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

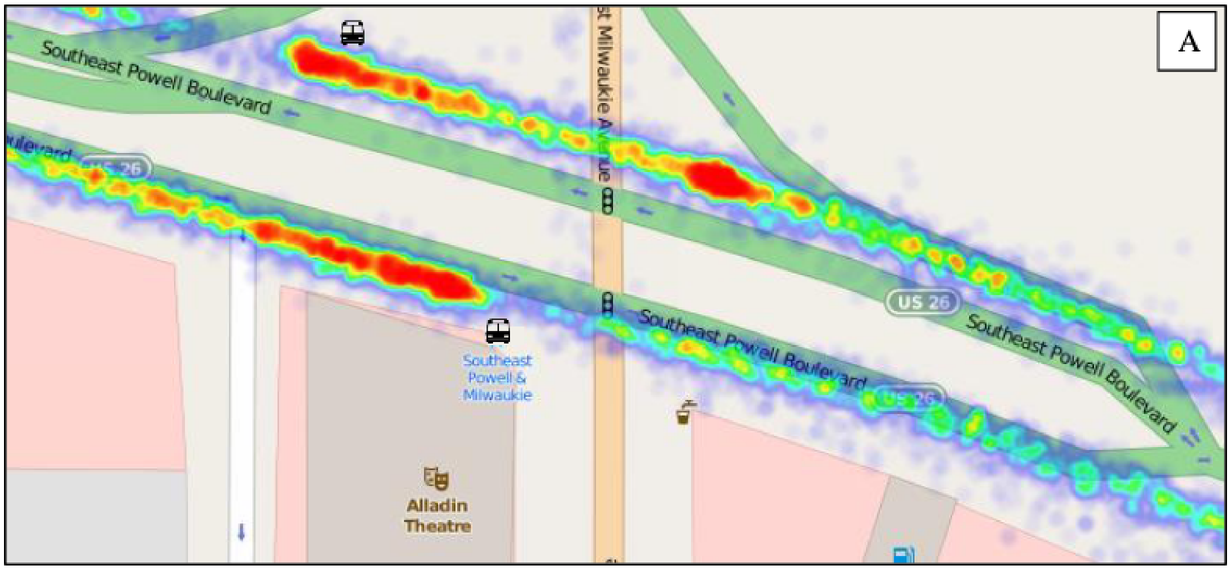
Your job is to enhance your Kafka consumer to perform steps A-E so that your pipeline ingests high-quality data daily into a database server in preparation for visualization.

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:


A
Southeast Powell Boulevard
Southeast Powell Boulevard
Southeast Powell Boulevard
US 26
Milwaukie Avenue
Southeast Powell & Milwaukie
Alladin Theatre

This diagram shows vehicle speeds as a heatmap overlaid on a street map for a given latitutude/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.

Note to Winter 2021 Students

The course is new, the project is untested, and we might need to update this data specification again. Sorry about that, we will try to minimize changes and communicate changes via slack or email as well as updates to this document.

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 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 this document to learn how.
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 databsae

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
EVENT_NO_STOP
OPD_DATE
VEHICLE_ID
METERS
ACT_TIME

VELOCITY
DIRECTION
RADIO_QUALITY
GPS_LONGITUDE
GPS_LATITUDE
GPS_SATELLITES
GPS_HDOP
SCHEDULE_DEVIATION

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.

- Existence - something must exist within the data
- Limit - a value/field must fall within some range
- Intra-record - fields within a record must satisfy some relationship
- Inter-record - fields from separate records must satisfy some relationship
- Summary - Some property or relationship must be satisfied when examining all of the records as a whole
- Referential integrity - A field within each record must correspond precisely to a field within other records
- Statistical - The overall data set must conform to some statistical property (usually a distribution).

1. EVENT_NO_TRIP field is a 9 digit integer for all records.
2. EVENT_NO_STOP field is a 9 digit integer for all records.
3. OPD_DATE field is a date in 2020 for all records.
4. VEHICLE_ID field is a 4 digit integer for all records.
5. METERS field will be a larger integer than any record with the same EVENT_NO_TRIP and VEHICLE_ID at an earlier ACT_TIME for all records.
6. ACT_TIME field is in seconds between midnight up to at most 2am the next day.
7. VELOCITY field exists and is greater than or equal to 0 mph for all records.
8. DIRECTION field will be an integer within the range [0,360).
9. RADIO_QUALITY field is empty for all records.
10. GPS_LONGITUDE field must be near -122.676483 (Portland’s longitude).
11. GPS_LATITUDE field must be near 45.523064 (Portland’s latitude).
12. GPS_SATELLITES field is greater than 0 for all records.
13. GPS_HDOP field is a positive floating point number.
14. SCHEDULE_DEVIATION field will be populated for the majority of records.
15. An EVENT_NO_TRIP number should be associated with only 1 VEHICLE_ID.

16. Every record will have the same ordering of fields.
17. There are more breadcrumbs during the weekdays than during the weekends.
18. There are more breadcrumbs during working hours between 8am and 5pm daily.
19. A VELOCITY field will not be a negative number.
20. The OPD_DATE field is in the format DD-MMM-YY.

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.

Transformation that occurred during data validation:

1) I changed the data type of the EVENT_NO_TRIP field to be integer from string in order to validate with the between() function.
2) I did the same transformation in number 1 to the EVENT_NO_STOP field.
3) I applied the same transformation in number 1 to the VEHICLE_ID field.
4) Once again, I transformed the ACT_TIME field into an integer data type similar to number 1.
5) I replaced all empty strings with "0" and then changed the data type to float in the VELOCITY field.
6) I performed the same transformation in number 5 to the DIRECTION field but changed the type to integer instead of float.
7) I replaced all empty strings with "0" in the GPS_HDOP field.

Transformation that occurred after validation in order to streamline loading of the data:

1) I changed the OPD_DATE field into a datetime type using the pandas to_datetime method.
2) I changed the ACT_TIME field into a timedelta type using the pandas to_timedelta method
3) I updated the OPD_DATE field by adding the old value with the timedelta of the ACT_TIME field.
4) I renamed the following fields into their respective table heading names conforming to the schemas of both tables:
    a) OPD_DATE to tstamp
    b) GPS_LATITUDE to latitude
    c) GPS_LONGITUDE to longitude
    d) DIRECTION to direction
    e) VELOCITY to speed
    f) EVENT_NO_TRIP to trip_id
    g) VEHICLE_ID to vehicle_id
5) I dropped the following columns to avoid loading extraneous data:
    a) EVENT_NO_STOP
    b) METERS

c) ACT_TIME
d) GPS_SATELLITES
e) GPS_HDOP
f) RADIO_QUALITY
g) SCHEDULE_DEVIATION

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.