code editor interface. The terminal tab is active, displaying a MySQL session. The session starts with a connection to the root user on a MacBook Pro. It then displays standard MySQL welcome messages, copyright information, and a note about Oracle trademarks. The user runs the command `USE weather_db;`, which changes the database context. Then, the `SHOW TABLES;` command is run, listing five tables: alerts, locations, stations, weather_conditions, and weather_records. The output concludes with a timestamp of 0.004 seconds.

```
PROBLEMS 68 DEBUG CONSOLE SONARQUBE 49 SPELL CHECKER 1 TERMINAL CODE REFERENCE LOG PORTS
hardikmarlapudi@Hardiks-MacBook-Pro CSCE548-Project1 % mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 121
Server version: 9.6.0 Homebrew

Copyright (c) 2000, 2026, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

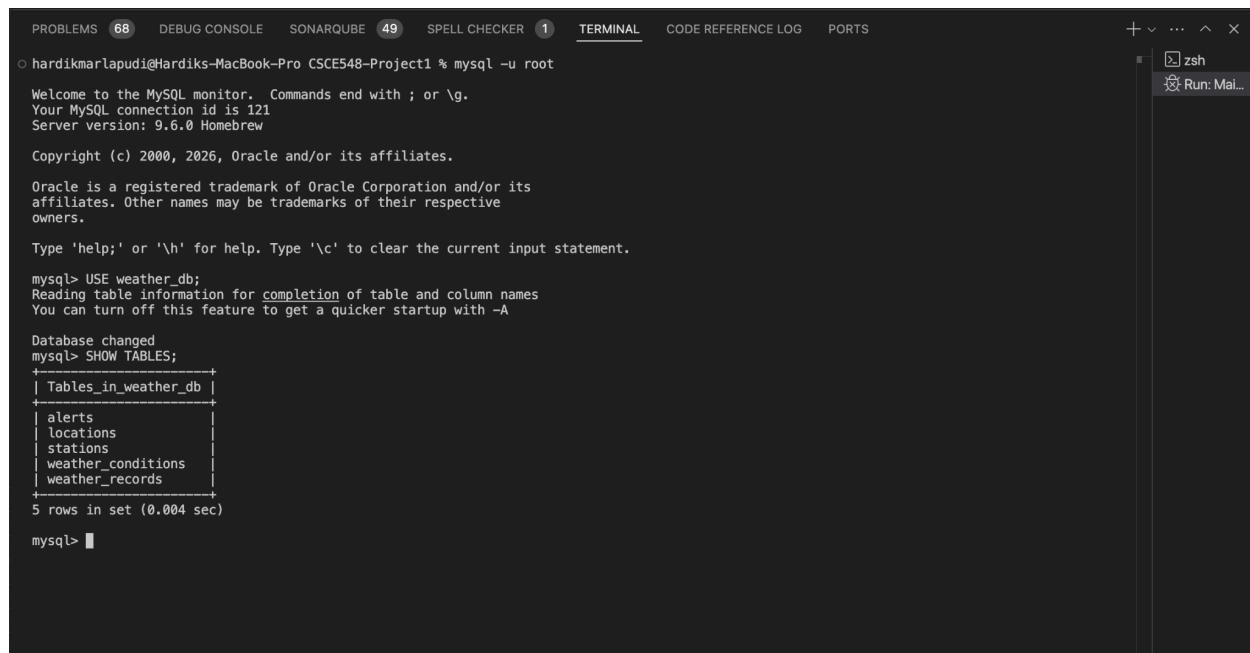
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> USE weather_db;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_weather_db |
+-----+
| alerts
| locations
| stations
| weather_conditions
| weather_records
+-----+
5 rows in set (0.004 sec)

mysql> 
```

Tables Created



The screenshot shows a terminal window within a code editor interface. The terminal tab is active, displaying a MySQL session. The session starts with a connection to the root user on a local MySQL monitor. It then displays standard MySQL welcome messages, copyright information, and a note about Oracle trademarks. The user runs the command `USE weather_db;`, which changes the database context. Following this, the `SHOW TABLES;` command is run, listing five tables: alerts, locations, stations, weather_conditions, and weather_records. The output concludes with a timestamp of 0.004 seconds.

```
PROBLEMS 68 DEBUG CONSOLE SONARQUBE 49 SPELL CHECKER 1 TERMINAL CODE REFERENCE LOG PORTS
hardikmarlapudi@Hardiks-MacBook-Pro CSCE548-Project1 % mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 121
Server version: 9.6.0 Homebrew

Copyright (c) 2000, 2026, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

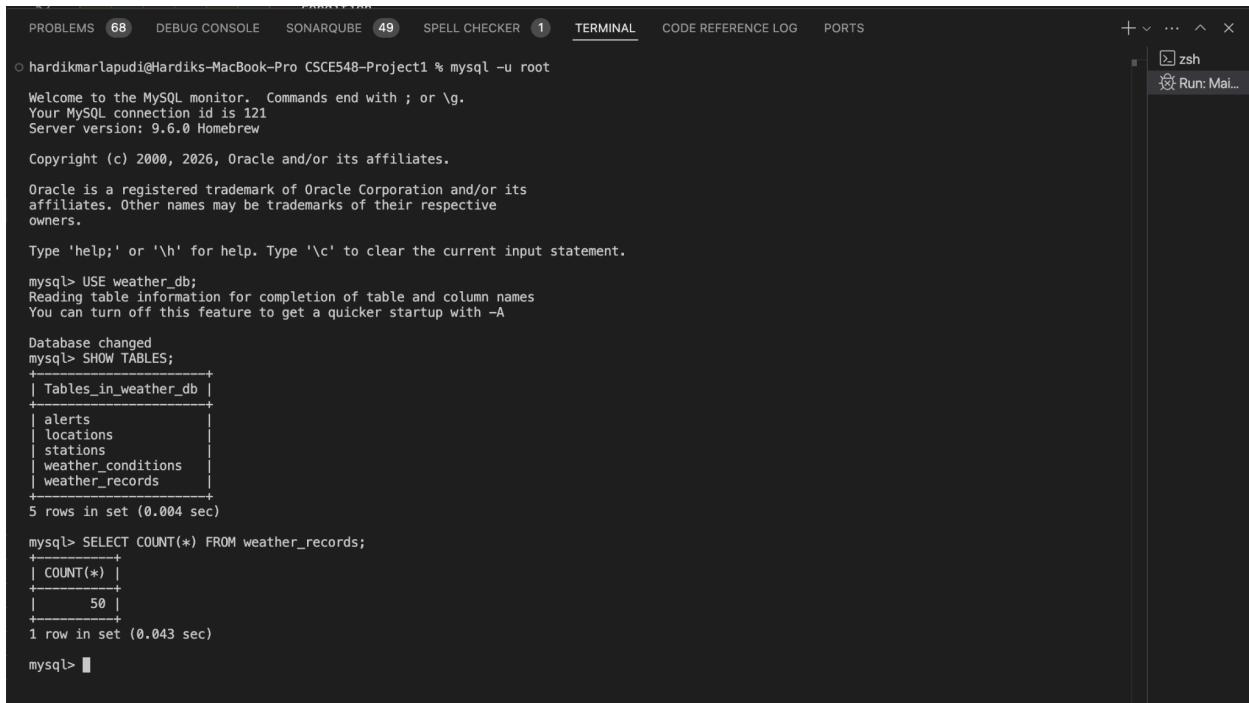
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> USE weather_db;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_weather_db |
+-----+
| alerts
| locations
| stations
| weather_conditions
| weather_records
+-----+
5 rows in set (0.004 sec)

mysql> 
```

50+ Rows Verification



A screenshot of a terminal window within a code editor interface. The terminal tab is active, showing a MySQL session on a MacBook Pro. The user has connected as root and selected the 'weather_db' database. They run 'SHOW TABLES;' to list five tables: alerts, locations, stations, weather_conditions, and weather_records. Then, they run 'SELECT COUNT(*) FROM weather_records;' which returns a single row with the value 50.

```
PROBLEMS 68 DEBUG CONSOLE SONARQUBE 49 SPELL CHECKER 1 TERMINAL CODE REFERENCE LOG PORTS
hardikmarlapudi@Hardiks-MacBook-Pro CSCE548-Project1 % mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 121
Server version: 9.6.0 Homebrew

Copyright (c) 2000, 2026, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> USE weather_db;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_weather_db |
+-----+
| alerts           |
| locations        |
| stations         |
| weather_conditions |
| weather_records  |
+-----+
5 rows in set (0.004 sec)

mysql> SELECT COUNT(*) FROM weather_records;
+-----+
| COUNT(*) |
+-----+
|      50   |
+-----+
1 row in set (0.043 sec)

mysql>
```

Console Application Running

The screenshot shows the Visual Studio Code (VS Code) interface with a Java project named "CSCE548-Project1". The "Main.java" file is open in the editor, displaying the main method and a menu loop. The code uses Scanner to read user input and System.out.println to display options. The terminal at the bottom shows the application running, printing the menu options and waiting for input. The status bar indicates the terminal command and Java environment details.

```
import java.sql.Date;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        WeatherRecordDAO weatherDAO = new WeatherRecordDAO();
        AlertDAO alertDAO = new AlertDAO();

        while (true) {
            System.out.println("1. View Weather Records");
            System.out.println("2. Add Weather Record");
            System.out.println("3. Delete Weather Record");
            System.out.println("4. View Alerts");
            System.out.println("5. Add Alert");
            System.out.println("6. Exit");
            System.out.print("Choice: ");

            int choice = scanner.nextInt();
            scanner.nextLine(); // consume newline

            switch (choice) {
                case 1:
                    weatherDAO.getAllRecords();
                    break;
                case 2:
                    weatherDAO.addRecord();
                    break;
                case 3:
                    weatherDAO.deleteRecord();
                    break;
                case 4:
                    alertDAO.getAllAlerts();
                    break;
                case 5:
                    alertDAO.addAlert();
                    break;
                case 6:
                    System.out.println("Exiting...");
                    return;
                default:
                    System.out.println("Invalid choice");
            }
        }
    }
}
```

TERMINAL

```
hardikmarapudi@hardiks-MacBook-Pro CSCE548-Project1 % /usr/bin/java -agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:61938 @var/folders/mp/g8vjs70s2cxfwlz4nwfjzgh000gn/T/cp_dnh4igo2d8m9aj6lt9ejsn18p.argfile Main

Weather Management System
1. View Weather Records
2. Add Weather Record
3. Delete Weather Record
4. View Alerts
5. Add Alert
6. Exit
Choice:
```

PROBLEMS 68 DEBUG CONSOLE SONARQUBE 49 SPELL CHECKER 1 TERMINAL CODE REFERENCE LOG PORTS

hardikmarapudi@hardiks-MacBook-Pro CSCE548-Project1 % /usr/bin/java -agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:61938 @var/folders/mp/g8vjs70s2cxfwlz4nwfjzgh000gn/T/cp_dnh4igo2d8m9aj6lt9ejsn18p.argfile Main

1. View Weather Records
2. Add Weather Record
3. Delete Weather Record
4. View Alerts
5. Add Alert
6. Exit

Data Retrieved from Database

The screenshot shows a Visual Studio Code (VS Code) interface with the following details:

- Project Structure:** The left sidebar shows a project named "CSCE548-PROJECT1" containing .vscode, bin, lib, screenshots (with four files: screenshot#1.png, screenshot#2.png, screenshot#3.png, screenshot#4.png), sql (with two files: data.sql and schema.sql), and src (with Java files: Alert.java, AlertDAO.java, DBConnection.java, Location.java, LocationDAO.java, Main.java, Station.java, StationDAO.java, WeatherCondition.java, WeatherConditionDAO.java, WeatherRecord.java, and WeatherRecordDAO.java). There are also README.md and Project_Overview.pdf files.
- Code Editor:** The main editor area displays Java code and a SQL query. The Java code includes imports for java.util.List and org.json.JSONObject, and defines classes like Station, WeatherRecord, DBConnection, Alert, and data.sql. The SQL query is a SELECT statement from a table named alerts, showing columns like location_name and message.
- Terminal:** The bottom right corner shows a terminal window with the title "hardik-CSCE548-Project1-fixes*". It contains a menu for the Weather Management System with options 1 through 6, and a choice of 4 is selected. It also shows log entries for weather records and alerts.
- Bottom Status Bar:** The status bar at the bottom provides various status indicators, including "AWS: 2 of 2 Connections Expired", "Java: Ready", "SonarQube focus: overall code", and "MS SQL".

Write the Overview Document Section

This project is based on modeling a Weather Management System, which stores and manages weather details based on location and station. The system monitors daily weather conditions such as temperature and humidity levels, and it is capable of generating location-based weather alerts. The application is based on the integration of a database with a Java-based console interface.

AI Prompt Used

ChatGPT was employed for creating the initial SQL schema, test data, and Java Data Access Object (DAO) code. The prompts were centered on creating a relational database with various tables, establishing foreign key relationships between them, and creating CRUD operations for the project. The result obtained from using ChatGPT was a starting point, which was later verified and corrected using manual debugging and testing.

Changes I Made Manually

Example:

- Removed unnecessary join tables generated by AI
- Replaced users table with stations to better fit the problem
- Fixed SQL column mismatches causing runtime errors
- Corrected DAO queries to match final schema
- Ensured 50 sequential weather records
- Debugged MySQL connection and missing-table errors

AI Effectiveness Analysis

ChatGPT helped hasten the development process, but it created overly complex schemas and used columns inconsistently. Debugging and refactoring were necessary to make it correct and adhere to foreign keys.