# DataEng S24: Data Transformation In-Class Assignment

**Submit**: Make a copy of this document and use it to record your results. Store a PDF copy of the document in your git repository along with any needed code. Submit the in-class activity submission form by Friday at 10:00 pm.

## A. [MUST] Initial Discussion Questions

Discuss the following questions among your working group members at the beginning of the week and place your own response into this space. If desired, also include responses from your group members.

1. In the lecture we mentioned the benefits of Data Transformation, but can you think of any problems that might arise with Data Transformation?

Data transformation in data engineering is a critical process that involves converting data from one format, structure, or type to another. While it is a key step in data integration, cleansing, and analysis, several challenges can emerge. Common problems include data loss, data corruption, and schema changes, which can occur when transformations are not handled correctly. For example, incorrect filtering or aggregation could lead to data loss, while errors in transformation logic or character encoding can cause data corruption. Schema changes, like renaming or reordering columns, might disrupt downstream processes that depend on a specific data structure.

Additional issues arise from data inconsistencies, redundancy, and incorrect mapping. These can occur when integrating data from multiple sources with varying formats or when mapping fields inaccurately. Performance bottlenecks and scalability challenges can result from complex transformations or large-scale data operations, while data security and privacy risks can increase if sensitive data isn't handled properly. Effective data transformation requires robust error handling, monitoring, and maintaining data lineage to ensure accuracy and traceability. By adopting best practices and modern frameworks, data engineers can minimize these risks and ensure smooth data processing.

2. Should data transformation occur before data validation in your data pipeline or after?

The sequence in which data transformation and validation occur in a pipeline can have a big impact on the accuracy and dependability of the data. Validating your data before transforming it

enables you to find and fix mistakes or inconsistencies as soon as possible. By ensuring that only clean data enters the transformation process, this method lowers the possibility of corrupted or lost data. When working with unprocessed data from outside sources, where the quality of the data may vary, it is especially helpful. You may keep a cleaner data flow and prevent issues later in the pipeline by verifying data up front.

However, when working with different data sources that need to be reformatted or standardized, it can be advantageous to alter the data before validating it. By aligning data into a single structure or format, this method facilitates validation against standardized standards. More thorough validation procedures can be used after transformation to guarantee data integrity, quality, and adherence to business regulations. This technique is especially helpful in pipelines where the incoming data needs to be significantly transformed before validation due to variations in its structure. In the end, merging the two strategies—first validation to identify fundamental mistakes, then transformation and finally comprehensive validation—provides a strong foundation for preserving data quality all the way through the pipeline.

# B. [MUST] Small Sample of TriMet data

Here is sample data for one trip of one TriMet bus on one day (February 15, 2023): bc_trip259172515_230215.csv It's in .csv format not json format, but otherwise, the data is a typical subset of the data that you are using for your class project.

We recommend that you use google Colab or a Jupyter notebook for this assignment, though any python environment should suffice.

Use the pandas.read_csv() method to read the data into a DataFrame.

# C. [MUST] Filtering

Some of the columns in our TriMet data are not generally useful for our class project. For example, our contact at TriMet told us that the EVENT_NO_STOP column is not used and can be safely eliminated for any type of analysis of the data.

Use pandas.DataFrame.drop() to filter the EVENT_NO_STOP column.

For this in-class assignment we won't need the GPS_SATELLITES or GPS_HDOP columns, so drop them as well.

Next, start over and this time try filtering these same columns using the usecols parameter of the read_csv() method.

Why might we want to filter columns this way instead of using drop()?

# D. [MUST] Decoding

Notice that the timestamp for each bread crumb record is encoded in an odd way that might make analysis difficult. The breadcrumb timestamps are represented by two columns, OPD_DATE and ACT_TIME. OPD_DATE merely represents the date on which the bus ran, and it should be constant, unchanging for all breadcrumb records for a single day. The ACT_TIME field indicates an offset, specifically the number of seconds elapsed since midnight on that day.

We're not sure why TriMet represents the breadcrumb timestamps this way. We do know that this encoding of the timestamps makes automated analysis difficult. So your job is to decode TriMet's representation and create a new "TIMESTAMP" column containing a pandas.Timestamp value for each breadcrumb.

Suggestions:

- Use DataFrame.apply() to apply a function to all rows of your DataFrame
- The applied function should input the two to-be-decoded columns, then it should:
    - create a datetime value from the OPD_DATE input using datetime.strptime()
    - create a timedelta value from the ACT_TIME
    - add the timedelta value to the datetime value to produce the resulting timestamp

# E. [MUST] More Filtering

Now that you have decoded the timestamp you no longer need the OPD_DATE and ACT_TIME columns. Delete them from the DataFrame.

# F. [MUST] Enhance

Create a new column, called SPEED, that is a calculation of meters traveled per second. Calculate SPEED for each breadcrumb using the breadcrumb's METERS and TIMESTAMP values along with the METERS and TIMESTAMP values for the immediately preceding breadcrumb record.

Utilize the pandas.DataFrame.diff() method for this calculation. diff() allows you to calculate the difference between a cell value and the preceding row's value for that same column. Use diff() to create a new dMETERS column and then again to create a new dTIMESTAMP column. Then use apply() (with a lambda function) to calculate SPEED = dMETERS / dTIMESTAMP. Finally, drop the unneeded dMETERS And dTIMESTAMP columns.

**Question**: What is the minimum, maximum and average speed for this bus on this trip? (Suggestion: use the Dataframe.describe() method to find these statistics)

# G. [SHOULD] Larger Data Set

Here is breadcrumb data for the same bus TriMet for the entire day (February 15, 2023):
bc_veh4223_230215.csv

Do the same transformations (parts C through F) for this larger data set. Be careful, you might need to treat each trip separately. For example, you might need to find all of the unique values for the EVENT_NO_TRIP column and then do the transformations separately on each trip.

**Questions**:
What was the maximum speed for vehicle #4223 on February 15, 2023?
Maximum Speed of the vehicle 4223 on Feb 15, 2023 was 17.4 meters per second

Where and when did this maximum speed occur?

```
max_speed_4223 = df1['SPEED'].max()
max_speed_4223 = df1.loc[df1['SPEED'] == max_speed_4223]
print(max_speed_4223)
```

```
                  EVENT_NO_TRIP  VEHICLE_ID  GPS_LONGITUDE  GPS_LATITUDE  \
EVENT_NO_TRIP
259172515     2356      259172515        4223    -122.660822     45.505452
259173647     6573      259173647        4223    -122.535638     45.489627

                  SPEED
EVENT_NO_TRIP
259172515     2356   17.4
259173647     6573   17.4
```

What was the median speed for this vehicle on this day?
Median Speed of the vehicle 4223 on Feb 15, 2023 was 17.4 meters per second

# H. [ASPIRE] Full Data Set

Here is breadcrumb data for all TriMet vehicles for the entire day (February 15, 2023):
bc_230215.csv

Do the same transformations (parts C through F) for the entire data set. Again, beware that simple transformations developed in parts C through F probably will need to be modified for the full data set which contains interleaved breadcrumbs from many vehicles.

**Questions**:
What was the maximum speed for any vehicle on February 15, 2023?
200.3333

Where and when did this maximum speed occur?

```
max_speed = df2['SPEED'].max()
max_speed_details = df2.loc[df2['SPEED'] == max_speed]
print(max_speed_details)
```

```
                                  EVENT_NO_TRIP  VEHICLE_ID  GPS_LONGITUDE  \
VEHICLE_ID EVENT_NO_TRIP
3244       258750515     1734683      258750515         3244    -122.879553

                                  GPS_LATITUDE           TIMESTAMP       SPEED
VEHICLE_ID EVENT_NO_TRIP
3244       258750515     1734683     45.509918 2023-02-15 18:05:38  200.333333
```

Which vehicle had the fastest mean speed for any single trip on this day? Which vehicle and which trip achieved this fastest average speed?

```
The vehicle and trip with the fastest mean speed were Vehicle ID: 3419, Trip ID: 258614729
This mean speed was 21.889053500310045 meters per second.
```