# pedestrian\_stats\_assignment

**Approach:**

This assignment of data modelling and transformation to derive aggregated stats has been performed using pyspark and SQL.

The data has been downloaded manually from `https://data.melbourne.vic.gov.au/Transport/Pedestrian-Counting-System-Monthly-counts-per-hour/b2ak-trbp` and `https://data.melbourne.vic.gov.au/Transport/Pedestrian-Counting-System-Sensor-Locations/h57g-5234` .The files are in csv format and placed in local to build dataframes from the csv files.

There are two datasets used:

Pedestrian Counting System - Monthly (counts per hour)

Pedestrian Counting System - Sensor Locations

**Stages of data processing:**

**Raw to staging:**

1.The datasets in their raw formats are placed in a raw zone in their native formats and loaded into spark. We have predefined the schema with datatypes to load data from csv.

2.We can perform data validation on top of the data.We have just added null checks and empty dataset check in spark.

3.Filter only the necessary columns for our analysis and stats.

4.Added additional columns like load\_Date and load\_source for auditing later for both tables. A row hash is also generated and stored in a column only for the monthly counts data.

5.The monthly count data table is saved in the staging zone in parquet format.As the monthly count table is loaded every month ,to cater for reprocessing and new additional data monthly we overwrite the spark partitions of the staging data for that new incremental month's data. As it’s not a deltalake implementation, we haven't provided implementation to upsert data for incremental loads every month.

6.The sensor location data is also saved in parquet in stage layer. Sensor dimension table doesn't change frequently and partitioning is not required as data set will be small. We overwrite the data if new changes come in.

7.Parquet is chosen as it has advantages as columnar format for better query performance and works best with Spark.

**Staging to aggregated:**

1.The datasets from the stage layer saved in parquet are loaded.

2.The monthly count dataset is joined with the sensor location data on sensor id columns to enrich the dataset and form the merged dataset.

3.Derive aggregated dimensions of location by day.

4.Derive aggregated dimensions of location and month

5.Derive aggregated dimensions of location and Year

6.The merged dataset along with the above 3 aggregated datasets are persisted in the aggregated zone for further queries.

7.These datasets can be saved as snowflake tables to build the aggregated schema. Using snowflake JDBC connector spark can easily persist these datasets in snowflake.

But for the sake of this assignment, the datasets have been saved as tables in hive metastore for local queries.

**Aggregated data analysis to derive stats:**

1.Top 10 (most pedestrians) locations by day

Table

Description automatically generated

2.Top 10 (most pedestrians) locations by Month

Table

Description automatically generated

3.Which location has shown most decline due to lockdowns in last 2 years?

Graphical user interface, text, application

Description automatically generated

4.Which location has most growth in last year?

Text

Description automatically generated with medium confidence

**Proposed architecture for a production use case:**

**Diagram

Description automatically generated**

This proposed architecture is more of a data lake which depicts that the files can be ingested into the raw layer using sftp drop or api consumption. The data from the raw layer gets into staging after validations and pre-processing with ETL using pyspark. Once the data is in staging it can be further transformed to build aggregated datasets for the aggregated zone. A datawarehouse like snowflake can be built on top of the aggregated zone to cater to reporting and dashboards,if any. Better incremental updates and ACID transactions can handled with a lakehouse like architecture with delta as the storage solution.

**Suggestions and improvements:**

1. More analytical stats can be derived from the given datasets like:
   1. What time of the day shows the highest footfall of pedestrians.
   2. Top 10 (most pedestrians) locations by time of the day
   3. Quarterly stats of pedestrian data.
   4. Per Yearly average count of pedestrians across all locations.
2. This data model and transformation could have been used in a delta lake with spark or a data warehouse. For incremental data loads in staging layer from raw and loading from staging to aggregated layer for the hourly dataset could have been done using upserts with MERGE command. As this is a monthly load, it only updates old data in target from source or inserts new month’s data in target. This would be a much better solution to implement incremental data load patterns.
3. The load frequency of the monthly counts data is every month.
4. The final aggregated derived stats can be stored as materialized views which gets refreshed when new data is loaded every month. As this is a batch load and no real-time streaming dashboard, materialized views will be a good option.
5. Test cases for the pyspark and spark should be added in production ready code with integration and unit tests triggers during CI/CD.
6. The workflows and jobs should be well orchestrated using airflow or any scheduling tool. We can make this whole process event driven using lambda in AWS.

Data Model in raw zone:

