

OpenStreetMap - Snowflake Data Marketplace Documentation



Sonra Intelligence Limited

GW 107, GreenWay Hub
DIT Grangegorman
Dublin 7
Ireland
hello@sonra.io
www.sonra.io

1. Overview

Sonra has made available OpenStreetMap (OSM) data sets on the Snowflake Data Marketplace for some of the bigger countries. The data is available in each Snowflake region.

You can request countries that are not yet available by sending an e-mail to hello@sonra.io

2. Core tables

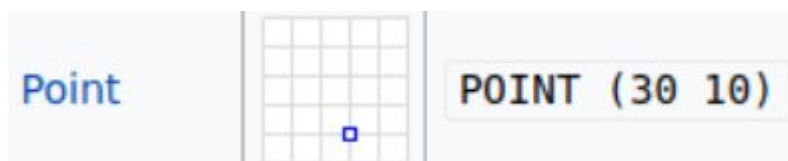
The OSM data model is quite simple. It contains three tables.

2.1. Node (V_OSM_NODE)

A [Node](#) is one of the three core objects in OpenStreetMap. It consists of a single point in space defined by its latitude and longitude. OSM assigns a Node ID to each Node.

| ID | TAGS | COORDINATES |
|------------|--|---|
| 4001249870 | [{"key": "historic", "value": "memorial"}, {"key": "mat... | {"coordinates": [-8.792285320000001e+01, 4.30387502000... |

In [Well-known text](#) representation of geometry (WKT) the equivalent of a Node would be a Point.



Each Node has a unique ID, e.g. ID 4001249870 is the Washington Monument in Milwaukee.

You can view a Node ID on the OSM website
<https://www.openstreetmap.org/node/<Node ID>>, e.g.
<https://www.openstreetmap.org/node/4001249870>

Node attributes

The attributes of a Node are stored as key / value pairs

Node: Washington Monument (4001249870)

change height tag

Edited 11 months ago by [shuui](#)

Version #5 · Changeset #74356043

Location: [43.0387502](#), [-87.9228532](#)

Tags

| | |
|-----------------------------|---|
| artist_name | Richard Henry Park |
| height | 10'6" |
| historic | memorial |
| inscription | RH PARK SC;F. GALLI FUSERO;WASHING TON;The Gift of Elizabeth A. Plankinton To the City of Milwaukee 1885 |
| lit | yes |
| material | bronze |
| memorial | statue |
| name | Washington Monument |
| start_date | 1885 |
| support | pedestal |
| wikidata | Q7972063 |
| wikipedia | en:Washington Monument (Milwaukee) |



2.1.1. Columns

| Column Name | Data Type | Description |
|---|---------------|---|
| ID | NUMBER | Unique identifier for a Node (not globally unique across all three objects) |
| TYPE | VARCHAR | Defaults to value 'Node' |
| TAGS | VARIANT | Attributes stored as key/value pairs |
| <p>Example for TAGS</p> <pre> TAGS [{"key": "historic", "value": "memorial"}, {"key": "material", "value": "bronze"}, {"key": "artist_name", "value": "Richard Henry Park"}, {"key": "lit", "value": "yes"}, { </pre> | | |
| LAT | NUMBER | Latitude coordinate of the object |
| LON | NUMBER | Longitude coordinate of the object |
| CHANGESET | NUMBER | Changeset ID |
| TIMESTAMP | TIMESTAMP_NTZ | Date and time when the object was created or changes are done |
| VERSION | NUMBER | Version of the object. Increments when a change is done |
| COUNTRY_IND | BOOLEAN | Some objects in OSM go across country boundaries, e.g. a network of roads in a Relation or multiple Nodes in a Way. We have included Nodes that are part of such a structure in the data. You can filter these Nodes by selecting COUNTRY_IND = 'T' |
| COORDINATES | GEOGRAPHY | Contains the longitude and latitude of the Node |

| | | |
|---------|---------|--------------------|
| GEOHASH | VARCHAR | Geohash for a Node |
|---------|---------|--------------------|

2.1.2. Sample queries

Extracting and pivoting tags

In the example below we pivot the data on the key shop and turn values for shop, name, website into columns.

```
SELECT n.id,
       n.COORDINATES,
       ont.shop,
       ont.name,
       ont.website,
       TYPE,
       concat(n.ID, '_', n.TYPE) AS ID_TYPE
FROM
  (SELECT ID,
         max(CASE
              WHEN lower(k)='shop' THEN v
              END) AS shop,
         max(CASE
              WHEN lower(k)='name' THEN v
              END) AS name,
         max(CASE
              WHEN lower(k)='website' THEN v
              END) AS website
   FROM
     (SELECT ID,
            value:key::string AS K,
            value:value::string AS V
      FROM V_OSM_NODE ,
           LATERAL flatten(INPUT => tags))
   WHERE ID IN
     (SELECT ID
      FROM V_OSM_NODE ,
           LATERAL flatten(INPUT => tags)
      WHERE value:key::string='shop')
   GROUP BY ID) AS ont
INNER JOIN
  (SELECT ID,
         coordinates,
         TYPE
   FROM V_OSM_NODE ) AS n ON n.ID = ont.ID;
```

| ID | COORDINAT... | SHOP | NAME | WEBSITE | TYPE | ID_TYPE |
|------------|-----------------|----------|-----------------|-----------------|------|----------------|
| 2277800332 | {"coordinate... | seafood | Moonfish | www.moonfis... | node | 2277800332... |
| 725122523 | {"coordinate... | optician | Gilna Opticians | NULL | node | 725122523_n... |
| 2706419555 | {"coordinate... | gift | Cardland | http://www.c... | node | 2706419555_... |

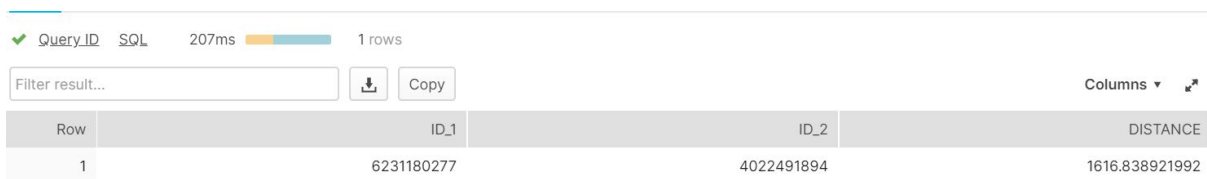
Calculating the distance between two Nodes

```

SELECT
  A.ID AS ID_1,
  B.ID AS ID_2,
  ST_DISTANCE(A.COORDINATES, B.COORDINATES) AS DISTANCE
FROM (SELECT
  ID,
  COORDINATES
FROM V_OSM_NODE
WHERE ID = '6231180277') AS A
INNER JOIN (SELECT
  ID,
  COORDINATES
FROM V_OSM_NODE
WHERE ID = '4022491894') AS B;

```

For more information on ST_DISTANCE please refer to the following link
https://docs.snowflake.com/en/sql-reference/functions/st_distance.html



Query ID: SQL 207ms 1 rows

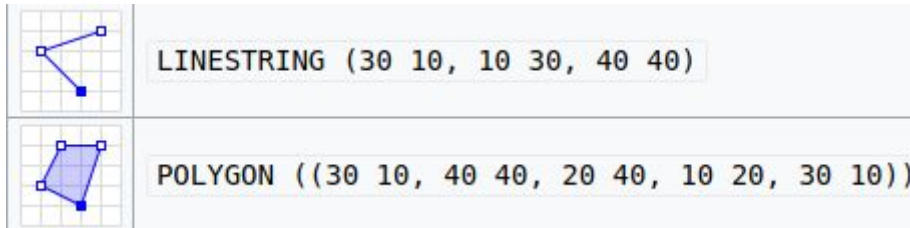
Filter result... [Download] [Copy] Columns ▾

| Row | ID_1 | ID_2 | DISTANCE |
|-----|------------|------------|----------------|
| 1 | 6231180277 | 4022491894 | 1616.838921992 |

2.2. Way (V_OSM_WAY)

In OSM a Way consists of multiple Nodes.

A Way can either be a LineString or a Polygon in [WKT representation](#).



2.2.1. Columns

| Column Name | Data Type | Description |
|-------------|---------------|--|
| ID | NUMBER | Unique identifier for a Way (IDs are not globally unique across the three core objects) |
| TYPE | VARCHAR | Defaults to value 'Way' |
| TAGS | VARIANT | Attributes stored as key/value pairs |
| NDS | ARRAY | Ordered list of Nodes when type is Way |
| CHANGESET | NUMBER | Changeset ID |
| TIMESTAMP | TIMESTAMP_NTZ | Date and time when the object was created or changes are done |
| VERSION | NUMBER | Version of the object. Increments when a change is made |
| COUNTRY_IND | BOOLEAN | Some objects in OSM go across country boundaries, e.g. a network of roads in a Relation. We have included such Ways that are part of such a Relation in the data. You can filter these Ways by selecting COUNTRY_IND = 'T' |

| | | |
|-------------|-----------|--|
| COORDINATES | GEOGRAPHY | Contains the coordinates of the Way. It can be a Linestring or a Polygon |
| GEOHASH | VARCHAR | Geohash for a object |

2.2.2. Sample queries

Distance between two Ways

```
SELECT
  A.ID AS ID_1,
  B.ID AS ID_2,
  ST_DISTANCE(A.COORDINATES, B.COORDINATES) AS DISTANCE
FROM (SELECT
  ID,
  COORDINATES
FROM V_OSM_WAY
WHERE ID = '676400806') AS A
INNER JOIN (SELECT
  ID,
  COORDINATES
FROM V_OSM_WAY
WHERE ID = '676400902') AS B;
```

For more information on ST_DISTANCE please refer to the following link
https://docs.snowflake.com/en/sql-reference/functions/st_distance.html

✓ Query ID SQL 534ms 1 rows

Filter result...  Copy Columns ▾ 

| Row | ID_1 | ID_2 | DISTANCE |
|-----|-----------|-----------|----------------|
| 1 | 676400806 | 676400902 | 4692.929256592 |

Check if Node is part of a Way

```
SELECT
  WAY.ID as WAY_ID,
  NODE.ID as NODE_ID,
  INDEX
FROM V_OSM_WAY WAY, LATERAL FLATTEN(NDS)
INNER JOIN
V_OSM_NODE NODE
ON
VALUE:ref=NODE.ID
```



```
WHERE NODE.ID='3654701350';
```

✓ Query_ID SQL 2.71s 1 rows

Filter result... [Download] Copy Columns ▾

| Row | WAY_ID | NODE_ID | INDEX |
|-----|-----------|------------|-------|
| 1 | 360929174 | 3654701350 | 28 |

A Way is an ordered list of Nodes. We can see that Node (365791350) is present in Way (360929174) at position 29 as index starts from 0.

Length of a Way

```
SELECT
    ID,
    ST_LENGTH(COORDINATES) AS LENGTH
FROM V_OSM_WAY
WHERE ID='676400813';
```

For more information on ST_LENGTH please refer to the following link
https://docs.snowflake.com/en/sql-reference/functions/st_length.html

✓ Query_ID SQL 105ms 1 rows

Filter result... [Download] Copy Columns ▾

| Row | ID | LENGTH |
|-----|-----------|---------------|
| 1 | 676400813 | 528.340481545 |

List the number of Nodes in a Way

```
SELECT
    ID,
    ST_NPOINTS(COORDINATES) AS NUM_OF_NODES
FROM V_OSM_WAY
WHERE ID='676400813';
```

For more information on ST_NPOINTS please refer to the following link
https://docs.snowflake.com/en/sql-reference/functions/st_npoints.html

✓ Query_ID SQL 241ms 1 rows

Filter result... [Download] Copy Columns ▾

| Row | ID | NUM_OF_NODES |
|-----|-----------|--------------|
| 1 | 676400813 | 50 |

List the dimension of a Way

```
SELECT
```

```

ID,
ST_DIMENSION(COORDINATES) AS DIMENSION
FROM V_OSM_WAY
WHERE ID='676400813';
    
```

✓ Query_ID SQL 184ms 1 rows

Filter result... [Download] [Copy] Columns ▾

| Row | ID | DIMENSION |
|-----|-----------|-----------|
| 1 | 676400813 | 1 |

For further information on ST_DIMENSION refer to the Snowflake docs:
https://docs.snowflake.com/en/sql-reference/functions/st_dimension.html

2.3. Relation (V_OSM_RELATION)

A Relation is a complex object type. It can consist of Nodes, Ways, or even other Relations. A Relation can have another Relation as a parent or child. A Relation can be part of a recursive Relationship with children, grandchildren and so on.

An example for a Relation with a Relation as a parent

OpenStreetMap Edit History Export GPS Traces User Diaries

Search Where is this? Go

Relation: Ashland Express (8682175)

Fix Bus Route

Edited about 2 months ago by Zol87
Version #12 · Changeset #87218352

Tags

| | |
|--------------------|---|
| description | Southbound service - Limited to 74th Street |
| from | Irving Park/Broadway |
| name | Ashland Express |
| network | CTA |
| operational_status | operating |
| operator | Chicago Transit Authority |

Parent

Part of

Relation [Ashland Express \(8682177\)](#) (as south)

An example for a Relation with a Relation as a child

Relation: Metro Purple Line (D) (7935318) ×

California Rail; LACMTA

Edited about 1 month ago by [stevea](#)
Version #3 · Changeset [#87473898](#)

Tags

| | |
|--------------|--|
| colour | purple ■ |
| name | Metro Purple Line (D) |
| network | Metro Rail |
| operator | Los Angeles County Metropolitan Transportation Authority |
| ref | 805 |
| route_master | subway |
| type | route_master |
| wikidata | Q3916689 |

Child Relation

Members

- Relation [Metro Purple Line \(D\) - Wilshire/Western → Union Station \(2810819\)](#)
- Relation [Metro Purple Line \(D\) - Union Station → Wilshire/Western \(7935312\)](#)

In table Relation we have included Relations that have a member inside the country. We also have included any recursive parent and child Relations of this Relation even if they

are outside the country.. We have flagged these Relations as being outside the country with the column country_ind = false.

2.3.1. Columns

| Column Name | Data Type | Description |
|-------------|---------------|---|
| ID | NUMBER | Unique identifier for a Relation (not globally unique across all three objects) |
| TYPE | VARCHAR | Defaults to value 'Relation' |
| TAGS | VARIANT | Attributes stored as key/value pairs |
| MEMBERS | ARRAY | Ordered list of one or more Nodes, Ways and/or Relations when type is Relation |
| CHANGESET | NUMBER | Changeset ID |
| TIMESTAMP | TIMESTAMP_NTZ | Date and time when the object was created or changes were made |
| VERSION | NUMBER | Version of the object. Increments when a change is done |
| COUNTRY_IND | BOOLEAN | There are some Relations which have members inside as well as outside the country boundary. For eg a travel link between two countries. You can filter these Relations by selecting COUNTRY_IND = 'T' |

2.3.2. Sample queries

Give me all of the recursive children (Node, Way, Relation) of a Relation

```
WITH t AS
(
SELECT
    ID,
    VALUE:ref AS MEMBERS,
    VALUE:type::STRING AS TYPE,
    INDEX,
    1 AS LVL
FROM V_OSM_RELATION, LATERAL FLATTEN(MEMBERS)
WHERE ID='9313587'
UNION ALL
SELECT
    R.ID,
    R.MEMBERS,
    R.TYPE,
    R.INDEX,
    LVL+1
FROM
(
    SELECT
        ID,
        VALUE:ref AS MEMBERS,
        VALUE:type::STRING AS TYPE,
        INDEX
    FROM V_OSM_RELATION, LATERAL FLATTEN(MEMBERS)
) AS R
INNER JOIN
t
ON
t.MEMBERS=R.ID
)
SELECT ID, MEMBERS, TYPE, INDEX, LVL FROM t ORDER BY LVL, ID, INDEX;
```

✓ Query_ID SQL 1.96s 725 rows

Filter result...

Download Copy

Columns ▾

| Row | ID | MEMBERS | TYPE | INDEX | LVL |
|-----|---------|---------|----------|-------|-----|
| 1 | 9313587 | 6585129 | relation | 0 | 1 |
| 2 | 9313587 | 8219357 | relation | 1 | 1 |
| 3 | 9313587 | 7935053 | relation | 2 | 1 |
| 4 | 9313587 | 7935245 | relation | 3 | 1 |
| 5 | 9313587 | 7935318 | relation | 4 | 1 |
| 6 | 9313587 | 6584634 | relation | 5 | 1 |
| 7 | 6584634 | 6584633 | relation | 0 | 2 |
| 8 | 6584634 | 2351006 | relation | 1 | 2 |

All children of Relation 9313587.

Give me the geography coordinates of a Relation

We get all of the members of a Relation and their geo coordinates.

```

SELECT
    ST_COLLECT(COORDINATES)
FROM
    (
    SELECT
        RELATION.ID,
        COORDINATES,
        INDEX,
        LVL
    FROM
        (
    WITH t AS
        (
    SELECT
            ID,
            VALUE:ref AS MEMBERS,
            VALUE:type::STRING AS TYPE,
            INDEX,
            1 AS LVL
        FROM
            V_OSM_RELATION,
            LATERAL FLATTEN(MEMBERS)
            WHERE ID='9313587'
        UNION ALL
        SELECT
            R.ID,
            R.MEMBERS,
    
```

```
        R.TYPE,
        R.INDEX,
        LVL+1
FROM
(
    SELECT
        ID,
        VALUE:ref AS MEMBERS,
        VALUE:type::STRING AS TYPE,
        INDEX
    FROM
        V_OSM_RELATION,
        LATERAL FLATTEN(members)
) AS R
INNER JOIN
t
ON
t.MEMBERS=R.ID
)
SELECT ID, MEMBERS, TYPE, INDEX, LVL FROM t ORDER BY LVL, ID, INDEX) RELATION INNER
JOIN
V_OSM_WAY W ON RELATION.MEMBERS=W.ID AND RELATION.TYPE=W.TYPE
UNION ALL
SELECT
    RELATION.ID,
    COORDINATES,
    INDEX,
    LVL
FROM
(
    WITH t AS
    (
        SELECT
            ID,
            VALUE:ref AS MEMBERS,
            VALUE:type::STRING AS TYPE,
            INDEX,
            1 AS LVL
        FROM
            V_OSM_RELATION,
            LATERAL FLATTEN(MEMBERS)
            WHERE ID='9313587'
    )
    UNION ALL
    SELECT
        R.ID,
        R.MEMBERS,
        R.TYPE,
        R.INDEX,
```

```

LVL+1
FROM
(
SELECT
    ID,
    VALUE:ref AS MEMBERS,
    VALUE:type::STRING AS TYPE,
    INDEX
FROM V_OSM_RELATION, LATERAL FLATTEN(members)
) AS R
INNER JOIN
t
ON
t.MEMBERS=R.ID
)
SELECT ID, MEMBERS, TYPE, INDEX, LVL FROM t ORDER BY LVL, ID, INDEX) RELATION INNER
JOIN
V_OSM_NODE N
ON RELATION.MEMBERS=N.ID AND RELATION.TYPE=N.TYPE) ORDER BY LVL, INDEX;

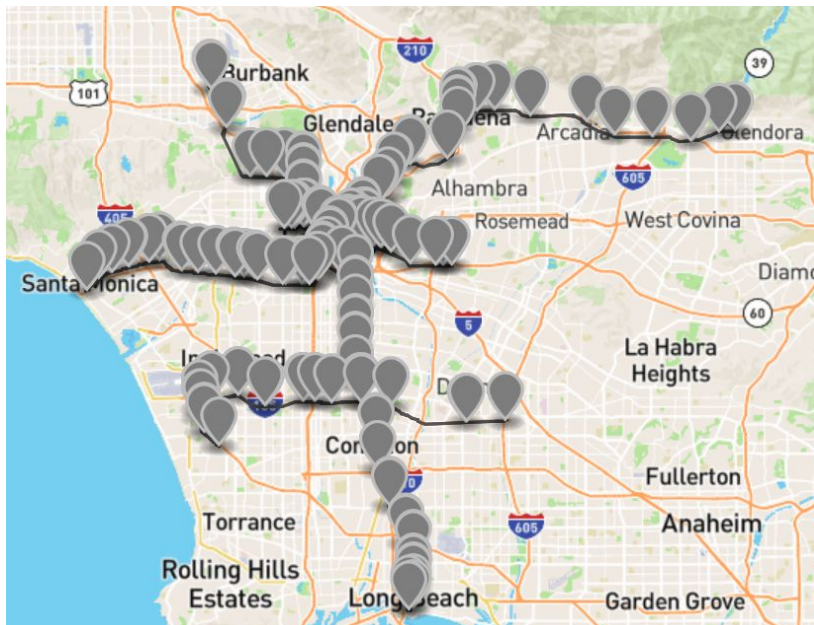
```

✓ Query ID SQL 33.2s 1 rows

Filter result... Columns ▾

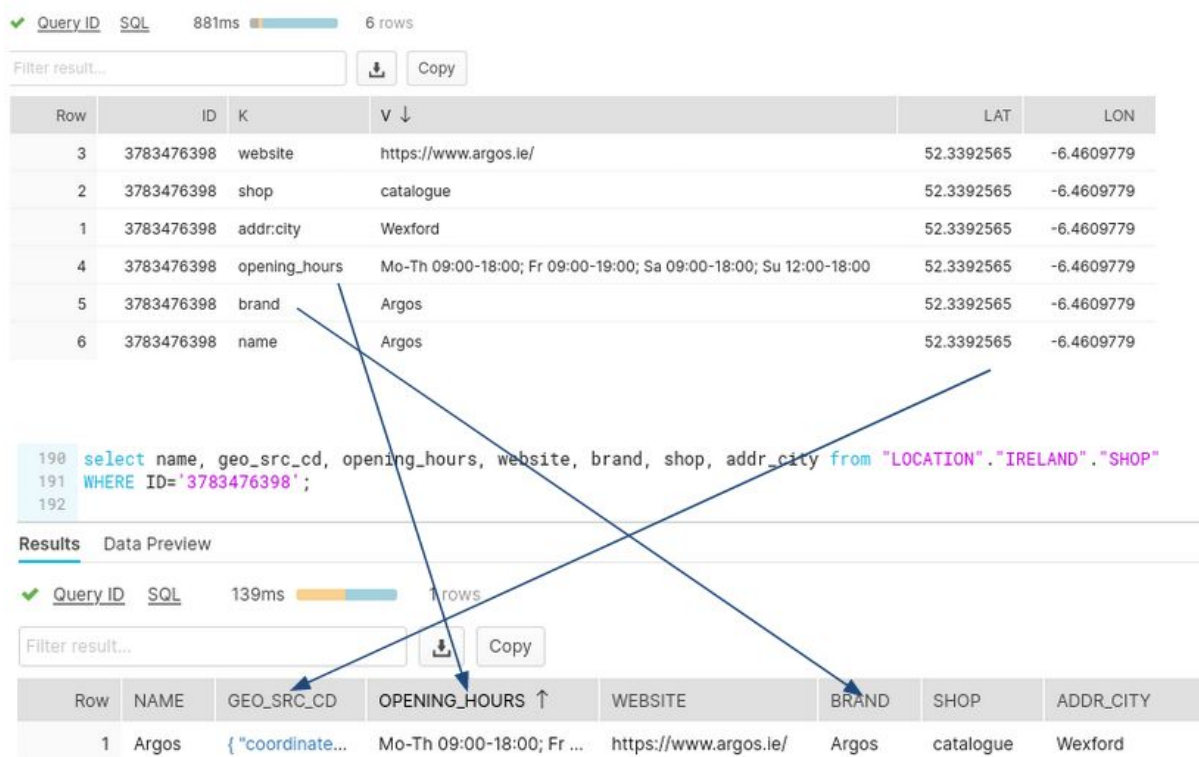
| Row | ST_COLLECT(COORDINATES) |
|-----|--|
| 1 | { "geometries": [{ "coordinates": [[-118.449714, 34.0323628], [-118.4469138, 34.032937], [-118.4468261, 34.032955], [... |

Here we have plotted the geo coordinates on a map



3. Feature tables

We have processed some of the key/value attributes for some of the more popular OSM objects such as Shop (retail) and Amenity. We have pivoted the key/value pairs to table rows and columns for easy consumption.



The screenshot shows a database query interface. The top part displays a query result for ID 3783476398 with 6 rows. The columns are Row, ID, K, V, LAT, and LON. The data is as follows:

| Row | ID | K | V | LAT | LON |
|-----|------------|---------------|---|------------|------------|
| 3 | 3783476398 | website | https://www.argos.ie/ | 52.3392565 | -6.4609779 |
| 2 | 3783476398 | shop | catalogue | 52.3392565 | -6.4609779 |
| 1 | 3783476398 | addr:city | Wexford | 52.3392565 | -6.4609779 |
| 4 | 3783476398 | opening_hours | Mo-Th 09:00-18:00; Fr 09:00-19:00; Sa 09:00-18:00; Su 12:00-18:00 | 52.3392565 | -6.4609779 |
| 5 | 3783476398 | brand | Argos | 52.3392565 | -6.4609779 |
| 6 | 3783476398 | name | Argos | 52.3392565 | -6.4609779 |

The bottom part shows the SQL query used to retrieve this data:

```
190 select name, geo_src_cd, opening_hours, website, brand, shop, addr_city from "LOCATION"."IRELAND"."SHOP"
191 WHERE ID='3783476398';
192
```

Below the query is a 'Results Data Preview' section showing a pivoted view of the data for the same ID. The columns are Row, NAME, GEO_SRC_CD, OPENING_HOURS, WEBSITE, BRAND, SHOP, and ADDR_CITY. The data is as follows:

| Row | NAME | GEO_SRC_CD | OPENING_HOURS | WEBSITE | BRAND | SHOP | ADDR_CITY |
|-----|-------|------------------|---------------------------|-----------------------|-------|-----------|-----------|
| 1 | Argos | { "coordinate... | Mo-Th 09:00-18:00; Fr ... | https://www.argos.ie/ | Argos | catalogue | Wexford |

Blue arrows in the image point from the 'K' column of the top table to the corresponding column headers in the bottom table: 'website' to 'WEBSITE', 'shop' to 'SHOP', 'addr:city' to 'ADDR_CITY', 'opening_hours' to 'OPENING_HOURS', 'brand' to 'BRAND', and 'name' to 'NAME'.

3.1. Amenity (V_OSM_AMENITY)

We have created a separate table Amenity that includes all of the Amenities for a country. Amenities can be of type Node, Way, and Relation.

As per OSM wiki an Amenity is split further into different categories and category types:

<https://wiki.openstreetmap.org/wiki/Key:amenity>

- Sustenance, e.g. bar, cafe, restaurant etc.
- Education e.g. college, driving_school, kindergarten etc
- Transportation e.g. bicycle_parking, bicycle_repair_station, bicycle_rental etc
- Financial e.g. atm, bank, bureau_de_change etc
- Healthcare e.g. baby_hatch, clinic, dentist etc

- Entertainment, Arts, Culture e.g. arts_centre, brothel, casino etc
- Others e.g. animal_boarding, animal_shelter, baking_oven etc
- Miscellaneous e.g. weighbridge, smoking_area, payment_centre etc (Keys that not present in the above types)

For each of these categories we have created a separate Table

| |
|--|
| V_OSM_AMENITY |
| V_OSM_AMENITY_EDUCATION |
| V_OSM_AMENITY_ENTERTAINMENT_ARTS_AND_CULTURE |
| V_OSM_AMENITY_FINANCIAL |
| V_OSM_AMENITY_HEALTHCARE |
| V_OSM_AMENITY_MISC |
| V_OSM_AMENITY_OTHERS |
| V_OSM_AMENITY_SUSTENANCE |
| V_OSM_AMENITY_TRANSPORTATION |

3.1.1. Columns

Each of these tables has 50+ columns. You can look up the description for each column on the OpenStreetMap Wiki.

https://wiki.openstreetmap.org/wiki/Category:Key_descriptions

For example: A description for the attribute Brand can be found here:

<https://wiki.openstreetmap.org/wiki/Key:brand>

3.1.2. Sample queries

Get all amenities from education category in a radius of 2,000 metres from a point

```
SELECT  
    *  
FROM V_OSM_AMENITY_EDUCATION
```

```
WHERE ST_DWITHIN(ST_POINT(-6.2330597, 53.3533353), COORDINATES, 2000);
```

For more information on ST_DWITHIN please refer to the following link
https://docs.snowflake.com/en/sql-reference/functions/st_dwithin.htm

✓ Query_ID SQL 277ms 77 rows

Filter result... [Download] [Copy] Columns ▾

| Row | ID | COORDINATES | AMENITY | NAME | ACCESS | ADDR_CITY | ADDR_COUNTRY | ADDR_COUNTY | ADDR_HOUSE |
|-----|-----------|----------------|--------------|------------------|--------|-----------|--------------|-------------|--------------|
| 65 | 130073184 | {"coordinat... | university | Dublin City ... | NULL | Dublin 5 | NULL | NULL | Water Dep... |
| 66 | 311466843 | {"coordinat... | university | Trinity Colle... | NULL | Dublin | NULL | NULL | NULL |
| 67 | 43148943 | {"coordinat... | library | Berkeley Lib... | NULL | NULL | NULL | NULL | NULL |
| 68 | 159677466 | {"coordinat... | school | Ardscoil Rís | NULL | NULL | NULL | NULL | NULL |
| 69 | 70799689 | {"coordinat... | school | Mount Temp... | NULL | NULL | NULL | NULL | NULL |
| 70 | 292900330 | {"coordinat... | school | The Model S... | NULL | NULL | NULL | NULL | NULL |
| 71 | 125770009 | {"coordinat... | kindergarten | Bright Horiz... | NULL | NULL | NULL | NULL | NULL |
| 72 | 121270000 | {"coordinat... | school | Crafts Col... | NULL | NULL | NULL | NULL | NULL |

Get the TOP geohashes (precision 5) by the number of bars

```
SELECT
    ST_GEOHASH(COORDINATES, 5) AS GEOHASH,
    COUNT(*) AS COUNT_OF_BAR
FROM V_OSM_AMENITY
WHERE AMENITY='bar'
GROUP BY GEOHASH
ORDER BY COUNT_OF_BAR DESC;
```

For more information on ST_GEOHASH please refer to the following link
https://docs.snowflake.com/en/sql-reference/functions/st_geohash.html

✓ Query_ID SQL 145ms 300 rows

Filter result... [Download] [Copy] Columns ▾

| Row | GEOHASH | COUNT_OF_BAR |
|-----|---------|--------------|
| 1 | gc7×3 | 70 |
| 2 | gcey9 | 31 |
| 3 | gc1zp | 18 |
| 4 | gc7×9 | 16 |

3.2. Shop (V_OSM_SHOP)

We have created a separate table named Shop that includes all of the shops for a country. Shops can be of type Node, Way, and Relation.

As per OSM wiki a Shop is split further into different categories and category types:

<https://wiki.openstreetmap.org/wiki/Key:shop>

- Food, beverages e.g. alcohol, bakery, beverages etc
- General store, department store, mall e.g. department_store, general, kiosk etc
- Clothing, shoes, accessories e.g. baby_goods, bag, boutique etc
- Discount store, charity e.g. charity, second_hand, variety_store etc
- Health and beauty e.g. beauty, chemist, cosmetics etc
- Do-it-yourself, household, building materials, gardening e.g. agrarian, appliance, bathroom_furnishing etc
- Furniture and interior e.g. antiques, bed, candles etc
- Electronics e.g. computer, electronics, hifi etc
- Outdoors and sport, vehicles e.g. atv, bicycle, boat etc
- Art, music, hobbies e.g. art, collector, craft etc
- Stationery, gifts, books, newspapers e.g. anime, books, gift etc
- Others e.g. bookmaker, cannabis, copyshop etc
- Miscellaneous e.g. fancy_dress, estate_agent, appliance_repair etc (Keys that not present in the above types)

For each of these categories we have created a separate Table

| |
|--|
| V_OSM_SHOP |
| V_OSM_SHOP_FOOD_BEVERAGES |
| V_OSM_SHOP_GENERAL_STORE_DEPARTMENT_STORE_MALL |
| V_OSM_SHOP_CLOTHING_SHOES_ACCESSORIES |
| V_OSM_SHOP_DISCOUNT_STORE_CHARITY |
| V_OSM_SHOP_HEALTH_AND_BEAUTY |
| V_OSM_SHOP_DO_IT_YOURSELF_HOUSEHOLD_BUILDING_MATERIALS_GARDENING |
| V_OSM_SHOP_FURNITURE_AND_INTERIOR |
| V_OSM_SHOP_ELECTRONICS |
| V_OSM_SHOP_OUTDOORS_AND_SPORT_VEHICLES |
| V_OSM_SHOP_ART_MUSIC_HOBBIES |
| V_OSM_SHOP_STATIONERY_GIFTS_BOOKS_NEWSPAPERS |
| V_OSM_SHOP_OTHERS |

V_OSM_SHOP_MISC

3.2.1. Columns

Each of these tables has 50+ columns. You can look up the description for each column on the OpenStreetMap Wiki.

https://wiki.openstreetmap.org/wiki/Category:Key_descriptions

For example: A description for the attribute Brand can be found here:

<https://wiki.openstreetmap.org/wiki/Key:brand>

3.2.2. Sample queries

Get all food and beverage Shops in a radius of 2,000 metres from a point

```
SELECT
    *
FROM V_OSM_SHOP_FOOD_BEVERAGES
WHERE ST_DWITHIN(ST_POINT(-6.2330597, 53.3533353), COORDINATES, 2000);
```

For more information on ST_DWITHIN please refer to the following link

https://docs.snowflake.com/en/sql-reference/functions/st_dwithin.html

Query ID: SQL 735ms 108 rows

Filter result... [Download] [Copy] Columns ▾

| Row | ID | COORDINATES | SHOP | NAME | ADDR_CITY | ADDR_COUNTR | ADDR_COUNTY | ADDR_HOUSEN | ADDR_HOUSENI |
|-----|------------|----------------|-------------|------------------|-----------|-------------|-------------|-------------|--------------|
| 1 | 1508826450 | {"coordinat... | convenience | Ballybough ... | NULL | NULL | NULL | NULL | NULL |
| 2 | 1508841440 | {"coordinat... | convenience | Tesco Express | Dublin | NULL | NULL | NULL | 18 |
| 3 | 1508853551 | {"coordinat... | convenience | J. Tallon | NULL | NULL | NULL | NULL | NULL |
| 4 | 1508868461 | {"coordinat... | convenience | Griffin's Cen... | NULL | NULL | NULL | NULL | NULL |
| 5 | 3450108623 | {"coordinat... | bakery | St. Peter's B... | Dublin | NULL | NULL | NULL | NULL |
| 6 | 3676877493 | {"coordinat... | convenience | North Stran... | Dublin | NULL | NULL | NULL | NULL |
| 7 | 3677080034 | {"coordinat... | convenience | K&A Stores | Dublin | NULL | NULL | NULL | NULL |

Get the TOP geohashes (precision 5) by number of supermarkets

```
SELECT
    ST_GEOHASH(COORDINATES, 5) AS GEOHASH,
    COUNT(*) AS COUNT_OF_SUPERMARKETS
FROM
```

```
V_OSM_SHOP  
WHERE SHOP='supermarket'  
GROUP BY GEOHASH  
ORDER BY COUNT_OF_SUPERMARKETS DESC;
```

For more information on ST_GEOHASH please refer to the following link
https://docs.snowflake.com/en/sql-reference/functions/st_geohash.html

✓ Query ID SQL 473ms 550 rows

Filter result... [Download] [Copy] Columns ▾

| Row | GEOHASH | COUNT_OF_SUPERMARKETS |
|-----|---------|-----------------------|
| 1 | gc7x3 | 42 |
| 2 | gc3ge | 37 |
| 3 | gc7x9 | 29 |
| 4 | gce84 | 17 |
| 5 | gcey9 | 17 |