# Section 1 Overview

## 1.1 Purpose

Combine Traffic, Weather, and CHP Incident data into a single data warehouse, enabling systems to determine traffic rates and CHP incidents with respect to weather; and potentially other determinations (See Section 2.5).

# Section 2 Integration Strategy

## 2.1 Data Sources

Data is extracted from the following data sources:

- Caltrans Performance Measurement System (PeMS) - Traffic Data
- National Oceanic and Atmospheric Administration - Hourly Precipitation
- California Highway Patrol - Incident Reports

### 2.1.1 Data Quality

#### 2.1.1.1 Station 5min Readings

The station data conformed very tightly to the specifications provided by CalTrans. The data was very high quality and complete. We were able to collect data beginning January 1st, 2008 to the present for every online station for the entirety of San Diego County. If a station was online and reporting for the day it appeared that there were no gaps in data reporting for every 5 minute record.

Something observed was that even if no car traffic had occurred in the previous 5 minutes, a record was still reported for that station. This led our team  to the determination between if there was NO data versus missing data a bit difficult.

#### 2.1.1.2 Station Metadata

The station metadata was some of the most challenging data. This appeared to be due to a lot of constant updates to individual station fields, as well as stations not all having required fields; specifically latitude and longitude which are important for the geolocation queries.. Additionally, as a district's station list was reported every time that a change was made and this was not in a very consistent manner. It appears all stations that were reported for the 5 minute readings are present in the stations metadata file.

#### 2.1.1.3 CHP Incident Reports

Similar to the Station readings, the CHP Incident Reports conformed to the data specification provided by CalTrans. However, unlike the reliability and availability of the station readings, there were a number of large gaps in the availability of the data. The data appeared to be largely hand-generated, and this resulted in a large amount of poor-quality data. The amount of time spent cleaning and determining how best to handle various situations is discussed further in Section 3.1.

#### 2.1.1.4 Weather Stations

The weather station data was very clean and well-documented.  The documentation for the weather stations can be found at http://www.ncdc.noaa.gov/homr/reports under the section “MSHR, Standard Version”.  Using the documentation, the weather station was parsed without needing any cleaning whatsoever.

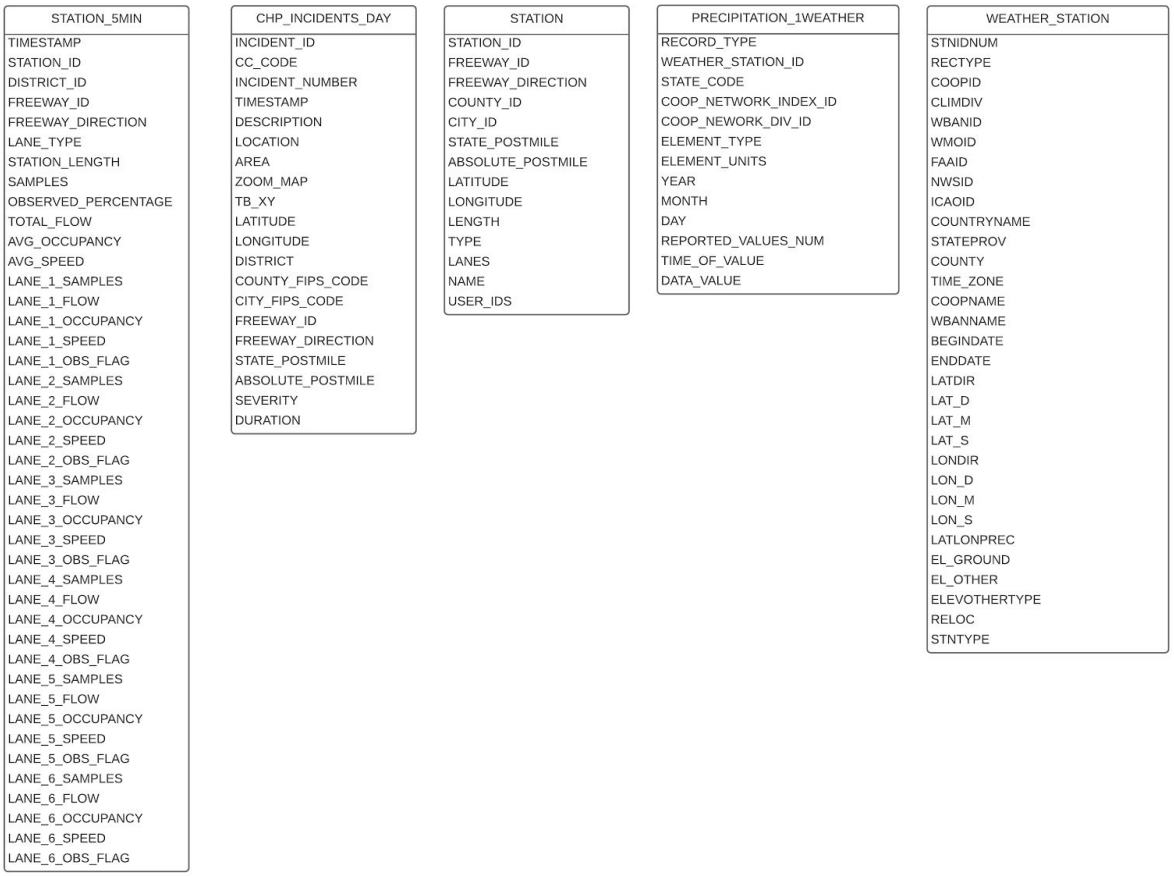
#### 2.1.1.5 Hourly Precipitation Observations

The vast majority of the hourly observation precipitation data was also very clean.  All observations for 2013 were clean, but 201 and 2012 precipitation data contained a number of invalid reports containing time information that did not conform to the documentation.  According to the documentation

(ftp.ncdc.noaa.gov/pub/data/hourly_precip-3240/dsi3240.pdf), the time format should be 0100, 0400, 1500, etc. The “bad” observations reported times with ‘g’ prepended to the time field. The documentation never indicates this is a valid time format, so the decision was made to drop these observations because there were only a handful of them.

When joining the precipitation observations to their associated weather station metadata, every observation had a valid weather station identifier. This was a quick and dirty sanity check that the precipitation data was consistent.

## 2.2 Source Schema




STATION_5MIN

This dataset contains the standard PeMS rollup of raw detector data for 5 minute intervals.

CHP_INCIDENTS_DAY

This dataset contains CHP Incidents from all Caltrans Districts where each downloadable file contains all incidents that occurred in one day.

STATION

This dataset contains the descriptive information for a recording station.

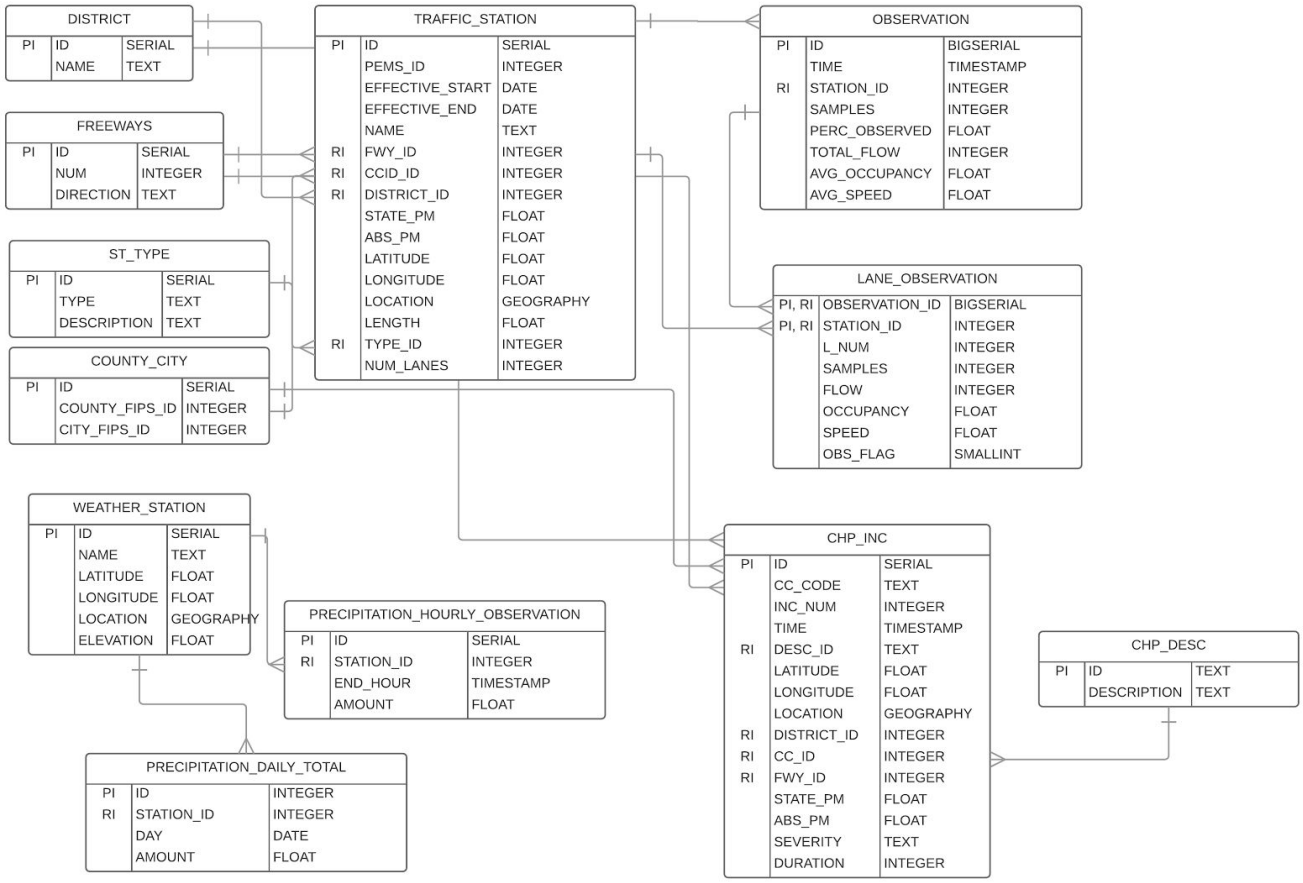
PRECIPITATION_1WEATHER

This table contains hourly precipitation data taken by observers at principal (primary) stations, secondary stations, and cooperative observer stations operated by the National Weather Service (NWS) and the Federal Aviation Agency (FAA).

WEATHER_STATION

This table contains more than 50,000 weather stations documented in the NCDC Station History Database. They are located on all continents but most are US sites. It contains information about the present and historical names, identifiers, and locations.

## 2.3 Target Schema




ST_TYPE

This table contains enumeration of all possible traffic station types. The enumerated list of station types were not found in the documentation, but rather were found in the 5 minute data files. All possible enumerations were found and resulted in the values found in this table.

FREEWAYS

This table contains freeway metadata.

COUNTY_CITY

This table contains county and city metadata.

TRAFFIC_STATION

This table contains traffic station metadata.

OBSERVATION
This table contains aggregate traffic throughput, average occupancy, average speed across all lanes for a particular station at 5m granularity.

LANE_OBSERVATION
This table contains traffic throughput, occupancy, speed for a particular lane at 5m granularity.

CHP_DESC
This table contains CHP incident codes and their corresponding description.

CHP_INC
This table contains CHP incident data.

WEATHER_STATION
This table contains weather station metadata. The primary key is the Cooperative Network Identifier Number assigned by NOAA.

PRECIPITATION_HOURLY_OBSERVATION
This table contains hourly precipitation data. Each row contains the amount of rain recorded for the previous hour at a particular weather station. For example, if the timestamp is '2010-01-01 02:00:00', the data in its row corresponds to the reading of precipitation for '2010-01-01 01:00:00' until '2010-01-01 01:59:59'. Rows with 'amount' values of zero denotes a trace amount collected that hour. During the ELT process, any hour with invalid data (either determined by our validator or indicated by the data itself) is ignored and not included in this table. The rain amount is reported in hundredths of an inch.

PRECIPITATION_DAILY_TOTAL
This table contains daily precipitation totals at a particular weather station. Rows with 'amount' values of zero denotes a trace amount collected that day. Daily totals with an error in any hourly reading is omitted from this table. The rain amount is reported in hundredths of an inch.

## 2.4 Schema Content and Semantics

### 2.4.1 Source Schema and Semantics

The source schema is a set of files collected from various sources around the web. The CHP Incidents and Station 5 minute readings were obtained from the PEMs database. There were certain constraints present in the the specifications provided by PEMs. Things such as station freeway direction (N, S, E, W), Lane Types (ML, FF, HV, OR..), and ranges for percentages (0-1/100). One of the reasons for selecting to work with Traffic for the San Diego area is that multiple team members have lived and grown up here their whole lives. This gives them first hand experience with the expectations of how freeways are connected within the area. The precipitation data was a collection of flat files separated by month or state depending on the year. Each file contained one record per row. Each record

contained the hourly, daily, and station identifiers.  From each row, between 2 and 25 records were written to the database.

### 2.4.2 Target Schema and Semantics

One of the goals of the target schema was to try and adhere to Third Normal Form (3NF). This required moving any information that was duplicated across rows into their own domain tables - information such as Station Types, City/County Code Combinations, the Freeway numbers and directions. Additionally because most of our queries were centered around using a stations  summary recording instead of looking at individual lane readings we decided to move the Lane observations to their own table. This also reduced the size of the observation table for querying.

Also by looking at the CHP Incident data it seemed to contain a lot of the same information for describing the location of an incident - freeway, direction, city, county - by putting this information into a domain knowledge table more freeways, directions, cities and counties could be added without having to worry about updating the already present set of data.

Another major change from the source schema to the target was to use our own internal unique identifiers for Observations and Stations. This decision was made to handle the temporal changes and updates to the station metadata.

## 2.5 Queries

NOTE: All queries made within the month of January 2010, in the San Diego geographic area

1. What traffic station has the largest difference in average speed over the first two weeks of the month?
2. How significant is the difference in traffic throughput on a rainy Monday vs a non-rainy Monday?
3. Does trace amount of precipitation affect the number of CHP traffic incidents on a given day? (Trace precipitation is defined as a weather station registering precipitation but less than the unit granularity of the sensor)
4. Identify the top 5 freeways with respect to traffic speed.
5. Is the traffic throughput of one freeway indicative of others?

# Section 3 Implementation

The entire task was completed using Clover ETL. That is, all sources were extracted, translated, validated, and persisted using a single Clover ETL graph. The graph persisted data directly to a Postgres/PostGIS instance.

## 3.1 Phase 1 - Traffic

### 3.1.1 Station 5min Readings

The 5 minute sensor data was the most challenging from the onset. One of the early challenges was that each row in the daily file was variable length and a single Metadata could not be used to describe it. The number of fields in the row was dependent upon the number of lanes that a particular station was reading. The additional downside was that as part of the station information included in the reading

didn't include the number of lanes. However the information for each lane was a constant 5 fields so it could be inferred how many lanes there were at a particular station. The summary information was an average for all lanes at that particular station over the last 5 minutes.

Since we had downloaded the entire station metadata information in a separate file it was determined that the station information recorded for each reading in the data was a duplication and could be thrown away. All that was needed was the Station Identifier (Station_ID) to be able to tie the two files together.

Since most of our queries and analytics were aimed more at the summary information for a station on a highway and less about individual lanes we decided to create a foreign key to split out observations between summary and individual lanes. This allowed us to not worry about handling various sets of NULL fields in a single observation depending on how many lanes there were and that the size of data being queried would be much smaller.

By doing this split it required that for each observation that was recorded a new Unique ID needed to be generated during the ETL phase. To ensure that there wasn't duplication of readings a Unique Index constraint was added to the Lane Observation table as a combination of Lane Number, Observation ID and Station ID.

A custom Java Reader was written in CloverETL to handle the reading, ID generation and splitting of the station readings into the three disparate sets - Station MetaData, Station Summary Readings, Lane Readings. As noted the station metadata from this input source was not stored in the warehouse and was discarded. However it is accessible via the CloverETL instead of being totally thrown away should it need to be used later.

As for the Observations - Summary and Lane - additional filtering and validation was done on the data to ensure accuracy. First any Lane readings that contained all NULL or 0 inputs were discarded as there was no value to be gained from them. After that the data was validated to ensure that it met with the constraints we had laid out - NOT NULL Observation ID, Number of samples was greater than 0, and that the Occupancy and Interval fields were accurate. Rather than try to fix or determine what values should have been rows that didn't match were discarded. Running the data through the validator it appeared that all data was valid and matched the constraints we set as well as those provided by PEMS.

The most difficult of the (Summary) Observation and similarly the Lane Observation tables was handling the temporal aspect of the Station ID. This is discussed further in Section 3.1.2.

After all ETL processing the data was written to the PostgreSQL database. An interesting case occasionally would arise when trying to write data to the referencing table prior to the referenced data row being available. Thankfully, CloverETL has a nice mechanism available to control the phase of execution for any component. by tuning the components in CloverETL to ensure that the data for Lane Observations was always done last as it referenced the most other tables and the Summary Observation rows were done just prior. Additionally by controlling the phasing of execution it also allowed for CloverETL to finish processing certain steps and clean up the memory from those

components and at the same time not begin processing steps with higher memory usage until others had finished. This resulted in more optimal times for doing the ETL processing.

### 3.1.2 Station Metadata

The station metadata required some of the most extensive work and tricks within CloverETL to get it to conform to the target schema that we had outlined. Thankfully the data extracted from the PEMs database was a nice standard format and could be easily read without having to write custom code.

To avoid having to find a master list of all the various Freeways and directions within San Diego it was decided to use the Station Metadata file to generate the entries and ensure that they conformed to some basic guidelines for how the highway system in the US works. A separate entry was created for a freeway number and direction as the sensors are recorded for both sides of the highway. Then the freeway direction and number were validated to ensure that even numbered freeways only had directions that went East/West and conversely that Odd numbered freeways went North/South. From there the data was assigned a Unique identifier and then stored into the PostgreSQL database. It was decided that rather than try and use triggers within the Database to do reverse joins that the work be done in the ETL processing. Thus, the Station rows were updated to remove the Freeway number and direction and replaced with the foreign key.

In much a similar manner as the Freeways the County/City codes were removed and generated using the station metadata. This was done as it was determined that these codes were also present in the CHP data and could be used as a join lookup table for finding incidents near particular stations.
The most difficult part of the station data was due to the fact that whenever any information for a particular station was updated, removed or added an entirely new list of stations was put out on that date. This brought about a lot of discussion in our group about how best to handle this. Some of the ideas discussed were:

1. Using only the most recent version of the station list.
2. Using the first version of the station list
3. Melding information and keeping a single record for each station with only newest information
4. Doing temporal analysis and keeping a record for each station at a given time.

Ideas One and Two about using a single list were discarded as it was discovered that when stations were brought online that the observation readings were being discarded due to not having a station entry. Idea three was discarded as we felt that even though for this particular assignment we were focusing only on a single month that in the course of the capstone project we would be working across multiple years. This could lead us to doing analytics during a certain time period and thus having the ability to look at data for a particular time period would be valuable.

After finally settling and arriving at the conclusion that we wanted to do some sort of temporal based strategy there was much discussion about how best to implement this between the key tables - Traffic_Stations, Observations and Lane Observations. An early idea was to create a new table that had a history log of the old station records and that whenever a modification was made to a station that the old record would be moved over. This would allow us keep using the same PEMs Unique Station Identifier and be able to retrieve the historical information. However when we began to look at the difficulties added to the queries with the joins and temporal modifiers that idea was discarded.

Ultimately what we decided upon was that during our ETL processing we would generate a new internal Unique Station Identifier for every unique station. Thus moving the PEMs ID to a new field in the station record and then replacing the PEMs ID in the other two tables with our generated one. However to accomplish this a lot of tricks were required.

One of the first challenges was determining the date that a station was added. Thankfully the metadata files all followed a standardized naming convention including the date. CloverETL had the ability to include the source file name in each record. This allowed me to parse out the effective start date for each station. From there we needed to remove any duplicate stations. We decided that if any field had changed - excluding the User information which was discarded anyway - that we would keep those rows.

From there we had to somehow figure out which stations had been updated and correctly set their effective end dates. After doing a bunch of research online about this what was decided was to self join the data on the PEMS ID and as part of the transformation if the base row (input 0) had a lower effective start date than the target join row (input 1) then the end date on the base row was updated to reflect the start date for the target row. By joining the rows against themselves this allowed us to get all possible combinations when a row had been updated. Thus if during the month a station had been updated three times the resulting joined gave us nine rows. See Table 1 and Table 2 for examples.

**Table 1: Before Join**

| ID | PEMS ID | Effective Start | Effective End |
|---|---|---|---|
| 1 | 1235 | 01-20-2010 | *null* |
| 2 | 1235 | 01-10-2010 | *null* |
| 3 | 1235 | 01-05-2010 | *null* |

**Table 2: After Join**

| ID | PEMS ID | Effective Start | Effective End |
|---|---|---|---|
| 1 | 1235 | 01-20-2010 | *null* |
| 1 | 1235 | 01-20-2010 | *null* |
| 1 | 1235 | 01-20-2010 | *null* |
| 2 | 1235 | 01-10-2010 | 01-20-2010 |
| 2 | 1235 | 01-10-2010 | *null* |
| 2 | 1235 | 01-10-2010 | *null* |
| 3 | 1235 | 01-05-2010 | 01-20-2010 |
| 3 | 1235 | 01-05-2010 | 01-10-2010 |
| 3 | 1235 | 01-05-2010 | *null* |

Note: Rows highlighted are the target rows

Figuring out how to do the join was just the first of three steps to correctly handle the temporal operation. From there the data had to be deduplicated to remove the extra station rows generated from the join. The problem was that the Clover standard deduplication component only allowed to keep First, Last or Unique. Keeping first would record that for ID 3 that it was replaced by ID 1 which is in correct. Keeping last would discard all the effective end date information just calculated. Choosing unique would result in having a single Row for ID 1 but for IDs 2 and 3 there would be two and three rows respectively. I attempted to make use of components like group by and aggregation to solve the problem but ultimately ended up write a custom java deduplication component that would return back the lowest effective end date excluding null. Finally the Station Metadata had been transformed and could be stored into PostgreSQL.

However this only solved two-thirds of the steps to processing the complete set of data as the Station IDs within the Observation records were the old PEMs IDs and not the new generated ones. Thankfully there was a simple suggestion posted online for how best to do this in CloverETL. The trick was to join the Station Metadata records with the Observation records on the PEMs ID and as part of the resulting metadata add the generated ID, Effective Start and Effective End timestamps. From there a simple filter could be applied to check that the timestamp for the observation was within the effective date for that station. Lastly a simple reformat was done to replace the PEMs ID with the Generated ID and remove the effective dates from the record. Since the Lane Observation records were all tied to a unique Observation record a simple join on the Observation ID to replace the PEMs ID in those records with the Generated ID.

### 3.2.3 CHP Incidents

Compared to the amount of processing required for the Traffic Station data the CHP incident data was fairly trivial. The records were all in a standard tab delimited format that was easy for Clover to parse and process. The first thing that was done was to isolate the CHP incidents for just the San Diego Area - District 11. From there the Freeway and City/County codes were replaced with the foreign indexes created using the Station Metadata. We validated that both on a longitude and latitude were present for where the incident occurred as we needed that information for performing the geospatial queries. Anything missing that was discarded. It was found that the number of rows without latitude and longitude was extremely low and thus the amount of time trying to determine it would not have been worth it.

After processing the CHP incident data we noticed that the ‘Description’ field seemed to be using a lot of the same repeated values which appeared to be linked to common CHP Traffic Codes. We decided to split out these into a separate table so that they could be looked up more easily and more analytical questions could be asked about incident type. We did some investigation on the web looking for a master list of codes, but it appeared that there isn’t a completely standardized list. Some of the terms involve a bit of slang and do vary. A future scope/project a more comprehensive list could be built from those found online as well as part of the complete state of California’s incident logs.

## 3.2 Phase 2 - Weather Stations

The Weather Station data was contained in a fixed-width file.  Each weather station was represented by a single line in the flat file. A Custom Java Transform was used in Clover ETL to gain greater access of

index control, as well as facilitate geographic coordinate conversion. The Weather Station subgraph was then incorporated into the larger Clover ETL graph.

### 3.3 Phase 3 - Weather Precipitation

The majority of the hourly precipitation data was reported in fixed-width format and lent itself well to built-in Clover components. Therefore, it was decided to use Clover to transform, validate, and load the precipitation data. The one minor issue with using Clover was that each record reported a variable length of hourly observations between 2 and 25. Clover does not natively support variable-length records, so a custom Java transform was required for the extraction step. Once the data was parsed, the rest of the transformations and loading was performed using built-in Clover components.

See GitHub repo for Clover ETL project

## Section 4 Results

See GitHub repo for results in IPython Notebook

## Section 5 Significance of Work

Consolidation of CHP, Traffic, and Weather data into a single data warehouse enables one to query across different data source with respect to another (ex: traffic with respect to weather) more easily. This opens the door to potentially many other applications that are outside the scope of this document.

## Section 6 Future Scope

- Expand current schema to account for aggregate tables that will store analytical results from Capstone Project work
- Create Web Service that serves as interface to database
- Expand Freeway Analysis to multiple years and freeways across California

## Section 7 Lessons Learned

- Handling Temporal effectivity dates on Station Metadata. Temporal information is a growing field as it becomes cheaper to store larger histories of information. This experience in dealing with where to put the complexity of the time based information was very interesting. Everyone in the group have very disparate ideas on how best to do this. Whether the work should be on the query side, the ETL side or in the Schema. Once we finally decided upon the best strategy still figuring out the implementation within an ETL job required learning a lot about how temporal processing is done in industry.
- Processing and analyzing geospatial data can be complicated. For the weather station data, multiple geo-locations were reported for the same station identifier. From the documentation, we learned “similar” weather stations are assigned the same identifier to simplify reporting. For this task, we decided to simply average latitude, longitude, and elevation. This may not be the optimal solution, but we did not have the time to properly analyze results using different station geo-location aggregation methods.

# Appendix

Github repo: https://github.com/conwaywong/dse203_final