Homework 1

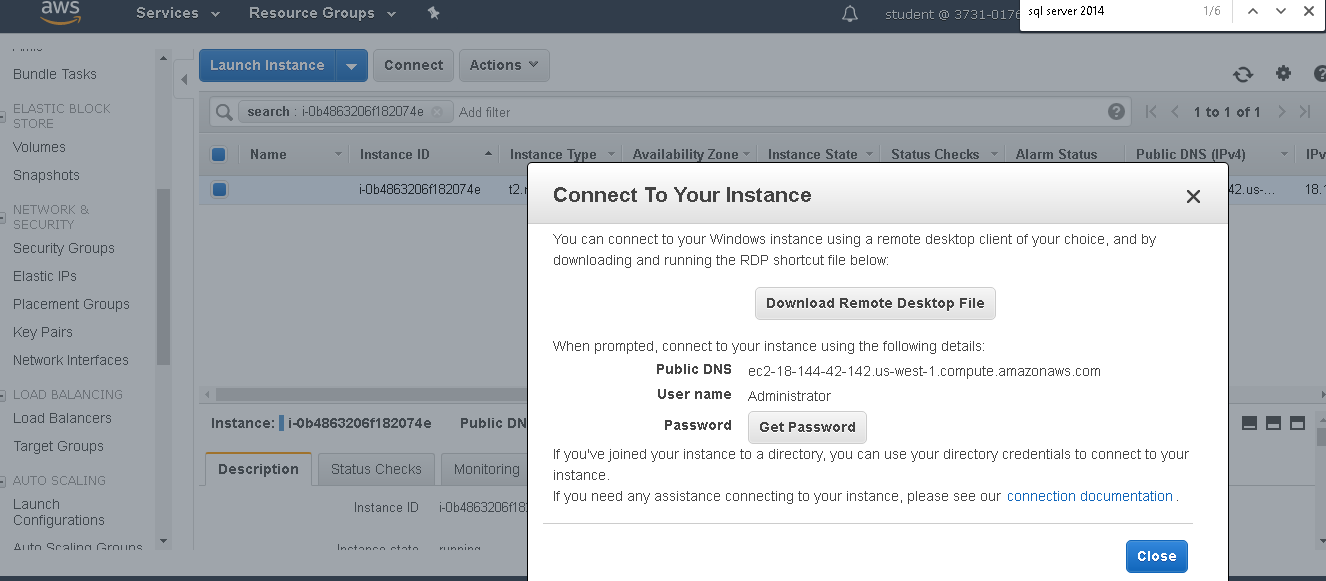
Step 1:

Launched a SQL Server EC2 instance on region other than North Virginia or Ohio. So used N California.  
  
Deployed an EC2 Instance with SQL Server on the AWS Cloud. Deployed an EC2 with SQL Server pre-installed.

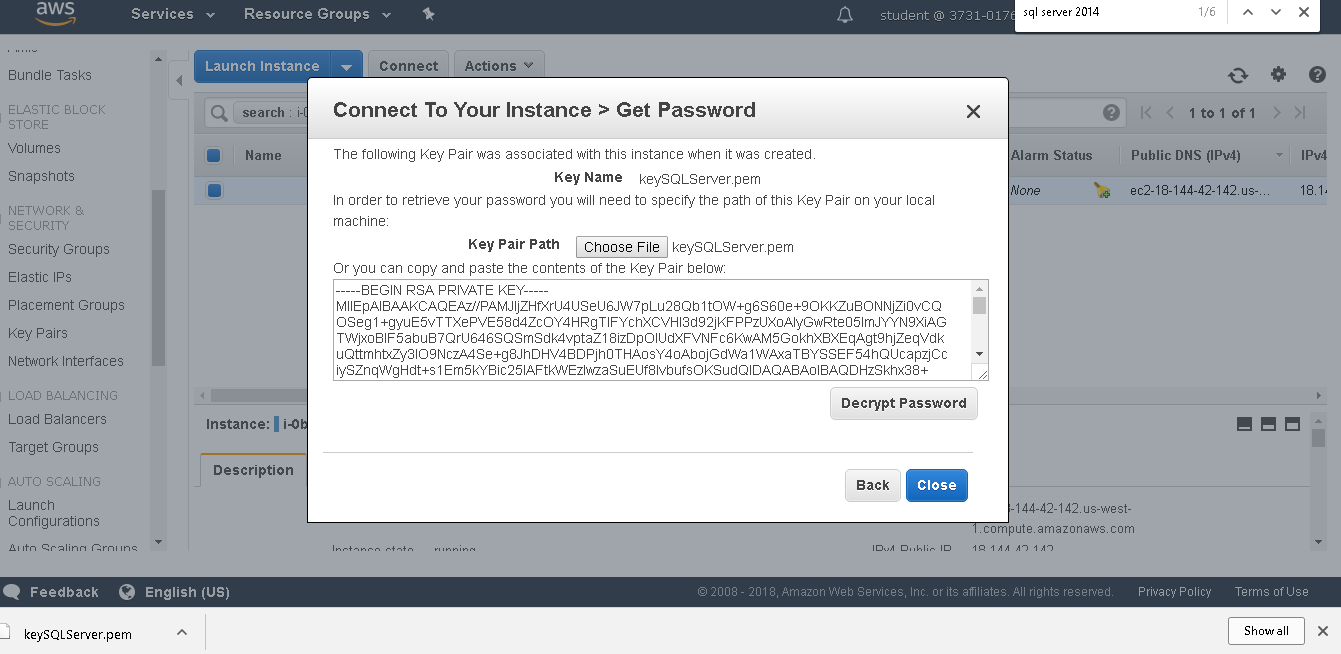
Default :Used 50 GB of default space.

Created a key called keySQLserver.pem

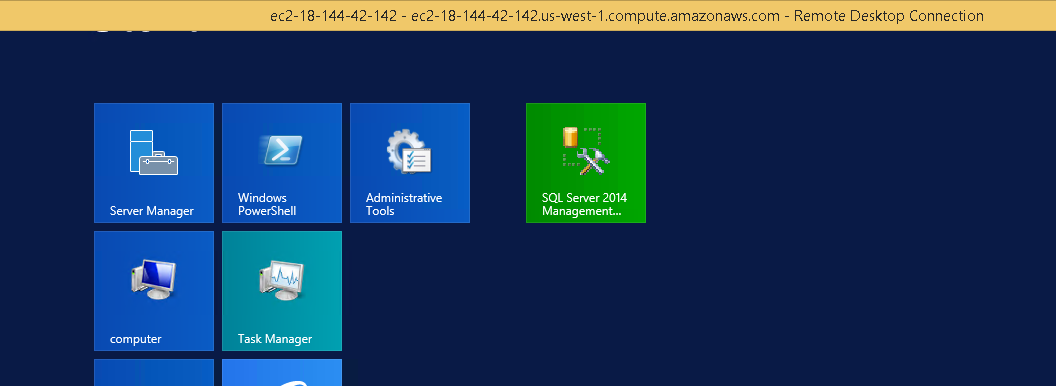
Default: Ports for SQL and remote access were not changed

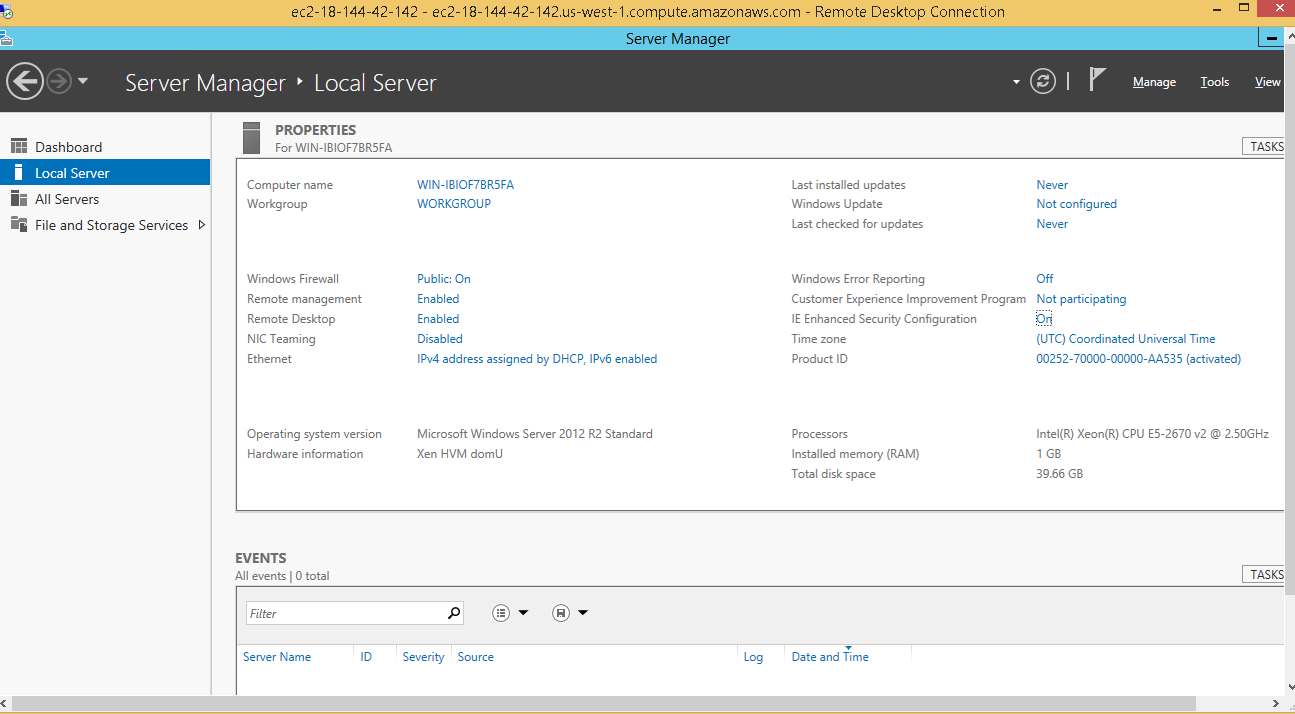


Created password using the key.pem file



Logged in

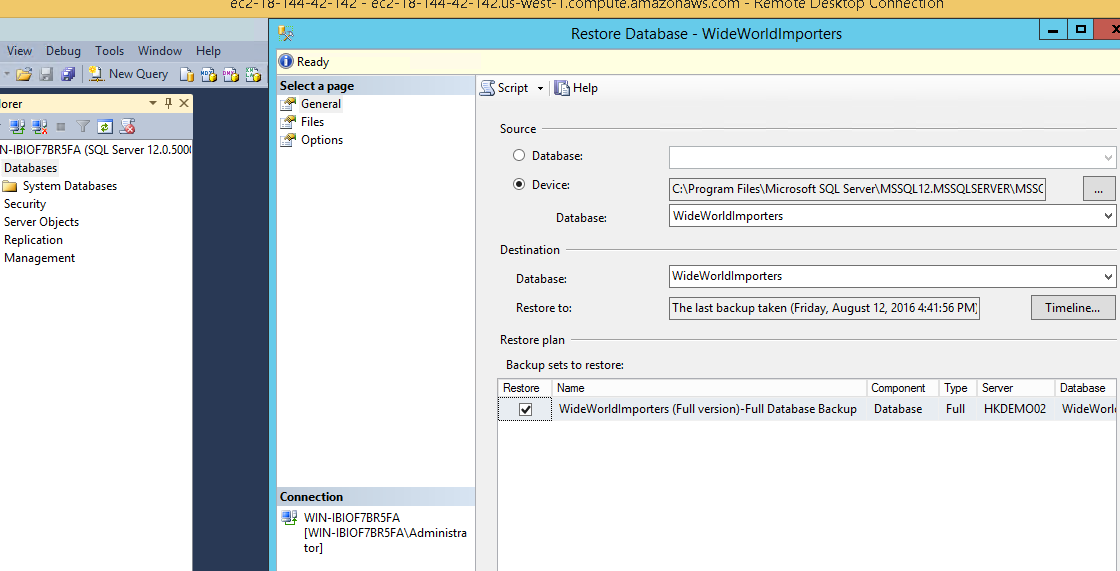




EC2 is read

Downloaded from the Github the Wide World Importers Sample database backup: https://github.com/Microsoft/sql-server-samples/releases/tag/wide-world-importers-v1.0.

Reinstated the backup



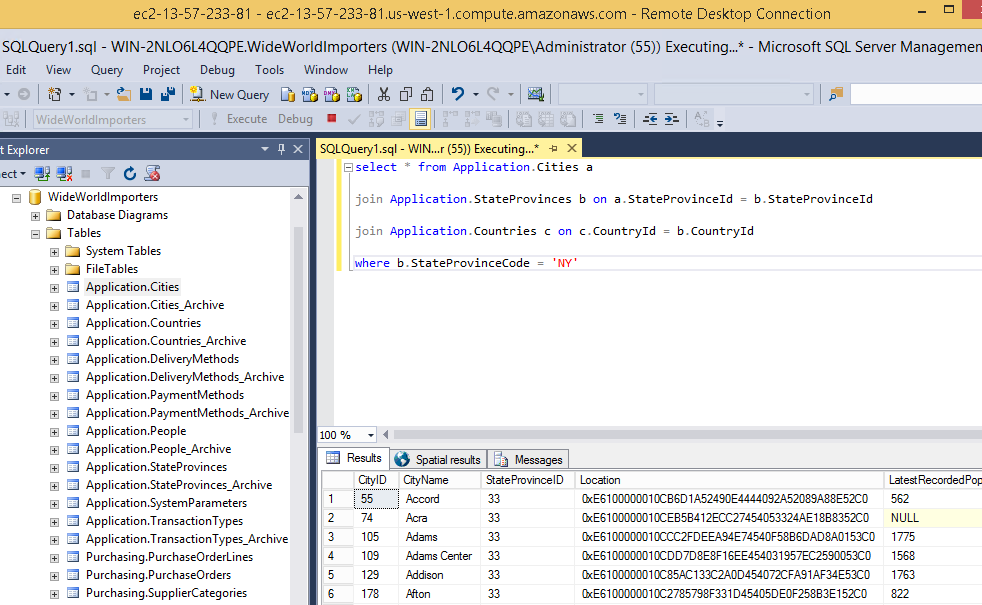
Part 2

Experiment with geospatial data queries.  
  
Familiarized myself with the SQL Server geography data types and wrote some sample queries with geospatial data using the two links provided below.  
  
Checked Introduction to SQL Server Spatial Data  
https://www.red-gate.com/simple-talk/sql/t-sql-programming/introduction-to-sql-server-spatial-data/

Checked geo Distance Using SQL Server  
https://www.codeproject.com/Tips/1185780/Calculate-Geo-Distance-Using-SQL-Server

Ran the sample geospatial queries I have posted on the Blackboard under Session 1:

Accessed the database Wide World Importers from SQL Server Management Studio.



In the Wide World Importers database and the tables that store City, StateProvince and Country information are:

Many tables are used and joined to get the details::

Application.Cities  
Application.StateProvinces

Application.Countries

Identify the relationships between them (Primary and Foreign keys).

StateProvinceId

CountryId

Find the coordinates of the location of the server you just launched. Without a GPS you can only find a rough estimate of your geolocation.

Current location:  
 Latitude: 38.8977  
 Longitude: -77.03660000000002

Found using internet the location using internet.

Write and execute a SQL that gives you the 500 closest cities based on the server's geolocation.

-- Set current location (for example New York)

declare @mylocation as geography

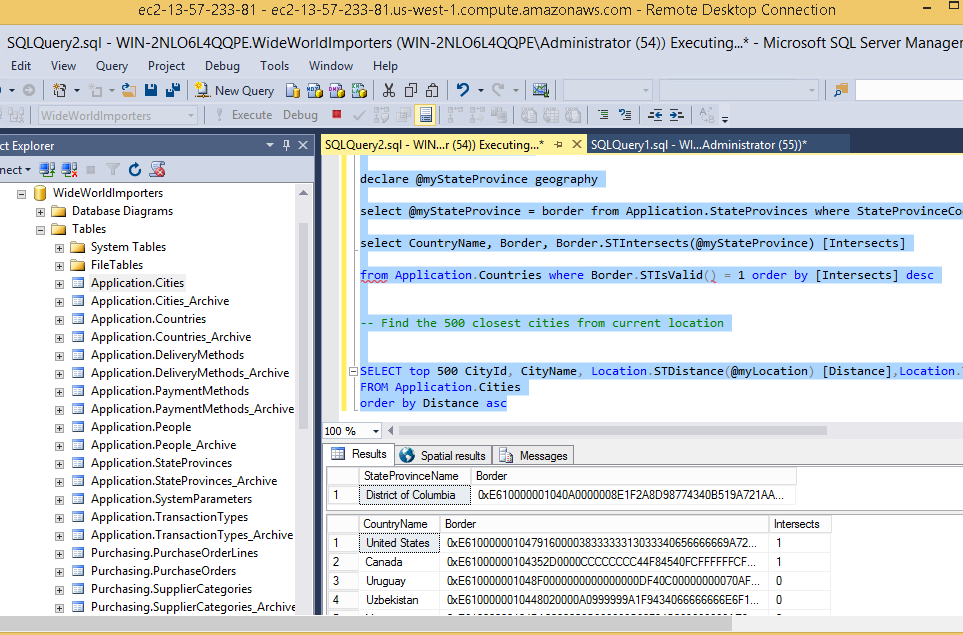
set @myLocation = geography::Point(38.8977, -77.03660000000002, 4326)

SELECT top 500 CityId, CityName, Location.STDistance(@myLocation) [Distance],Location.ToString() [Location],Location

FROM Application.Cities

order by Distance asc

Other codeDECLARE  
@GEO1 GEOGRAPHY,  
@LAT VARCHAR(10),  
@LONG VARCHAR(10)  
  
SET @LAT='38.897723.012034'  
SET @LONG='-77.03660000000002'  
  
SET @geo1= geography::Point(@LAT, @LONG, 4326)  
SELECT LOC\_ID,LOC\_NAME,(@geo1.STDistance(geography::Point(ISNULL(LAT,0), ISNULL(LONG,0), 4326))) as DISTANCE FROM LOCATION\_MASTER  
  
  
:



Write and execute a SQL that gives you the State or Province the server is at based on its geolocation.  
Write and execute a SQL that gives you the Country the server is at based on its geolocation

**-- Set current location (for example New York)**

declare @mylocation as geography

set @myLocation = geography::Point(40.7479864, -74.0008974, 4326)

**-- Select my State or Province**

select StateProvinceName, Border

from Application.StateProvinces

where Border.STIntersects(@mylocation) = 1

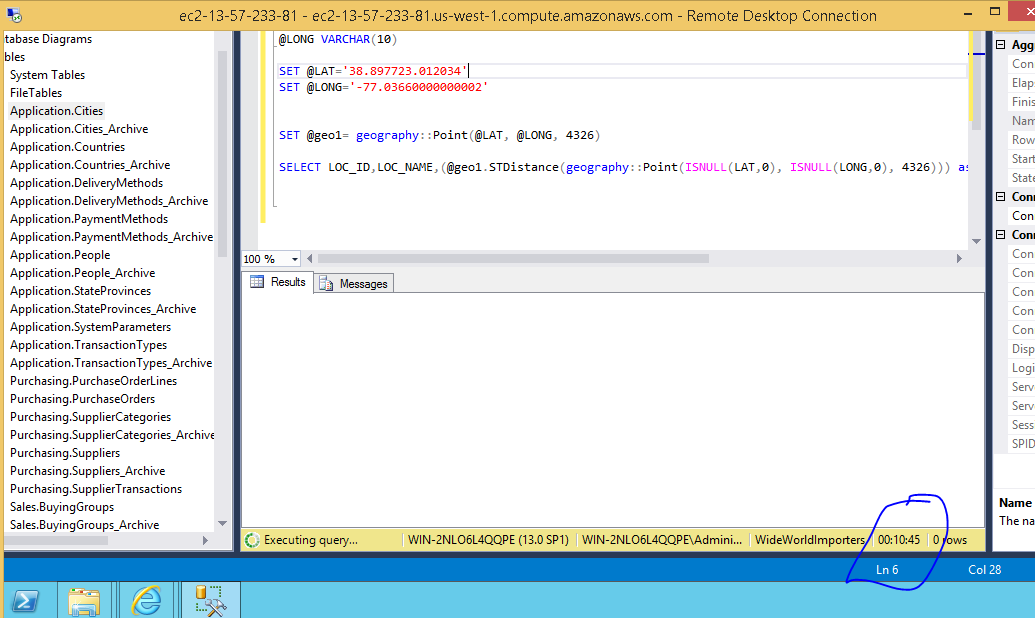
**-- select my Country/ies**

declare @myStateProvince geography

select @myStateProvince = border from Application.StateProvinces where StateProvinceCode = 'NY'

select CountryName, Border, Border.STIntersects(@myStateProvince) [Intersects]

from Application.Countries where Border.STIsValid() = 1 order by [Intersects] desc



Some of the queries took too long. I am still waiting for my $30 credit to get faster computer.