Lewis University  
CPSC 50900: Database Systems  
Term Project

World temperature by cities and states

Avinash Kommuru, avinashkommuru@lewisu.edu

Abhishek Manne, amanne, abhishekmanne@lewisu.edu

Madhu Sudhan Reddy Jakka , madhusudhanreddyja@lewisu.edu

https://github.com/a4v5i6/Database-system

Table of Contents

[Initial Proposal 2](#_Toc85814569)

[Data Sources 2](#_Toc85814570)

[Data Storage Alternatives 2](#_Toc85814571)

[Relational Database Design Process 3](#_Toc85814572)

[Relational Database Design 3](#_Toc85814573)

[Data Definition Language (DDL) Scripts 3](#_Toc85814574)

[Data Manipulation Language Scripts 4](#_Toc85814575)

[Indexes 5](#_Toc85814576)

[Views 5](#_Toc85814577)

[Triggers 5](#_Toc85814578)

[Transactions 5](#_Toc85814579)

[Database Security 6](#_Toc85814580)

[Locking and Concurrent Access 6](#_Toc85814581)

[Backing Up Your Database 6](#_Toc85814582)

[Python Programming 7](#_Toc85814583)

[PHP Programming 7](#_Toc85814584)

[Suggested Future Work 8](#_Toc85814585)

[Activity Log 8](#_Toc85814586)

# Initial Proposal

# We will store global land temperature changes by major cities and by major states.

We are interested in this data because global warming is the world’s biggest problem. Temperature fluctuations is abrupt and people are facing natural disasters.

It is important to keep track of temperature changes over the years to make amend to environment.

The data will come from

<https://data.world/data-society/global-climate-change-data>

Governments will use this data to keep control of air pollution

Title

World temperature by cities and states

# Data Sources

Gathered data in csv format. Data files contain world population by cities and states.

Dates(date), average temperature(int ), average temperature uncertainty(int), state(string), city(string), country(string)

# Data Storage Alternatives

**Network Database Model:**

Structure is like hierarchical model, except that a child can have multiple parents.

**Advantages:**  
A network schema that the database administrators use.

Has one or more subschemas that provide simpler access to certain nodes.

Standard languages are used.

**Disadvantages**

It has flexibility problem. Not all relations can be handled by assigning them in the form of owners and members.

The structure of the Network Model is quite complicated and so the programmer has to understand it well in order to implement or modify it.

**NoSQL Model**:  
Means “Not Only SQL”

Data is organized as text files and scripts need to be written to make sense all of the text files.

**Advantage:**  
No overheads such as coming up with fancy ways of organizing data.

Highly efficient to take in data.

**Disadvantage**:  
highly dependent on computational power to access data.

# Relational Database Design Process

date----🡪 averageTemperature, AveragTemperatureChange

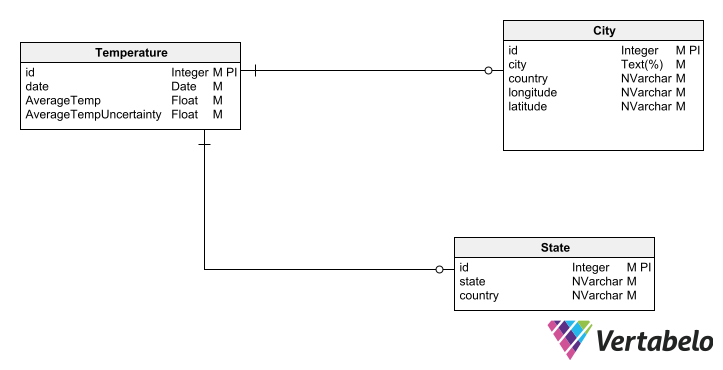
State----🡪country

City----🡪 country, Latitude, Longitude

Naming Entity Sets  
State

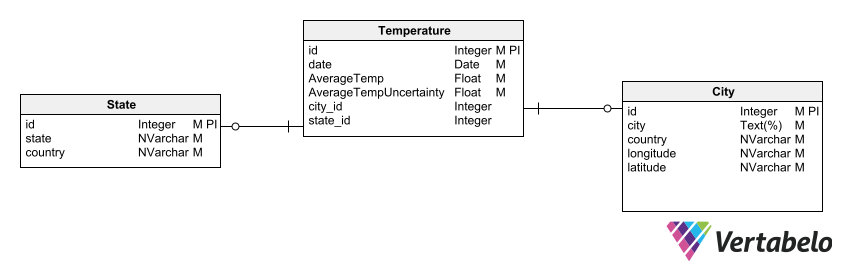
City

Temperature

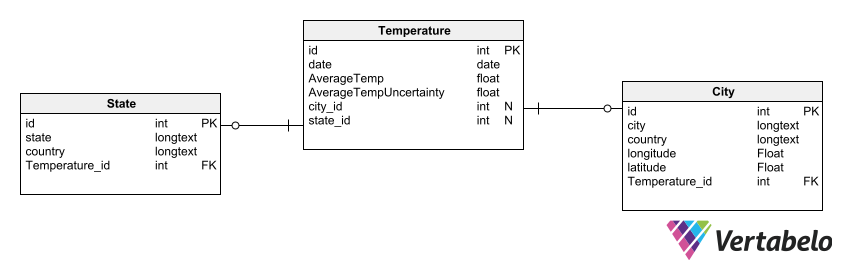


# Relational Database Design

*Normalised*

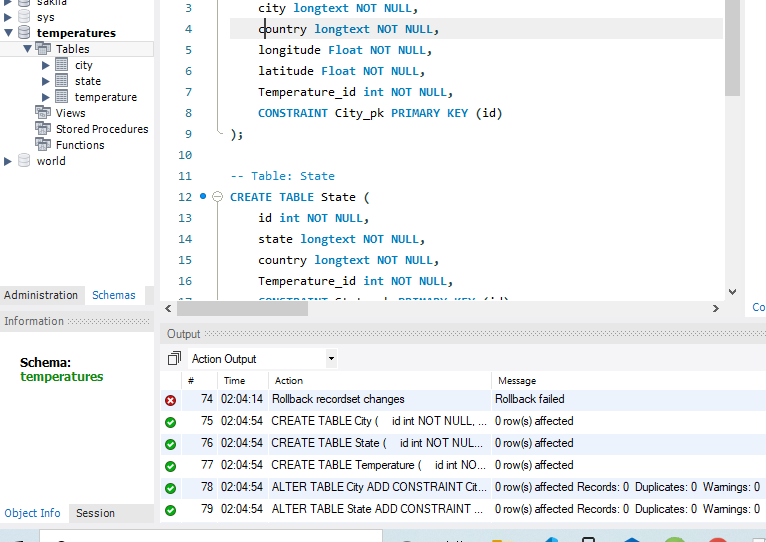


Physical DB model

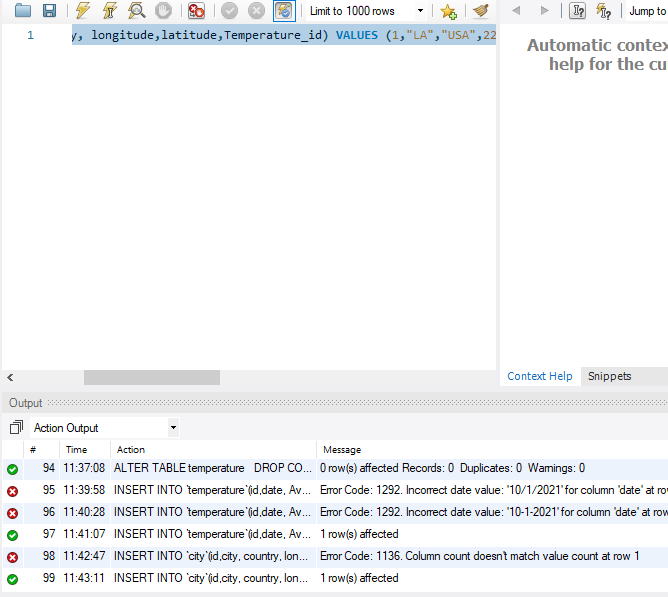
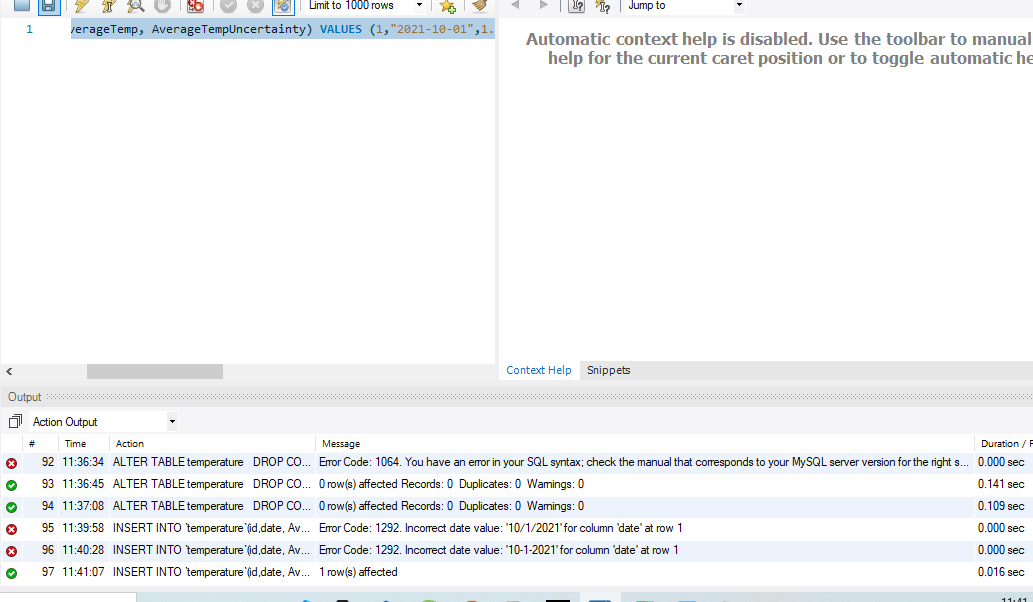


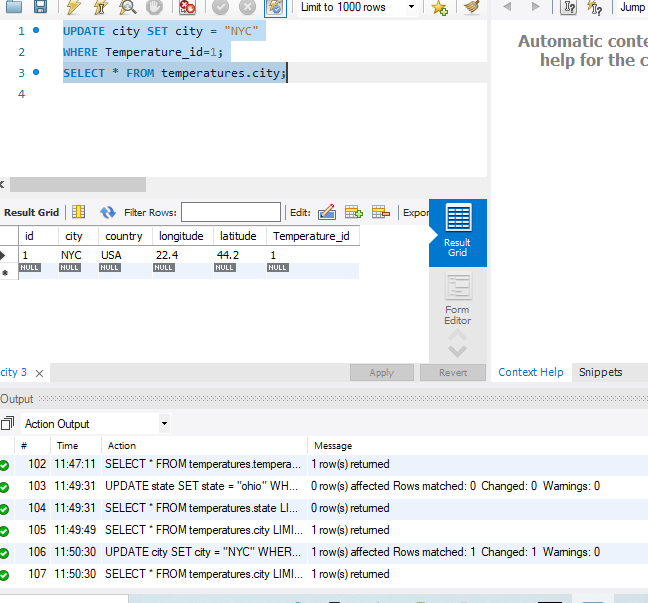
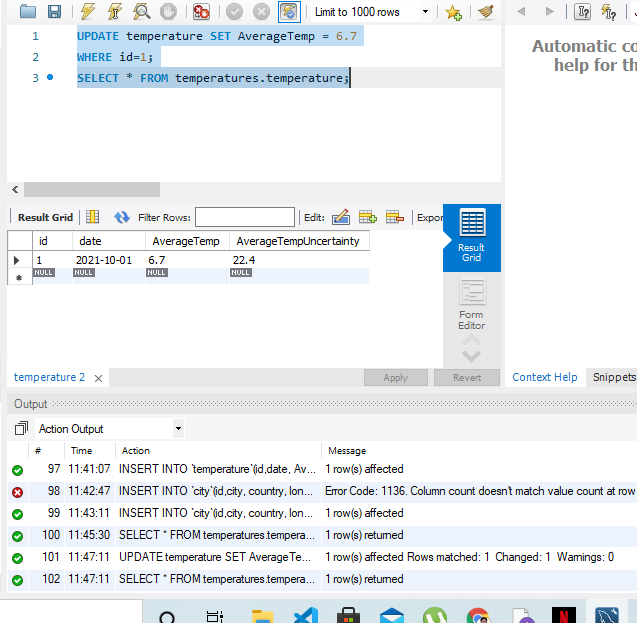
# Data Definition Language (DDL) Scripts

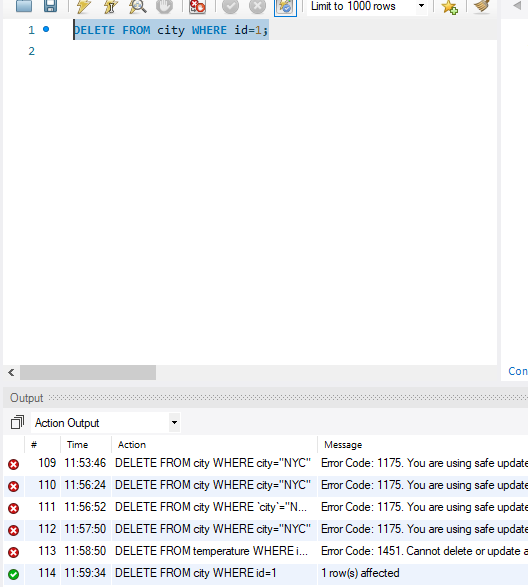
Generated three scripts to write three tables in sql

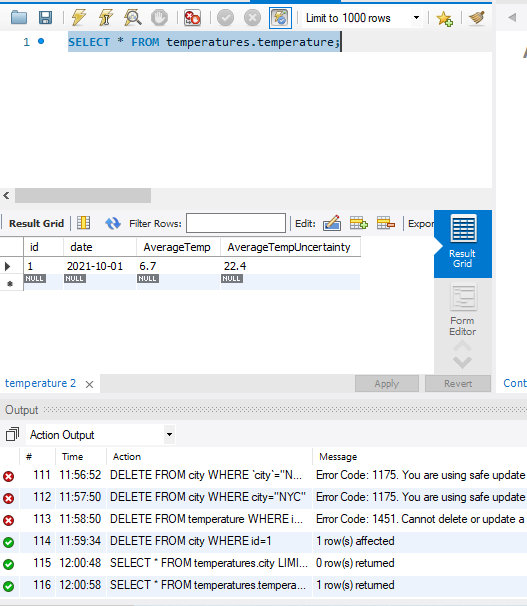


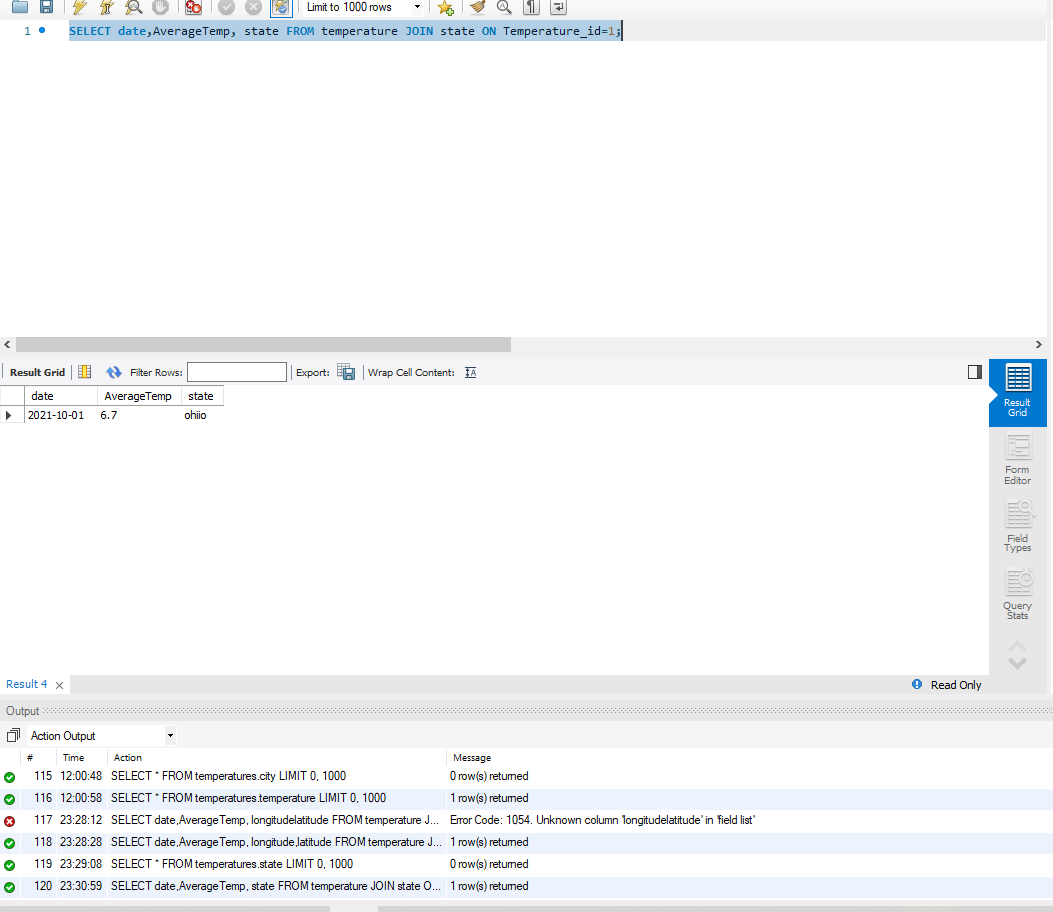
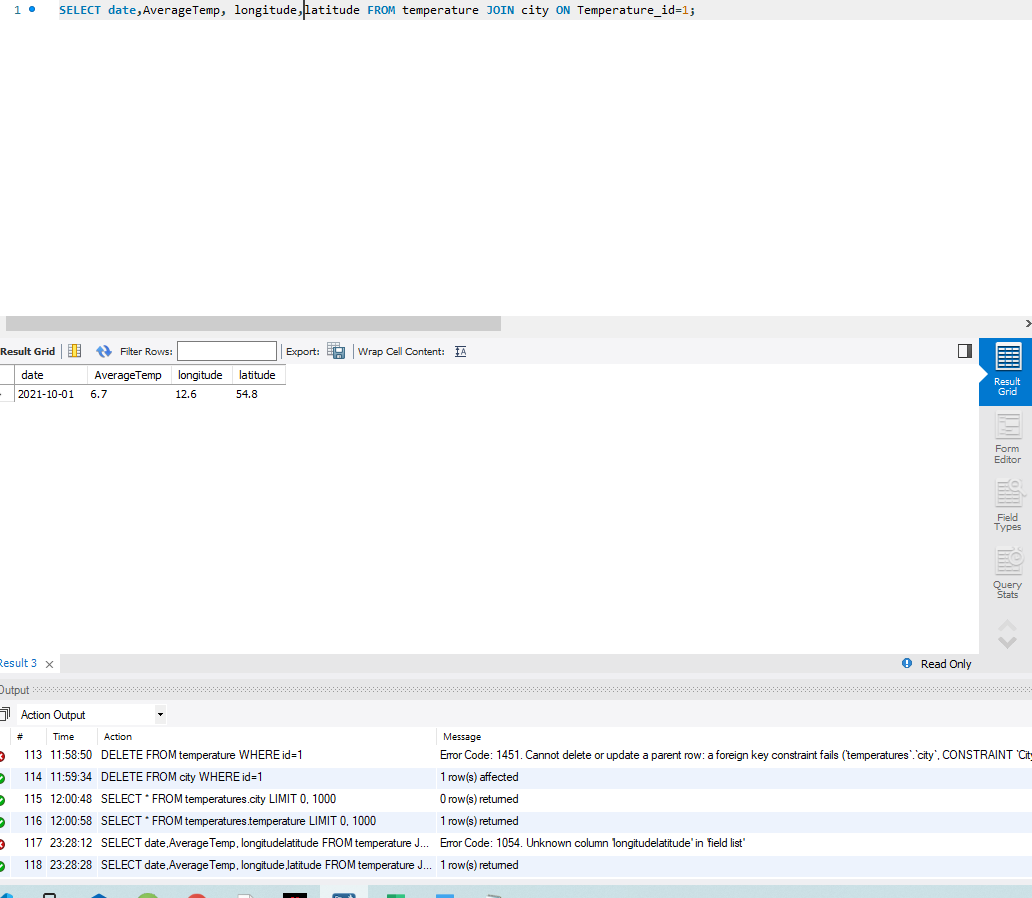
# Data Manipulation Language Scripts

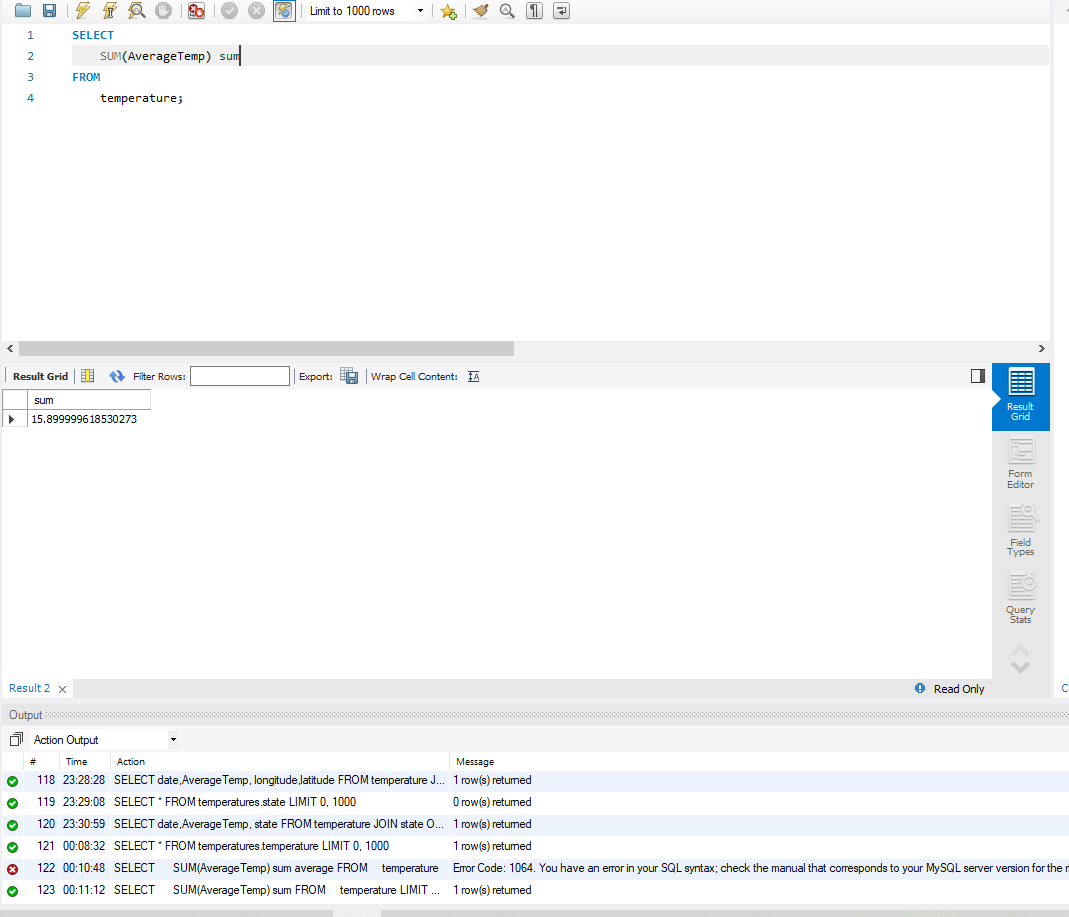
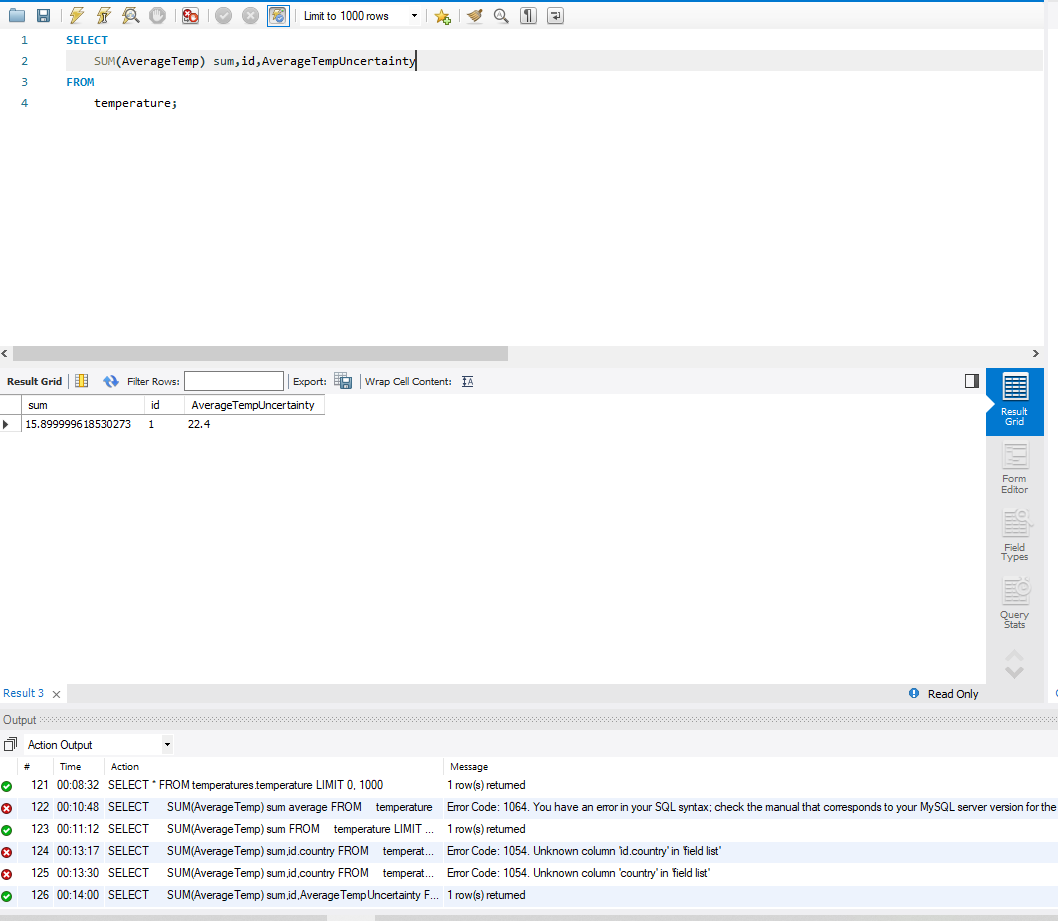
**INSERT COMMANDS:**  


**UPDATE COMMANDS:**  


**DELETE COMMAND**  


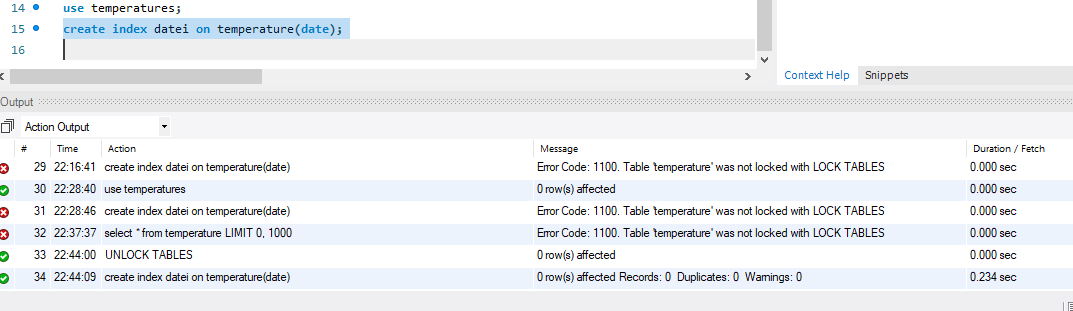
**SELECT COMMAND:**  


**JOIN COMMANDS:**  


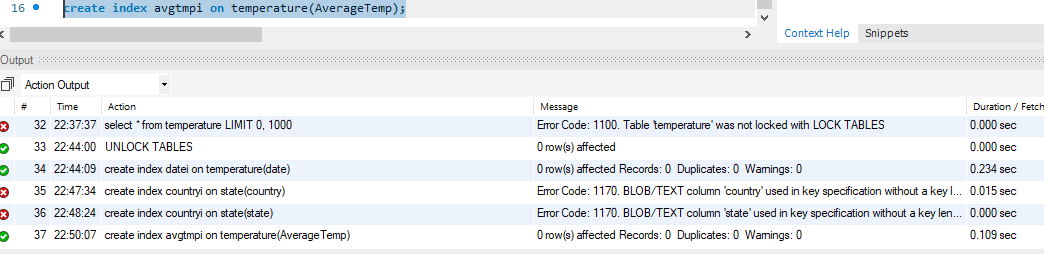
**SUMMARY COMMAND:**  
  
**COMMAND OF CHOICE**  


# Indexes

This index is applied on date since we would want to query by date.

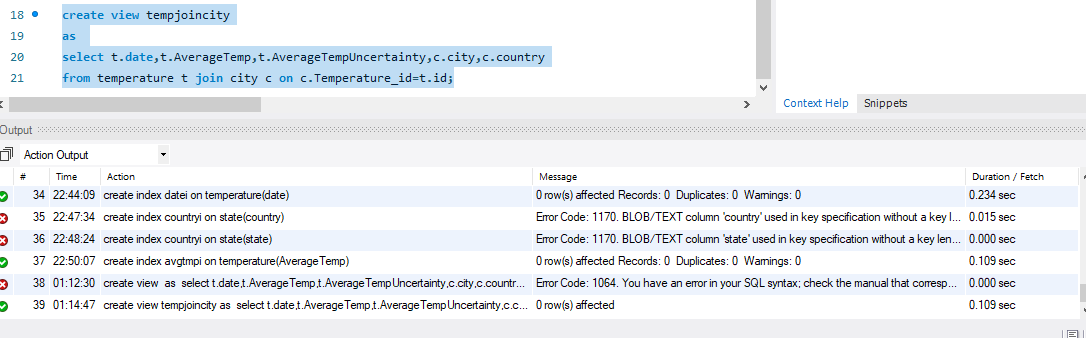


This index is applied on average temperature since queries will be sorted by it.

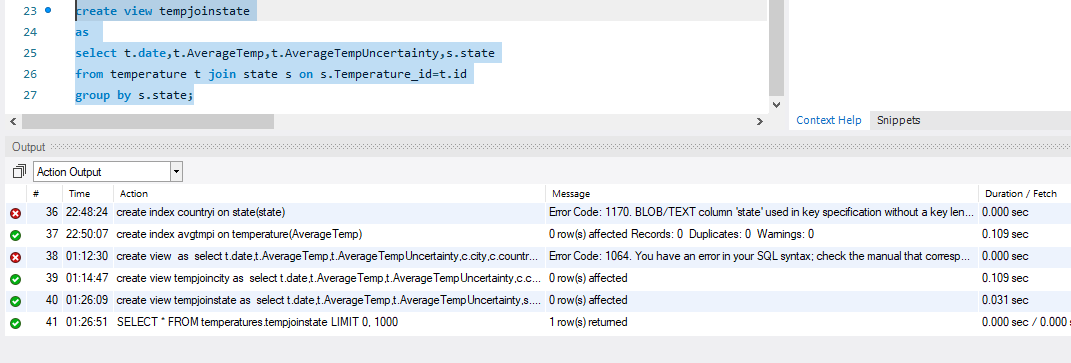


# Views

This view will be used in join query of city and temperature



This view shows result of a join and group the result according to state



# Triggers

CREATE TRIGGER avgtempwinter

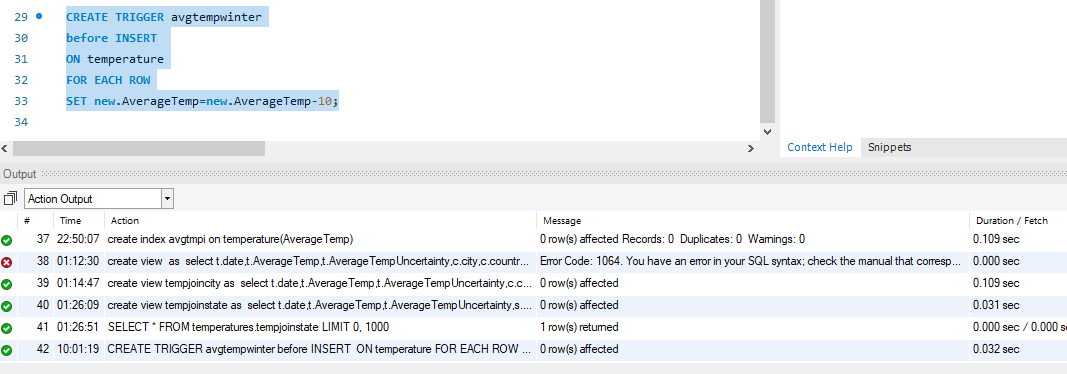
before INSERT

ON temperature

FOR EACH ROW

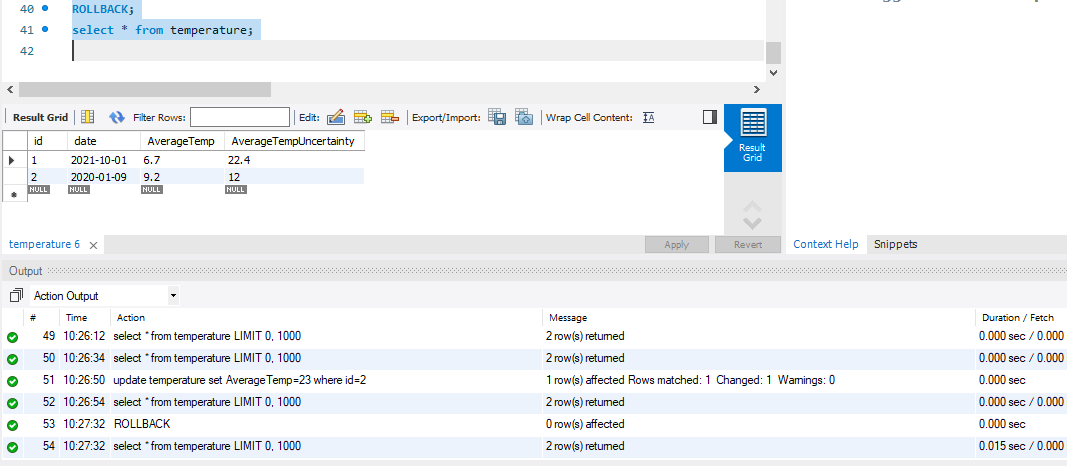
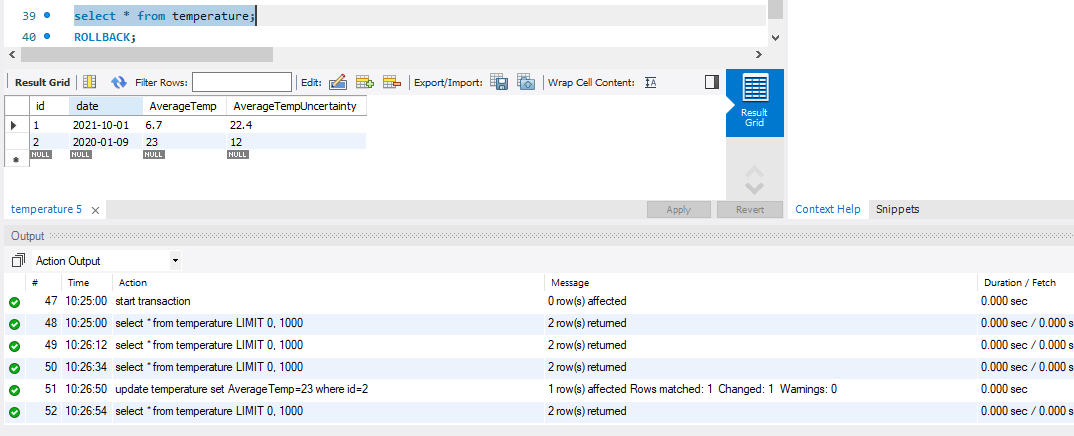
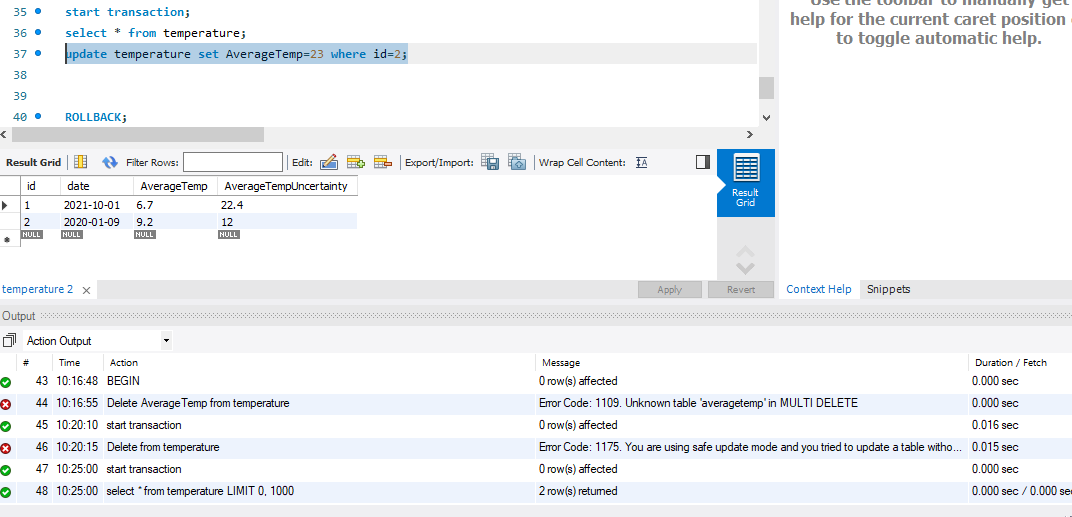
SET new.AverageTemp=new.AverageTemp-10;

Purpose: This trigger is supposed to subtract 10 from avg temperature



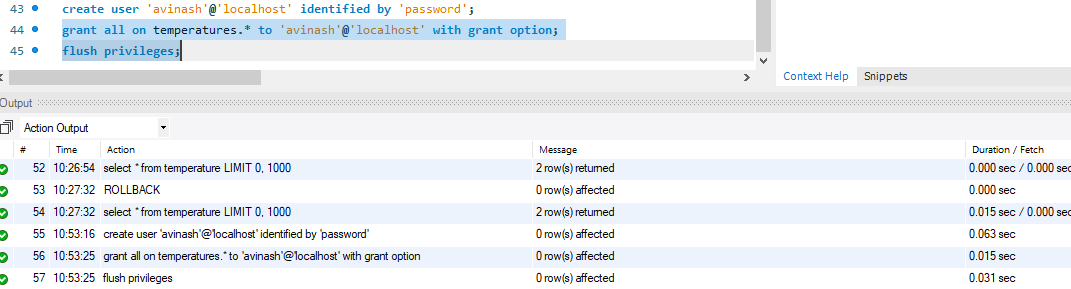
# Transactions

Transaction should follow rules of atomicity, consistency, isolation, and durability.

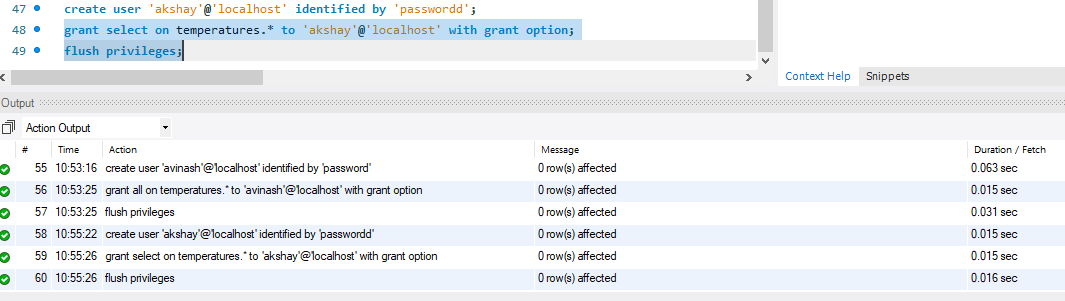


# Database Security

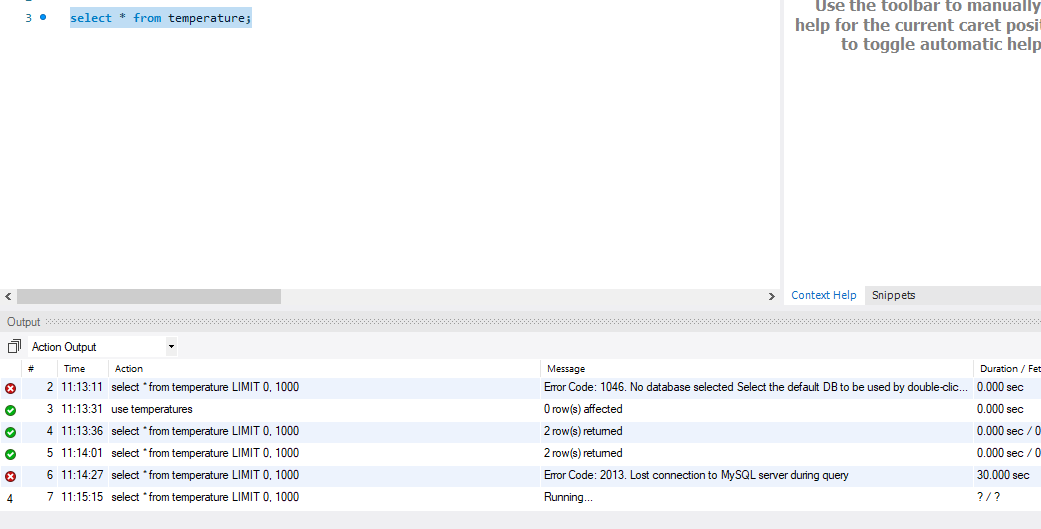
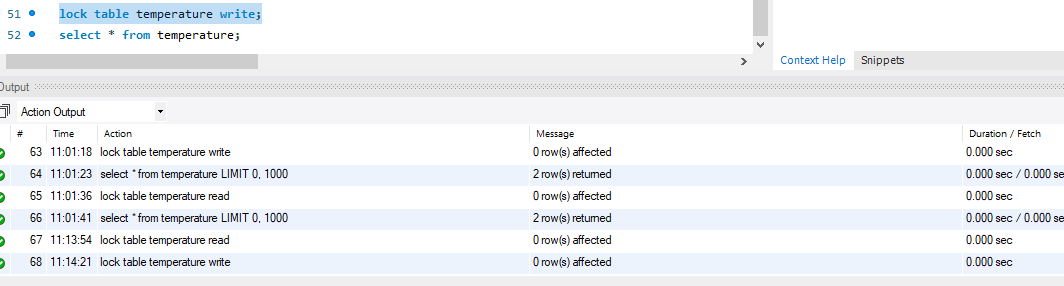
Avinash will have all the privileges as a database administrator



Abhishek will have privilege to select tables from database and no other privilege.



# Locking and Concurrent Access



# Backing Up Your Database

mysqldump -h localhost -u avinash -p password temperatures > temperatures.sql

# Python Programming

Python script:

import mysql.connector

from mysql.connector import errorcode

# a function that takes connection and query and return everything in the list

def select(conn,query):

cursor = conn.cursor()

cursor.execute(query)

results = []

for row in cursor.fetchall():

results.append(row)

cursor.close()

return results

# a function that takes connection and query and commits the query

def execute(conn,query): # update, delete, and insert

cursor = conn.cursor()

cursor.execute(query)

conn.commit()

# a function that prints the resulttothe screen

def show(rows):

for row in rows:

print(row)

# trying to establish connection to the database

try:

conn = mysql.connector.connect(

user="root",

password="",

host="localhost",

database="globaltemp")

except mysql.connector.Error as err:

print("Cannot connect.")

exit()

# storing query in variable

rows = select(conn,"select \* from city c join state s join temperature t where c.Temperature\_id=t.id & s.Temperature\_id=t.id ")

# function call

show(rows)

print("Now insert a record")

# function call

execute(conn,"insert into city values (null,'NYC','Usa',14.8,122.6,1)")

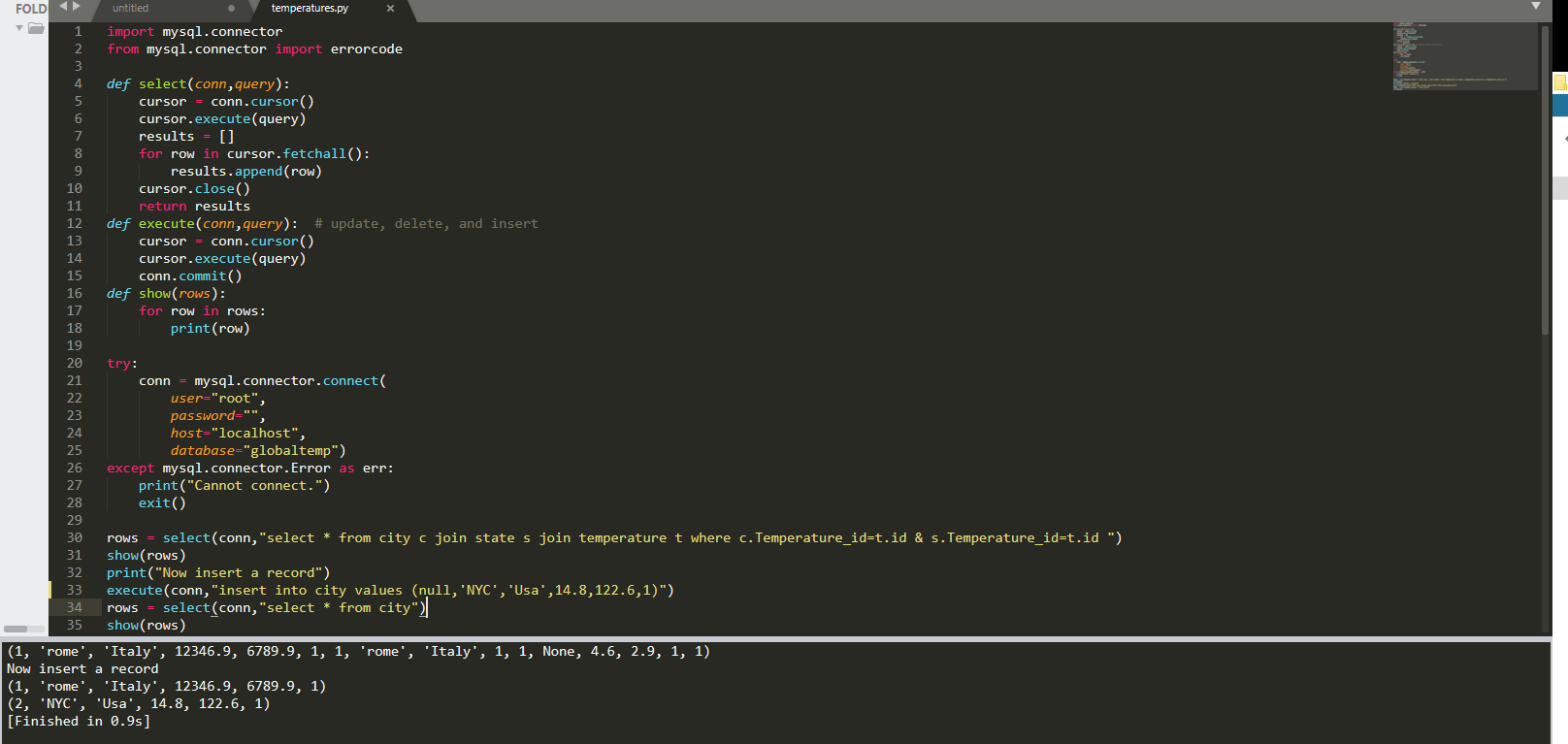
# function call

rows = select(conn,"select \* from city")

# function call

show(rows)

Screenshot of running program:



# PHP Programming

Search form:

<form action="search.php" method="POST">

    <input type="text" name="search" placeholder="search" >

    <button type="submit" name="submit">Go</button>

</form>

Php script that searches through database:

<?php

if(isset($\_POST['submit'])){

  $x= $\_POST['search'];

  $db->query("SELECT \* FROM temperature where

  id LIKE '%$x%' OR AverageTemp LIKE '%$x%'

  ");

 $result = $db->single();

 echo $result->id . "\t" . $result->date . "\t" .

    $result->AverageTemp . "\t" . $result->AverageTempUncertainty ."\t" . $result->AverageTempUncertainty."\t" ."\n";

}

?>

Result page complete:

<?php

include\_once("config.php");

include\_once("Database.php");

$db = new Database();

?>

<h1>result page</h1>

<div>

<?php

if(isset($\_POST['submit'])){

  $x= $\_POST['search'];

  $db->query("SELECT \* FROM temperature where

  id LIKE '%$x%' OR AverageTemp LIKE '%$x%'

  ");

 $result = $db->single();

 echo $result->id . "\t" . $result->date . "\t" .

    $result->AverageTemp . "\t" . $result->AverageTempUncertainty ."\t" . $result->AverageTempUncertainty."\t" ."\n";

}

?>

</div>

SQL injection: somebody can run some Sql on search field to cause unwanted damage to database.

We have used PDOs to prevent that from happening  
Before search  


After search



# Suggested Future Work

My database is not yet placed on remote server and has limited data storage capacity.  
I would migrate the database to cloud.  
NoSQL databases emerged in the late 2000s as the cost of storage dramatically decreased. Gone were the days of needing to create a complex, difficult-to-manage data model in order to avoid data duplication. Developers (rather than storage) were becoming the primary cost of software development, so NoSQL databases optimized for developer productivity.

# Activity Log

Avinash Kommuru identified data source with descriptions.

Abhishek Manne put data in csv files.

Abhishek Manne described the techniques to store data in relational databases

Avinash Kommuru did all the relational database design.

Abhishek Manne completed all the DDL work.

Madhu Sudhan Reddy Jakka completed DML work and helped in previous tasks as well.

Avinash Kommuru with indexes

Abhishek-views

Madhu-triggers, security

Avinash-locking

Abhishek-backup and future work

Avinash Kommuru made modified python script to work for our database

Abhishek Manne did the job with the main page of php website.

Madhu Sudhan Reddy Jakka completed the website by working on search form and result page