

# SQL GPS PROJECT

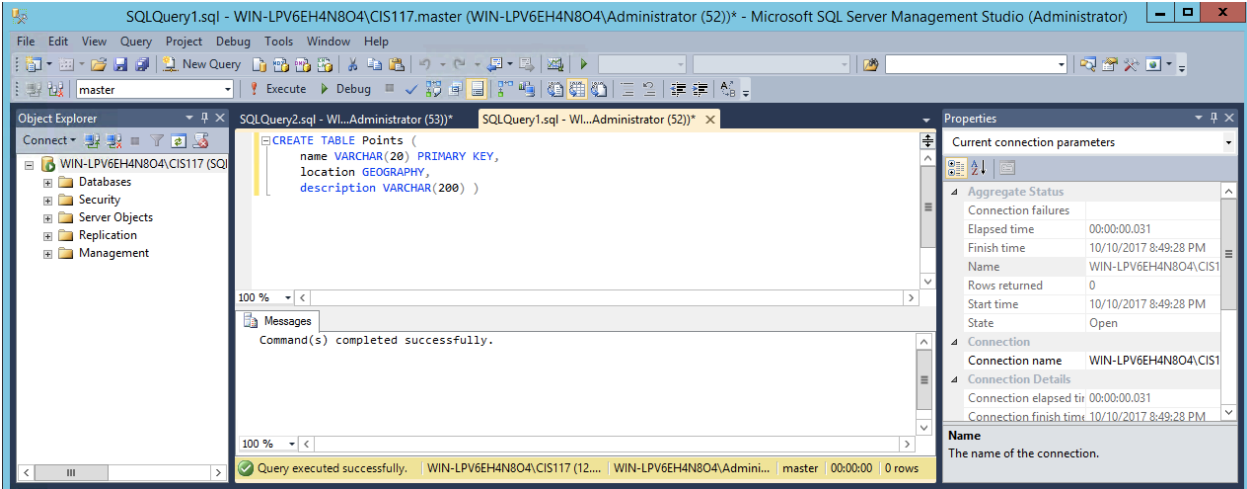
By  
Nate Boyle  
10/10/2017

## 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.

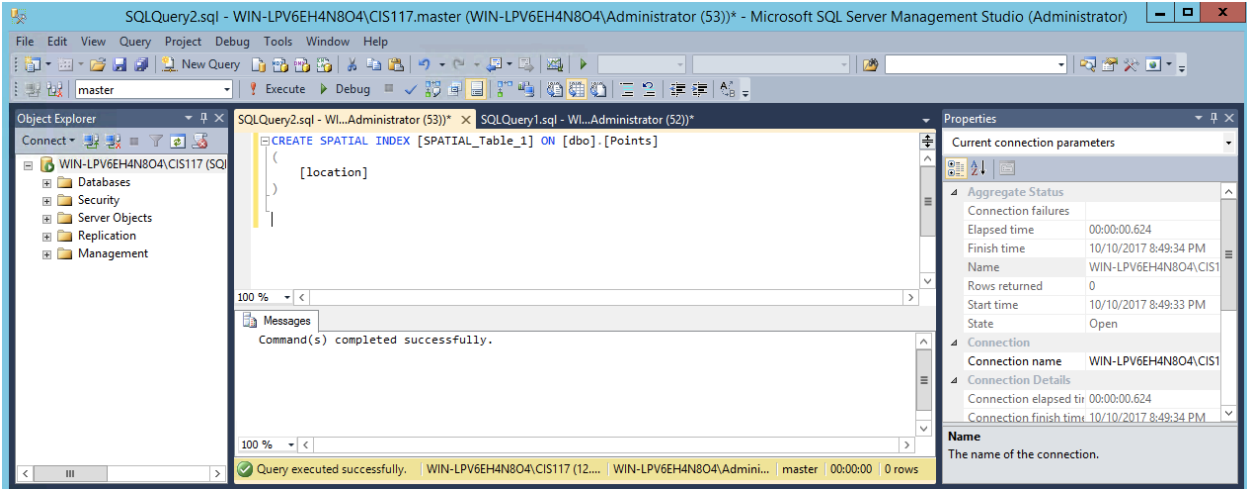
First we create a table specifying the names of the columns and their data types:

```
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:

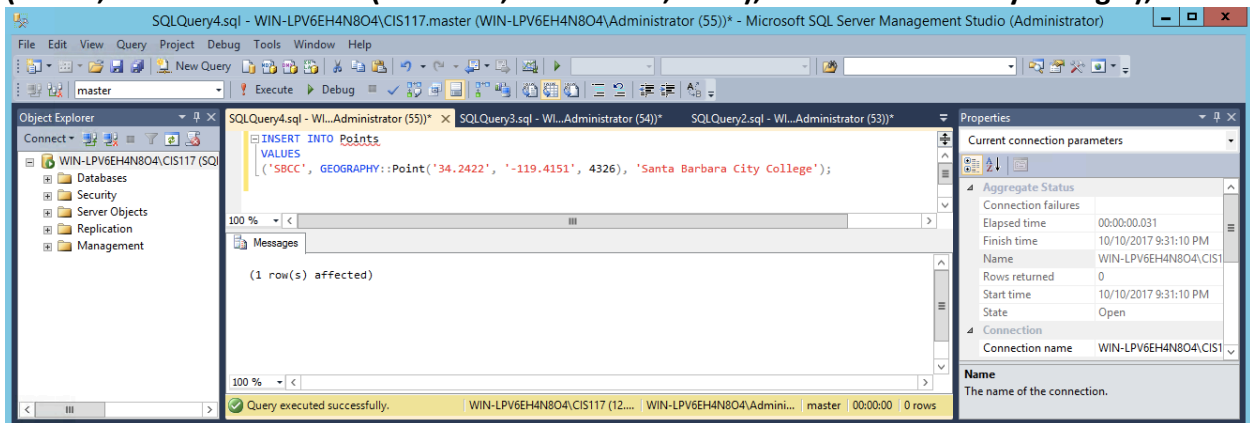
```
CREATE SPATIAL INDEX [SPATIAL_Table_1] ON [dbo].[Points]  
(  
    [location]  
)
```



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**

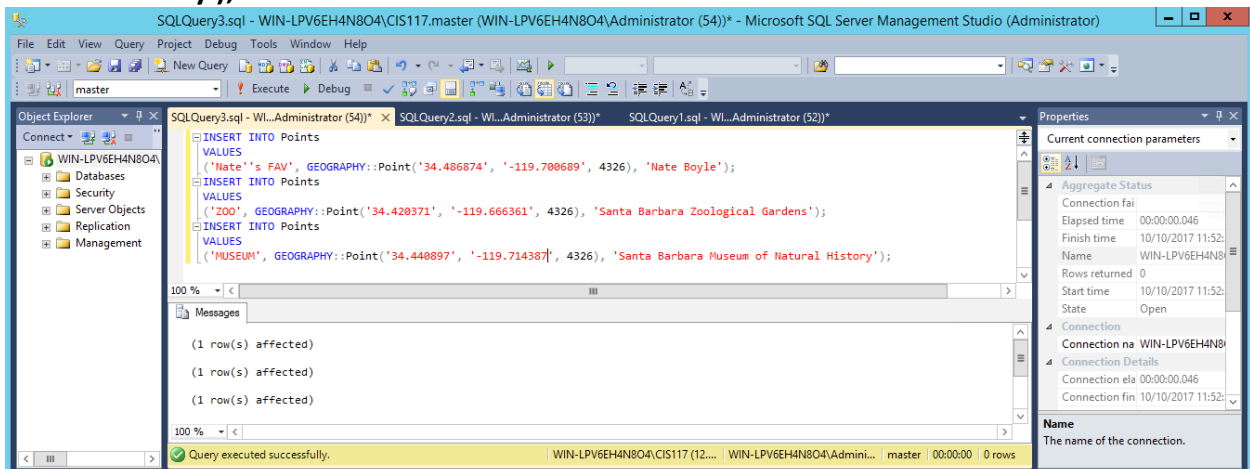
**('Nate"s FAV', GEOGRAPHY::Point('34.486874', '-119.700689', 4326), 'Nate Boyle');**

**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**

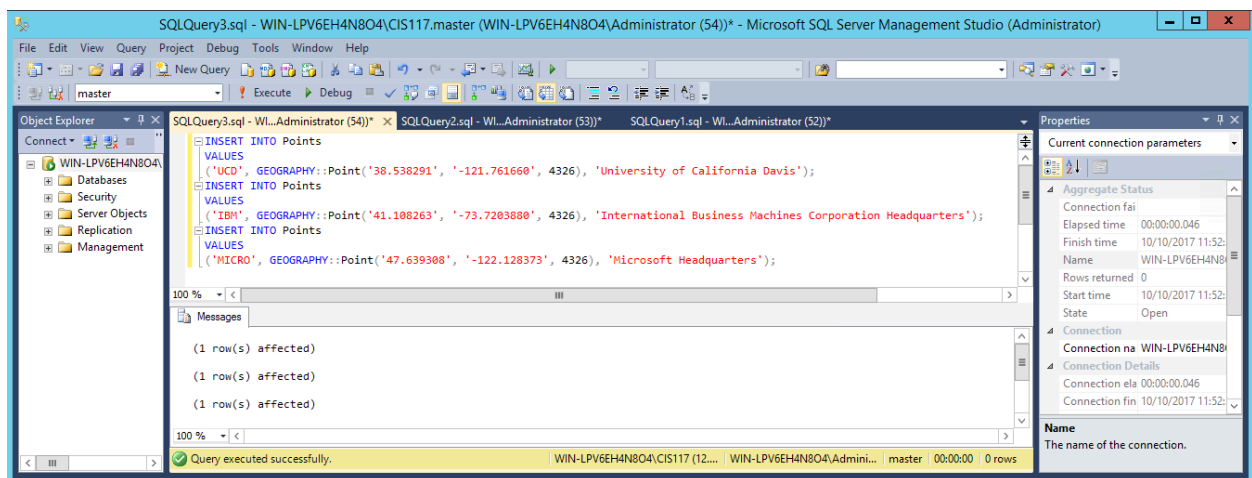
**('UCD', GEOGRAPHY::Point('38.538291', '-121.761661', 4326), 'University of California Davis');**

**INSERT INTO Points  
VALUES**

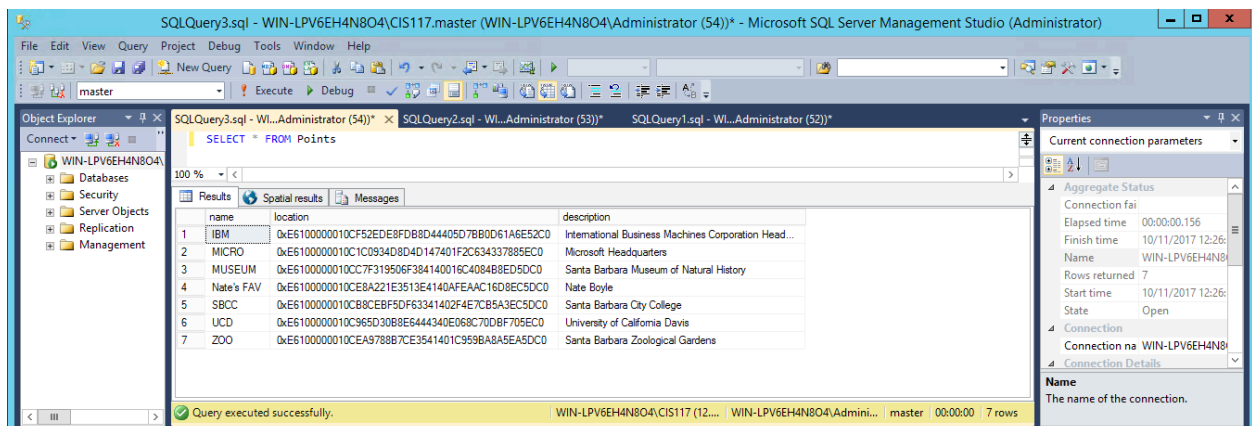
**('IBM', GEOGRAPHY::Point('41.108263', '-73.7203881', 4326), 'International Business  
Machines Corporation Headquarters');**

**INSERT INTO Points  
VALUES**

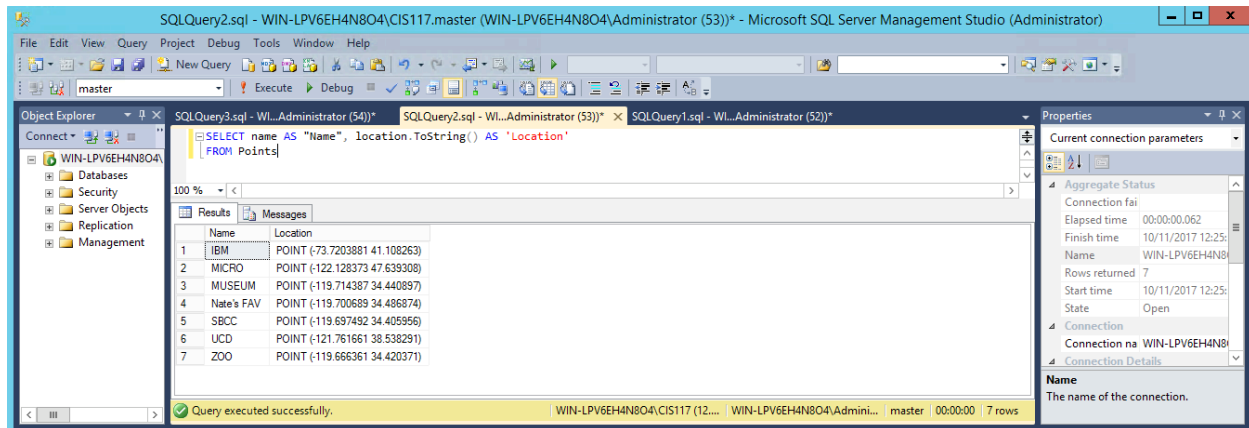
**('MICRO', GEOGRAPHY::Point('47.639308', '-122.128373', 4326), 'Microsoft Headquarters');**



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:



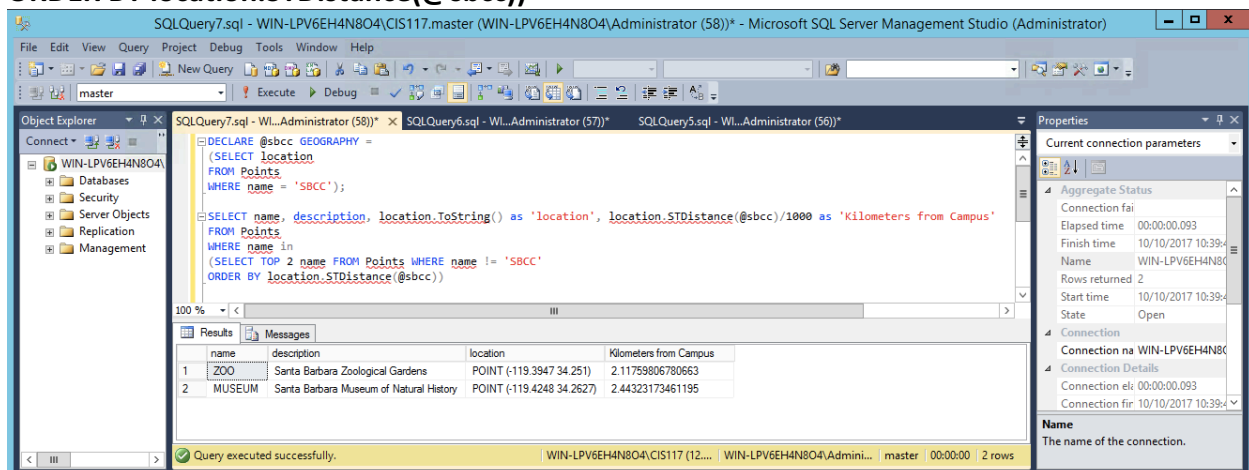
To address this, we use `.ToString()` attached to 'location':



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\*\*\*

The screenshot shows the Microsoft SQL Server Management Studio interface. The main window displays a T-SQL query in the Query Editor:

```
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));
```

The Results pane shows the output of the query:

Google Maps URL
1 https://www.google.com/maps/dir/34.420371,-119.66636/34.405956,-119.69749
2 https://www.google.com/maps/dir/34.440837,-119.71438/34.405956,-119.69749

The Properties pane on the right shows connection details for the current connection.

The screenshot shows Google Maps with a route from Santa Barbara to the Santa Barbara Zoo. The starting point is "221 Por La Mar Cir, Santa Barbara, CA 93102" and the destination is "Unnamed Road, Santa Barbara, CA 93102".

The map shows several routes with their respective distances and estimated travel times:

- via E Cabrillo Blvd: 12 min, 2.8 miles. Fastest route now due to traffic conditions. This route has restricted usage or private roads.
- via E Cabrillo Blvd and E Montecito St: 12 min, 2.9 miles.
- via US-101 N: 14 min, 3.1 miles. Heavy traffic, as usual.

The map also shows the Santa Barbara Zoo location and the route from the starting point to the zoo.

\*\*\*Distance between SBCC and the Santa Barbara Museum of Natural History\*\*\*

The screenshot shows the Microsoft SQL Server Management Studio interface. The main window displays a SQL query in the Query Editor:

```
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));
```

The Results pane shows two rows of Google Maps URLs:

Google Maps URL
1 https://www.google.com/maps/dir/34.420371,-119.66636/34.405956,-119.69749
2 https://www.google.com/maps/dir/34.440897,-119.71438/34.405956,-119.69749

The Properties pane on the right shows connection details for the current connection.

The screenshot shows Google Maps with directions from 2559 Puesta Del Sol, Santa Barbara, CA to Unnamed Road, Santa Barbara, CA 931C. The map displays three route options:

- via US-101 S**: 14 min, 3.8 miles (Fastest route, despite the usual traffic)
- via E Los Olivos St and De La Vina St**: 16 min, 3.6 miles (Some traffic, as usual)
- via Foothill Rd and US-101 S**: 16 min, 6.1 miles

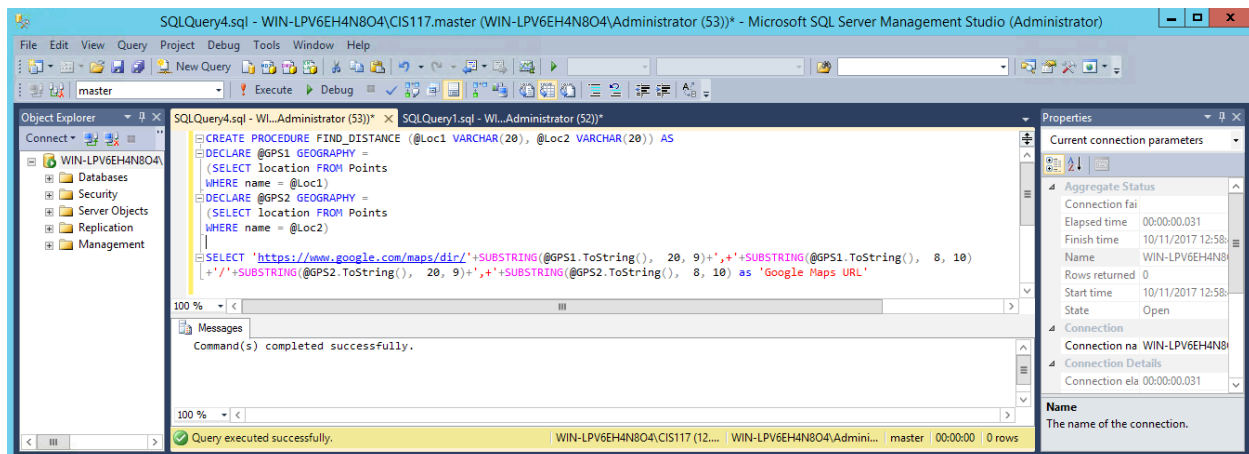
The map also shows landmarks like Elings Park and Santa Barbara Bowl, and major roads like US-101 and State St.



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:

The image displays two overlapping windows. The top window is Microsoft SQL Server Management Studio (SSMS) running a query in the 'master' database. The query is `EXEC FIND_DISTANCE 'IBM', 'MICRO'`. The results pane shows a single row with a Google Maps URL: `https://www.google.com/maps/dir/41.108263,+73.720388/47.639308,-122.12837`. The status bar indicates the query executed successfully.

The bottom window is Google Maps. The starting point is marked with a red pin at coordinates 41.108263, 73.720388 (IBM headquarters). The destination is marked with a blue pin at coordinates 47.639308, -122.12837 (Microsoft headquarters). A blue route is shown across the United States, with a callout indicating a travel time of 43 hours and a distance of 2,874 miles. The route is described as 'via I-80 W, I-94 W and I-90 W'. Additional information includes: 'Fastest route now, avoids road closures on I-80 W', 'This route has tolls', and 'Your destination is in a different time zone'. The map shows the United States, Canada, and Mexico, with state and provincial boundaries labeled.