DataEng: Project Assignment 3

Data Integration

Genevieve LaLonde
Data Engineering Winter 2021
Bruce Irvin

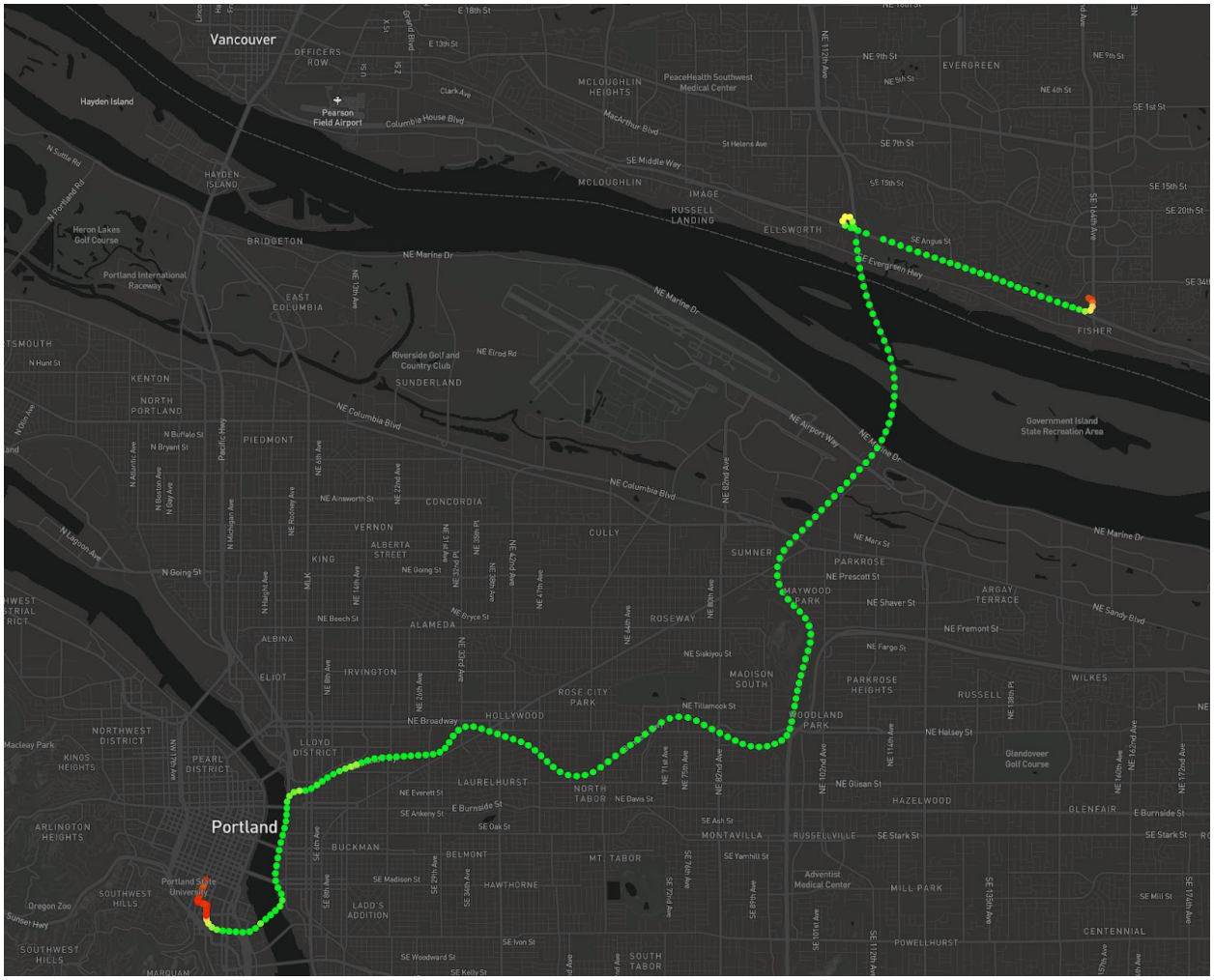
Assignment date: February 16
Due date: February 28, 2021 @10pm PT
Submit: assignment submission form

Submission

Visualization 1.

A visualization of speeds for a single trip for any bus route that crosses the Glenn Jackson I-205 bridge. You choose the day, time and route for your selected trip. To find a trip that traverses this bridge, consider finding a trip that includes breadcrumb sensor points within this bounding box: [45.592404, -122.550711, 45.586158, -122.541270]. Any bus trip that includes breadcrumb points within that box either crosses the bridge or goes swimming in the Columbia river!

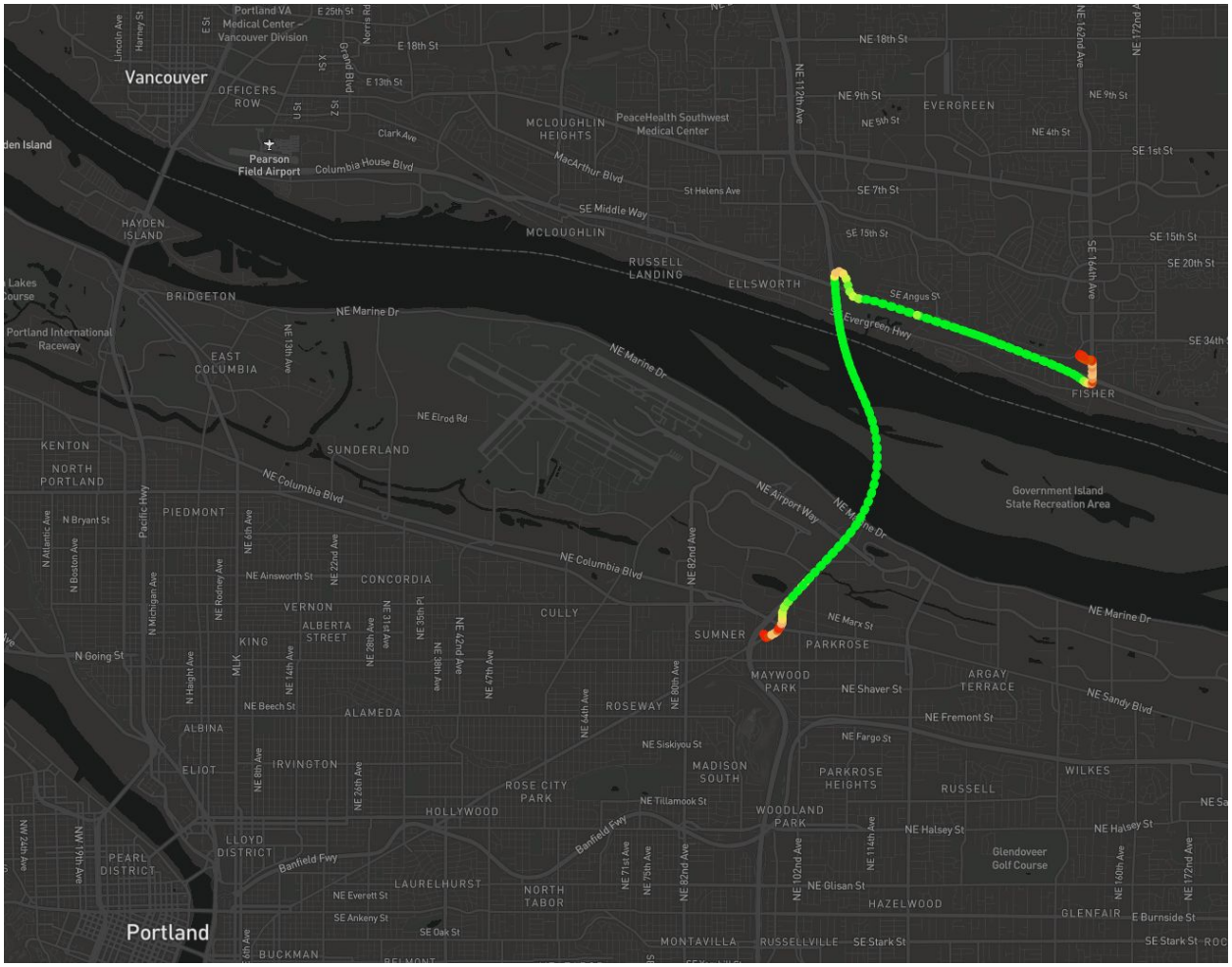
- One trip
- On route 164
- Which crosses the Glenn Jackson I-205 bridge
- On a weekday schedule
- Returning to the start
- On 2020-10-19
- Starting at 15:34:47
- Ending at 15:59:06



Visualization 2.

All outbound trips that occurred on route 65 on any Friday (you choose which Friday) between the hours of 4pm and 6pm.

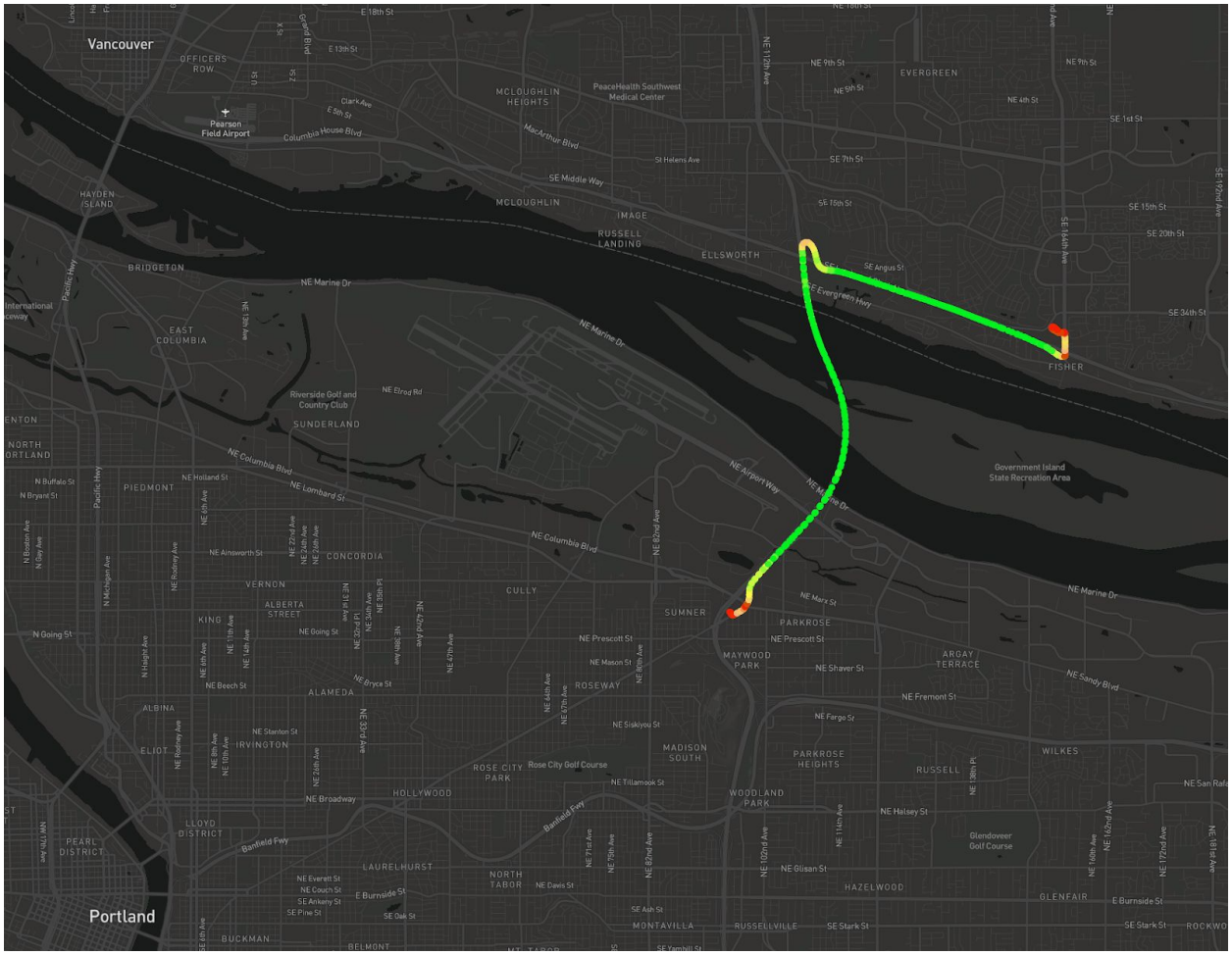
- All outbound trips
- On route 65
- On Friday 2020-10-18
- 4:00 pm - 6:00 pm



Visualization 3.

All outbound trips for route 65 on any Sunday morning (you choose which Sunday) between 9am and 11am.

- All outbound trips
- On route 65
- On Sunday 2020-10-25
- 9:00 am - 11:00 am



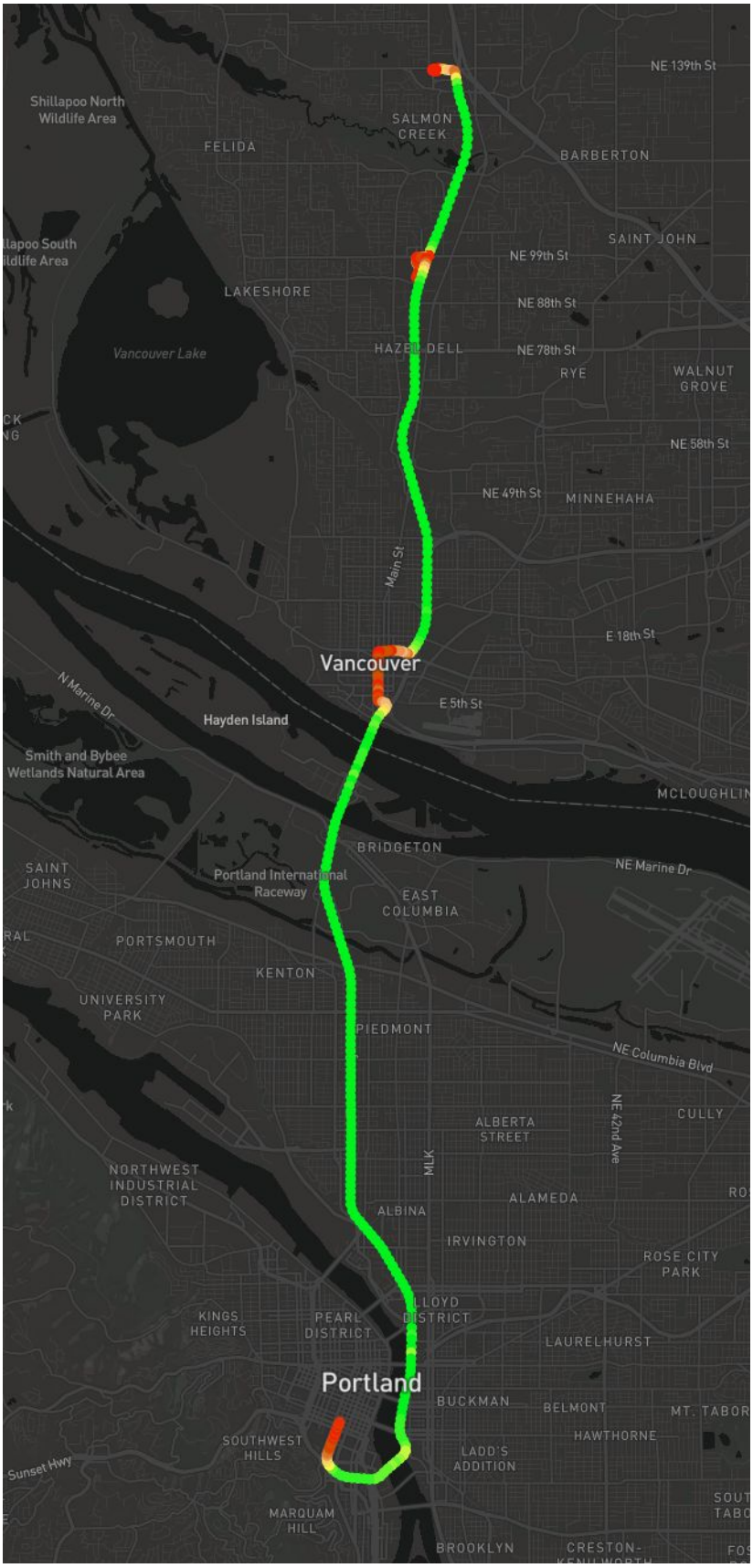
Visualization 4.

The longest (as measured by time) trip in your entire data set. Indicate the date, route #, and trip ID of the trip along with a visualization showing the entire trip.

The longest duration trip I have full data for is: 170770658.
A lot of the other very long trips are from before we started ingesting Stop Event data.

```
ctran=> select t.*, ib.*
ctran-> from trip t inner join (
ctran(> select max(tstamp), min(tstamp), max(tstamp) - min(tstamp) as trip_length, trip_id
ctran(> from breadcrumb b
ctran(> group by trip_id
ctran(> ) ib on t.trip_id = ib.trip_id
ctran-> order by ib.trip_length desc
ctran-> limit 20;
  trip_id   | route_id | vehicle_id | service_key | direction |         max         |         min         | trip_length |  trip_id
------------+----------+------------+-------------+-----------+---------------------+---------------------+-------------+-----------
 169302880 |          |    1254260 |             |           | 2020-10-01 22:02:44 | 2020-10-01 16:30:18 | 05:32:26    | 169302880
 169026975 |          |       2269 |             |           | 2020-09-29 10:14:19 | 2020-09-29 07:02:14 | 03:12:05    | 169026975
 168692921 |          |       2251 |             |           | 2020-09-24 14:06:46 | 2020-09-24 11:16:41 | 02:50:05    | 168692921
 167674081 |          |       4026 |             |           | 2020-09-11 12:34:30 | 2020-09-11 09:50:21 | 02:44:09    | 167674081
 170770658 |      105 |    1254260 | Weekday     | Back      | 2020-10-20 19:23:25 | 2020-10-20 17:27:30 | 01:55:55    | 170770658
 167690539 |          |       4032 |             |           | 2020-09-11 13:09:26 | 2020-09-11 11:17:18 | 01:52:08    | 167690539
 167688567 |          |       4020 |             |           | 2020-09-11 12:51:28 | 2020-09-11 11:00:11 | 01:51:17    | 167688567
 167859443 |          |       4029 |             |           | 2020-09-14 09:55:47 | 2020-09-14 08:04:49 | 01:50:58    | 167859443
 168094965 |          |       6001 |             |           | 2020-09-17 23:11:42 | 2020-09-17 21:20:58 | 01:50:44    | 168094965
 170809733 |       60 |       2291 | Weekday     | Back      | 2020-10-20 11:47:22 | 2020-10-20 09:59:34 | 01:47:48    | 170809733
 168425778 |          |       2220 |             |           | 2020-09-21 17:48:10 | 2020-09-21 16:01:00 | 01:47:10    | 168425778
 167954916 |          |       4025 |             |           | 2020-09-15 09:49:51 | 2020-09-15 08:05:25 | 01:44:26    | 167954916
 167132327 |          |       4030 |             |           | 2020-09-04 17:41:18 | 2020-09-04 16:00:59 | 01:40:19    | 167132327
 171423657 |       47 |       2231 | Weekday     | Out       | 2020-10-28 06:51:37 | 2020-10-28 05:13:25 | 01:38:12    | 171423657
 168124157 |          |       2232 |             |           | 2020-09-17 17:38:32 | 2020-09-17 16:00:52 | 01:37:40    | 168124157
 170239217 |          |       2243 |             |           | 2020-10-13 06:52:09 | 2020-10-13 05:14:47 | 01:37:22    | 170239217
 168621282 |          |       2220 |             |           | 2020-09-23 18:00:06 | 2020-09-23 16:22:48 | 01:37:18    | 168621282
 169466935 |          |       2215 |             |           | 2020-10-02 19:23:40 | 2020-10-02 17:46:55 | 01:36:45    | 169466935
 171329672 |       47 |       2248 | Weekday     | Out       | 2020-10-27 06:49:11 | 2020-10-27 05:13:08 | 01:36:03    | 171329672
 168499819 |          |       2248 |             |           | 2020-09-22 06:47:53 | 2020-09-22 05:13:11 | 01:34:42    | 168499819
(20 rows)
```

All datapoints on the longest duration trip are mapped below.


NE 139th St
Shillapoo North Wildlife Area
SALMON CREEK
FELIDA
BARBERTON
SAINT JOHN
NE 99th St
LAKESHORE
NE 88th St
Vancouver Lake
HAZEL DELL
NE 78th St
RYE
WALNUT GROVE
NE 58th St
NE 49th St
MINNEHAHA
Main St
E 18th St
Vancouver
N Marine Dr
E 5th St
Hayden Island
Smith and Bybee Wetlands Natural Area
MCLOUGHLIN
BRIDGETON
NE Marine Dr
SAINT JOHNS
Portland International Raceway
EAST COLUMBIA
PORTSMOUTH
KENTON
UNIVERSITY PARK
PIEDMONT
NE Columbia Blvd
NE 42nd Ave
CULLY
ALBERTA STREET
MLK
NORTHWEST INDUSTRIAL DISTRICT
ALAMEDA
ALBINA
IRVINGTON
ROSE CITY PARK
KINGS HEIGHTS
PEARL DISTRICT
LLOYD DISTRICT
LAURELHURST
Portland
BUCKMAN
BELMONT
MT. TABOR
HAWTHORNE
SOUTHWEST HILLS
LADD'S ADDITION
Sunset Hwy
MARQUAM HILL
BROOKLYN
CRESTON-

Visualization 5a, 5b, 5c, …. Three or more additional visualizations of your choice. Indicate why you chose each particular visualization.

5a) Longest Route - Distance

What is the longest route in terms of estimated distance traveled?
Map the longest trip for the longest route. Use a bounding box of the movement on the x and y axis to estimate length of the route.

I've defined the route length by getting the length of the hypotenuse of the bounding box of the route's maximum positions in either axis.
length_estimate = sqrt((max(x) - min(x))^2 + (max(y) - min(y))^2)

When I first did this I was only multiplying the movement of the x and y axis. But after considering whether multiplying or adding would be a better representation (the area or half the perimeter of the bounding box), I realized the hypotenuse is much more accurate, and not weighted by the shape of the box (long and skinny vs square), nor by whether the box is more aligned with the x/y axis. However this is still just an estimate. For a more precise measurement of bus distance, I think we could convert the Geo Points of a trip to a Line String and pull its length. This would be a good area for future work.

Query Description:
Calculate the hypotenuse of the bounding box of points in the trip as the trip length. Order the result by the length of the trip, with longest first. Also return the trip info like the route id.

Query Text:

```
select t.*, ib.*
from trip t inner join (
  select
  min(latitude) S,
  max(latitude) N,
  max(latitude) - min(latitude) as y_movement,
  min(longitude) E,
  max(longitude) W ,
  max(longitude) - min(longitude) as x_movement,
  sqrt((max(latitude) - min(latitude))^2 + (max(longitude) - min(longitude))^2) as total_movement,
  trip_id
  from breadcrumb
  group by trip_id
  order by 7 desc
  ) ib
on t.trip_id = ib.trip_id
where route_id > 0
```

and total_movement > 0
order by ib.total_movement desc
;

Data Selection:
From the result, I picked the longest trip, for the top 3 routes occurring in the data. This redacted result is below.

trip_id	route_id	vehicle_id	service_key	direction	s	n	y_movement	e	w	x_movement	total_movement	trip_id
171076865	47	2267	Weekday	Out	45.655968	45.86684	0.21087200000000195	-122.585415	-122.385082	0.20033300000000054	0.29086132653379865	171076865
170817612	105	1254280	Weekday	Back	45.517453	45.72181	0.20435699999999457	-122.677218	-122.543822	0.13339599999999052	0.2440415461862903	170817612
170872807	190	2251	Weekday	Out	45.498058	45.721833	0.22377499999999628	-122.689352	-122.652702	0.03664999999999452	0.2267564180458801	170872807

The points of most interest to our map are:

trip_id	route_id
171076865	47
170817612	105
170872807	190

I mapped each of these separately so we can clearly see each route without overlapping.

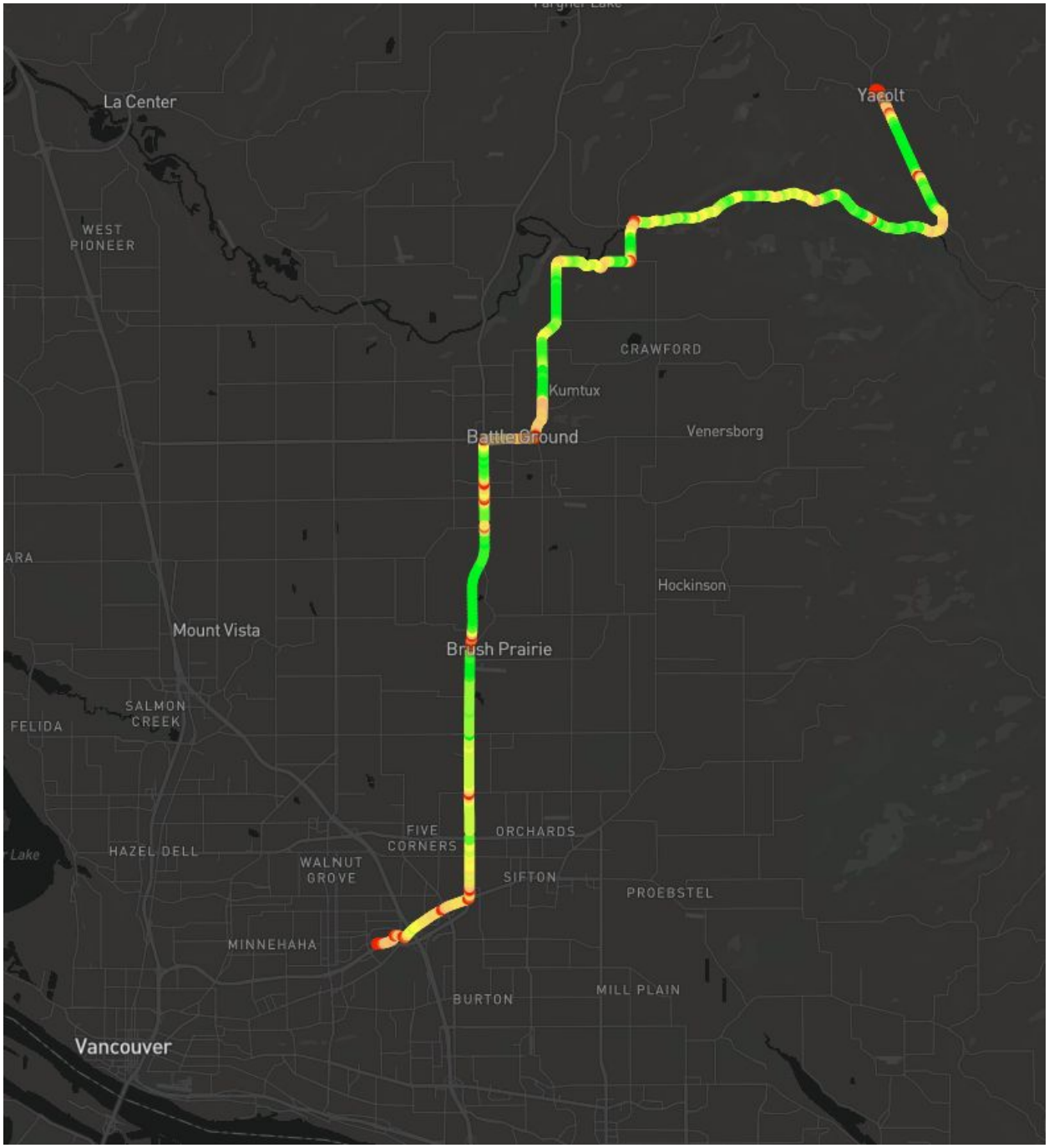
Queries to pull the data:

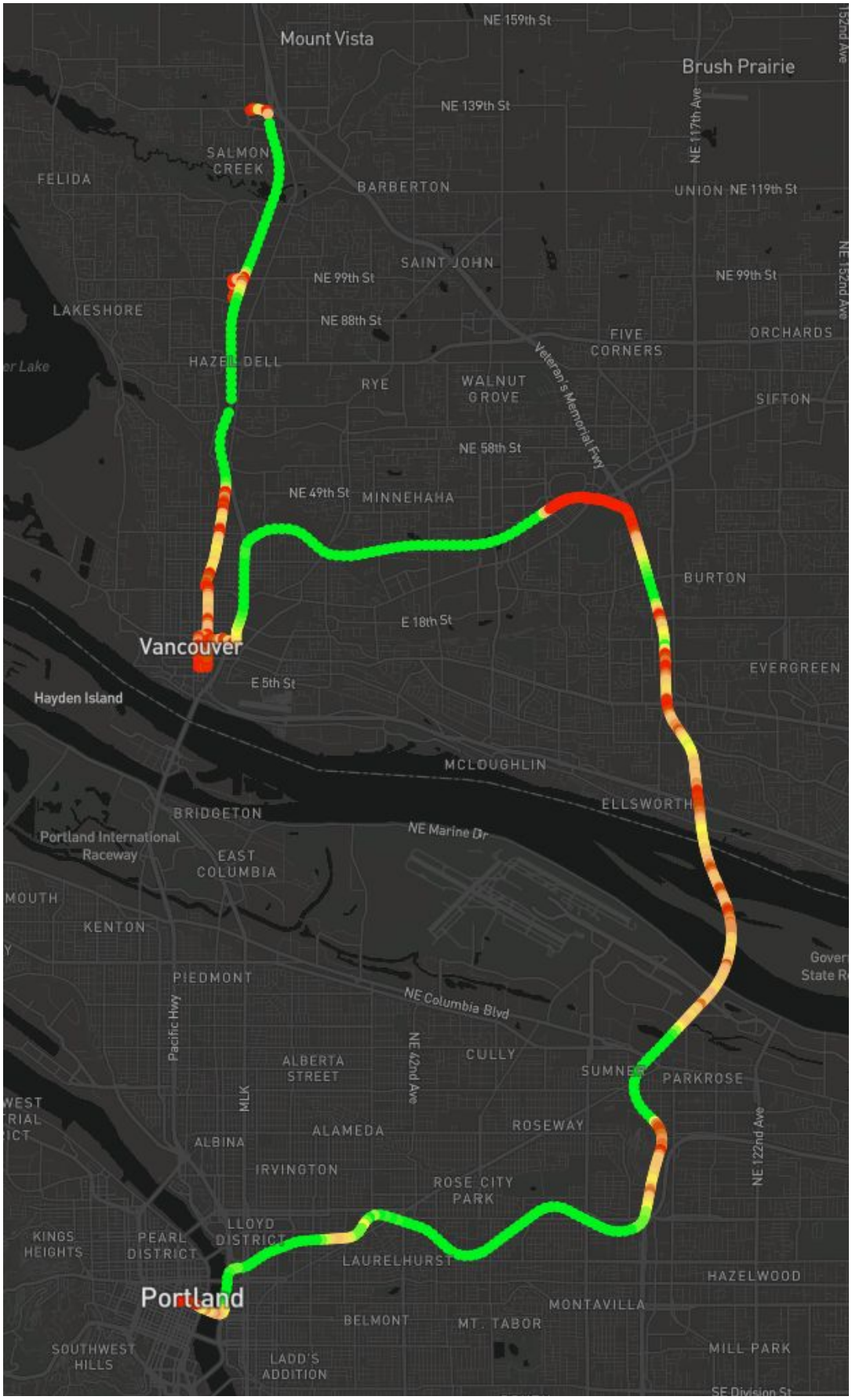
```
select longitude, latitude, speed from
  breadcrumb where trip_id = 171076865;
select longitude, latitude, speed from
  breadcrumb where trip_id = 170817612;
select longitude, latitude, speed from
  breadcrumb where trip_id = 170872807;
```

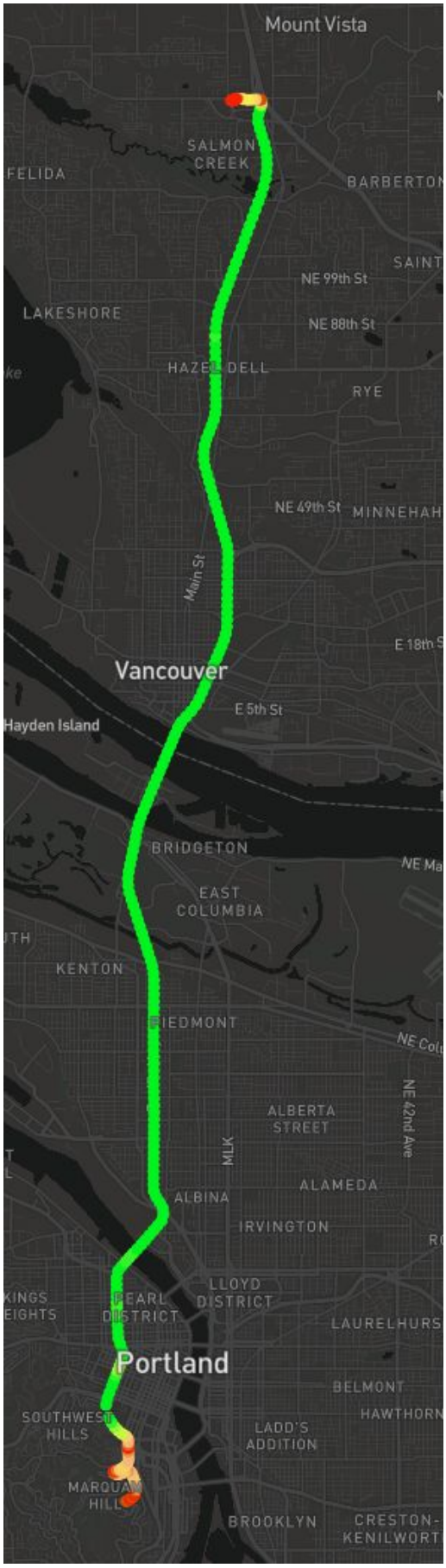
Maps:
On following pages, in this order:

trip_id	route_id
171076865	47
170817612	105
170872807	190

Improvements:
Now that I look at the maps. For most of them you could simply use the start/end point of the route instead of calculating the min/max of the x and y coordinates. That would be a suitable simplification. However this is more precise like this.


La Center
Yacolt
WEST
PIONEER
CRAWFORD
Kumtux
Battle Ground
Venersborg
Hockinson
Mount Vista
Brush Prairie
SALMON
CREEK
FELIDA
HAZEL DELL
FIVE
CORNERS
ORCHARDS
WALNUT
GROVE
SIFTON
PROEBSTEL
MINNEHAHA
BURTON
MILL PLAIN
Vancouver


Mount Vista
NE 159th St
Brush Prairie
NE 139th St
NE 117th Ave
SALMON CREEK
FELIDA
BARBERTON
UNION
NE 119th St
SAINT JOHN
NE 99th St
NE 99th St
NE 152nd Ave
LAKESHORE
NE 88th St
FIVE CORNERS
ORCHARDS
HAZEL DELL
Veteran's Memorial Fwy
RYE
WALNUT GROVE
SIFTON
NE 58th St
NE 49th St
MINNEHAHA
BURTON
E 18th St
Vancouver
EVERGREEN
E 5th St
Hayden Island
MCLOUGHLIN
ELLSWORTH
BRIDGETON
NE Marine Dr
Portland International Raceway
EAST COLUMBIA
KENTON
PIEDMONT
NE Columbia Blvd
Pacific Hwy
ALBERTA STREET
CULLY
SUMNER
PARKROSE
MLK
NE 42nd Ave
ALAMEDA
ROSEWAY
ALBINA
NE 122nd Ave
IRVINGTON
ROSE CITY PARK
LLOYD DISTRICT
KINGS HEIGHTS
PEARL DISTRICT
LAURELHURST
HAZELWOOD
Portland
MONTAVILLA
BELMONT
MT. TABOR
SOUTHWEST HILLS
LADD'S ADDITION
MILL PARK


Mount Vista
SALMON CREEK
FELIDA
BARBERTON
SAINT
NE 99th St
LAKESHORE
NE 88th St
HAZEL DELL
RYE
NE 49th St
MINNEHAH
Main St
E 18th S
Vancouver
E 5th St
Hayden Island
BRIDGETON
NE Ma
EAST COLUMBIA
KENTON
PIEDMONT
NE Colu
ALBERTA STREET
NE 42nd Ave
MLK
ALAMEDA
ALBINA
IRVINGTON
LLOYD DISTRICT
PEARL DISTRICT
LAURELHURS
Portland
BELMONT
HAWTHORN
SOUTHWEST HILLS
LADD'S ADDITION
BROOKLYN
CRESTON-KENILWORT

5b) Shortest Route - Distance

What is the shortest in terms of estimated distance traveled?
Map the shortest trips.

At first glance this seems easy. Simply use the same query as in 5a, just filter it in the other direction.

Query Description:
Calculate the hypotenuse of the bounding box of points in the trip as the trip length. Order the result by the length of the trip, with shortest first. Also return the trip info, like the route id.

Query Text:

```
select t.*, ib.*
from trip t inner join (
  select
  min(latitude) S,
  max(latitude) N,
  max(latitude) - min(latitude) as y_movement,
  min(longitude) E,
  max(longitude) W ,
  max(longitude) - min(longitude) as x_movement,
  sqrt((max(latitude) - min(latitude))^2 + (max(longitude) - min(longitude))^2) as total_movement,
  trip_id
  from breadcrumb
  group by trip_id
  order by 7 desc
  ) ib
on t.trip_id = ib.trip_id
where route_id > 0
and total_movement > 0
order by ib.total_movement
;
```

Example Result:

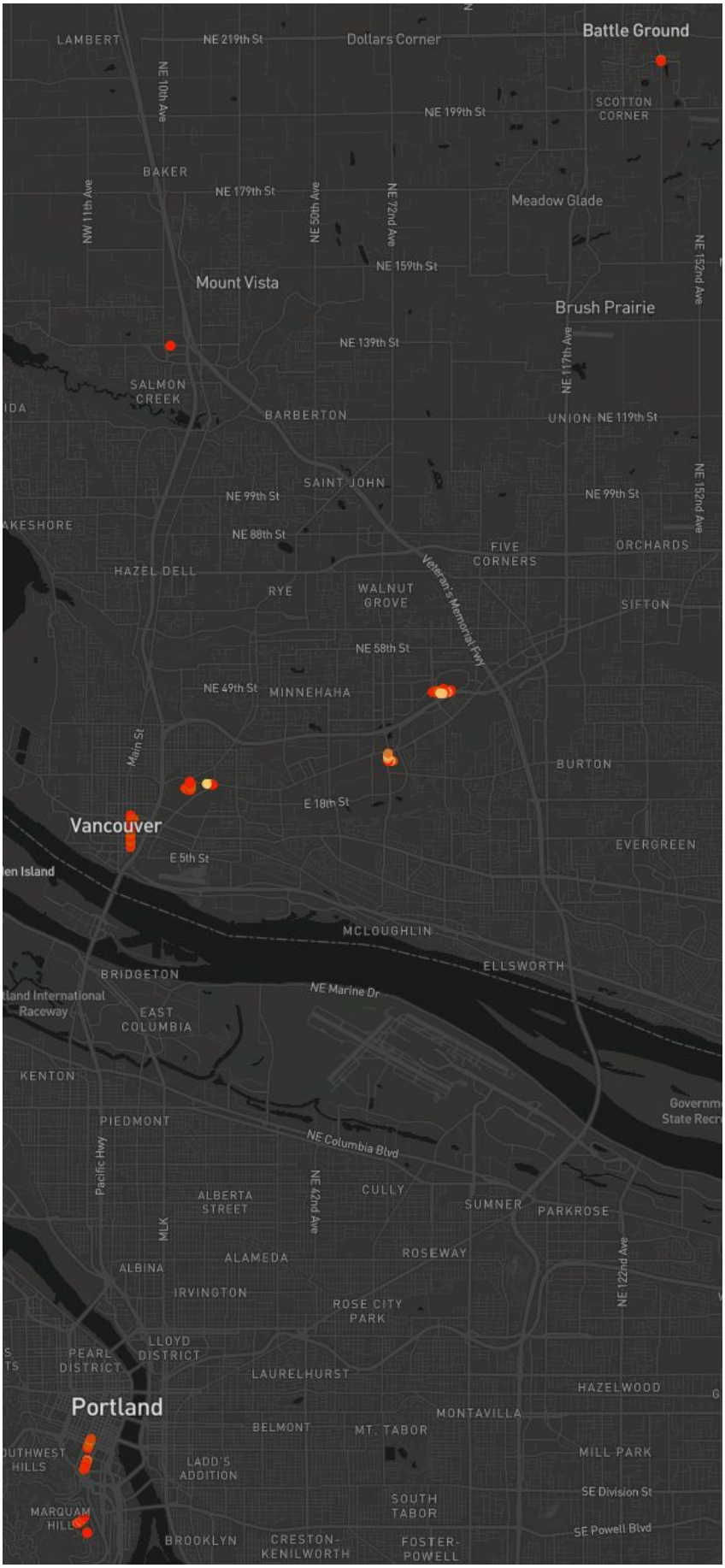
trip_id	route_id	vehicle_id	service_key	direction	s	n	y_movement	e	w	x_movement	total_movement	trip_id
171243872	190	2246	Weekday	Out	45.496162	45.496308	0.000146000000000766	-122.68237	-122.68229	0.000000001118224e-05	0.000166481230179388 44	171243872
171194612	41	2231	Weekday	Out	45.63264	45.632657	1.6999999999771169e-05	-122.670503	-122.67024	0.000262999999989688 15	0.000263548856940352 07	171194612
171425201	190	2242	Weekday	Out	45.496112	45.49639	0.000270000000001554 8	-122.682353	-122.682258	9.500000000173259e-05	0.000293783934212192 13	171425201
171448991	50	6004	Weekday	Out	45.65625	45.656347	9.699999999668307e-05	-122.584282	-122.583957	0.000325000000036834 5	0.000339166625719204 2	171448991
171424072	134	2232	Weekday	Out	45.721458	45.721682	0.000224000000020805 6	-122.659967	-122.659703	0.000264000000013742 5	0.000346225360137835 13	171424072
170990962	50	6010	Weekday	Out	45.655887	45.656102	0.000214999999971896 6	-122.587432	-122.587052	0.000384999999943865 6	0.000448964851266457	170990962
170978926	72	2902	Weekday	Out	45.656855	45.656227	0.000171999999991728 1	-122.588027	-122.58741	0.000616999999912079	0.000640852556445178 7	170978926
171329689	47	2248	Weekday	Out	45.775588	45.776177	0.000668999999949237	-122.527042	-122.526907	0.000135000000021828	0.000682485164669961	171329689
170876985	80	2289	Weekday	Out	45.656	45.65618	0.000180000000029104	-122.588480	-122.58767	0.000737999999983511	0.000759634122454798 1	170876985
170681624	7	2232	Weekday	Out	45.655763	45.65607	0.000306999999993911	-122.587457	-122.586732	0.000725000000027512	0.000787328773258025 6	170681624
171345414	134	4025	Weekday	Out	45.588998	45.589712	0.000714000000021018	-122.68289	-122.682523	0.000366999999999711963	0.000802798231189436	171345414
171255642	190	1254282	Weekday	Out	45.498882	45.49816	7.800000000280289e-05	-122.6856	-122.68464	8.980959999999920783	0.000963163537588094	171255642
171355552	74	2241	Weekday	Out	45.656813	45.65623	0.000216999999999245557	-122.58044	-122.587492	0.000948000000008164 5	0.000972510894425785 7	171355552
171426727	72	2262	Weekday	Out	45.655998	45.656153	0.000155000000006556 55	-122.588372	-122.587377	0.000995000000031070	0.001007000496528367 0	171426727
171422221	80	2245	Weekday	Out	45.655968	45.656235	0.000267000000009054	-122.588490	-122.587333	0.001165000000003047	0.001195284584909838 3	171422221
170623254	50	6009	Sunday	Back	45.655833	45.656625	0.000791999999998781 73	-122.586123	-122.585055	0.001068000000036216	0.001329619494443057 2	170623254
170810398	60	2292	Weekday	Out	45.630815	45.631897	0.001082000000038021	-122.670513	-122.66966	0.000853000000086432	0.001377000850807546 2	170810390
170905299	60	2291	Weekday	Out	45.630728	45.631907	0.001179000000004852	-122.67052	-122.669802	0.000717999999920028	0.001380422036911052 8	170905299
170802666	134	2228	Weekday	Out	45.50838	45.509752	0.001371999999963761	-122.68314	-122.682513	0.000626999999994382	0.001508488361152577 7	170802666
171457736	60	2294	Weekday	Out	45.638575	45.631853	0.001277999999999242	-122.670585	-122.669628	0.000877000000026812	0.001549971935238673 8	171457736
170704974	50	6004	Weekday	Back	45.655862	45.656175	0.000312999999998453 4	-122.585628	-122.584025	0.001603000000029077	0.001633272175728338 2	170704974
171345036	190	1254282	Weekday	Out	45.498325	45.499312	0.000987000000028696	-122.684575	-122.683047	0.001527999999933125	0.001019052775480593	171345036
171457732	60	2294	Weekday	Back	45.628452	45.63044	0.001987999999972142	-122.670553	-122.67042	0.001259999999945193 8	0.001991988955789555	171457732
171338818	60	2293	Weekday	Back	45.628438	45.630438	0.001999999999995339	-122.670563	-122.67045	0.001129999999891952	0.002003189706438487	171338818
170905294	60	2291	Weekday	Back	45.628497	45.630507	0.002009999999998513	-122.670572	-122.670447	0.000125000000011255	0.002013883865124893 5	170905294
171522296	134	2230	Weekday	Out	45.512370	45.514165	0.001787000000008205	-122.682180	-122.681255	0.000933000000034033	0.002015901287267579 6	171522296
171361154	60	2291	Weekday	Back	45.628478	45.630505	0.002026999999998224 7	-122.67056	-122.670438	0.000121999999990407 56	0.002030668116652857 3	171361154
171566814	50	6003	Weekday	Back	45.655813	45.656473	0.000659999999996330 2	-122.5859	-122.583963	0.001936999999998079 1	0.002046355852278981 6	171566814
170806553	72	2267	Weekday	Out	45.65573	45.656178	0.000447999999998671 7	-122.588238	-122.586235	0.002003000000019754	0.002052489464042805 7	170806553
170785513	80	2285	Weekday	Out	45.65573	45.656113	0.000382999999993561	-122.587493	-122.58547	0.002022999999994113	0.002058936133049223	170785513
171203157	60	2293	Weekday	Back	45.62841	45.630407	0.002076999999999845	-122.670577	-122.670445	0.000131999999935017	0.002801190284420078 4	171203157
171119328	74	2246	Weekday	Out	45.655738	45.655968	0.000230000000195807	-122.587305	-122.585233	0.002071999999902974	0.002084726360989704	171119328
171120834	60	2293	Weekday	Back	45.628283	45.63038	0.002096999999991273	-122.670567	-122.67045	0.000117000000083135	0.002108261412300156 7	171120834
170810394	60	2292	Weekday	Back	45.62829	45.630453	0.002163000000008236	-122.670545	-122.670515	3.000000000952241e-05	0.002163208834301726 4	170810394
170809934	60	2291	Weekday	Back	45.628277	45.630443	0.002166000000025548	-122.670553	-122.670437	0.000115999999913452 5	0.002169183962471384	170809934
170567074	60	1298380	Saturday	Back	45.626345	45.62854	0.002195000000008391	-122.670582	-122.670508	7.399999999790987e-05	0.002196247026179297	170567074
170633158	164	2248	Weekday	Out	45.50841	45.51039	0.001980000000032014	-122.683132	-122.682128	0.001003999999946758	0.002220083603601126	170633158
170566806	60	4015	Saturday	Back	45.628275	45.630498	0.002223000000008752	-122.670532	-122.670393	0.000138999999999811925	0.002227341464661663 97	170566806
171091671	50	6003	Weekday	Back	45.655808	45.656342	0.000534000000018108	-122.5062	-122.584032	0.002160000000011716	0.002323796452893262 4	171091671
170901573	72	2268	Weekday	Out	45.655715	45.656195	0.000479999999963915	-122.588465	-122.586203	0.002102000000047912	0.002234171082379935 7	170901573
170877257	60	2293	Weekday	Back	45.628237	45.630473	0.002236000000034574	-122.670558	-122.670437	0.000120999999992323 2	0.002239271533337069 5	170877257
171339538	47	2902	Weekday	Out	45.655735	45.656083	0.000348000000245745	-122.587452	-122.585227	0.002249999999957026	0.002252049955037096	171339538
170621802	60	4038	Sunday	Back	45.628273	45.63053	0.002257000000001755	-122.67057	-122.67043	0.000140000000008053 5	0.002261337873369145	170621802
170998738	60	2292	Weekday	Back	45.628167	45.630433	0.002265999999998769	-122.670555	-122.670412	0.000142999999994231 08	0.002270507652484961 6	170998738
171435768	190	2403	Weekday	Out	45.642888	45.64435	0.001542000000005984	-122.600713	-122.599013	0.001699999999995987	0.002295168996531715	171435768
170969487	164	2245	Weekday	Out	45.50845	45.510532	0.002081999999994366	-122.683125	-122.682142	0.000983000000050632	0.002382392885670579	170969487
171339229	72	2401	Weekday	Out	45.655733	45.656215	0.000482000000052805	-122.588277	-122.586002	0.002275000000115733	0.002325499731252978	171339229
170685686	72	2269	Weekday	Out	45.655748	45.656195	0.000446999999994891	-122.588312	-122.585983	0.002329000000083134	0.002371587959086226	170685686
170716794	6	2901	Weekday	Out	45.637377	45.63985	0.001672999999996705	-122.655733	-122.654025	0.001707999999936035	0.002398056122807711 7	170716794
170684224	6	2262	Weekday	Out	45.630448	45.630673	0.000225000000003638	-122.650680	-122.648197	0.002411000000009267	0.002421475596173503 5	170684224
170002137	41	1776	Weekday	Out	45.594137	45.594887	0.000749999999964757	-122.506242	-122.503927	0.002314999999995848	0.002433459471611453	170002137
171453993	72	2271	Weekday	Out	45.65573	45.656182	0.000452000000027835	-122.588377	-122.585945	0.002431999999988795	0.002473646700722855	171453993
171542879	67	4029	Weekday	Out	45.594107	45.59489	0.000782999999984238	-122.506223	-122.503863	0.002360000000101318	0.002486501357338329 4	171542879

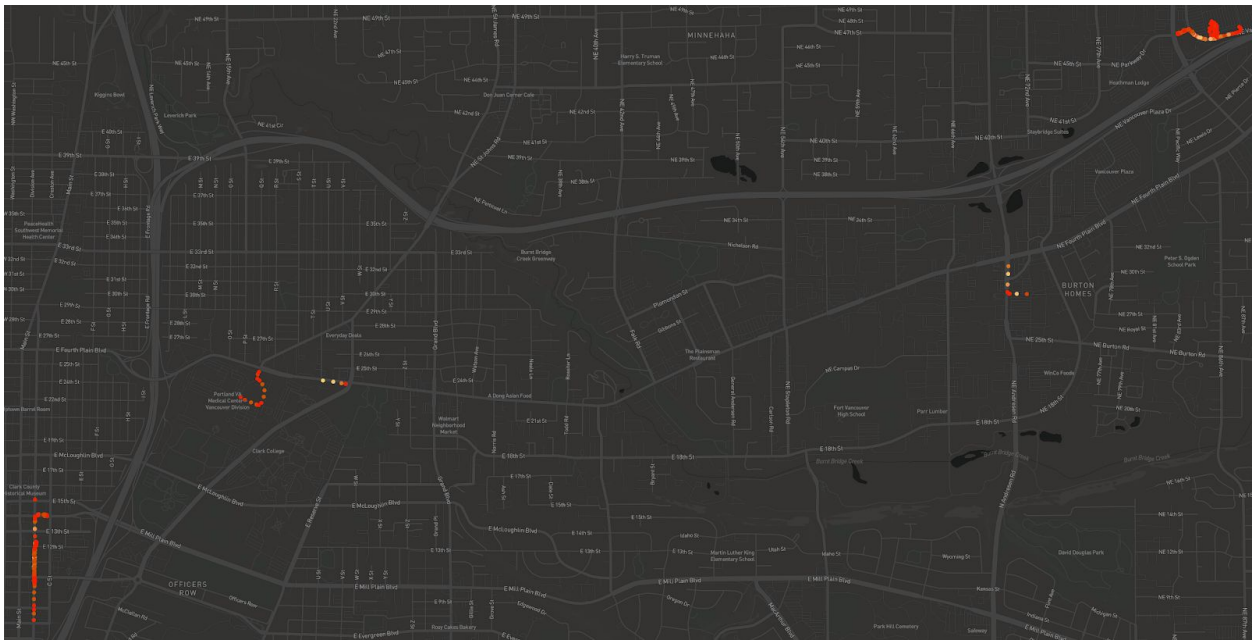
We can already see, something strange is going on. If this were really a representation of the shortest trips, we would see many of the same route id, the shortest route. Let's map it anyway to check it out. Map the first 50, the "shortest trips".

Queries to pull the data:
Pull the trip IDs for the first 50:

select longitude, latitude, speed from
breadcrumb where trip_id in
(171243872, 171194612, 171425201, 171448991, 171424072, 170990962, 170978926, 171329689, 170876985, 170681624, 171345414, 171255642, 171355552, 171426727, 171422221, 170623254, 170810398, 170905299, 170802666, 171457736, 170704974, 171345036, 171457732, 171338818, 170905294, 171522296, 171361154, 171566814, 170806553, 170785513, 171203157, 171119328, 171120834, 170810394, 170809934, 170567074, 170633158, 170566806, 171091671, 170901573, 170877257, 171339538, 170621802, 170998738, 171435768, 170969487, 171339229, 170685686, 170716794, 170684224)

Map:
All the data points, and a zoom in on downtown Vancouver.


Battle Ground
LAMBERT
NE 219th St
Dollars Corner
NE 10th Ave
NE 199th St
SCOTTON CORNER
BAKER
NE 179th St
NW 11th Ave
NE 50th Ave
NE 72nd Ave
Meadow Glade
NE 152nd Ave
NE 159th St
Mount Vista
Brush Prairie
NE 139th St
NE 117th Ave
SALMON CREEK
BARBERTON
UNION
NE 119th St
SAINT JOHN
NE 99th St
NE 99th St
NE 152nd Ave
NE 88th St
FIVE CORNERS
ORCHARDS
HAZEL DELL
RYE
WALNUT GROVE
Veteran's Memorial Fwy
SIFTON
NE 58th St
NE 49th St
MINNEHAHA
Main St
BURTON
E 18th St
Vancouver
E 5th St
EVERGREEN
MCLOUGHLIN
ELLSWORTH
BRIDGETON
NE Marine Dr
EAST COLUMBIA
KENTON
PIEDMONT
NE Columbia Blvd
Pacific Hwy
NE 42nd Ave
CULLY
ALBERTA STREET
SUMNER
PARKROSE
MLK
ALAMEDA
ROSEWAY
NE 122nd Ave
ALBINA
IRVINGTON
ROSE CITY PARK
LLOYD DISTRICT
PEARL DISTRICT
LAURELHURST
HAZELWOOD
Portland
MONTAVILLA
BELMONT
MT. TABOR
MILL PARK
LADD'S ADDITION
SE Division St
SOUTH TABOR
MARQUAM HILL
SE Powell Blvd
BROOKLYN
CRESTON-KENILWORTH
FOSTER-POWELL



Analysis and Future Work:

There is no continuity between these points. They do not represent routes. I'm going to suspect these are error readings, where a trip was unexpectedly cut off. While my first inclination is to toss this and do a different visualization or filter these out, this query could be used for data validation, after the data has been loaded into the database. We could compare the average route length estimate for the route, and for any trips where the length is significantly smaller, those trips and their breadcrumbs should be dropped. A lot of these are in downtown Vancouver or other city centers, so it's possible they represent a bus starting up and shutting down, for example if a driver has a break scheduled at the and of a trip. This does not represent movement in traffic, and they are misrepresetingly slow moving (very red/orange due to slow speeds) so we don't want them mucking up our dataset, whose purpose is to investigate traffic movement problems.

To find the actual shortest route, you could do it very precisely by dropping what may be bad datapoints as described above. Alternatively, but a similar concept of data could likely have been gathered by collecting the max length trip, for the route with the shortest average trip. Something like:

5c) Slowest 50 trips

Map the slowest 50 trips.
Slowest is defined as the average speed of the trip. Don't include very shortened trips that may have been miscalculations.

We tried to find shortest trip, we may as well try to find the slowest. First off there is a similar issue, where many datapoints show a speed of 0. So we'll average the speed across the trip. Additionally, now that we know a lot of trips are bad data, let's filter them out. Only keep the trips, where its length is as expected.

Query Description:
What is the trip with the slowest average speed per trip, where the length of the trip is between the max length of trips on that route, and the average length of trips on that route?
I used some CTEs for this to make it easier for me to conceptualize it.

Query text:

```
with greatest_cte as (
  select
  long_routes.route_id,
  max(long_routes.length_est) as max_length
  from (
        select t.*, ib.*
        from trip t inner join (
        select
        sqrt((max(latitude) - min(latitude))^2 + (max(longitude) - min(longitude))^2) as
length_est,
        trip_id
        from breadcrumb
        group by trip_id
        ) ib
        on t.trip_id = ib.trip_id
        where route_id > 0
  ) long_routes
  group by 1
),
middling_cte as
(
  select
  long_routes.route_id,
  avg(long_routes.length_est) as avg_length
  from (
        select t.*, ib.*
        from trip t inner join (
```

```
        select
        sqrt((max(latitude) - min(latitude))^2 + (max(longitude) - min(longitude))^2) as
length_est,
        trip_id
        from breadcrumb
        group by trip_id
        ) ib
        on t.trip_id = ib.trip_id
        where route_id > 0
  ) long_routes
  group by 1
),
this_trip_cte as
(
  select t.route_id, ib.*
  from trip t inner join (
        select
        avg(speed) as trip_speed,
        sqrt((max(latitude) - min(latitude))^2 + (max(longitude) - min(longitude))^2) as
length_est,
        trip_id
        from breadcrumb
        group by trip_id
        ) ib
  on t.trip_id = ib.trip_id
  where route_id > 0
)
select
  tc.trip_id,
  tc.trip_speed,
  tc.route_id,
  tc.length_est,
  gc.max_length,
  mc.avg_length
from
  this_trip_cte tc
  join greatest_cte gc
  on tc.route_id = gc.route_id
  join middling_cte mc
  on gc.route_id = mc.route_id
where tc.route_id in (6, 60, 19)
and tc.length_est <= gc.max_length
and tc.length_est >= mc.avg_length
order by 2;
```

Example result:

trip_id	trip_speed	route_id	length_est	max_length	avg_length
171525855	4.526041666666667	6	0.05356358735186373	0.0936146677236058	0.03389456739974154
171269108	5.197530864197531	6	0.039727726464017316	0.0936146677236058	0.03389456739974154
170879378	5.276923076923077	19	0.052188512088390634	0.08867950993323866	0.051085317247486536
171357318	5.342541436464089	6	0.03994323783820918	0.0936146677236058	0.03389456739974154
171180798	5.394636015325671	19	0.05215056801224857	0.08867950993323866	0.051085317247486536
171161934	5.950920245398773	6	0.03985348800294327	0.0936146677236058	0.03389456739974154
170684028	6.099290780141844	6	0.03975776190884689	0.0936146677236058	0.03389456739974154
170972035	6.233576642335766	6	0.03971924703465229	0.0936146677236058	0.03389456739974154
171453472	6.238805970149253	6	0.039762912732833625	0.0936146677236058	0.03389456739974154
171226796	6.318181818181818	60	0.040892009402815184	0.17028464015583014	0.040124873466136676
170612966	6.4144736842105265	6	0.039751089658018476	0.0936146677236058	0.03389456739974154
171453579	6.486486486486487	6	0.03934347341047446	0.0936146677236058	0.03389456739974154
171122635	6.4944444444444445	60	0.04102605440448915	0.17028464015583014	0.040124873466136676
171453459	6.5225225225225225	6	0.03938004499744239	0.0936146677236058	0.03389456739974154
171452929	6.544776119402985	6	0.039719769309490366	0.0936146677236058	0.03389456739974154
170716771	6.598214285714286	6	0.0394041425360303	0.0936146677236058	0.03389456739974154
170880147	6.612903225806452	6	0.03972066376584783	0.0936146677236058	0.03389456739974154
170684140	6.62962962962963	6	0.03937365044035934	0.0936146677236058	0.03389456739974154
170684016	6.636363636363637	6	0.039366031715182076	0.0936146677236058	0.03389456739974154
171332096	6.724770642201835	6	0.03935969472696099	0.0936146677236058	0.03389456739974154
170684089	6.738095238095238	6	0.039751963787973965	0.0936146677236058	0.03389456739974154
170795828	6.763285024154589	19	0.05224671037491292	0.08867950993323866	0.051085317247486536
170879715	6.770642201834862	6	0.039380877593065904	0.0936146677236058	0.03389456739974154
170880135	6.773584905660377	6	0.039315969325444136	0.0936146677236058	0.03389456739974154
171250837	6.776923076923077	6	0.03973812976221975	0.0936146677236058	0.03389456739974154
170781886	6.79047619047619	6	0.03936851083035415	0.0936146677236058	0.03389456739974154
170879658	6.790909090909091	6	0.03935176166323295	0.0936146677236058	0.03389456739974154
171331978	6.803738317757009	6	0.039360980678841945	0.0936146677236058	0.03389456739974154
170972633	6.841666666666667	6	0.03977123046877937	0.0936146677236058	0.03389456739974154
171453042	6.869158878504673	6	0.039346837357035765	0.0936146677236058	0.03389456739974154
170781180	6.88785046728972	6	0.039362226982219264	0.0936146677236058	0.03389456739974154
170880210	6.895161290322581	6	0.03973975523326968	0.0936146677236058	0.03389456739974154
170683490	6.902912621359223	6	0.039364709385436035	0.0936146677236058	0.03389456739974154
171332049	6.903225806451613	6	0.039759679639048376	0.0936146677236058	0.03389456739974154
171250893	6.9105691056910565	6	0.03970971575068446	0.0936146677236058	0.03389456739974154
171617798	6.9245283018867925	6	0.039427110723462076	0.0936146677236058	0.03389456739974154
170880199	6.951456310679611	6	0.03932570366820631	0.0936146677236058	0.03389456739974154
170781311	7	6	0.03976439800876713	0.0936146677236058	0.03389456739974154
171356667	7.008064516129032	6	0.0535661632376305 5	0.0936146677236058	0.03389456739974154
170972684	7.009803921568627	6	0.0393609695129587	0.0936146677236058	0.03389456739974154
171269568	7.01	6	0.03939009678841408	0.0936146677236058	0.03389456739974154
171617327	7.024390243902439	6	0.039758134689646445	0.0936146677236058	0.03389456739974154
171437179	7.04	6	0.03935641229837957	0.0936146677236058	0.03389456739974154
171332152	7.049019607843137	6	0.03934743540563274	0.0936146677236058	0.03389456739974154
171617682	7.058333333333334	6	0.03972776339035063	0.0936146677236058	0.03389456739974154
171123269	7.076335877862595	60	0.04092922899346972	0.17028464015583014	0.040124873466136676
170972741	7.076923076923077	6	0.03934681732998686	0.0936146677236058	0.03389456739974154
171357131	7.079207920792079	6	0.0393681735669907	0.0936146677236058	0.03389456739974154
171269824	7.125	6	0.03975122117620888	0.0936146677236058	0.03389456739974154
170781949	7.137254901960785	6	0.039343626332615164	0.0936146677236058	0.03389456739974154
171162273	7.138613861386139	6	0.039458683632383626	0.0936146677236058	0.03389456739974154
171269704	7.142857142857143	6	0.039746454684672854	0.0936146677236058	0.03389456739974154
171269753	7.15	6	0.039379949123901484	0.0936146677236058	0.03389456739974154

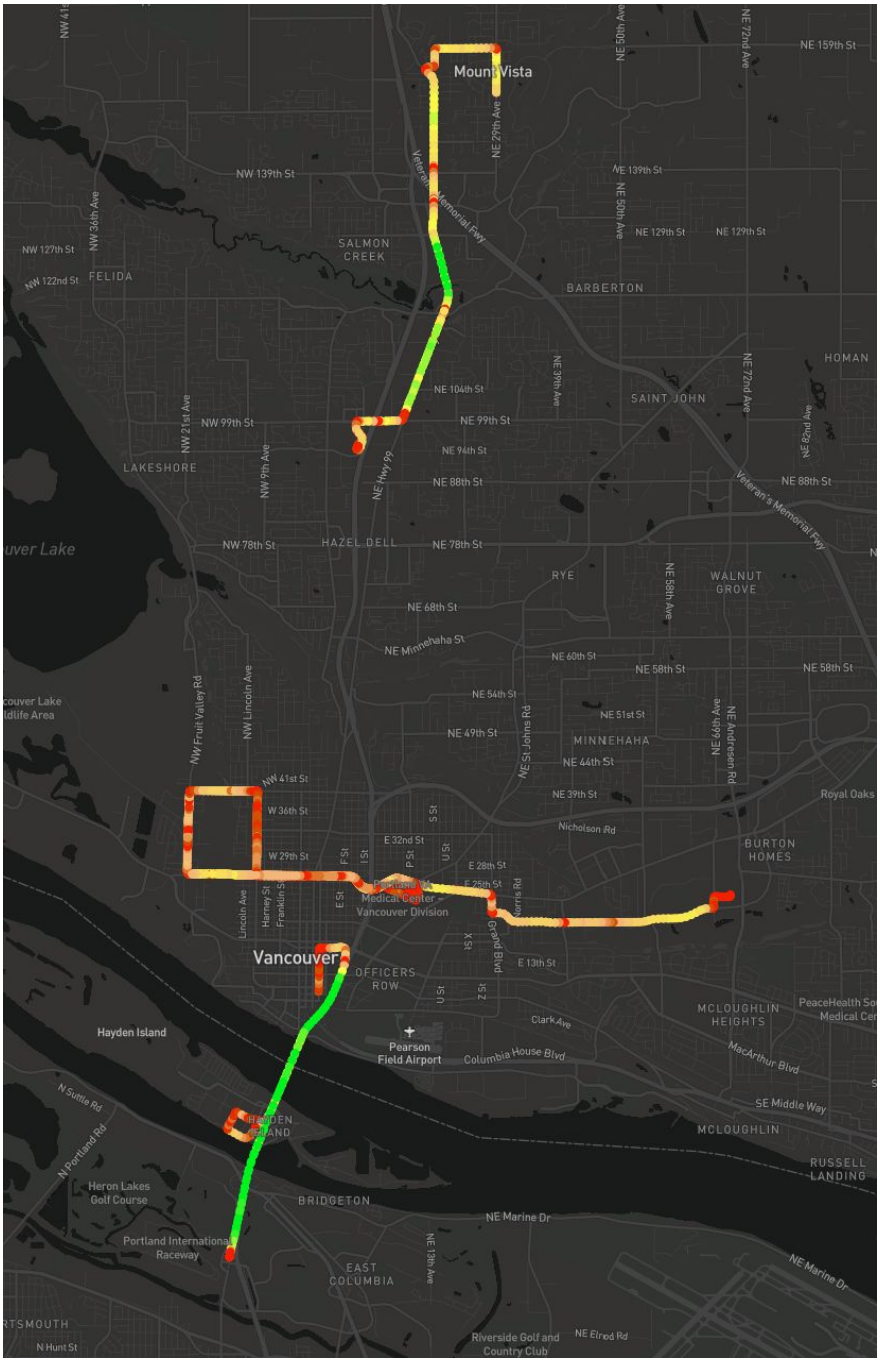
Next I did a simple query of the breadcrumb data for the first 50 trips in that list, as seen below. Alternatively you could use the previous query as the in-list, and only select the trip ID out of that subquery. But I liked taking that intermediate step, so I can visually validate results as seen above.

Data Generating Query:

```
select longitude, latitude, speed from
breadcrumb where trip_id in
(171525855, 171269108, 170879378, 171357318, 171180798, 171161934, 170684028,
170972035, 171453472, 171226796, 170612966, 171453579, 171122635, 171453459,
```

171452929, 170716771, 170880147, 170684140, 170684016, 171332096, 170684089, 170795828, 170879715, 170880135, 171250837, 170781886, 170879658, 171331978, 170972633, 171453042, 170781180, 170880210, 170683490, 171332049, 171250893, 171617798, 170880199, 170781311, 171356667, 170972684, 171269568, 171617327, 171437179, 171332152, 171617682, 171123269, 170972741, 171357131, 171269824, 170781949)

Map:



SQL Performance

I was interested to find that there was a significant performance difference between these 2 CTE below. I ran them on their own as a debugging step before running the full query above. The first one hangs because it takes so long. I thought I would be improving results, by selecting fewer columns so the project is not as large, and not nesting things as much, which was making it easier for me to think of this conceptually. However it seems operating on just one table at a time to pull the trip ids and calculate their length, and subsequently pulling data for those, is a much better plan.

```
# This was very very not performant, it hung.

with this_trip_cte as
(
  select
  b.trip_id,
  max(route_id),
  sqrt((max(latitude) - min(latitude))^2 + (max(longitude) - min(longitude))^2) as length_est
  from breadcrumb b
  inner join trip t
  on b.trip_id = b.trip_id
  where route_id > 0
  group by b.trip_id
)
select * from this_trip_cte limit 1

# This was very fast, comparatively.

with this_trip_cte as
(
        select t.*, ib.*
        from trip t inner join (
        select
        sqrt((max(latitude) - min(latitude))^2 + (max(longitude) - min(longitude))^2) as
length_est,
        trip_id
        from breadcrumb
        group by trip_id
        ) ib
        on t.trip_id = ib.trip_id
        where route_id > 0
  )
select * from this_trip_cte limit 1
```

Your Code

Provide a reference to the repository where you store your code. If you are keeping it private then share it with Bruce (bruce.irvin@gmail.com), David and Aman (github references TBD).

https://github.com/coding-gen/dataeng/tree/main/project/4_final