Najel Alarcon
Kevin Gao
Albert Ong
Devesh Singh
Akash Thiagarajan

Local Traffic Analysis and Forecasting

Summary

We have created a data warehouse to monitor and analyze traffic data in the Bay Area. The warehouse integrates traffic, weather, and accident data. The Bay Area is heavily congested due to its dense population, which leads to high volumes of traffic and bottlenecking, leading the roads to be congested. Our data warehouse evaluates data from 2021-2023 traffic, weather, and accident datasets. By interpreting these three datasets together and forming a star schema, we were able to optimally query our data warehouse and generate insights into traffic patterns and the impact that weather has on the traffic and on possibly being a factor in accidents. By using data analytics and visualizations we can help the public transportation department make data-driven decisions on improving road safety and traffic flow.

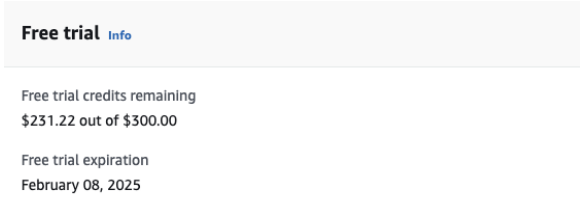
Significance

The project addresses the issues of traffic congestion in the Bay Area by building a data warehouse that enables data-driven decision-making to help transportation planners and policymakers make decisions on the city's infrastructure. This information can even help the public avoid rush hour to lessen traffic or avoid harsh weather, improving accident rates that are a result of weather conditions. Roads can receive improvements based on where accidents occur frequently by surveying blind spots or poorly designed infrastructure. For example, if an accident occurs in a common area of the road the government can take action to fix the problem. Identifying accident-prone areas and taking preventive measures for adverse weather conditions can greatly decrease traffic and lower accident rates. Not only will drivers likely be more cautious, accidents due to rain or other weather conditions can signal government entities to create roads that are less prone to rainwater collection, or to prep/salt the roads in cases of extreme frost. Aside from infrastructure, using our data-driven analysis, we can increase the frequency of public transportation around congested areas, which may promote the usage of public transportation rather than people using their vehicles, reducing the number of vehicles on the roads.

Lessons learned

One lesson learned was time management. We could have had a significantly better project execution if we could acquire data

faster that way we had more time to work on loading the data into the dimensions and executing that portion well. We learned how the data loading process works and it is not as easy as it seems, especially given that we were using 3 separate datasets and having to connect them within a star schema. We extensively used Redshift and AWS user IAM to collaborate which was quite the learning experience. It took some time to properly understand the correct IAM permissions to be given to each of the group members. During our initial setup, we assigned the policies but had to go back as some members of the group were not able to access the same things we needed as the admin of our AWS setup. After further exploration and reading we discovered which policies were needed, especially using "AmazonRedshiftQueryEditorV2FullAccess", which allowed us to perform our queries and set up the staging tables. Data extraction, figuring out the correct columns to use (which ones are important), and identifying useful information are some skills we picked up from looking at a lot of data. Another lesson we learned, is that while ChatGPT is there to help us, one of our group members was reading into the issues we faced and got a better understanding thus when we ran into a similar technical issue and were able to pinpoint the issue and come up with some temporary solutions that are not best practice for us to use while we make progress on the project. One of our biggest lessons learned was for us to use three different datasets one for the traffic, accident, and weather data; we had to devise a way to link the three data files properly without the need for excessive joins which had created our initial fact table with over 7 million rows. In our initial fact table creation, we didn't connect the 3 datasets correctly so we lacked a common key between all 3 datasets and had confusing queries that were not good at generating insights. Our solution ended up generating IDs given the date as it was the common factor between all three data sets, thus reducing our 7 million row fact table to around 500000 rows. However, while our current solution is not optimal, the best practice solution would be to hold the last auto-generated ID so if there is an additional batch load of data it will start from there rather than creating a random ID that way the integration will be continuous. One of the biggest learning points that we learned using AWS after our initial setup and some testing was how much charges we were incurring at first since we were not being too cautious with our best practices. After noticing that, we became very cautious as we did not want to exceed our free $300 credit limit.



Innovation

Our project innovates by using modern tools and technologies to manage complex traffic data in the San Francisco Bay Area. We are able to conclude next steps in road safety via actionable insights from the queries and information organized and outputted with our data warehouse and star schema. Amazon Redshift was used as

the data warehouse for scalable, high performance for integrating a large column of traffic, weather, and accident data. Additionally, PowerBI was used for visual insights and predictive analytics and PowerQuery was used for ETL and data preparation. While popular navigation tools available like Google Maps and Apple Maps have traffic prediction already for the individual user, our project and its conclusions can be used on a higher level, telling road officials which roads have more accidents and pinpointing causes of accidents (weather, traffic, roads). This project is one step toward accurate accident prediction based on temporal, geographical, and climate-based factors to prevent ANY accident that is not the complete fault of the driver.

Teamwork

To complete the project on time, our team needed to effectively collaborate and communicate during project implementation to meet the project requirements. Tasks, such as data collection, data cleaning, data warehouse management, and technical writing, were listed using Notion and then divided amongst the team members to eliminate overlap and ensure progress. Constant communication was needed, not just to solve technical problems but also how to manage our busy schedules and plan for team meetings, task coordination, and status updates. Our form of communication varied from Discord, WhatsApp, and Notion. The notion was primarily used to keep track of our progress and divert our workload accordingly and could be picked up by any team member. We used Discord and WhatsApp for our primary communication. WhatsApp was used for our “on-call” communication to text each other for those who were not at home and ask for a fast update as some of us were at work. Discord was our project-heavy communication platform as it allowed for a log of our messages allowing us to circle back to help refresh our memories, and being able to host calls and share screens which helped in our pair programming when issues were arising through our queries, scripts, and troubleshooting. Our teamwork throughout technical problems consisted of ensuring that each of us did not have questions before moving forward. We used analogies in our explanation to help each other better understand the current issue before explaining what needed to be done. Speaking in this way it helped us answer questions and catch each other up to speed. Our busy schedules did not prevent us from losing much time as all of us always had our phones with us. Thus, when one group member had a question, we got a response within the hour via WhatsApp. Additionally, the process of learning new technologies was delegated between different team members and collaboration was needed to integrate each facet and create a finalized product. Team member support was also needed to handle the stress of the workload to meet deadlines, especially when one member had a personal emergency.

Technical difficulty

We are all relatively new to data warehouses so we faced technical

difficulties at many points throughout the project. Most of them were from setup difficulties as it was our first time using AWS in a team-based setting to this scale of a project. We were able to circumvent some issues with the help of ChatGPT, speeding up the process. Some were just lots of trial and error. We faced a lot of difficulty trying to load the data into the table because we had to create a star schema and we were trying to figure out how to load the data while following industry standards which was pretty difficult. We were getting an excessive number of rows or not generating the fact table correctly as stated above in lessons learned. We did manage to execute the dimension tables correctly so when we queried we used as few joins as possible. As we were having technical issues figuring out some of the errors we were getting during our load process of the tables. One of our team members who was familiar with AWS was able to pinpoint the issue quickly and explained it to our team members. The issue that occurred was during our cleaning process we had made a slight error in setting the data type for our “Date” when we were combining our three columns given the year, month, and day into one column called date. The issue after inspection running our Python script was it was saying the datatype was an object instead of a date object. After fixing this issue, we were able to properly load the data after that. Another issue was when we were parsing through our data during the loading phase, we noticed some discrepancies in our year 2022 as it was not showing all the months, but rather only two months' worth of data. After further inspection, there was some sort of an error during our Python scripts that resulted in it missing where we reran the scripts and had another team member ensure our consistency by using Excel. After rerunning our scripts, we were able to fix the error for the missing dates from our original data upload to AWS Redshift.

Credit Statement

Najel – Conceptualization, Data Curation, Software, Methodology, Investigation

Kevin – Supervision, Validation, Investigation, Visualization, Project Administration, Writing-review and editing

Albert –Writing original draft, Supervision, Methodology, Data Curation

Devesh – Data curation, Methodology, resources

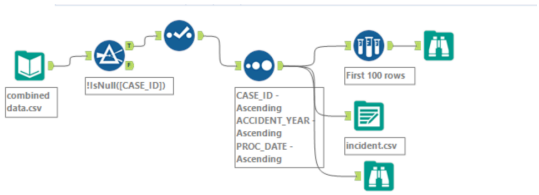
Akash – Data Curation, Investigation, Project Administration, Supervision, Methodology, Software, Writing

Appendix

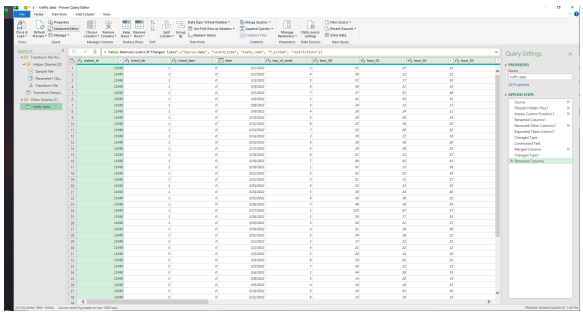
Criteria	Comments
Presentation Skills Includes time management	See slides
Code Walkthrough	See slides
Discussion/Q&A	See presentation
Demo	See slides
Version Control	https://github.com/Najel-A/Traffic-Data
Significance to the real world	See significance section
Lessons learned	See the lessons learned section
Innovation	See innovation section
Teamwork	See teamwork section
Technical difficulty	See the technical difficulty section
Practiced pair programming	Yes, members used pair programming during project implementation (discord screenshare)
Practiced agile/scrum	Notion
Used Grammarly/other tools for language	Grammarly
Slides	See slides

Report	This is the report
Used unique tools	https://prezi.com/view/2Twm0kFTPtrVxrxCdony/ Prezi
Performed substantial analysis using database techniques	-Joins to uncover insights -Optimized indexing -Aggregation queries -queries for insight and chart generation for data analysis
Used a new database or data warehouse tool not covered in the HW or class	Amazon Redshift, Power Query, Python
User-appropriate data modeling techniques	-Star schema, -Surrogate keys for joins Check slides for more info
Used ETL tool	Power Query, Alteryx, Python
Demonstrated how Analytics support business decisions	Actionable insights into traffic congestion patterns
Used RDBMS	Created tables using SQL queries before creating in redshift($300 was constraint)
Used Data warehouse	Amazon Web Services Redshift

Includes DB connectivity/API calls	Redshift Query, DBeaver
Used NoSQL	Amazon Web Services S3
Elevator Pitch	https://youtu.be/y6byhXGaz34



Using Power Query to clean data

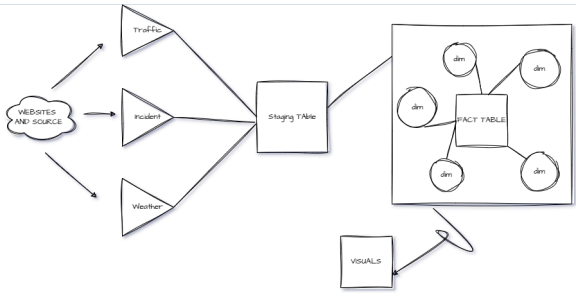


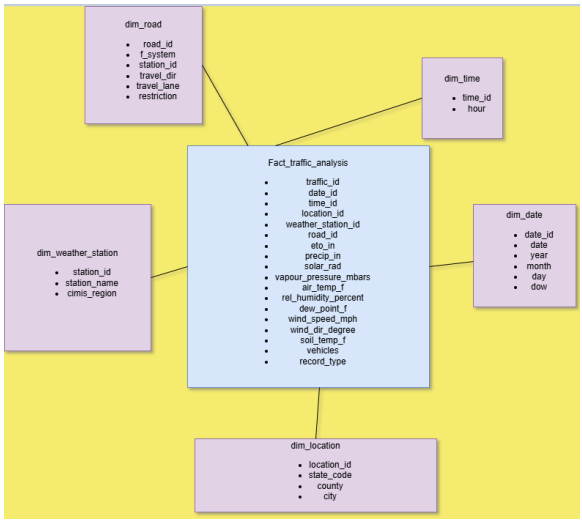
Using python and chat gpt to convert columns to rows (data cleaning)

```python
import pandas as pd

df = pd.read_csv('combined_cleaned_2021-2023_traffic_data.csv')


df_long = pd.melt(df,
                  id_vars=['date','station_id',
                           'travel_dir', 'travel_lane', 'day_of_week'],
                  value_vars=[f'hour_{i:02}' for i in range(24)],
                  var_name='hour',
                  value_name='value')


df_long['hour'] = df_long['hour'].str.extract('(\d+)').astype(int)


df_long.to_csv('updated_combined_cleaned_2021-2023_traffic_data.csv', index=False)
```

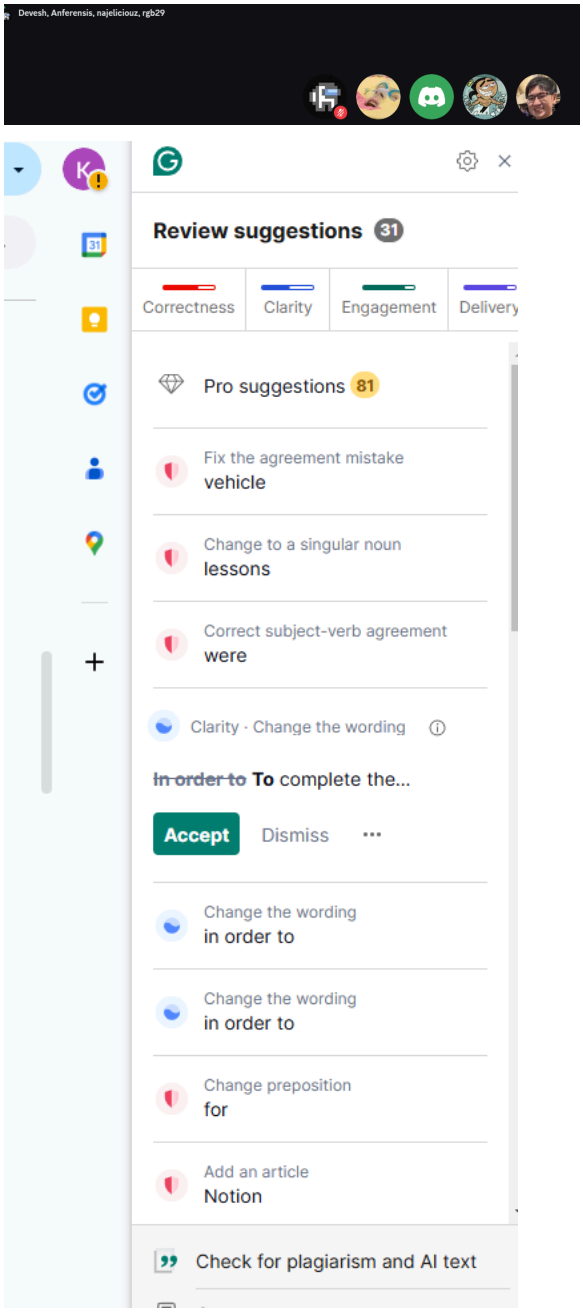
Used alteryx to clean Incident data

Model of warehouse





Discord for teamwork



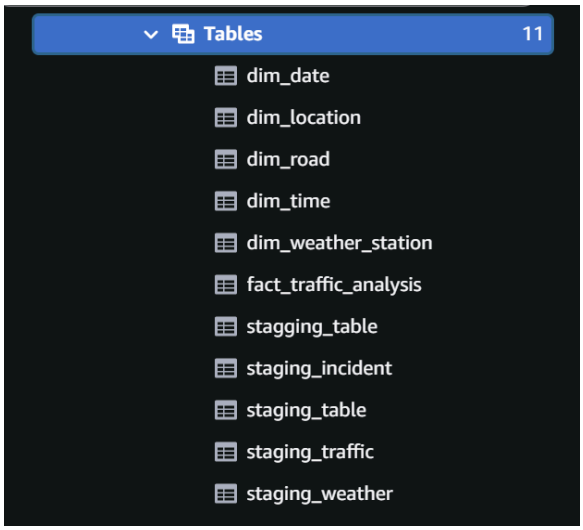
Grammarly for writing

Used to append every month's CSV file of a given year into one combined CSV file

```python
import pandas as pd
import glob

def append_csv_files(output_file_path, input_pattern):
    try:
        all_files = glob.glob(input_pattern)

        dataframes = []

        for file in all_files:
            df = pd.read_csv(file)
            dataframes.append(df)

        big_dataframe = pd.concat(dataframes, ignore_index=True)

        big_dataframe.to_csv(output_file_path, index=False)

        print(f"Successfully appended {len(all_files)} CSV files into {output_file_path}")

    except Exception as e:
        print(f"An error occurred: {e}")

output_file_path = 'updated_combined_2021_traffic_data.csv'
input_pattern = 'traffic_2021/*.csv'
append_csv_files(output_file_path, input_pattern)
```

Used to replace traffic data columns to create a Date column

```python
import pandas as pd
import numpy as np

csv_file_path = 'combined_2023_traffic_data.csv'

df = pd.read_csv(csv_file_path)

df.rename(columns={'year_record': 'year', 'month_record': 'month', 'day_record': 'day'}, inplace=True)
df['date'] = pd.to_datetime(df[['year', 'month', 'day']]).dt.strftime('%m/%d/%Y')

df.drop(columns=['year', 'month', 'day'], inplace=True)


float_columns = df.select_dtypes(include='float64').columns
if not float_columns.empty:
    # Replace non-finite values with 0 before conversion
    df[float_columns] = df[float_columns].replace([np.inf, -np.inf], np.nan).fillna(0)
    # Convert to int64
    df[float_columns] = df[float_columns].astype('int64')

column_to_move = 'date'
first_column = df.pop(column_to_move)
df.insert(0, column_to_move, first_column)

df.to_csv('updated_2023_file.csv', index=False)

print(df.head())
print(df.dtypes)
```



Used DBeaver to test connectivity for power bi and possible Informatica use