DataEng S22: Project Assignment 2

Validate, Transform, Enhance and Store

Due date: May 8, 2022 @10pm PT

By now your pipeline should be automatically gathering and transferring C-Tran breadcrumb records daily but not yet doing much with the data. The next step in building your data pipeline is to get your data in shape for analysis. This includes:

A. understanding the contents of the bread crumb data
B. reviewing the needed schema for the data
C. validating the data
D. transforming the data
E. storing the data into a database server
F. example queries

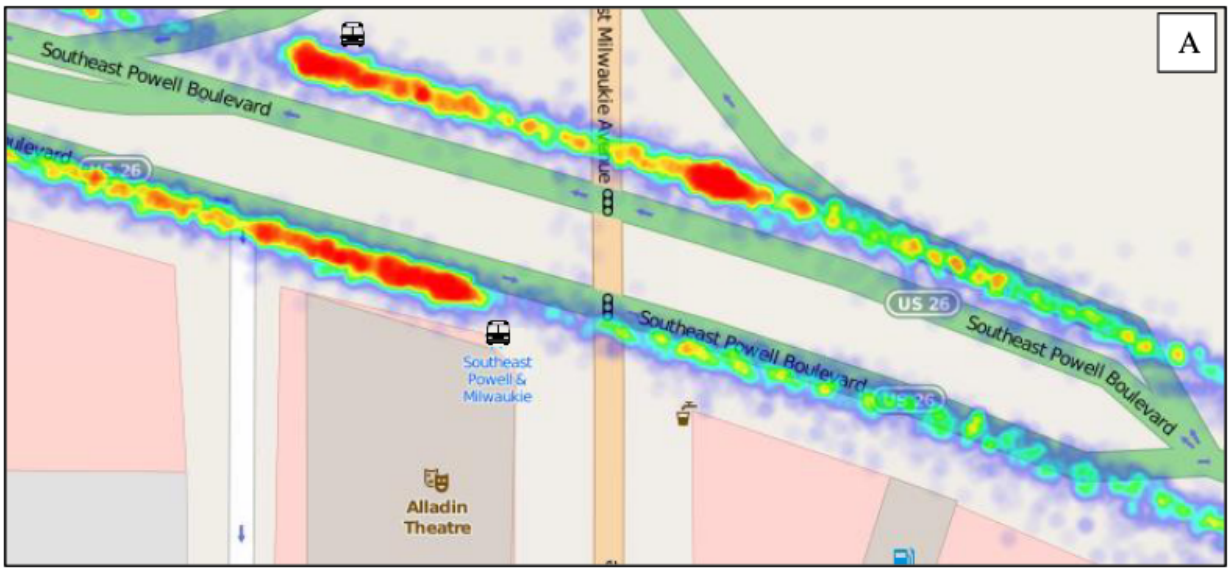
Your job is to enhance your Kafka consumer to perform steps A-E so that your pipeline ingests high-quality data daily into a database server in preparation for visualization.

A. Review the Data

Have a look at a few of the data records and see if you can determine the meaning of each record attribute. Also, have a look at the only documentation that we have for the data (provided to us by C-Tran). The documentation is not very good, so we are counting on you to give better descriptions of each field. In your submission document, provide an improved description of each data field.

B. Database Schema

We plan to develop a visualization application that builds visualizations similar to this:



This diagram shows vehicle speeds as a heatmap overlaid on a street map for a given latitude/longitude rectangle, route, date range and time range. Colors in the heatmap correspond to average speeds of buses at each GPS location for the selected route, date range and/or time range.

To achieve this we need you to build database tables (or views) that conform to the following schema.

Table: **BreadCrumb**

tstamp	latitude	longitude	direction	speed	trip_id

The BreadCrumb table keeps track of each individual breadcrumb reading. Each row of the BreadCrumb table corresponds to a single GPS sensor reading from a single bus.

tstamp: the moment at which the event occurred, i.e., at which the bus's GPS location was recorded. This should be a Postgres "timestamp" type of column.

latitude: a float value indicating fractional degrees of latitude. Latitude and Longitude together indicate the location of the bus at time=tstamp.

longitude: a float value indicating the fractional degrees of longitude. Latitude and Longitude indicate the location of the bus at time=tstamp.

direction: an integer value in the range 0..359 indicating the forward facing direction of the bus at time=tstamp. Direction is measured in degrees, 0 equals north.

speed: a float value indicating the speed of the bus at time=tstamp. Speed is measured in miles per hour.

trip_id: this integer column indicates the identifier of the trip in which the bus was participating. A trip is a single run of a single bus servicing a single route. This column is a foreign key referencing trip_id in the Trip table.

Table: **Trip**

trip_id	route_id	vehicle_id	service_key	direction

The Trip table keeps track of information about each bus trip. A “trip” is a single run of a single bus over a single route.

trip_id: unique integer identifier for the trip.

route_id: integer identifying the route that the trip was servicing. For this project we do not have a separate Route table, but if we did, then this would be a foreign key reference to the Route table. Note that you will not have enough information to properly populate this field during assignment #2.

vehicle_id: integer identifying the vehicle that serviced the trip. A trip is serviced by only one vehicle but a vehicle services potentially many trips.

service_key: one of ‘Weekday’, ‘Saturday’, or ‘Sunday’ depending on the day of the week on which the trip occurred OR depending on whether the trip occurred on a Holiday recognized by C-Tran. Note that you will not have enough information to properly populate this field during assignment #2.

direction: either ‘0’ or ‘1’ indicating whether the trip traversed the route from beginning to end (‘0’) or from end to beginning (‘1’). Note that this “direction” has a completely different meaning than the same-named column in the BreadCrumb table (sorry about that). Also note that you will not have enough information to properly populate this field during assignment #2.

See the ER diagram and DDL code for the schema. Please implement this schema precisely in your database so that the visualization tool will work for you.

C. Data Validation

Create 20 distinct data validation assertions for the bread crumb data. These should be in English (not python). Include them in your submission document (see below).

Then implement at least 10 of the assertions in your Kafka consumer code. Implement a variety of different types of assertions so that you can experience with each of the major types of data

validation assertions: existence, limit, intra-record check, inter-record check, summary, referential integrity, and distribution/statistical assertions.

D. Data Transformation

Add code to your Kafka consumer to transform your data. Your transformations should be driven by either the need to resolve validation assertion violations or the need to shape the data for the schema. Describe any/all needed transformations in your submission document.

E. Storage in Database Server

1. Install and configure a PostgreSQL database server on your Virtual Machine. Refer to this document to learn how. For more information on PostgreSQL commands, you can see guides like this one: https://www.postgresqltutorial.com/postgresql-cheat-sheet/
2. Enhance your data pipeline to load your transformed data into your database server. You are free to use any of the data loading techniques that we discussed in class. The data should be loaded ASAP after your Kafka consumer receives the data, validates the data and transforms the data so that you have a reliable end-to-end data pipeline running daily, automatically.

F. Example Queries

Answer the following questions about the C-Tran system using your sensor data database. In your submission document include your query code, number of rows in each query result (if applicable) and first five rows of the result (if applicable).

1. How many vehicles are there in the C-Tran system?
2. How many bread crumb reading events occurred on October 2, 2020?
3. How many bread crumb reading events occurred on October 3, 2020?
4. On average, how many bread crumb readings are collected on each day of the week?
5. List the C-Tran trips that crossed the I-5 bridge on October 2, 2020. To find this, search for all trips that have bread crumb readings that occurred within a lat/lon bounding box such as [(45.620460, -122.677744), (45.615477, -122.673624)].
6. List all bread crumb readings for a specific portion of Highway 14 (bounding box: [(45.610794, -122.576979), (45.606989, -122.569501)]) during Mondays between 4pm and 6pm. Order the readings by tstamp. Then list readings for Sundays between 6am and 8am. How do these two time periods compare for this particular location?
7. What is the maximum velocity reached by any bus in the system?
8. List all possible directions and give a count of the number of vehicles that faced precisely that direction during at least one trip. Sort the list by most frequent direction to least frequent.
9. Which is the longest (in terms of time) trip of all trips in the data?

10. Devise three new, interesting questions about the C-Tran bus system that can be answered by your bread crumb data. Show your questions, their answers, the SQL you used to get the answers and the results of running the SQL queries on your data (the number of result rows, and first five rows returned).

Submission

Congratulations! Your data pipeline is now working end-to-end!

To submit your completed Assignment 2, create a google document containing the table shown below. Share the document as viewable by anybody at PSU who has the link. You do NOT need to share it with the individual instructors. Then include the URL of the document when using the DataEng project assignment submission form.

DataEng Project Assignment 2 Submission Document

Construct a table showing each day for which your pipeline successfully, automatically processed one complete day's worth of sensor readings. The table should look like this:

Date	Day of Week	# Sensor Readings	# updates/insertions into your database
04-28-22	Thursday		
04-29-22	Friday		
04-30-22	Saturday		
05-01-22	Sunday		
05-02-22	Monday		
05-03-22	Tuesday		
05-04-22	Wednesday		
05-05-22	Thursday		
05-06-22	Friday		
05-07-22	Saturday		

Documentation of Each of the Original Data Fields

For each of the fields of the bread crumb data, provide any documentation or information that you can determine about it. Include bounds or distribution data where appropriate. For example, for something like "Vehicle ID", say something more than "It is the identification number for the vehicle". Instead, add useful information such as "the integers in this field range from <min val> to <max val>, and there are <n> distinct vehicles identified in the data. Every vehicle is used on weekdays but only 50% of the vehicles are active on weekends."

EVENT_NO_TRIP: This field in the bread crumb data is of type "Number", and holds information in regards to the trip index.
EVENT_NO_STOP: This field is used to determine the stop index, or the point index. This is of type "Number".
OPD_DATE: This field is used to determine what date this vehicle even occurred on. This is of type "Date".

VEHICLE_ID: This is the vehicle ID number, and determines what vehicle all of this data is coming from. This is of type “Number”.
METERS: This is the odometer reading at every stop that the vehicle makes, we will know how many miles the vehicle has traveled based on this. This is of type “Number”.
ACT_TIME: The act time is the actual time the reading of this data was made. This is of type “Number”.
VELOCITY: The velocity is the directional speed of the bus in motion. This is also of type “Number”.
DIRECTION: The direction is the geo orientation of the bus, and is of type “Varchar2”. This tells us what direction the bus is facing.
RADIO_QUALITY: Radio quality is the signal strength, and this column isn’t populated at all. The type for this is “Number”.
GPS_LONGITUDE: The GPS longitude, describes the longitude on the map of the vehicle during the stop. This is of type “Number’.
GPS_LATITUDE: The GPS Latitude, describes the latitude on the map of the vehicle during the stop. This is of type “Number”.
GPS_SATELLITES: This describes the count of satellites, and is of type “Number”.
GPS_HDOP: This refers to the horizontal dilution of precision, which is the effects on accuracy of the combined errors in a two-dimensional fix obtained from crossing two lines of position. Thi is of type “Number”.
SCHEDULE_DEVIATION: This refers to the current schedule deviation, which is defined as the absolute value of the difference between actual schedule and planned schedule. y= |Actual - Planned|.

Data Validation Assertions

List 20 or more data validation assertion statements here. These should be English language sentences similar to “The speed of a C-Tran bus should not exceed 100 miles per hour” . You will only implement a subset of them, so feel free to write assertions that might be difficult to evaluate. Create assertions for all of the fields, even those, like RADIO_QUALITY, that might not be used in your database schema.

1. Data must have latitude and longitude
2. If the longitude exists, so will the latitude
3. Direction must be between 0 and 365
4. There should be a timestamp for each stop
5. ACT_TIME must be within 24 hours.
6. EVENT_NO_TRIP must not be null or empty
7. Latitude and longitude must be close to Clark County in the State of Washington
8. Each bus has a unique Vehicle ID
9. Each piece of data must have an OPD_DATE
10. GPS_LONGITUDE must be within valid range
11. Meters cannot be negative

12. GPS_LATITUDE must be within valid range
13. Every breadcrumb needs a GPS_SATELLITES.
14. Direction degrees should vary more than 100% repeatedly (going in circles)
15. Busses must make at least 2 stops
16. Meters must be within a valid range of where the bus is traveling
17. Every bus trip is for a known bus route
18. Vehicle ID must be unique across all records
19. Total # vehicle ID must be about 100 vehicles.

Data Transformations

Describe any transformations that you implemented either to react to validation violations or to shape your data to fit the schema. For each, give a brief description of the transformation along with a reason for the transformation.

> One thing we saw is that there is no speed value in the data, only velocity. We need to transform this to fit the scheme. 1 meter per second = 2.2369 miles per hour, accurate to 5 significant figures.
>
> We also needed to make sure that the tstamp, had the right type of timestamp, and not an int. We had to combine the ACT_TIME, and OPD_DATE, to create the tstamp.
>
> When trying to insert the data, we had to make sure our queries were correct, and we had issues with some of the parameters such as “DIRECTION” being empty. We had to validate that data to make sure it was correct.

Example Queries

Provide your responses to the questions listed in Section E above. For each question, provide the SQL you used to answer the questions along with the count of the number of rows returned (where applicable) and a listing of the first 5 rows returned (where applicable).

Your Code

Provide a reference to the repository where you store your python code. If you are keeping it private then share it with Bruce (bruce.irvin@gmail.com) and Genevieve (sgl@pdx.edu).