SDS Bootcamp - London 2023

Using SQL to Scale your Spatial Analytics

In this session you will learn:

- How different clouds are supporting geospatial data
- How CARTO is extending these clouds to offer analytical capabilities to extract insighful knowledge from you geospatial data
- How to do sptial analytics using SQL

The session is organized into 3 subsections:

1. **SQL Basics for spatial analytics**
   You'll learn how to make basic operations on geospatial data such us selecting events happening in an area or using spatial indexes for visualizing large amounts of data.
2. **Analyzing crime data for city management, Real Estate investment, and insurance underwriting**
   You'll learn how to use advanced functions to build advanced analytics workflows.
3. **Other advanced analyses**
   You'll learn about other interesting analytics functions and their applications to real-world use cases.

Requirements

Platform
You only need to have a CARTO trial account to follow this session.

Data
Note that all data used in this guide is publicly available through CARTO's Data Observatory or BigQuery public project `bigquery-public-data`. In addition, we have given public access to samples to of data for a smooth experience during the workshop.

Geospatial Data Support on the Cloud

- BigQuery
- Snowflake
- Redshift
- Databricks

Section 1. SQL Basics for Spatial Analytics

Explore POI data

Let’s start visualizing some OSM POI data in London. Note we have made a sample publicly available. If you’re interested in accessing the entire dataset, you can find it as public data through:

- CARTO data Observatory (note this requires subscription).
- BigQuery public data project `bigquery-public-data`

Copy-paste the following code on your Builder console.

```
SELECT geometry AS geom
FROM `cartobq.docs.sdsb_london_osm_pois`
```

Next, let’s filter only those POI within a specific bounding box. Find your bounding box of interest here.

- ST_INTERSECTSBOX

```
SELECT geometry AS geom
FROM `cartobq.docs.sdsb_london_osm_pois`
WHERE ST_INTERSECTSBOX(geometry, -0.130119,51.449300,-0.096817,51.549751)
```

Let’s now learn how to filter OSM POI data. We’d like to filter restaurants and fast food locations within a bounding box.

```
SELECT CAST(id AS STRING) AS id , tag.value, geometry as geom
FROM `cartobq.docs.sdsb_london_osm_pois`, UNNEST(all_tags) as tag
WHERE ST_INTERSECTSBOX(geometry, -0.130119,51.449300,-0.096817,51.549751) AND
     (tag.value in ('fast_food', 'restaurant') AND tag.key = 'amenity')
```

We can also select a buffer area. Suppose we’d like to know all restaurants and fast food locations around Wallacespace St Pancras.

- ST_GEOGPOINT
- ST_BUFFER
- ST_INTERSECTS

```
SELECT ST_BUFFER(ST_GEOGPOINT(-0.128716, 51.526864), 3000) AS geom
```

```
SELECT CAST(id AS STRING) AS id , tag.value, geometry as geom
FROM `cartobq.docs.sdsb_london_osm_pois`, UNNEST(all_tags) as tag
WHERE ST_INTERSECTS(geometry, ST_BUFFER(ST_GEOGPOINT(-0.128716, 51.526864), 3000))
AND
     (tag.value in ('fast_food', 'restaurant') AND tag.key = 'amenity')
```

Explore polygon data

Let’s now analyze some polygon data and combine it with point data.

We’ll use the Retail Centre Boundaries dataset developed by the Consumer Data Research Centre (CDRC).

```
SELECT *
FROM `cartobq.docs.sdsb_london_retail_centres`
WHERE ST_INTERSECTSBOX(geom, -0.230793,51.431280,0.083004,51.568064)
```

We are now interested in combining the two datasets to know the POI density in each retail centre. Note how the table join is performed with a geometric operation.

```
SELECT t2.geoid, ANY_VALUE(t2.RC_name) AS RC_name, ANY_VALUE(t2.Classification) AS
Classification, ANY_VALUE(t2.geom) AS geom,
COUNT(t2.Classification)/ST_AREA(ANY_VALUE(t2.geom)) AS poi_density
FROM `cartobq.docs.sdsb_london_osm_pois` t1
JOIN `cartobq.docs.sdsb_london_retail_centres` t2
ON ST_INTERSECTS(t1.geometry, t2.geom)
GROUP BY t2.geoid
```

Aggregate data using spatial indexes

What are spatial indexes?

Explore POI density using spatial indexes.

- H3_FROMGEOGPOINT

```
SELECT h3, COUNT(h3) as n_pois
FROM (
 SELECT`carto-un`.carto.H3_FROMGEOGPOINT(geometry, 9) AS h3
 FROM `cartobq.docs.sdsb_london_osm_pois`
)
GROUP BY h3
```

Section 2. Analyzing crime data for city management, Real Estate investment, and insurance underwriting

Understand the data

We start by taking a look at crime data in Chicago. This data is publicly available in BigQuery's public project `bigquery-public-data`. In particular, crime data of the city of Chicago is availablew in the following table.

```
`bigquery-public-data.chicago_crime.crime`
```

Let's see the most common crimes in 2019.

```
SELECT primary_type, COUNT(unique_key) AS ncount
FROM `bigquery-public-data.chicago_crime.crime`
WHERE year = 2019
GROUP BY primary_type
ORDER BY ncount DESC
```

We take 2019 burglary as our data of interest. This data has been made available in table `cartobq.docs.sdsb_chicago_burglary_crime_sample`.

```
SELECT date, description, location_description, ST_GEOGPOINT(longitude, latitude) AS
geom
FROM `cartobq.docs.sdsb_chicago_burglary_crime_sample`
```

We can add a widget with crime descriptions to evaluate them. We can observe that burglary crimes seem to be equally distributed throughout the city.

Cluster crime data

Suppose we'd like to open 5 new police stations or that we'd like to organize police in 5 areas so that police officers are as close as possible no burglary crimes. In order to identify these areas, we run a k-means cluster analysis.

- ST_CLUSTERKMEANS

```
WITH clustered_points AS
(
   SELECT `carto-un`.carto.ST_CLUSTERKMEANS(ARRAY_AGG(ST_GEOGPOINT(longitude,
latitude)), 5) AS cluster_arr
   FROM `cartobq.docs.sdsb_chicago_burglary_crime_sample`
)
SELECT cluster_element.cluster, cluster_element.geom AS geom FROM clustered_points,
UNNEST(cluster_arr) AS cluster_element
```

We can calculate the new police stations as the centroids or center-medians of all crime fallin in the same cluster.

- ST_CENTERMEDIAN

```
WITH clustered_points AS
(
   SELECT `carto-un`.carto.ST_CLUSTERKMEANS(ARRAY_AGG(ST_GEOGPOINT(longitude,
latitude)), 5) AS cluster_arr
   FROM `cartobq.docs.sdsb_chicago_burglary_crime_sample`
)
```

```
SELECT cluster_element.cluster,
`carto-un`.carto.ST_CENTERMEDIAN(ST_UNION_AGG(cluster_element.geom)) AS geom FROM
clustered_points, UNNEST(cluster_arr) AS cluster_element
GROUP BY cluster_element.cluster
```

High-density clustering

We're now interested in finding locations with high concentrations of burglary crime. This information can be valuable in order to identify where to allocate resources or take further action for many different purposes: city management, insurance, real estate.

- ST_CLUSTERDBSCAN

```
WITH crimes AS (
   SELECT ST_GEOGPOINT(longitude, latitude) AS geom
   FROM `cartobq.docs.sdsb_chicago_burglary_crime_sample`
)
SELECT geom, ST_CLUSTERDBSCAN(geom, 100, 10) OVER () AS cluster_num
FROM crimes
```

Spatial outliers

Let's now analyze the data aggregated using H3.

- H3_FROMLONGLAT

```
SELECT h3, COUNT(h3) as n_crimes
FROM (
 SELECT `carto-un`.carto.H3_FROMLONGLAT(longitude, latitude, 8) as h3
 FROM `cartobq.docs.sdsb_chicago_burglary_crime_sample`
)
GROUP BY h3
```

We'd like to know if there is higher concentrations of crime in specific parts of the city and if there are spatial outliers, i.e., locations with a low expected number of crimes experiencing high numbers of crime, or vice versa.

We start by calculating the spatial autocorrelation. It is important to note that we should fill up the gaps with 0 crime so that they are counted as areas with low (actually, non-existing crime). In order to fill these gaps, we need to define our area of interest and polyfill it with the corresponding grid.

For our area of interest, we take Chicago's boundary. This data has been made available for this workshop in table `cartobq.docs.sdsb_chicago_boundaries`. It was originally retrieved from Who's On First geography table, publicly available (through subscription) in CARTO's data Observatory.

- H3_POLYFILL
- MORANS_I_H3

```
WITH chicago_polyfilled AS (
SELECT h3
FROM UNNEST(`carto-un`.carto.H3_POLYFILL((SELECT geom h3array FROM
`cartobq.docs.sdsb_chicago_boundaries`), 8)) AS h3
),
aggregated_crime AS (
SELECT h3, COUNT(h3) AS n_crimes
FROM (
 SELECT `carto-un`.carto.H3_FROMLONGLAT(longitude, latitude, 8) as h3
 FROM `cartobq.docs.sdsb_chicago_burglary_crime_sample`
)
GROUP BY h3
)
SELECT `carto-un`.carto.MORANS_I_H3(input, 1, 'exponential')
FROM (
SELECT ARRAY_AGG(STRUCT(cp.h3 AS index, CAST(IFNULL(aggt.n_crimes, 0) AS FLOAT64) AS
value)) AS input
FROM chicago_polyfilled cp
LEFT JOIN aggregated_crime aggt
USING(h3)
)
```

We can see there is a relatively high spatial autocorrelation, which makes sense. However, there seem to be areas which are not similar to their surroundings. We'd like to see where these areas are and understand what's happening there, i.e., what are the characteristics of these locations compared to the surrounding areas.

For this analysis, we'll use LOCAL_MORANS_I_H3. Since these values depend very much on the original ones (i.e., number of crime, a good practice is to use the p-value because it gives us an objective criteria to determine what's significant and what's not).

```
WITH chicago_agg_crime AS (
WITH chicago_polyfilled AS (
  SELECT h3
  FROM UNNEST(`carto-un`.carto.H3_POLYFILL((SELECT geom h3array FROM
`cartobq.docs.sdsb_chicago_boundaries`), 8)) AS h3
),
aggregated_crime AS (
  SELECT h3, COUNT(h3) AS n_crimes
  FROM (
     SELECT `carto-un`.carto.H3_FROMLONGLAT(longitude, latitude, 8) as h3
     FROM `cartobq.docs.sdsb_chicago_burglary_crime_sample`
  )
  GROUP BY h3
)
SELECT cp.*, IFNULL(aggt.n_crimes, 0) AS n_crimes
FROM chicago_polyfilled cp
LEFT JOIN aggregated_crime aggt
USING(h3)
)
SELECT i.* EXCEPT(index), index AS h3,
REPLACE(REPLACE(REPLACE(REPLACE(CAST(i.quad as string), '1', 'HH'), '2', 'LL'), '3',
'LH'), '4', 'HL') AS clust
FROM UNNEST((`carto-un`.carto.LOCAL_MORANS_I_H3(
      ARRAY(SELECT AS STRUCT h3, CAST(n_crimes AS FLOAT64) AS value FROM
chicago_agg_crime),
      1, 'uniform', 100
  ))) AS i
```

Finally, let's try to understand what is different in each of these four clusters (LL, HH, LH, HL). We'll describe them in terms of demographics and socioeconomic variables. For that, we use ACS data, opublicly available through CARTO's Data Observatory.

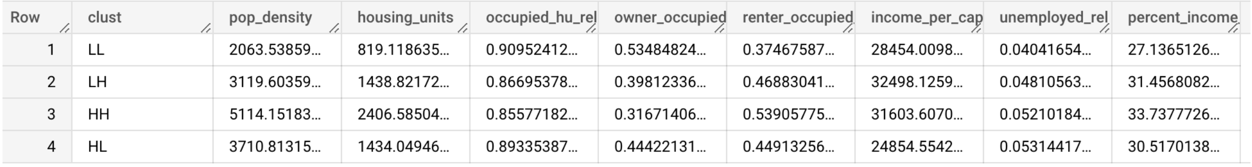
The following query enriches our grid with the selection of variables made.

- Sociodemographics - United States of America (Census Block Group, 2018, 5yrs)

```
CALL `carto-un`.carto.DATAOBS_ENRICH_GRID(
  -- grid type
  'h3',
  -- input query
  R'''
  SELECT *
  FROM `cartobq.docs.sdsb_chicago_h3_polyfill`
  ''',
  -- spatial index column name
  'h3',
  -- enrichment variables
  [('total_pop_3409f36f', 'sum'),
   ('income_per_capi_bfb55c80', 'avg'),
   ('housing_units_ba2efbd2', 'sum'),
   ('occupied_housin_aaffda96', 'sum'),
   ('housing_units_r_cd127087', 'sum'),
   ('owner_occupied__e07aeb89', 'sum'),
   ('percent_income__93d611a5', 'avg'),
   ('unemployed_pop_e1bb4940', 'sum')],
   -- filters
  NULL,
  -- output
  ['`cartobq.docs.sdsb_chicago_h3_polyfill_enriched`'],
  -- source
  '<my-project>.<my-dataset>'
)
```

At this point, we can compute some statistics by cluster and analyze the differences among the different clusters.

```
WITH enriched_clustered AS (
  SELECT t1.*, t2.clust
  FROM `cartobq.docs.sdsb_chicago_h3_polyfill_enriched` t1
  JOIN `cartobq.docs.sdsb_chicago_burglary_outliers` t2
  ON t1.h3=t2.h3
)
SELECT clust,
      AVG(total_pop_3409f36f_sum) AS pop_density,
      AVG(housing_units_ba2efbd2_sum) AS housing_units,
      SUM(occupied_housin_aaffda96_sum)/SUM(housing_units_ba2efbd2_sum) AS
occupied_hu_rel,
      SUM(owner_occupied__e07aeb89_sum)/SUM(housing_units_ba2efbd2_sum) AS
owner_occupied_hu_rel,
      SUM(housing_units_r_cd127087_sum)/SUM(housing_units_ba2efbd2_sum) AS
renter_occupied_hu_rel,
      AVG(income_per_capi_bfb55c80_avg) AS income_per_capita,
      SUM(unemployed_pop_e1bb4940_sum)/SUM(total_pop_3409f36f_sum) AS
unemployed_rel,
      AVG(percent_income__93d611a5_avg) AS percent_income_spent_on_rent
FROM enriched_clustered
GROUP BY clust
```



Row	clust	pop_density	housing_units	occupied_hu_rel	owner_occupied	renter_occupied	income_per_cap	unemployed_rel	percent_income
1	LL	2063.53859...	819.118635...	0.90952412...	0.53484824...	0.37467587...	28454.0098...	0.04041654...	27.1365126...
2	LH	3119.60359...	1438.82172...	0.86695378...	0.39812336...	0.46883041...	32498.1259...	0.04810563...	31.4568082...
3	HH	5114.15183...	2406.58504...	0.85577182...	0.31671406...	0.53905775...	31603.6070...	0.05210184...	33.7377726...
4	HL	3710.81315...	1434.04946...	0.89335387...	0.44422131...	0.44913256...	24854.5542...	0.05314417...	30.5170138...

Section 3. Other advanced analyses

Geographically Weighted regression

```
CALL `carto-un`.carto.GWR_GRID(
 -- input_table
 'cartobq.docs.sdsb_airbnb_berlin',
 -- regressors [ beds, bathrooms, review score ]
 ['bedrooms', 'bathrooms', 'review_scores_value'],
 -- target variable [ price ]
 'price',
 -- cell_column
 'h3_z8',
 -- algorithm parameters
 'h3', 3, 'gaussian', TRUE,
 -- output table
 'cartobq.docs.sdsb_airbnb_berlin_gwr_output'
)
```

Isochrones (15min city)

https://www.theguardian.com/us-news/2023/jan/25/15-minute-city-urban-planning-future-us-cities

Schools

```
SELECT CAST(id AS STRING) AS id , tag.value, geometry as geom
FROM `cartobq.docs.sdsb_london_osm_pois`, UNNEST(all_tags) as tag
WHERE (tag.value in ('school') AND tag.key = 'amenity')
```

Clinics

```
SELECT CAST(id AS STRING) AS id , tag.value, geometry as geom
FROM `cartobq.docs.sdsb_london_osm_pois`, UNNEST(all_tags) as tag
WHERE (tag.value in ('clinic') AND tag.key = 'amenity')
```

Supermarkets

```
SELECT CAST(id AS STRING) AS id , tag.value, geometry as geom
FROM `cartobq.docs.sdsb_london_osm_pois`, UNNEST(all_tags) as tag
WHERE (tag.value in ('supermarket') AND tag.key = 'shop')
```

Bank branches

```
SELECT CAST(id AS STRING) AS id , tag.value, geometry as geom
FROM `cartobq.docs.sdsb_london_osm_pois`, UNNEST(all_tags) as tag
WHERE (tag.value in ('bank') AND tag.key = 'amenity')
```

Subway stations

```
SELECT CAST(id AS STRING) AS id , tag.value, geometry as geom
FROM `cartobq.docs.sdsb_london_osm_pois`, UNNEST(all_tags) as tag
WHERE (tag.value in ('subway') AND tag.key = 'station')
```

15-minute isochrones

```
CALL `carto-un.carto.ROUTING_ISOLINES`(
-- Start locations
---- Notting Hill
[ST_GEOGPOINT(-0.205919, 51.511773),
---- New Cross Rd
ST_GEOGPOINT(-0.033704, 51.475854),
---- Victoria Park Rd
ST_GEOGPOINT(-0.044486, 51.538130)],
-- Max time (cost limit) (in seconds)
[900.],
-- Area of interest
ST_GEOGFROMTEXT('POLYGON ((0.083004 51.43128, 0el.083004 51.568064, -0.230793
51.568064, -0.230793 51.43128, 0.083004 51.43128))'),
-- Transportation mode
'foot',
-- do_network_table
"sub_carto_geography_glo_roadnetwork_v1",
--do_source
'<my-project>.<my-dataset>',
--output_table
'cartobq.docs.sdsb_london_isochrones',
-- Options
"""
{
  "TYPE":"time",
  "MAX_COST":"100000",
  "WITH_PATH":"True"
}
"""
)
```

```
SELECT CONCAT('start_', start_order) AS origin,
      start_cost,
      dest_cost,
      detailed_geography_chunked_agg AS geom
FROM `cartobq.docs.sdsb_london_isochrones`
```