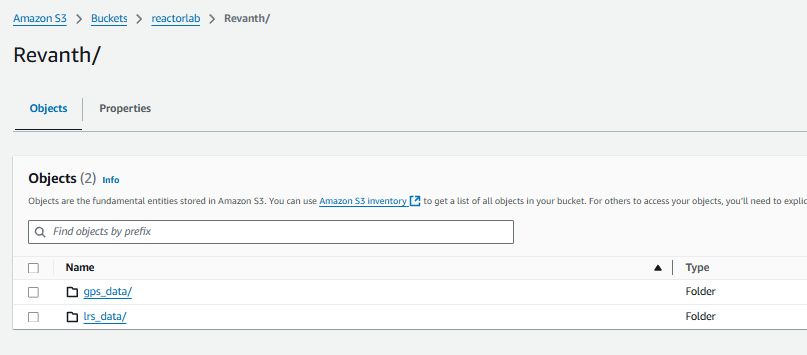
**Lecture 8**

**Activity 0: AWS account setup and folder creation**



**Activity 1: Creating Tables**

1. **Creating Revanth\_lrs\_data:**

**Script:**

CREATE EXTERNAL TABLE IF NOT EXISTS `aiml\_class`.`Revanth\_lrs\_data` (

`route\_id` varchar(50),

`segment\_start\_latitude` float,

`segment\_start\_longitude` float,

`segment\_end\_latitude` float,

`segment\_end\_longitude` float,

`segment\_start\_measure` float,

`geohash\_lrs` float)

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'

WITH SERDEPROPERTIES ('field.delim' = ',')

STORED AS INPUTFORMAT 'org.apache.hadoop.mapred.TextInputFormat' OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'

LOCATION 's3://reactorlab/Revanth/lrs\_data/'

TBLPROPERTIES ('classification' = 'csv');

1. **Creating Revanth\_gps\_data**

**Script:**

CREATE EXTERNAL TABLE IF NOT EXISTS `aiml\_class`.`revanth\_gps\_data` ( `datapointid` varchar(50), `journeyid` varchar(50), `latitude` float, `longitude` float, `month` int, `day` int, `hour` int, `geohash` varchar(10), `speed` float

) COMMENT "GPS\_data"

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'

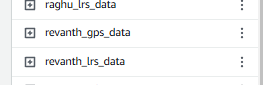
WITH SERDEPROPERTIES ('field.delim' = ',')

STORED AS INPUTFORMAT 'org.apache.hadoop.mapred.TextInputFormat' OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'

LOCATION 's3://reactorlab/Revanth/gps\_data/'

TBLPROPERTIES ('classification' = 'csv')

**Output:**



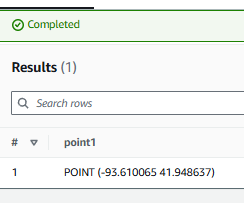
**Activity 2: GeoSQL**

1. Make point feature from Point 1: 41.948637, -93.610065 using ST\_POINT

**Script:**

SELECT ST\_POINT(-93.610065, 41.948637) AS point1;

**Output:**



1. Find distance (ST\_DISTANEC) between points below: (Ans: 0.048702)

Point 1: 41.948637, -93.610065, Point 2 : 41.997339, -93.610229

**Script:**

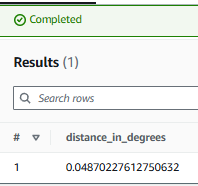
SELECT ST\_DISTANCE(

ST\_POINT(-93.610065, 41.948637),

ST\_POINT(-93.610229, 41.997339)

) AS distance\_in\_degrees;

**Output:**



1. Create a line string (ST\_LINESTRING) from above points and calculate distance from Point 3 : 41.972017, -93.619738 (Ans: 0.0095942)

**Script:**

SELECT ST\_DISTANCE(

ST\_POINT(-93.619738, 41.972017), -- Point 3

ST\_LINESTRING(

ARRAY[

ST\_POINT(-93.610065, 41.948637), -- Point 1

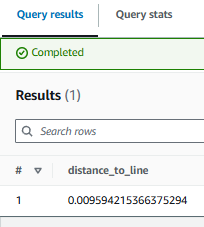
ST\_POINT(-93.610229, 41.997339) -- Point 2

]

)

) AS distance\_to\_line;

**Output:**



1. Get it in feet by multiplying 364567 (Ans: 3497.7 ft)

**Script:**

SELECT

ST\_DISTANCE(

ST\_POINT(-93.619738, 41.972017), -- Point 3

ST\_LINESTRING(

ARRAY[

ST\_POINT(-93.610065, 41.948637), -- Point 1

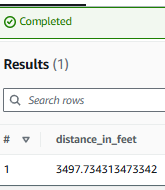
ST\_POINT(-93.610229, 41.997339) -- Point 2

]

)

) \* 364567 AS distance\_in\_feet;

**Output:**



1. Make point geometry column from latitude and longitude in gps\_data table
2. Create a buffer (ST\_BUFFER) around 41.502954, -94.295980 for 0.05 units and filter all datapoints (ST\_CONTAINS) from gps\_data within the buffer (Ans: 54,581 rows)

**Script:**

WITH unique\_gps\_points AS (

SELECT

DISTINCT datapointid, -- Ensure unique datapoints

journeyid,

latitude,

longitude,

month,

day,

hour,

geohash,

speed

FROM revanth\_gps\_data

)

SELECT COUNT(\*) AS num\_points\_in\_buffer

FROM (

SELECT

ST\_POINT(longitude, latitude) AS geom -- Create point geometry after deduplication

FROM unique\_gps\_points

) AS geom\_points

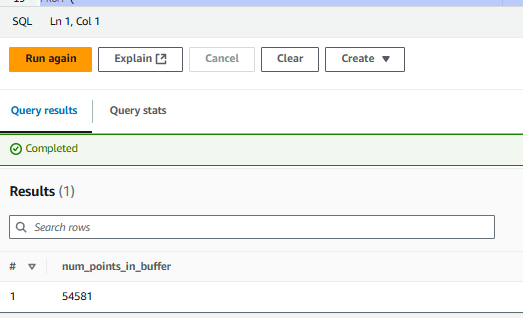
WHERE ST\_CONTAINS(

ST\_BUFFER(ST\_POINT(-94.295980, 41.502954), 0.05), -- Create buffer with a radius of 0.05 units

geom\_points.geom -- Check if the points are within the buffer

);

**Output:**



**Activity 3: Data conflation through spatial join**

1. Find distance from each point to all line strings in the database

**Script:**

select \*, st\_distance(

st\_point(a.longitude, a.latitude),

st\_linestring(array[

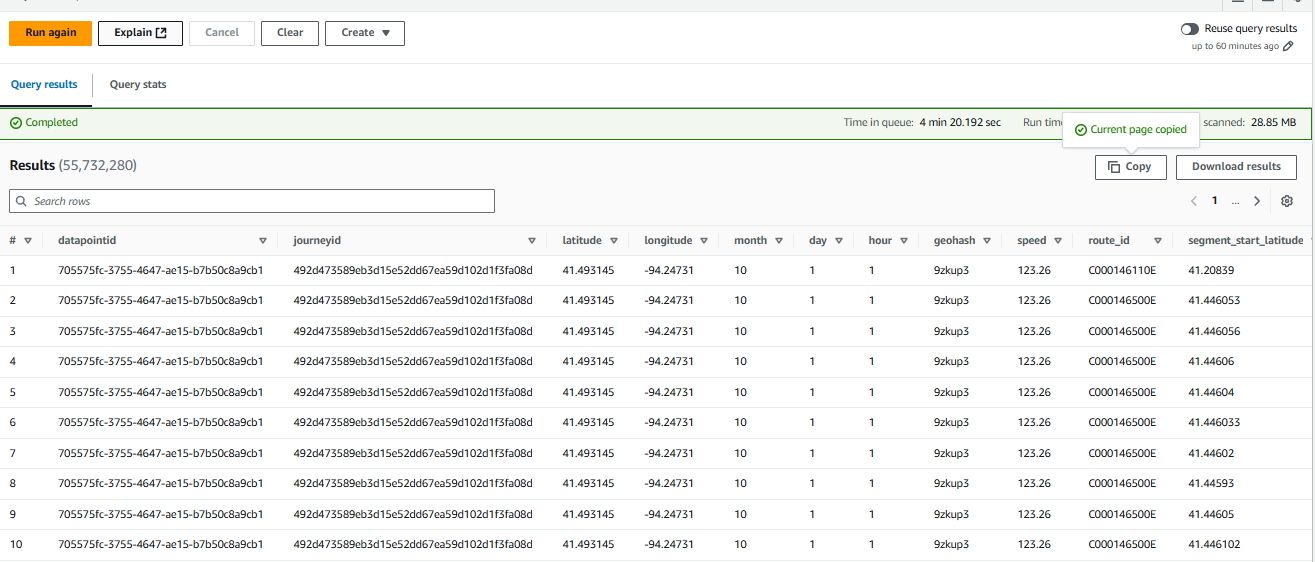
st\_point(b.segment\_start\_longitude, b.segment\_start\_latitude),

st\_point(b.segment\_end\_longitude,b.segment\_end\_latitude)])) distance from "raghu\_gps\_data" a

cross join (select \* from "aiml\_class"."raghu\_lrs\_data" where not segment\_start\_latitude = segment\_end\_latitude and not segment\_start\_longitude = segment\_end\_longitude) as b

where a.day =1 and a.hour = 1

**Output:**



1. Filter the nearest one for each point through a nested query

**Script:**

WITH filtered\_gps AS (

SELECT

datapointid,

journeyid,

latitude AS gps\_lat,

longitude AS gps\_lon,

day

FROM

revanth\_gps\_data

WHERE

day = 1 AND -- Use AND instead of a comma

hour = 1 -- Filter for hour = 1 as well

),

distance\_calculation AS (

SELECT

gps.datapointid,

gps.journeyid,

gps.gps\_lat,

gps.gps\_lon,

lrs.route\_id,

lrs.segment\_start\_latitude AS start\_lat,

lrs.segment\_start\_longitude AS start\_lon,

lrs.segment\_end\_latitude AS end\_lat,

lrs.segment\_end\_longitude AS end\_lon,

-- Haversine formula for distance to start of segment

6371 \* acos(

cos(radians(gps.gps\_lat)) \* cos(radians(lrs.segment\_start\_latitude)) \*

cos(radians(lrs.segment\_start\_longitude) - radians(gps.gps\_lon)) +

sin(radians(gps.gps\_lat)) \* sin(radians(lrs.segment\_start\_latitude))

) AS distance\_to\_start,

-- Haversine formula for distance to end of segment

6371 \* acos(

cos(radians(gps.gps\_lat)) \* cos(radians(lrs.segment\_end\_latitude)) \*

cos(radians(lrs.segment\_end\_longitude) - radians(gps.gps\_lon)) +

sin(radians(gps.gps\_lat)) \* sin(radians(lrs.segment\_end\_latitude))

) AS distance\_to\_end

FROM

filtered\_gps gps

CROSS JOIN

revanth\_lrs\_data lrs

)

SELECT \* FROM distance\_calculation;

**Output:**

