SQL GPS PROJECT

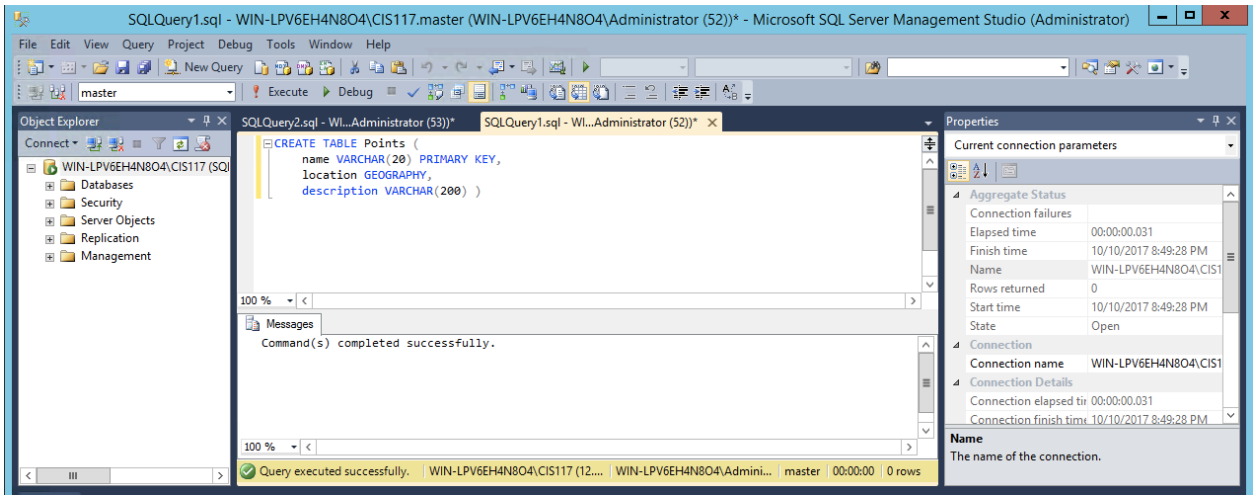
By
Nate Boyle
10/10/2017

Objectives:

- Create a table for entering and storing the name (acronym), location (GPS coordinates), and description (full name) of a given place.
- Create a spatial index for the table using ‘location’.
- Fill table up by entering data for specified places.
- Use various SQL operations to test the efficacy of the table.

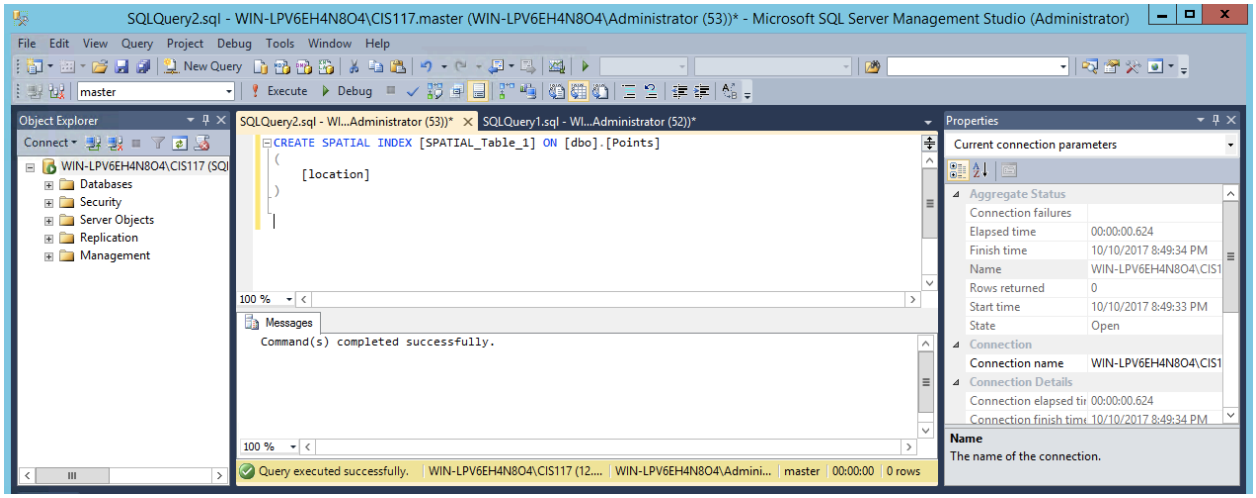
First we create a table specifying the names of the columns and their data types:

```
CREATE TABLE Points (
        name VARCHAR(20) PRIMARY KEY,
        location GEOGRAPHY,
        description VARCHAR(200) )
```



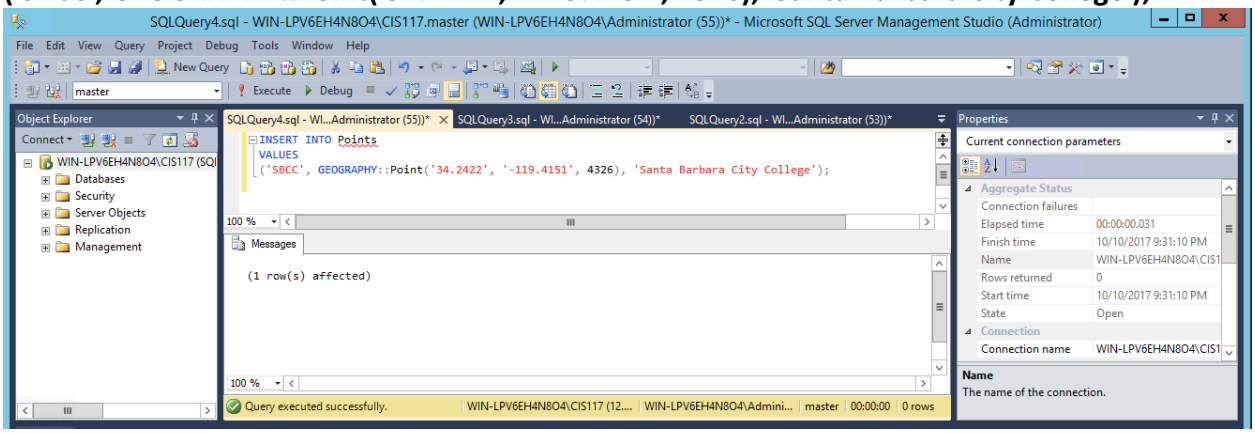
Then we create the spatial index is creating using data stored in the 'location' column:

```
CREATE SPATIAL INDEX [SPATIAL_Table_1] ON [dbo].[Points]
(
        [location]
)
```



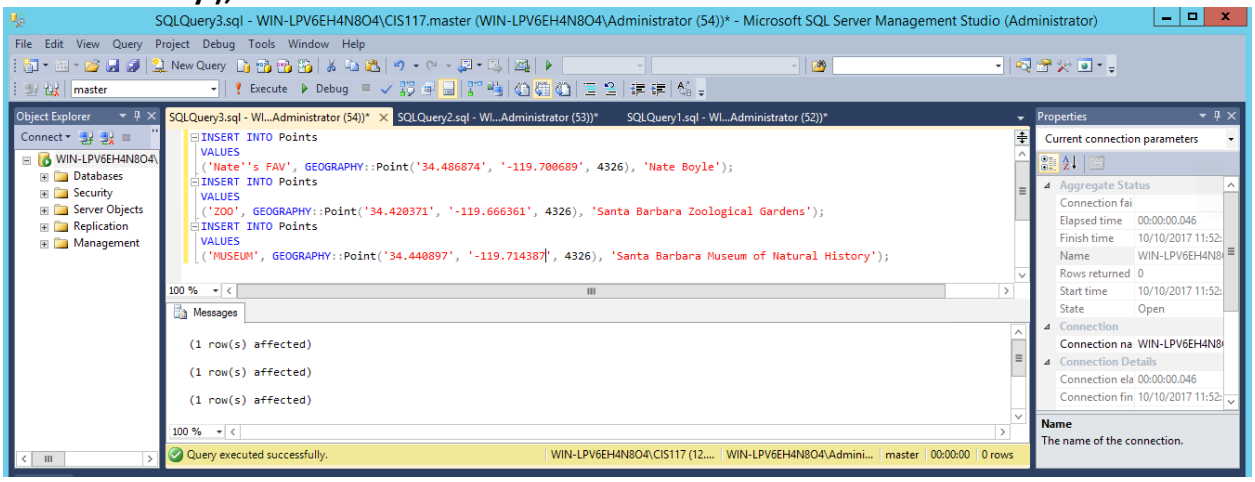
Now we enter in our first row of data, specifically for Santa Barbara City College SBCC:

INSERT INTO Points
VALUES
('SBCC', GEOGRAPHY::Point('34.2422', '-119.4151', 4326), 'Santa Barbara City College');



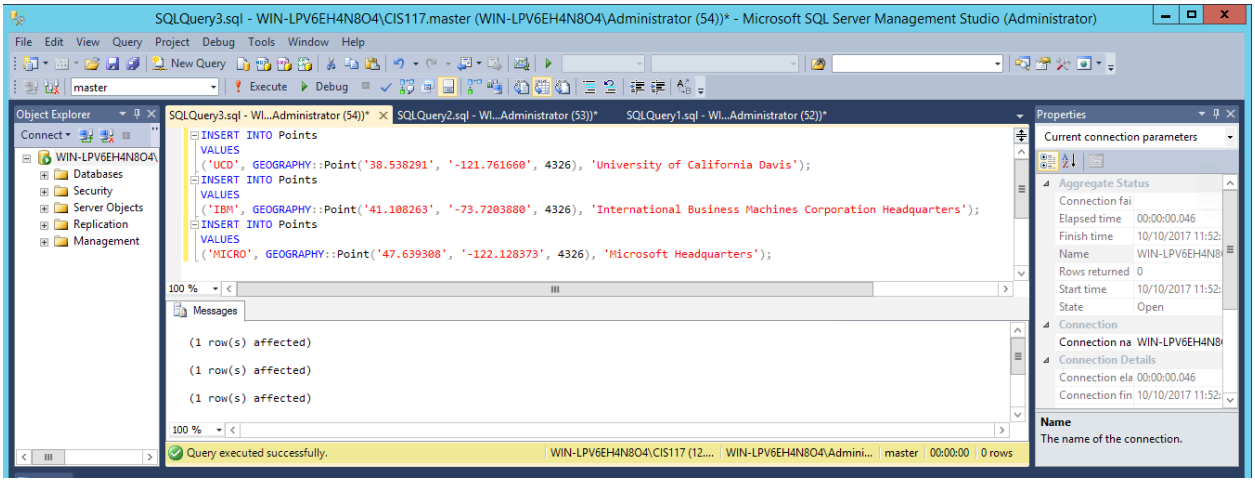
Then for three more locations within twenty miles of the original item, SBCC. (Nate's FAV is the Tunnel Road hiking trail in Santa Barbara County):

INSERT INTO Points
VALUES
('Nate''s FAV', GEOGRAPHY::Point('34.486874', '-119.700689', 4326), 'Nate Boyle');
INSERT INTO Points
VALUES
('ZOO', GEOGRAPHY::Point('34.420371', '-119.666361', 4326), 'Santa Barbara Zoological Gardens');
INSERT INTO Points
VALUES
('MUSEUM', GEOGRAPHY::Point('34.440897', '-119.714387', 4326), 'Santa Barbara Museum of Natural History');

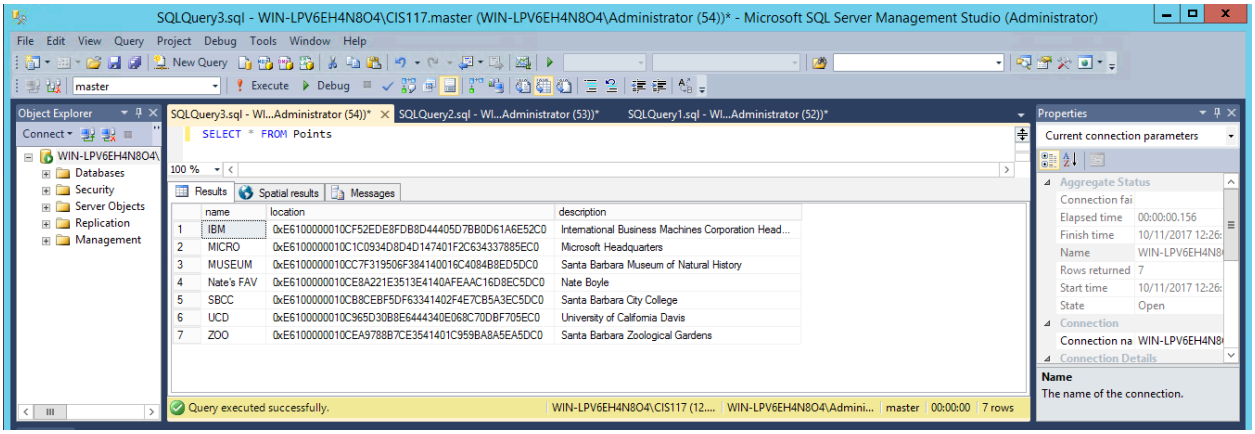


Finally, we enter in data for the last three places, which are all at least 40 miles from SBCC:

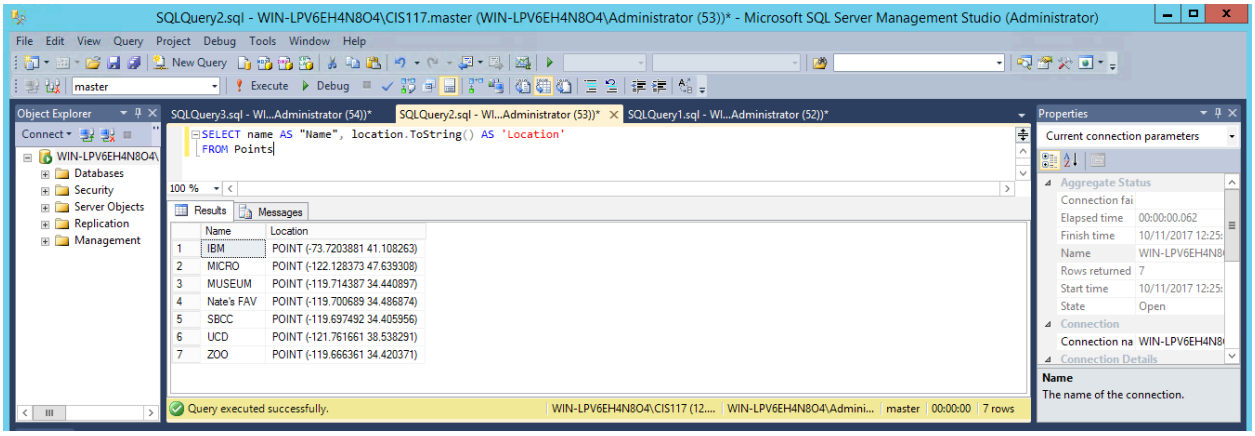
INSERT INTO Points
VALUES
('UCD', GEOGRAPHY::Point('38.538291', '-121.761661', 4326), 'University of California Davis');
INSERT INTO Points
VALUES
('IBM', GEOGRAPHY::Point('41.108263', '-73.7203881', 4326), 'International Business Machines Corporation Headquarters');
INSERT INTO Points
VALUES
('MICRO', GEOGRAPHY::Point('47.639308 ', '-122.128373', 4326), 'Microsoft Headquarters');



Here we use a SELECT * FROM query for the Points table to display our entered data. But, there is something funky going on with our 'location' column:



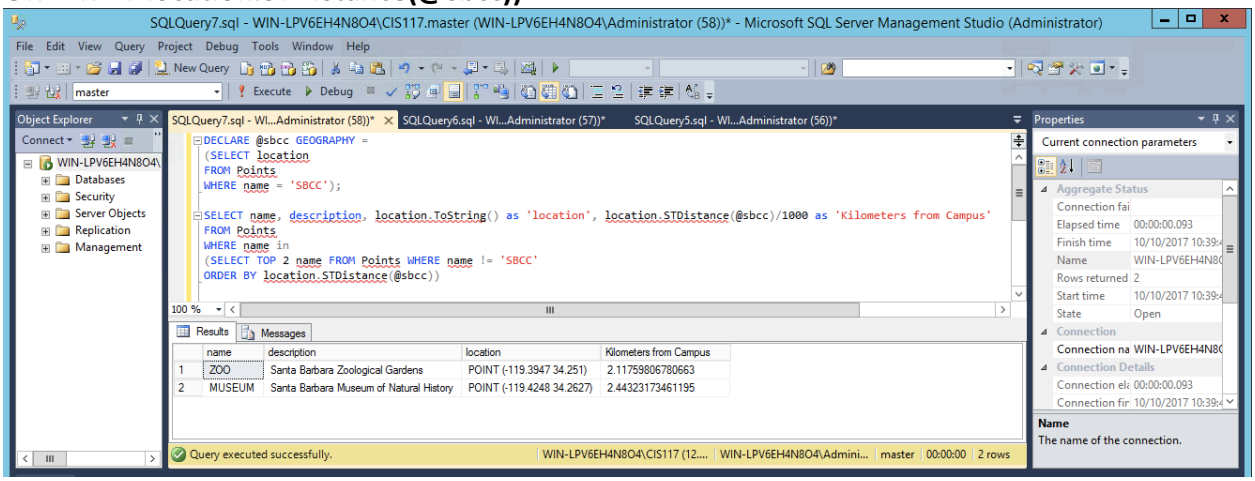
To address this, we use .ToString() attached to ‘location’:



Here we create a query used to find the two nearest places to SBCC. To do this we compare the data stored in the ‘location’ column with a predetermined variable that has ‘location’ data identical to SBCC. We also implement a TOP clause to accomplish our goal:

```
DECLARE @sbcc GEOGRAPHY =
(SELECT location
FROM Points
WHERE name = 'SBCC');

SELECT name, description, location.ToString() as 'location', location.STDistance(@sbcc)/1000
as 'Kilometers from Campus'
FROM Points
WHERE name in
(SELECT TOP 2 name FROM Points WHERE name != 'SBCC'
ORDER BY location.STDistance(@sbcc))
```

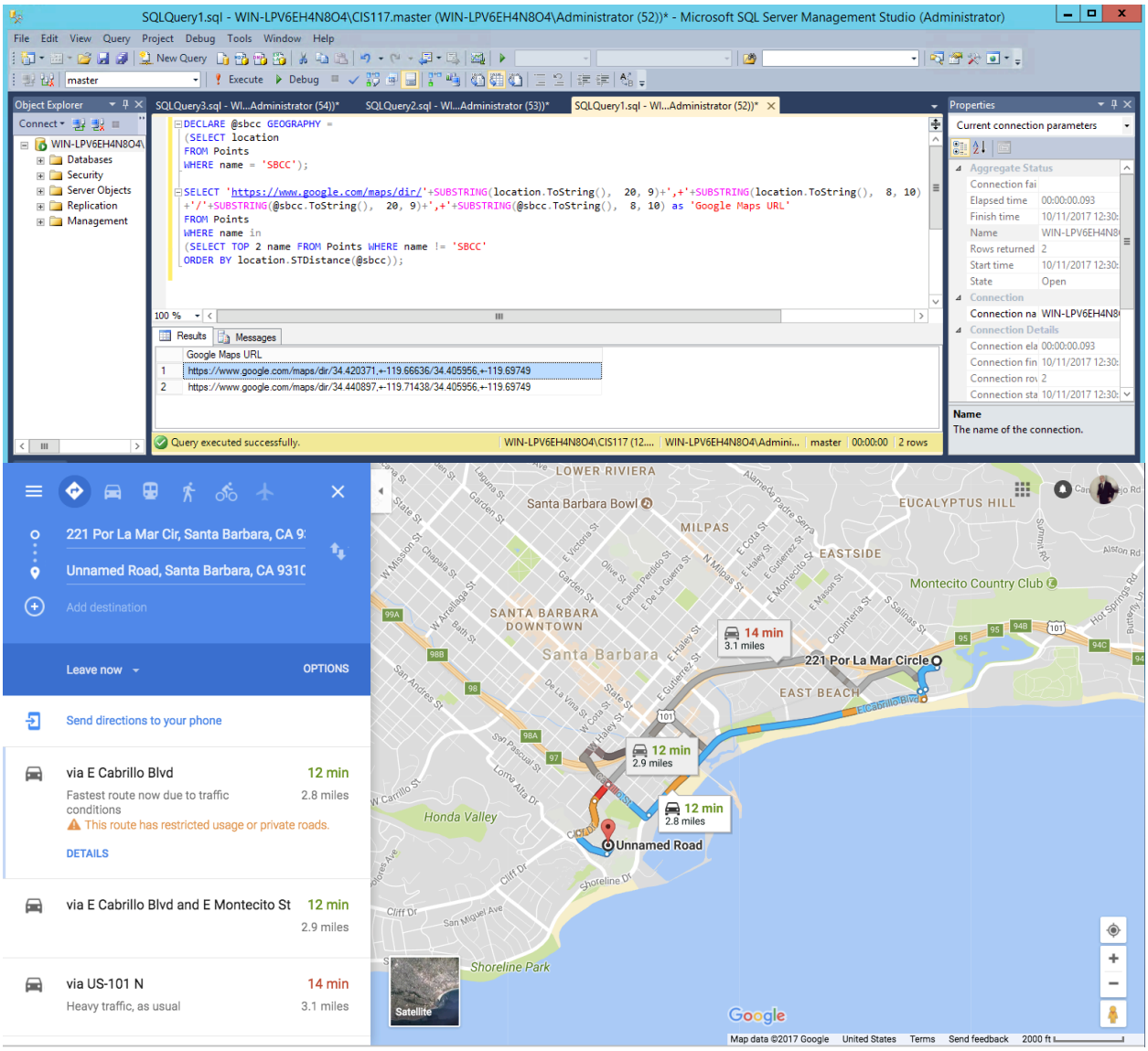


Here we create a query similar to the previous one in that it finds the two nearest places to SBCC, but it also prints out a concatenated string that be used as a Google Maps URL when copied and pasted to find the distance between SBCC and one of the two closest locations. In addition to the usual code and a screenshot of the code, a screenshot of the webpage found using the URL given is also provided:

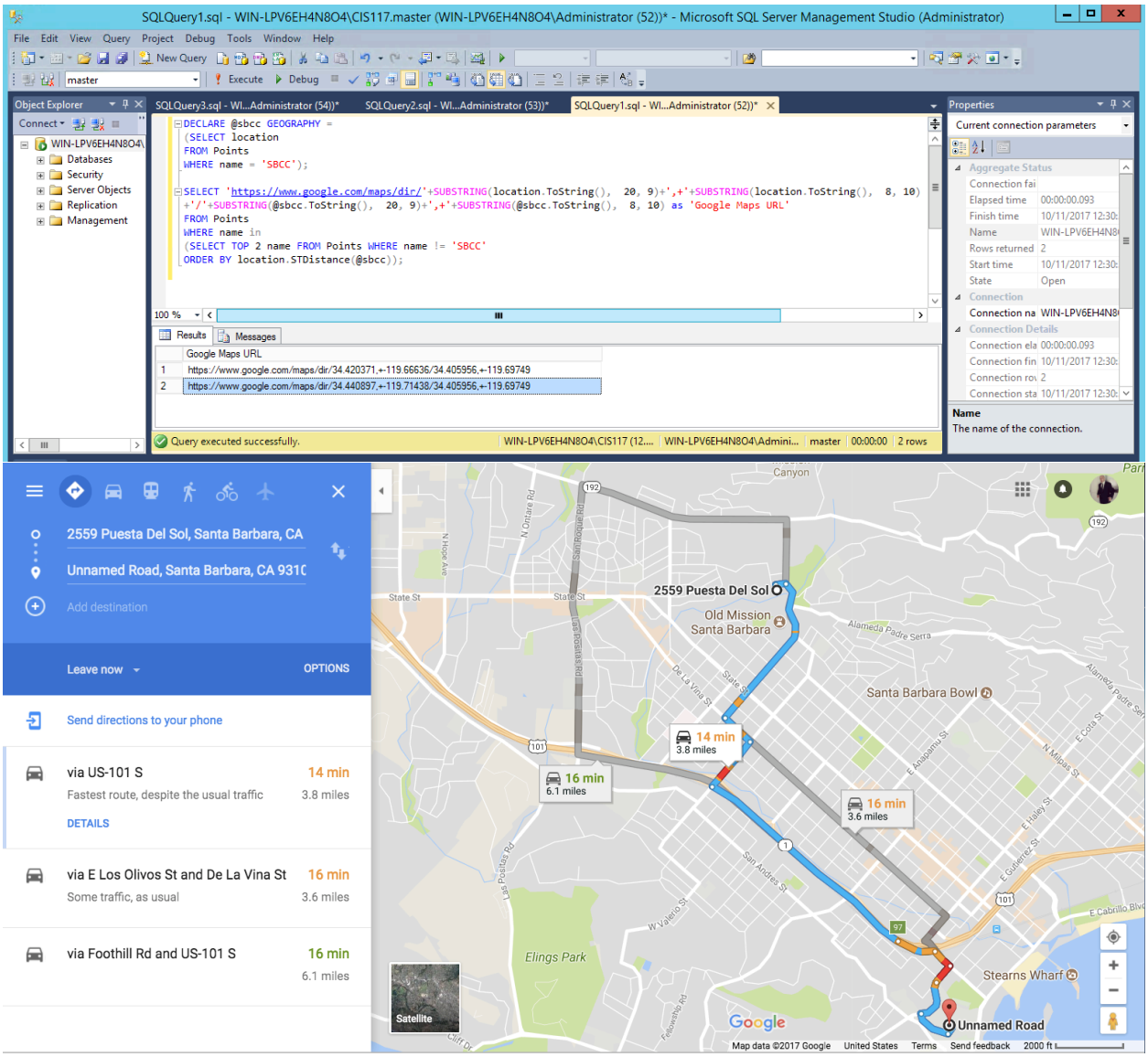
```
DECLARE @sbcc GEOGRAPHY =
(SELECT location
FROM Points
WHERE name = 'SBCC');

SELECT 'https://www.google.com/maps/dir/'+SUBSTRING(location.ToString(),  20,
9)+',+'+SUBSTRING(location.ToString(),  8, 10)
+'/'+SUBSTRING(@sbcc.ToString(),  20, 9)+',+'+SUBSTRING(@sbcc.ToString(),  8, 10) as 'Google
Maps URL'
FROM Points
WHERE name in
(SELECT TOP 2 name FROM Points WHERE name != 'SBCC'
ORDER BY location.STDistance(@sbcc));
```

Distance between SBCC and the Santa Barbara Zoo



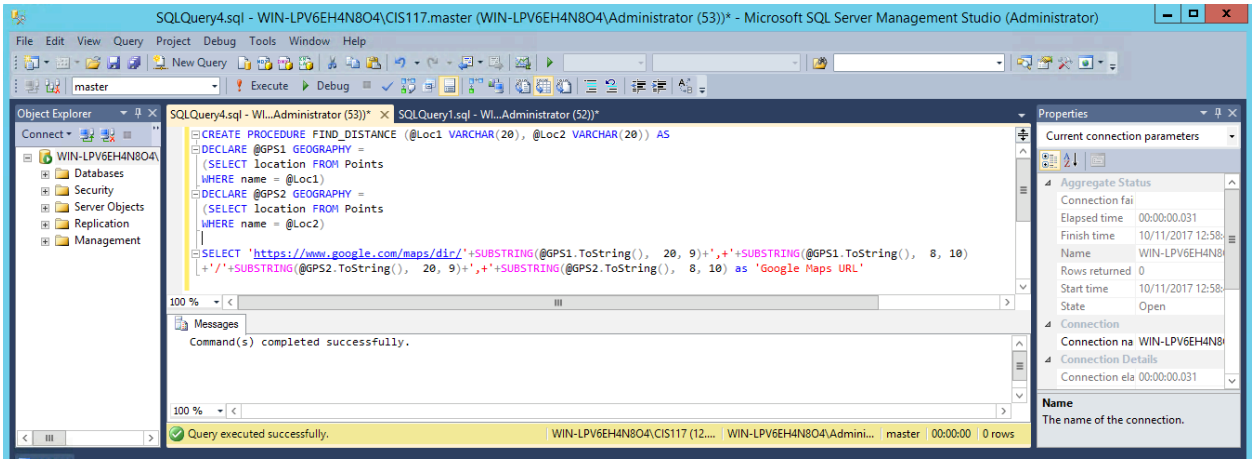
Distance between SBCC and the Santa Barbara Museum of Natural History



The previous query is useful, but has a one time use and is limited to comparing locations to a hard-coded SBCC location. To create a more flexible and even reusable query, we create a procedure that can take in any two locations and give a URL for finding the distance between them on Google Maps:

```
CREATE PROCEDURE FIND_DISTANCE (@Loc1 VARCHAR(20), @Loc2 VARCHAR(20)) AS
DECLARE @GPS1 GEOGRAPHY =
(SELECT location FROM Points
WHERE name = @Loc1)
DECLARE @GPS2 GEOGRAPHY =
(SELECT location FROM Points
WHERE name = @Loc2)

SELECT 'https://www.google.com/maps/dir/'+SUBSTRING(@GPS1.ToString(),  20, 9)+',+'+SUBSTRING(@GPS1.ToString(),  8, 10)
+'/'+SUBSTRING(@GPS2.ToString(),  20, 9)+',+'+SUBSTRING(@GPS2.ToString(),  8, 10) as 'Google Maps URL'
```



To test the efficacy of the procedure we compare the locations of IBM and Microsoft headquarters: