ETL PROJECT

Team 12: Victoria Traffic Analysis

An Nguyen , Seng Heang (Darren) Kim, Anjana Bandara

TABLE OF CONTENTS

OVERVIEW

The main focus of this project is to extract data from available sources and perform an ETL route on the extracted data. For this project we chose to analyse the behaviour of vehicle traffic in the state of Victoria. The three main focus areas for this project are,

- The traffic volume as a whole for Victoria
- The hourly traffic volume for Victoria
- Homogenous Traffic Flow

What we want to achieve through this analysis is, to outline the freeways and other linked roads to them with highest traffic volume. Furthermore, we will be outlining the highest traffic volume as per the time of the day. With the analysis of homogenous traffic flow, we will be creating a heat map with traffic flowing in and out of Victoria.

DATA EXTRACTION

We extracted the **Traffic Volume** data through three sources in two formats.
Traffic volume data was extracted through an API from Victoria Open Data Hub. The data was extracted in json format. We had to develop a query URL to extract the data from the API.

```
# API base url to get the data as a json format.
base_url = "https://opendata.arcgis.com/datasets/5512df2ff41e4941bacf868053dbfba9_0.geojson"
```

```
# Getting the data as a json response
responses = requests.get(base_url).json()
```

```
# Allocating features as the main response
features = responses['features']
```

Hourly traffic volume data was downloaded in a csv format through Kaggle.
Data source: https://www.kaggle.com/daralm/hourly-traffic-volume-in-victoria

Homogenous Traffic Flow was extracted from GeoJSON file, which is an open standard format designed for representing simple geographical features, along with their non-spatial attributes. It is based on the JSON format.

In order to read a GeoJSON file into GeoPandas DataFrame, we need to **import** geopandas as gdp

Then:

```
json_file = "Resources/homogenous_traffic_flow.geojson"
hmgns_flow = gpd.read_file(json_file)
hmgns_flow.head()
```

Data Source: https://discover.data.vic.gov.au/dataset/homogenous-traffic-flow

DATA TRANSFORMATION

Traffic Volume Data set

- Traffic volume data was extracted through the API and stored inside separate list. We extracted the data only needed for the analysis. For the data cleaning we used pandas library from python 3.

```
# Getting the data for required columns and appending into a list.
TIS_ID_LIST = []
HMGNS_FLOW_ID = []
HMGNS_LNK_ID = []
HMGNS_LNK_DESC = []
LGA_SHORT_NM = []
RGN_LONG_NM = []
ROAD_NBR = []
ALLVEHS_AADT = []
FLOW_LIST = []
geo_tag = []
for feature in features:
    tis = feature['properties']['TIS_ID']
    flow_id = feature['properties']['HMGNS_FLOW_ID']
    lnk_id = feature['properties']['HMGNS_LNK_ID']
    lnk_desc = feature['properties']['HMGNS_LNK_DESC']
    lga_name = feature['properties']['LGA_SHORT_NM']
    rgn_name = feature['properties']['RGN_LONG_NM']
    road_num = feature['properties']['ROAD_NBR']
    all_vechs = feature['properties']['ALLVEHS_AADT']
    flow = feature['properties']['FLOW']


    TIS_ID_LIST.append(tis)
    HMGNS_FLOW_ID.append(flow_id)
    HMGNS_LNK_ID.append(lnk_id)
    HMGNS_LNK_DESC.append(lnk_desc)
    LGA_SHORT_NM.append(lga_name)
    RGN_LONG_NM.append(rgn_name)
    ROAD_NBR.append(road_num)
    ALLVEHS_AADT.append(all_vechs)
    FLOW_LIST.append(flow)
```

- After storing the data in separate list, pandas were used to create a dataframe out of those lists.

```
# Creating a dataframe from the API data
traffic_df = pd.DataFrame({"TIS_ID":TIS_ID_LIST})
traffic_df["HMGNS_FLOW_ID"] = HMGNS_FLOW_ID
traffic_df["HMGNS_LNK_ID"] = HMGNS_LNK_ID
traffic_df["HMGNS_LNK_DESC"] = HMGNS_LNK_DESC
traffic_df["LGA_SHORT_NM"] = LGA_SHORT_NM
traffic_df["RGN_LONG_NM"] = RGN_LONG_NM
traffic_df["ROAD_NBR"] = ROAD_NBR
traffic_df["ALLVEHS_AADT"] = ALLVEHS_AADT
traffic_df["FLOW_LIST"] = FLOW_LIST
traffic_df
```

- We used pandas .isnull ().sum () function to find out all the missing values in the dataset. We found out there were eight rows with missing values. We dropped this rows using .dropna () function in pandas.
- Some of the columns were renamed after this step because we already had created the database tables and column names for each dataset.

```
new_df = new_df.rename(columns={"ALLVEHS_AADT":"DAILY_AVG_VEHICLES",
                                "FLOW_LIST":"FLOW"})
```

- All the duplicate values were dropped after the above step.
- All column headers were changed into lower case cause the database table headers are in lower case.

```
# Rename all columns to lowercase
new_df.columns = [x.lower() for x in new_df.columns]
new_df
```

- Finally, the cleaned dataframe was stored in a csv.

The Hourly Traffic Volume Data set

- The csv was read using the pandas read_csv function. The columns required for the analysis then extracted from the initial dataframe.
- The new dataframe headers are then renamed as per the table name headers in the database.

```
# Create a filtered dataframe from specific columns
hourly_cols = ["HMGNS_LNK_ID", "HMGNS_FLOW_ID", "PERIOD_TYPE","DOW",
               "0:00","1:00","2:00","3:00","4:00","5:00","6:00","7:00","8:00","9:00","10:00","11:00","12:00",
               "13:00","14:00","15:00","16:00","17:00","18:00","19:00","20:00","21:00","22:00","23:00"]
hourly_transformed= hourly_df[hourly_cols].copy()

# Rename the column headers
hourly_transformed = hourly_transformed.rename(columns={"HMGNS_LNK_ID":"hmgns_lnk_id", "HMGNS_FLOW_ID":"hmgns_flow_id",
                                                        "PERIOD_TYPE":"period_type","DOW":"dow",
                                                        "0:00":"t00","1:00":"t01","2:00":"t02","3:00":"t03","4:00":"t04","5:0
                                                        "6:00":"t06","7:00":"t07","8:00":"t08","9:00":"t09","10:00":"t10","11
                                                        "12:00":"t12","13:00":"t13","14:00":"t14","15:00":"t15","16:00":"t16"
                                                        "18:00":"t18","19:00":"t19","20:00":"t20","21:00":"t21","22:00":"t22"
hourly_transformed
```

- After this step, a function is created to transform day the day of the week column into the name of the day. And the function is applied for the whole day of the week column.

```
# Transform DOW value to proper day description

#Source: http://data.vicroads.vic.gov.au/Metadata/Typical%20Hourly%20Traffic%20Volumes.html
#Day of Week (DOW) where 1=Monday, 2=Tuesday,3=Wednesday,4=Thursday,5=Friday,6=Saturday and 7=Sunday


# Create function to derive day from DOW
def label_day (row):
   if row['dow'] == 1 :
      return 'Monday'
   if row['dow'] == 2 :
      return 'Tuesday'
   if row['dow'] == 3 :
      return 'Wednesday'
   if row['dow'] == 4 :
      return 'Thursday'
   if row['dow'] == 5 :
      return 'Friday'
   if row['dow'] == 6 :
      return 'Saturday'
   if row['dow'] == 7 :
      return 'Sunday'
   return "Undefined"

# Create new DAY column
hourly_transformed["day"] = hourly_transformed.apply (lambda row: label_day(row), axis=1)
```

```
# Review transformation
hourly_transformed.groupby("dow")["day"].apply(lambda x: list(np.unique(x)))
```

Activate Windows

- Then the Dataframe is checked for duplicate values and the remaining duplicate values are dropped after that.

```
# Are there any duplicate rows group by 'HMGNS_LNK_ID', 'HMGNS_FLOW_ID', 'PERIOD_TYPE', 'DOW'
data_groups = hourly_transformed.groupby(hourly_transformed.columns[:4].tolist())
size = data_groups.size().reset_index()
# Dataframe of duplicates
size[size[0] > 1]
# Number of duplicates
print(f"No. of duplicates: {len(size[size[0] > 1])}")
```

```
No. of duplicates: 0
```

```
# Are there any duplicate rows by all columns
data_groups = hourly_transformed.groupby(hourly_transformed.columns.tolist())
size = data_groups.size().reset_index()
# Dataframe of duplicates
size[size[0] > 1]
# Number of duplicates
print(f"No. of duplicates: {len(size[size[0] > 1])}")
```

```
No. of duplicates: 0
```

The Homogeneous Traffic Flow Data set

This type of data after extracted into the DataFrame is quite simpler than the rest of the data set that we have. We have selected relevant data for our analysis as shown below:

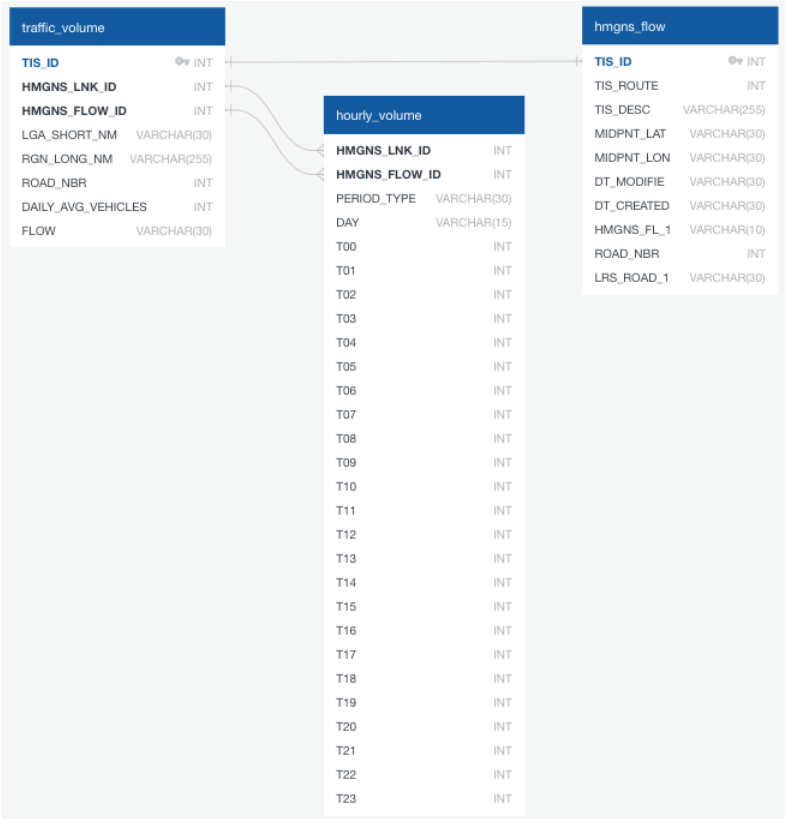
```
# Create a filtered dataframe from specific columns
hmgns_flow = hmgns_flow[['TIS_ID', 'TIS_ROUTE_', 'TIS_DESC', 'MIDPNT_LAT', 'MIDPNT_LON',
                         'DT_MODIFIE', 'DT_CREATED', 'HMGNS_FL_1', 'ROAD_NBR', 'LRS_ROAD_1']].copy()
hmgns_flow.head()
```

LOADING DATA

- PostgreSql was used to load the extracted and transformed data. We chose PostgreSql as our main database due to the compatibility it has with different data resources. And relational database seemed to be the best fit for our structured data.
- Data was loaded into the database using sqlalchemy create engine function.
- The dataset was merged together using the hmgns_lnk_id.

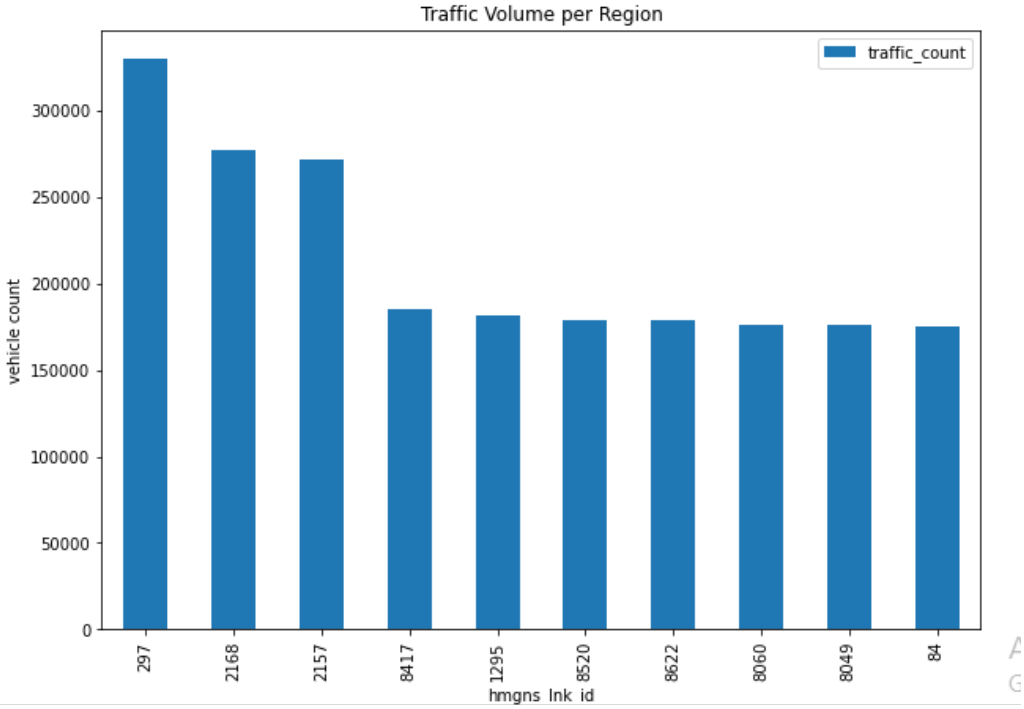
```
# Inner join to inscope_df and load data to database
volume_load=pd.merge(new_df, inscope_df, on='hmgns_lnk_id', how='inner')
volume_load.to_sql(name='traffic_volume', con=engine, if_exists='append', index=False)
```

ERD for the SQL



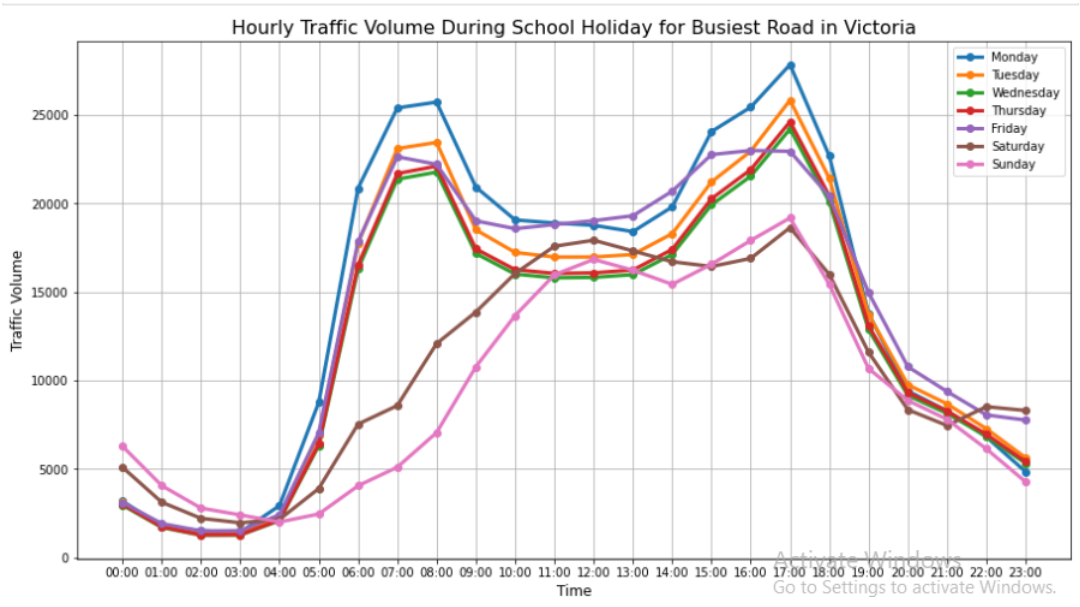
ANALYSIS

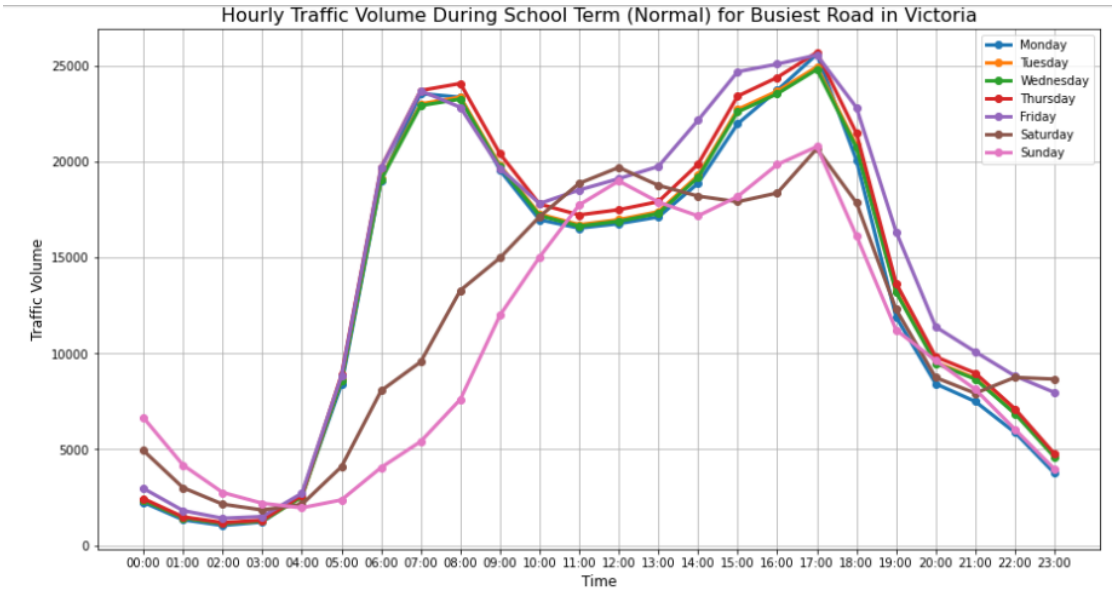
Traffic Volume Analysis



This graph shows the top ten link_id's with highest traffic volume in Victoria. Link id 297 has the highest traffic volume and link id 84 has the lowest volume. On average 1250 vehicles are passed on each link throughout Victoria.

The Hourly Traffic Volume





These graphs show the traffic volume during school term and school holidays. As per the graphs, there's a decrease in traffic volume during the school holidays. Weekends are always low with traffic but during seasonal functions traffic volume tend to increase.

The Homogeneous Traffic Flow

The homogenous traffic flow data is the extended dataset of traffic volume information which consists of latitude, longitude, and the traffic flow along a link that is representative of all travel along the whole link. Therefore, with the powerful tools in pandas, we are able to see the differences of the traffic flow in Victoria specifically on the map under various conditions.

APPENDIX

Extraction

[7]:
```
# Creating a dataframe from the API data
traffic_df = pd.DataFrame({"TIS_ID":TIS_ID_LIST})
traffic_df["HMGNS_FLOW_ID"] = HMGNS_FLOW_ID
traffic_df["HMGNS_LNK_ID"] = HMGNS_LNK_ID
traffic_df["HMGNS_LNK_DESC"] = HMGNS_LNK_DESC
traffic_df["LGA_SHORT_NM"] = LGA_SHORT_NM
traffic_df["RGN_LONG_NM"] = RGN_LONG_NM
traffic_df["ROAD_NBR"] = ROAD_NBR
traffic_df["ALLVEHS_AADT"] = ALLVEHS_AADT
traffic_df["FLOW_LIST"] = FLOW_LIST
traffic_df
```

[7]:

	TIS_ID	HMGNS_FLOW_ID	HMGNS_LNK_ID	HMGNS_LNK_DESC	LGA_SHORT_NM	RGN_LONG_NM	ROAD_NBR	ALLVEHS_AADT
0	14915	14915	2006	MARYSVILLE-WOODS POINT ROAD btwn LAKE MOUNTAI...	YARRA RANGES	METROPOLITAN SOUTH EAST REGION	4961	24
1	14140	14140	8786	STEELS CREEK ROAD btwn WILLOWBEND DRIVE & ELT...	YARRA RANGES	METROPOLITAN SOUTH EAST REGION	9999	373
2	12113	12113	6035	LATROBE ROAD btwn TANJIL EAST ROAD & GORDON S...	LATROBE	EASTERN REGION	5911	1100
3	12897	12897	7079	CASTERTON ROAD btwn GLENELG HIGHWAY & COLERAI...	SOUTHERN GRAMPIANS	SOUTH WESTERN REGION	2670	801
4	9893	9893	3475	HUTTON ROAD btwn CHAPEL ROAD & GREENS ROAD	DANDENONG	METROPOLITAN SOUTH EAST REGION	5168	12000

Transformation & Analysis – Filtering data

[15]:
```
new_df.isnull().sum()
```

[15]:
```
TIS_ID                0
HMGNS_FLOW_ID         0
HMGNS_LNK_ID          0
LGA_SHORT_NM          0
RGN_LONG_NM           0
ROAD_NBR              0
DAILY_AVG_VEHICLES    0
FLOW                  0
dtype: int64
```

[19]:
```
print(f"number of duplicate values in dataframe {new_df.duplicated().sum()}")
```

```
number of duplicate values in dataframe 0
```

[20]:
```
new_df.drop_duplicates(keep = 'first', inplace = True)
print(f"number of duplicate values in dataframe {new_df.duplicated().sum()}")
```

```
number of duplicate values in dataframe 0
```

[21]:
```
# Rename all columns to lowercase
new_df.columns = [x.lower() for x in new_df.columns]
new_df
```

[21]:

	tis_id	hmgns_flow_id	hmgns_lnk_id	lga_short_nm	rgn_long_nm	road_nbr	daily_avg_vehicles	flow
0	14915	14915	2006	YARRA RANGES	METROPOLITAN SOUTH EAST REGION	4961	24	EAST BOUND
1	14140	14140	8786	YARRA RANGES	METROPOLITAN SOUTH EAST REGION	9999	373	NORTH BOUND

Loading to SQL: Database Connection and Data Frame Loading

[34]:
```
engine = create_engine(f'postgresql://{username}:{password}@localhost:5432/traffic_db')
conn = engine.connect()
```

Check for tables

[35]:
```
# Confirm tables
engine.table_names()
```

```
<ipython-input-35-2d97e9123e98>:2: SADeprecationWarning: The Engine.table_names() method is deprecated and will
be removed in a future release.  Please refer to Inspector.get_table_names(). (deprecated since: 1.4)
  engine.table_names()
```

[35]:
```
['traffic_volume', 'hmgns_flow', 'hourly_volume']
```

Loading to SQL – Table Creation

[39]:
```
# Inner join to inscope_df and load data to database
volume_load=pd.merge(new_df, inscope_df, on='hmgns_lnk_id', how='inner')
volume_load.to_sql(name='traffic_volume', con=engine, if_exists='append', index=False)
```

[40]:
```
# No. of records in dataframe
print(f"No of records in dataframe: {len(volume_load.hmgns_lnk_id)}")

# No. of records loaded to database
cnt_traffic_volume=pd.read_sql("SELECT count(1) AS records FROM traffic_volume",conn)
print(f"No of records in database: {cnt_traffic_volume.records}")
```

```
No of records in dataframe: 10787
No of records in database: 0    10787
Name: records, dtype: int64
```
