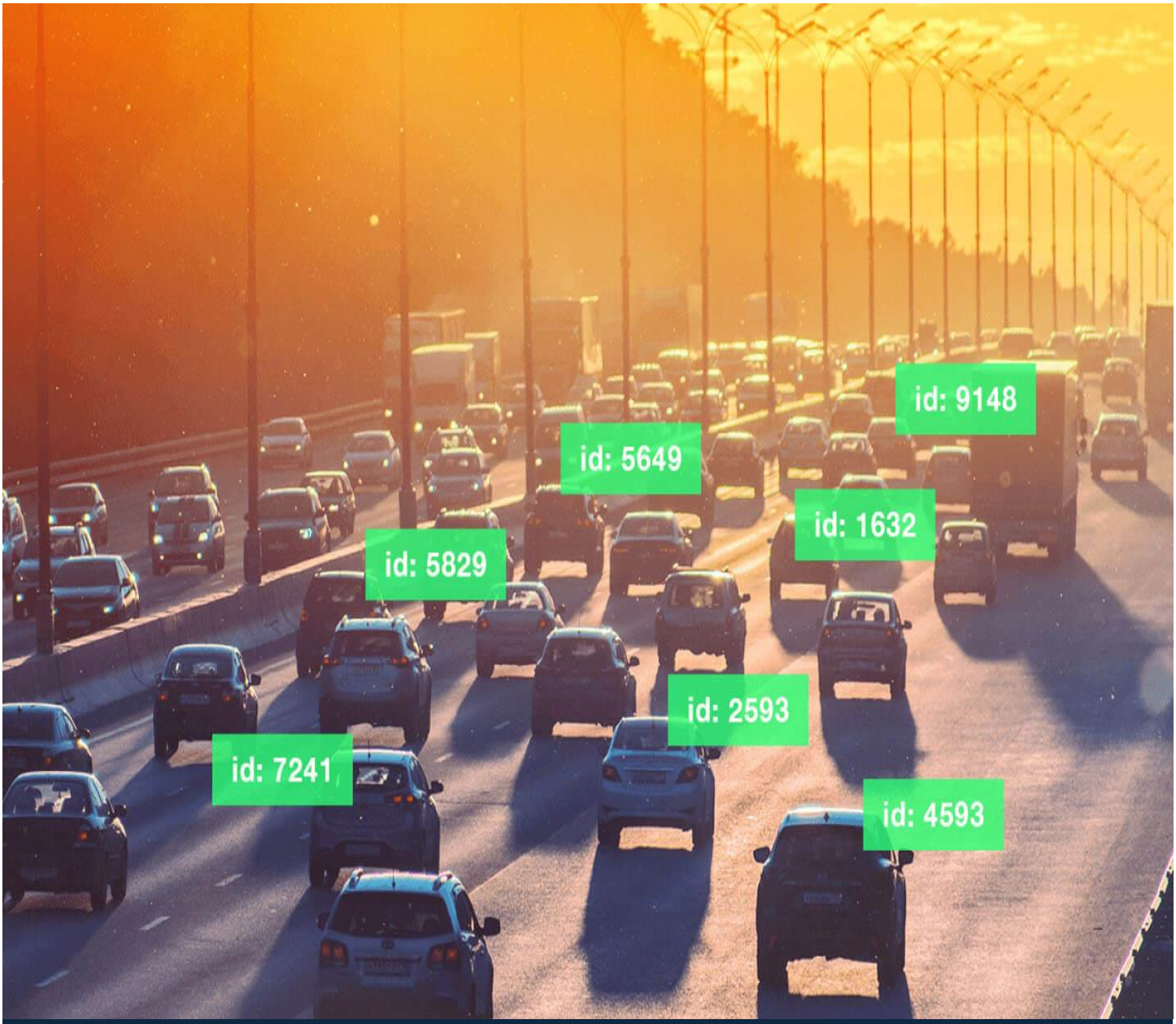


Real-Time Traffic Monitoring and Analysis

CAI1_AIS4_S10d

Final report

10/11/24

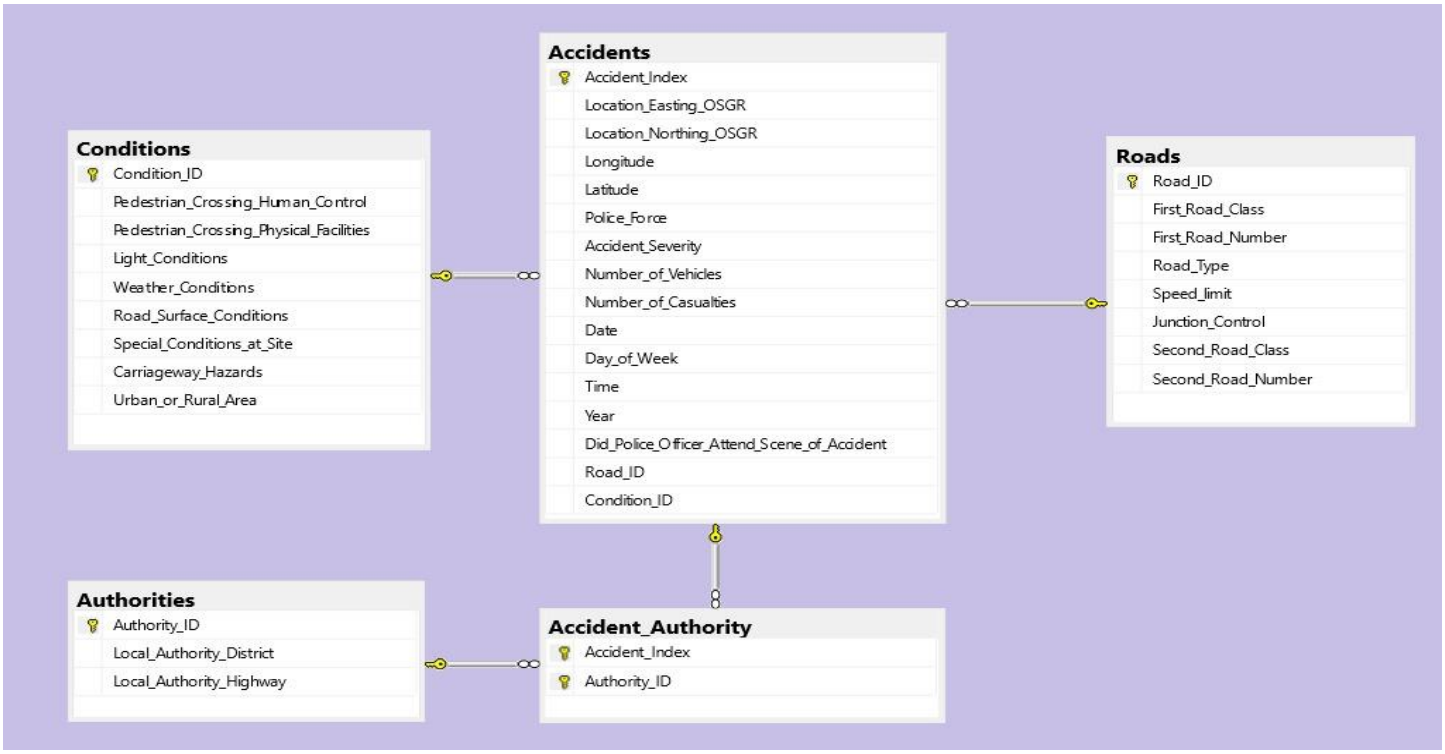
Team Members:

Name	Task
Gamal Sayed Ahmed	○ Created database schema ○ Scripted post-Loading data-python (SQL Alchemy) ○ Scripted data-python (Analysis) ○ Machine Learning Modeling (Classification model) ○ Final report
Abdelrahman Adel Mohamed	○ Loaded data into database (SSIS) ○ Created data warehouse schema (SSMS) ○ Azure services ○ Dashboard (power Bi)
Mahmoud Hamdy Mahmoud	○ SQL queries (SSMS) ○ Created data warehouse schema (SSMS) ○ Presentation
Omar Ahmed Ayad	○ SQL queries (SSMS) ○ Loaded data from database into data warehouse (SSIS) ○ Azure Services
Youssef Amr Said	○ Scripted pre-processing data-python (cleaning) (Python) ○ Presentation

1) Week 1: SQL Database and Data Collection

✓ Database Design:

➢ Designed a SQL database schema for managing traffic data

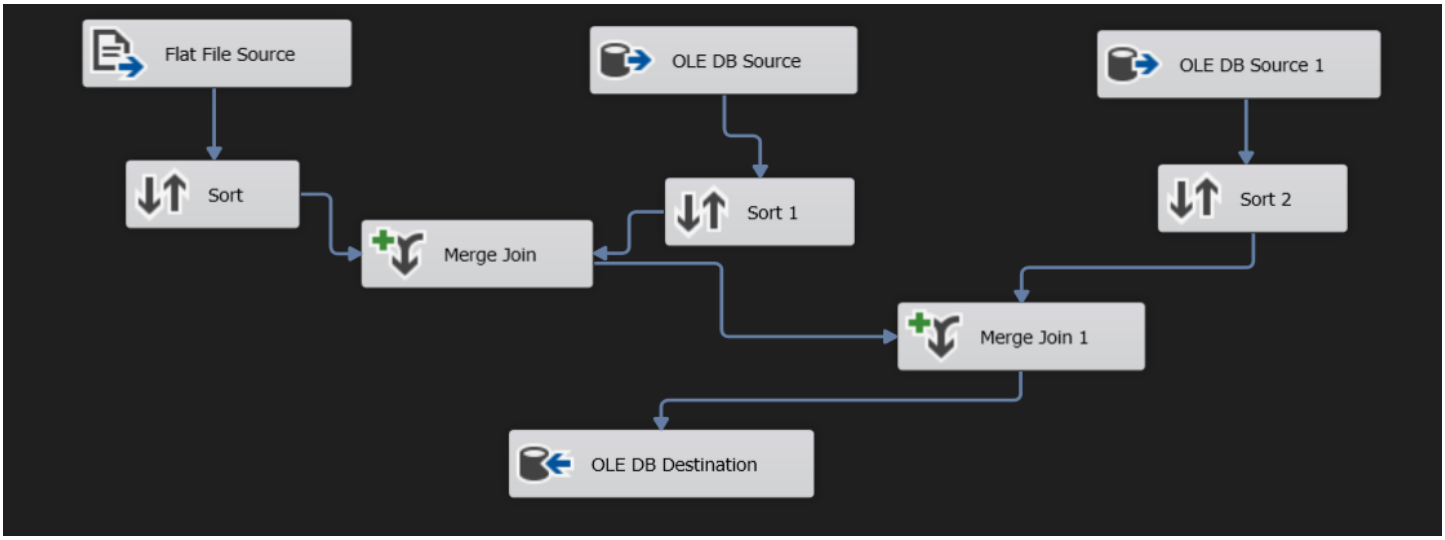


✓ Data Collection:

➢ used real-time traffic data to populate the database from Kaggle [https://www.kaggle.com/datasets/silicon99/dft-accident-data/data]

Accident_Index index numbers	Location_Easting... grid reference?	Location_Northi...	Longitude location longitude	Latitude location latitude	P 1 for
1780653 unique values	65.0k 656k	10.3k 1.21m	-7.52 1.76	49.9 60.8	1
200501BS00001	525680	178240	-0.191170	51.489096	1
200501BS00002	524170	181650	-0.211708	51.520075	1
200501BS00003	524520	182240	-0.206458	51.525301	1
200501BS00004	526900	177530	-0.173862	51.482442	1
200501BS00005	528060	179040	-0.156618	51.495752	1
200501BS00006	524770	181160	-0.203238	51.515540	1

- used SSIS to load the data into the database



- SQL queries

```sql
-----------------------------------------------
--- 4_Number of accidents by day of the week ---
-----------------------------------------------
SELECT
    CASE
        WHEN Day_of_Week = 1 THEN 'Sunday'
        WHEN Day_of_Week = 2 THEN 'Monday'
        WHEN Day_of_Week = 3 THEN 'Tuesday'
        WHEN Day_of_Week = 4 THEN 'Wednesday'
        WHEN Day_of_Week = 5 THEN 'Thursday'
        WHEN Day_of_Week = 6 THEN 'Friday'
        WHEN Day_of_Week = 7 THEN 'Saturday'
    END AS Day_Name,
    COUNT(Accident_Index) AS Total_Accidents
FROM Accidents
GROUP BY Day_of_Week
ORDER BY Day_of_Week;
-----------------------------------------------------------------------
----5_average number of vehicles involved in accidents by road type,----
-----------------------------------------------------------------------
SELECT
    R.Road_Type, AVG(A.Number_of_Vehicles) AS Average_Vehicles_Involved
FROM Accidents A JOIN Roads R
    ON A.Road_ID = R.Road_ID
GROUP BY R.Road_Type
ORDER BY Average_Vehicles_Involved DESC;
```

```sql
    Accident_Hour;
-----------------------------------------------------------------------
--- 16.Average Number of Accidents on Weekends vs. Weekdays ---
-----------------------------------------------------------------------
SELECT
    CASE
        WHEN DATEPART(WEEKDAY, Date) IN (1, 7) THEN 'Weekend'
        ELSE 'Weekday'
    END AS Day_Type,
    COUNT(Accident_Index) AS Total_Accidents
FROM
    Accidents
GROUP BY
    CASE
        WHEN DATEPART(WEEKDAY, Date) IN (1, 7) THEN 'Weekend'
        ELSE 'Weekday'
    END;


-----------------------------------------------------------------------
--- 17.Accidents Within a Specific Date Range ---
-----------------------------------------------------------------------
SELECT
    COUNT(*) AS Total_Accidents
FROM
    Accidents
WHERE
    Date BETWEEN '2005-01-01' AND '2005-12-31';
```

Tools used for week 1:

Microsoft SQL Server
SQL Management Studio

2) Week 2: Data Warehouse and Python Integration

- ✓ Data Warehouse:
  - ➢ Implemented Data warehouse using Star Shema

- Time dimension table:

```
CREATE TABLE Dim_Time (
    Time_ID INT PRIMARY KEY IDENTITY(1,1),
    Date DATE,
    Day_of_Week INT,
    Time TIME,
    Year INT
);
```

- Measures functions:

```
UPDATE Fact_Accidents
SET Casualties_Per_Vehicle = CASE
    WHEN Number_of_Vehicles = 0 THEN NULL
    ELSE Number_of_Casualties / Number_of_Vehicles
END;
```

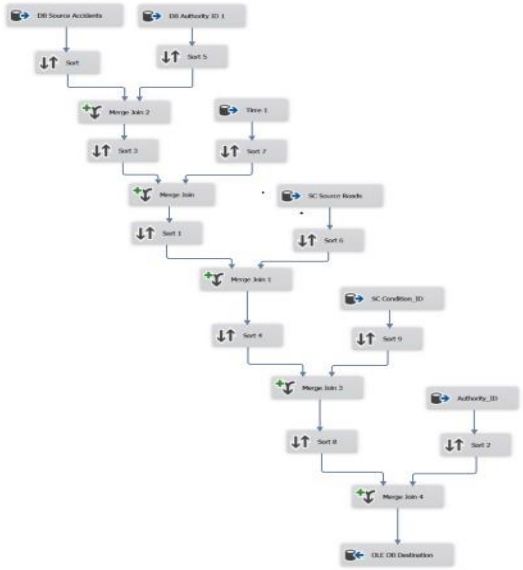
```
WITH LightConditionAccidents AS (
    SELECT f.Road_ID, COUNT(*) AS AccidentsCount
    FROM Fact_Accidents f
    JOIN Dim_Conditions c ON f.Condition_ID = c.Condition_ID
    WHERE c.Light_Conditions IN ('Daylight: Street light present', 'Darkness: Street lighting unknown','Darkness: Street lights pr
    GROUP BY f.Road_ID
)
UPDATE fa
SET fa.Accidents_Under_Light_Condition = lca.AccidentsCount
FROM Fact_Accidents fa
JOIN LightConditionAccidents lca ON fa.Road_ID = lca.Road_ID;
```

```
WITH CommonAccidentHour AS (
    SELECT f.Road_ID, DATEPART(HOUR, t.Time) AS Accident_Hour, COUNT(*) AS Frequency
    FROM Fact_Accidents f
    JOIN Dim_Time t ON f.Time_ID = t.Time_ID
    GROUP BY f.Road_ID, DATEPART(HOUR, t.Time)
)
UPDATE fa
SET fa.Most_Common_Accident_Hour = ca.Accident_Hour
FROM Fact_Accidents fa
```

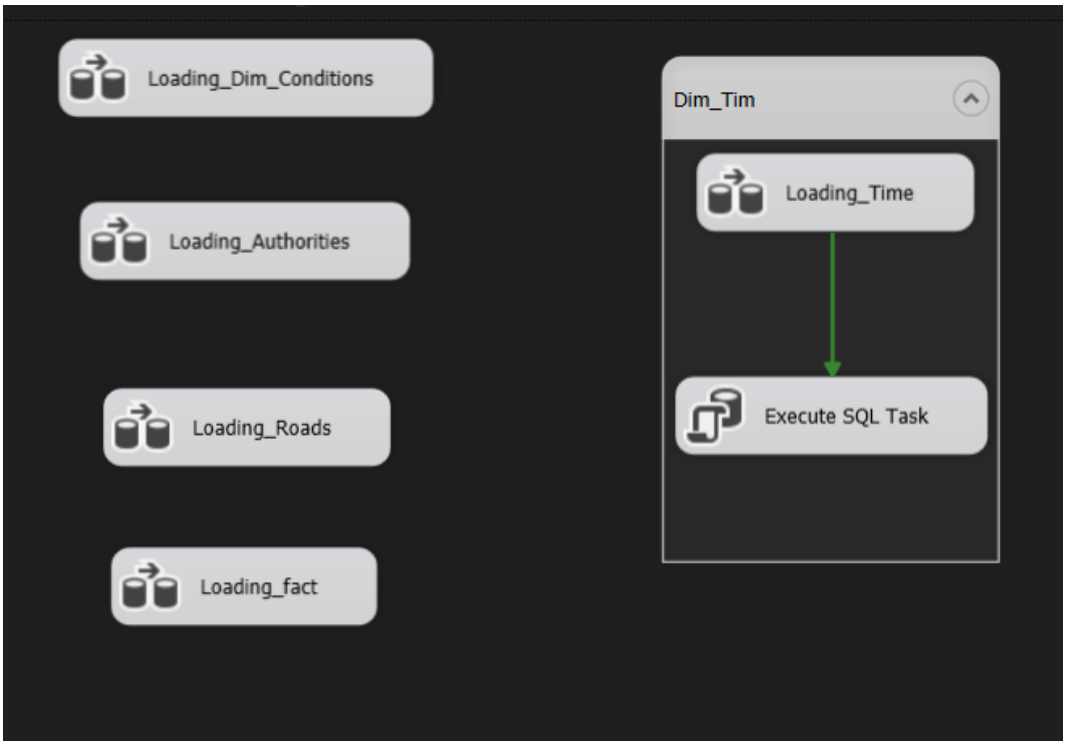
```
WITH HighRiskAreas AS (
    SELECT Road_ID
    FROM Fact_Accidents
    GROUP BY Road_ID
    HAVING COUNT(*) > 100
)
UPDATE fa
SET fa.Is_High_Risk_Area = CASE
    WHEN fa.Road_ID IN (SELECT Road_ID FROM HighRiskAreas) THEN 1
    ELSE 0
END
FROM Fact_Accidents fa;
```

✓ Data Loading:

- Using SSIS:
- Fact table data loading



➢ Other Tables data loading:



✓ Python Scripting:

➢ Pre-processing loading into database

```
roads.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 34392 entries, 1 to 49999
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Road_ID           0 non-null      float64
 1   1st_Road_Class    34392 non-null  int64
 2   1st_Road_Number   34392 non-null  int64
 3   Road_Type         34392 non-null  object
 4   Speed_limit       34392 non-null  int64
 5   Junction_Control  34392 non-null  object
 6   2nd_Road_Class    34392 non-null  int64
 7   2nd_Road_Number   34392 non-null  int64
dtypes: float64(1), int64(5), object(2)
memory usage: 2.4+ MB
```

```
roads.to_csv('new_roads_file.csv', index=False)
```

```
Accidents.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 16 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Accident_Index                          0 non-null      float64
 1   Location_Easting_OSGR                   49987 non-null  float64
 2   Location_Northing_OSGR                  50000 non-null  int64
 3   Longitude                               49987 non-null  float64
 4   Latitude                                50000 non-null  float64
 5   Police_Force                            50000 non-null  int64
 6   Accident_Severity                       50000 non-null  int64
 7   Number_of_Vehicles                      50000 non-null  int64
 8   Number_of_Casualties                    50000 non-null  int64
 9   Date                                    50000 non-null  object
 10  Day_of_Week                             50000 non-null  int64
 11  Time                                    49999 non-null  object
 12  Did_Police_Officer_Attend_Scene_of_Accident  50000 non-null  object
 13  Year                                    50000 non-null  int64
 14  Road_ID                                 0 non-null      float64
 15  Conditions_ID                           0 non-null      float64
dtypes: float64(6), int64(7), object(3)
memory usage: 6.1+ MB
```

```
Accidents.isna().sum()
```

- Pos-processing loading from database into python

Importing the needed lib and packages

```
import pandas as pd
from sqlalchemy import create_engine
import pyodbc
```

Define the connection string for SQLAlchemy

```
connection_string = 'mssql+pyodbc://DESKTOP-T4KINTS\SQLEXPRESS/Trafficuk?driver=SQL+Server'

# Create an SQLAlchemy engine
engine = create_engine(connection_string)

# Function to fetch data from a table and convert it to a Pandas DataFrame
def fetch_data(query, table_name):
    print(f"Fetching data from {table_name}...")
    return pd.read_sql(query, engine)
```

Queries to fetch data from each table

```
queries = {
    'Accidents': """
        SELECT [Accident_Index], [Location_Easting_OSGR], [Location_Northing_OSGR],
               [Longitude], [Latitude], [Police_Force], [Accident_Severity],
               [Number_of_Vehicles], [Number_of_Casualties], [Date], [Day_of_Week],
               [Time], [Year], [Did_Police_Officer_Attend_Scene_of_Accident],
               [Road_ID], [Condition_ID]
        FROM [Trafficuk].[dbo].[Accidents]
    """,
    'Roads': """
        SELECT [Road_ID], [First_Road_Class], [First_Road_Number], [Road_Type],
               [Speed_limit], [Junction_Control], [Second_Road_Class], [Second_Road_Number]
        FROM [Trafficuk].[dbo].[Roads]
    """,
    'Conditions': """
        SELECT [Condition_ID], [Pedestrian_Crossing_Human_Control],
               [Pedestrian_Crossing_Physical_Facilities], [Light_Conditions],
               [Weather_Conditions], [Road_Surface_Conditions],
               [Special_Conditions_at_Site], [Carriageway_Hazards], [Urban_or_Rural_Area]
        FROM [Trafficuk].[dbo].[Conditions]
    """,
    'Authorities': """
        SELECT [Authority_ID], [Local_Authority_District], [Local_Authority_Highway]
        FROM [Trafficuk].[dbo].[Authorities]
    """,
    'Accident_Authority':"""
        SELECT  [Accident_Index] ,[Authority_ID]
        FROM [Trafficuk].[dbo].[Accident_Authority]
    """
}
```

```
# Fetch and preprocess data from each table
accidents_df = fetch_data(queries['Accidents'], 'Accidents')
roads_df = fetch_data(queries['Roads'], 'Roads')
conditions_df = fetch_data(queries['Conditions'], 'Conditions')
authorities_df = fetch_data(queries['Authorities'], 'Authorities')
auth_acc_df = fetch_data(queries['Accident_Authority'],'Accident_Authority')
```

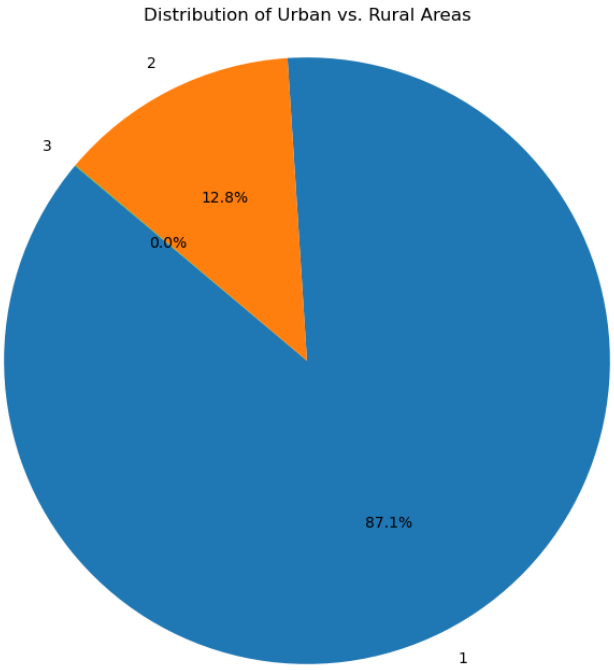
```
Fetching data from Accidents...
Fetching data from Roads...
Fetching data from Conditions...
Fetching data from Authorities...
Fetching data from Accident_Authority...
```

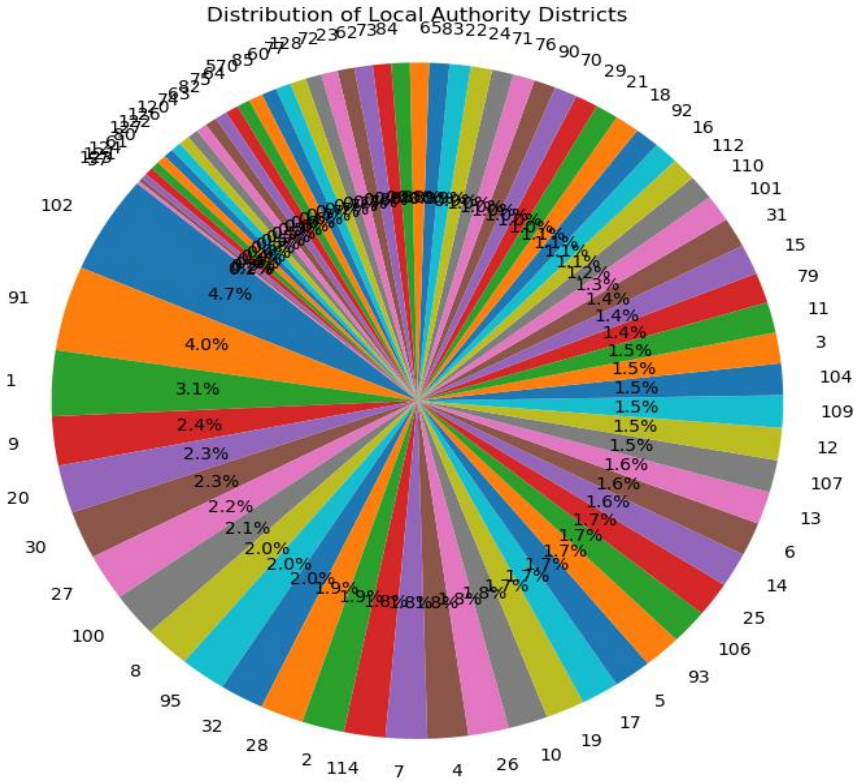
Tools used for week 2:

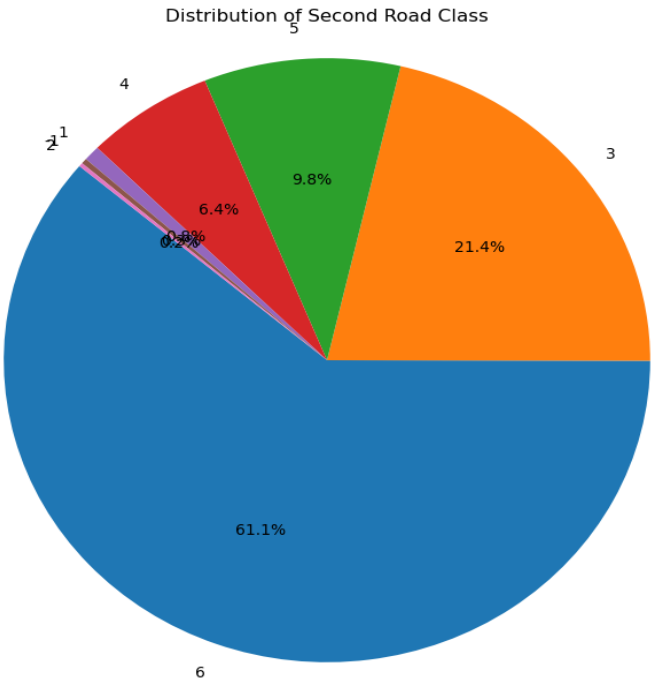
SSIS (visual studio for loading the data in the data warehouse)
SSMS (for implementing data warehouse)
Jupyter notebook (python: SQL Alchemy , pandas , NumPy)

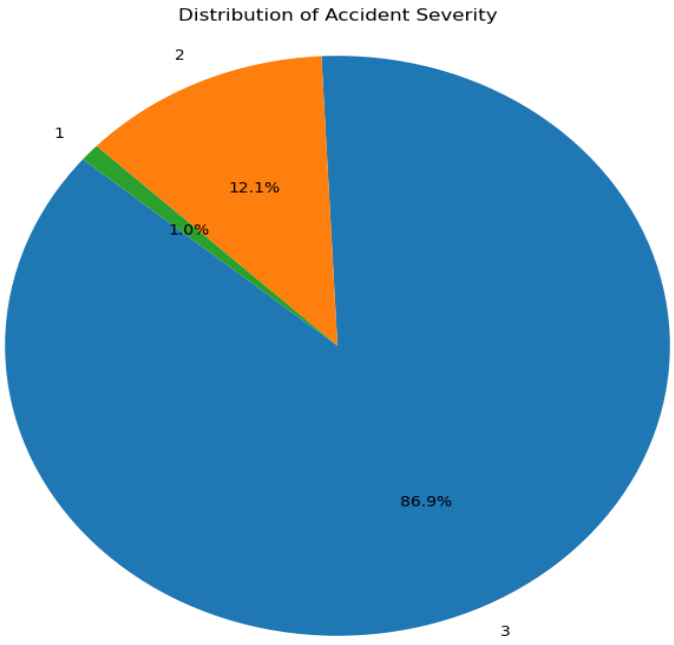
3) Week 3: Traffic Analysis and Azure Services

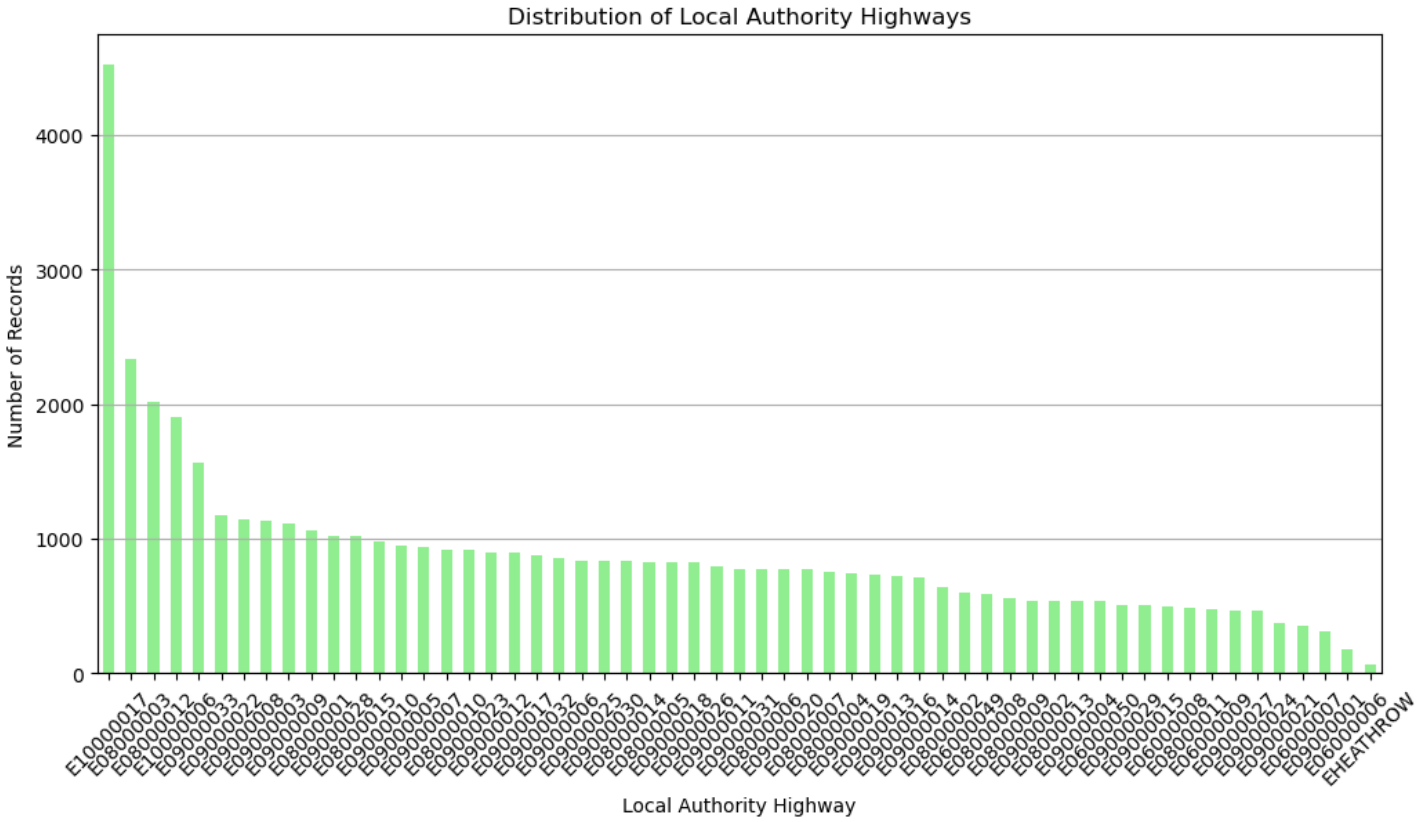
✓ Traffic Analysis:

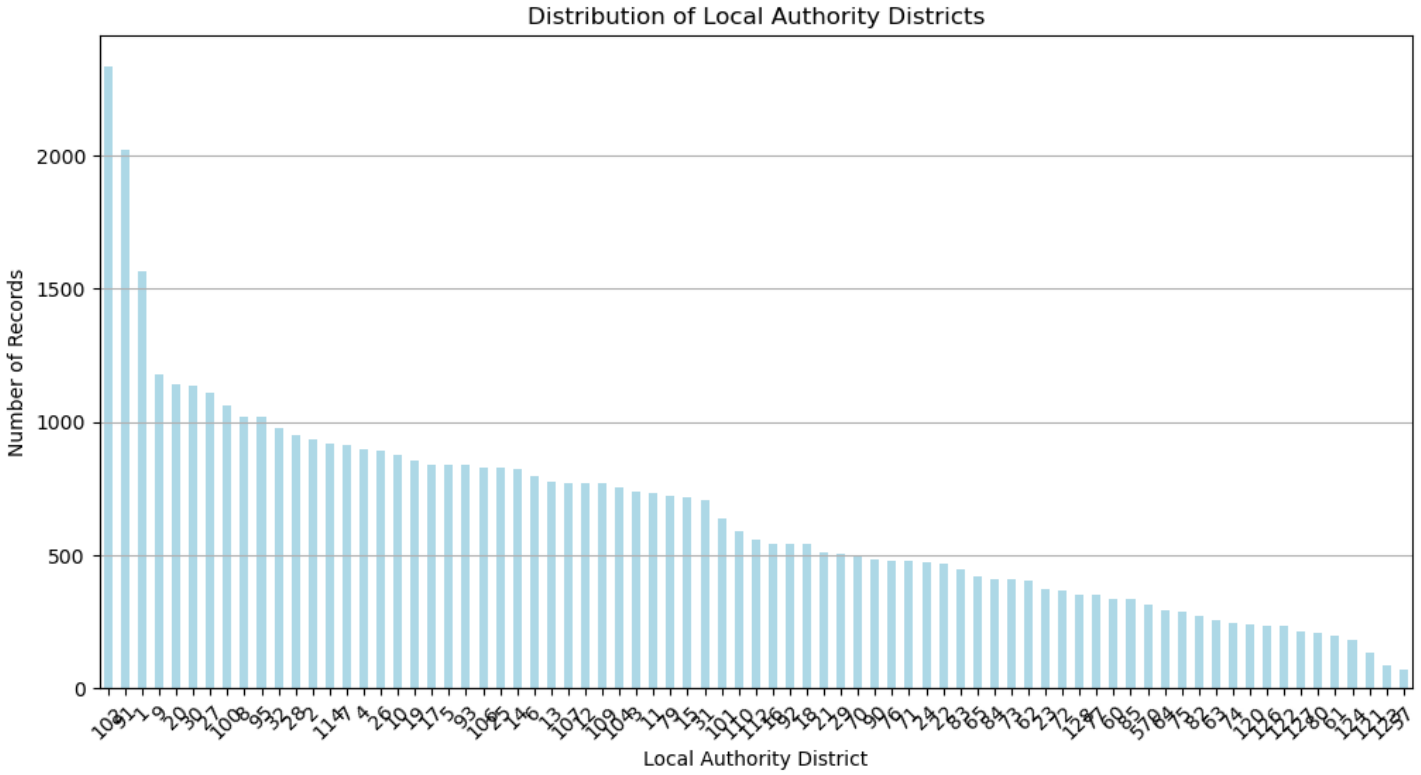


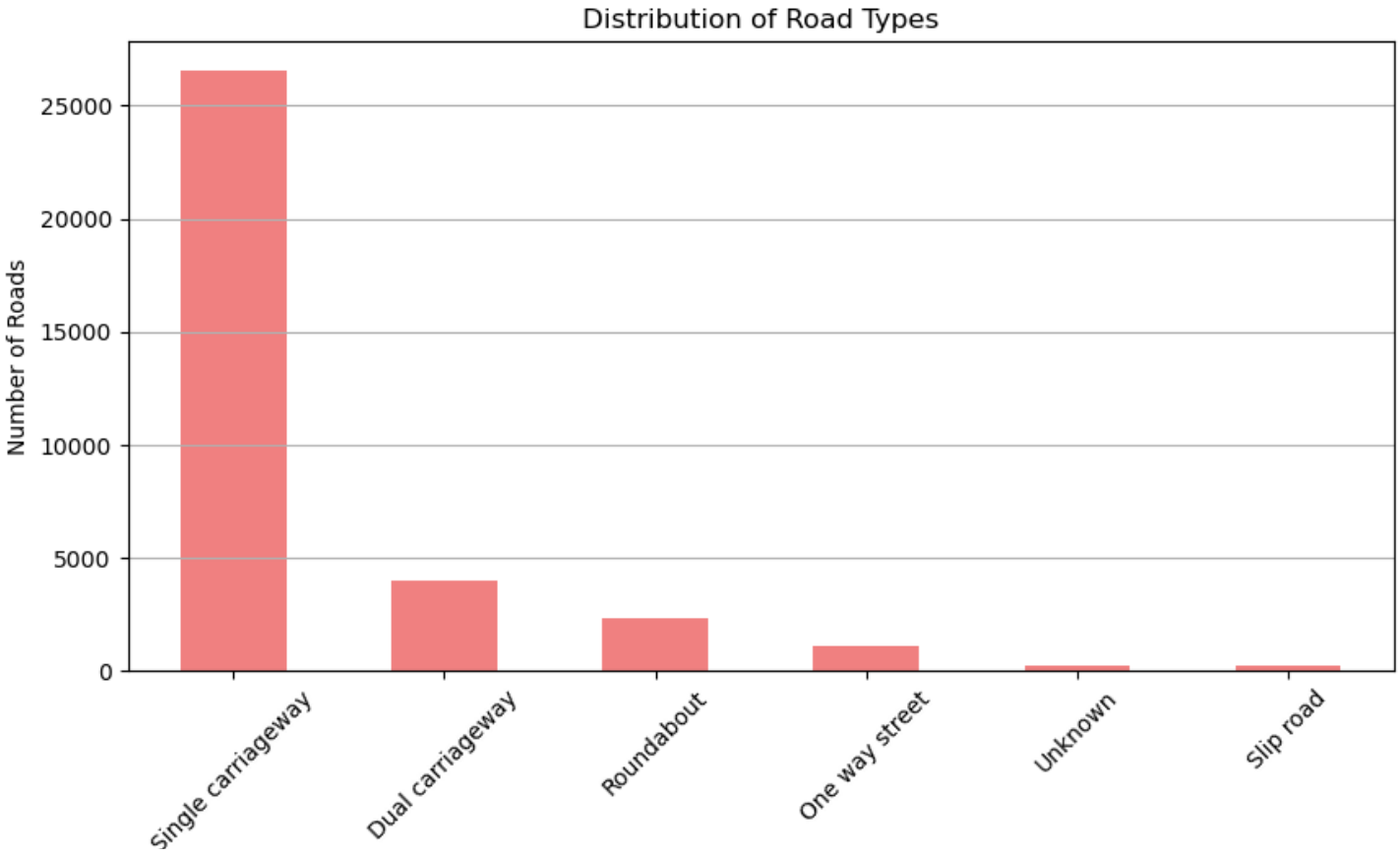


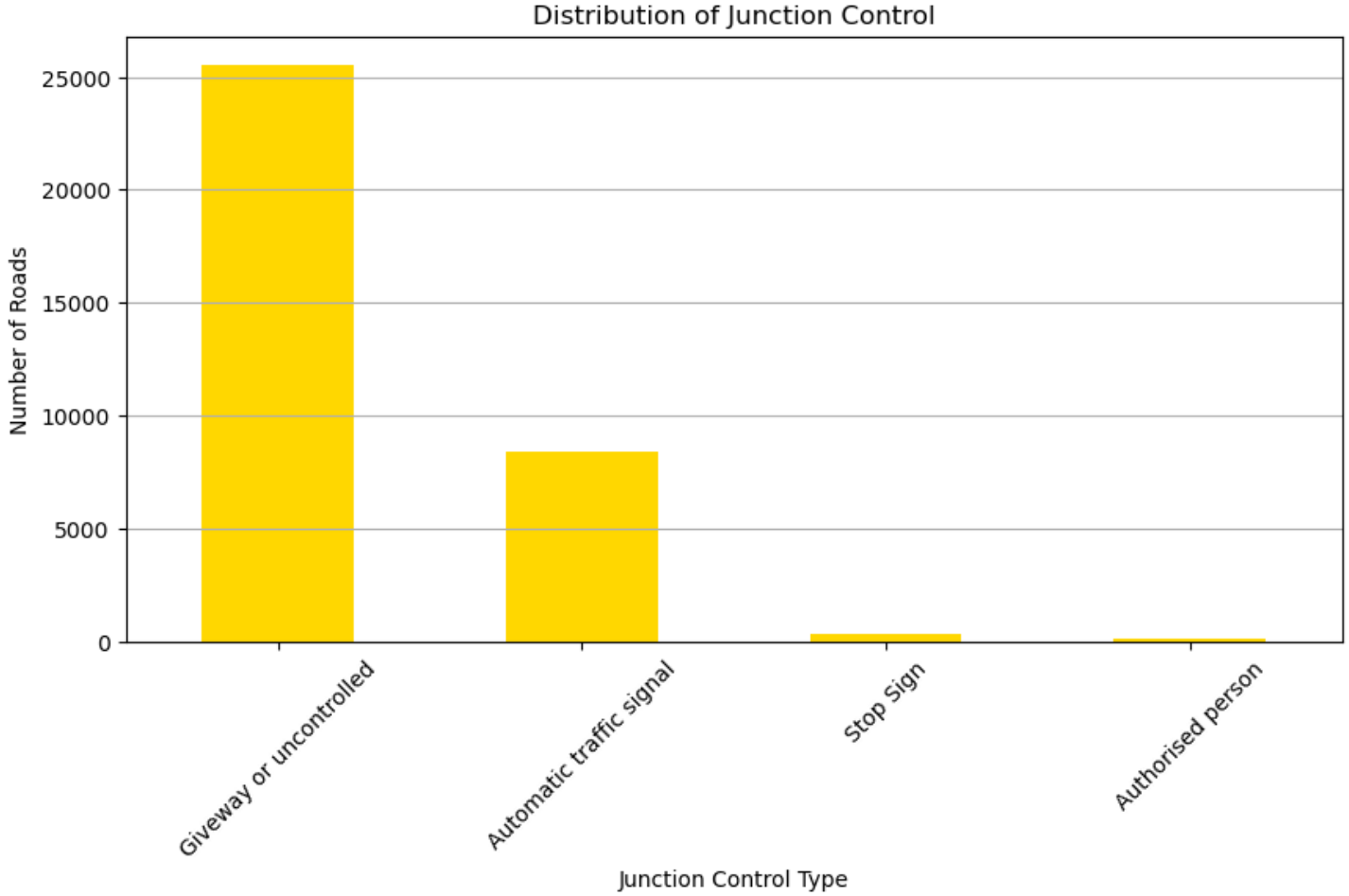


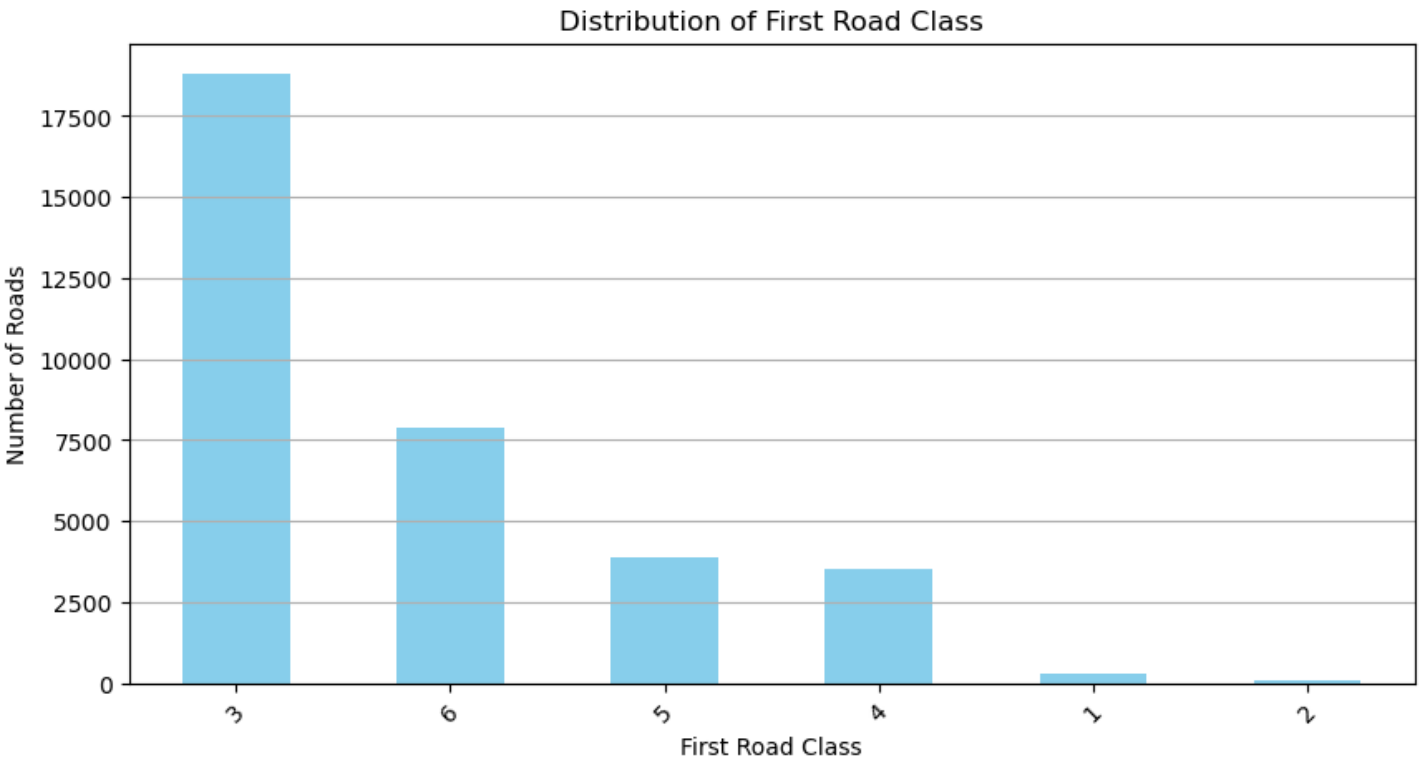


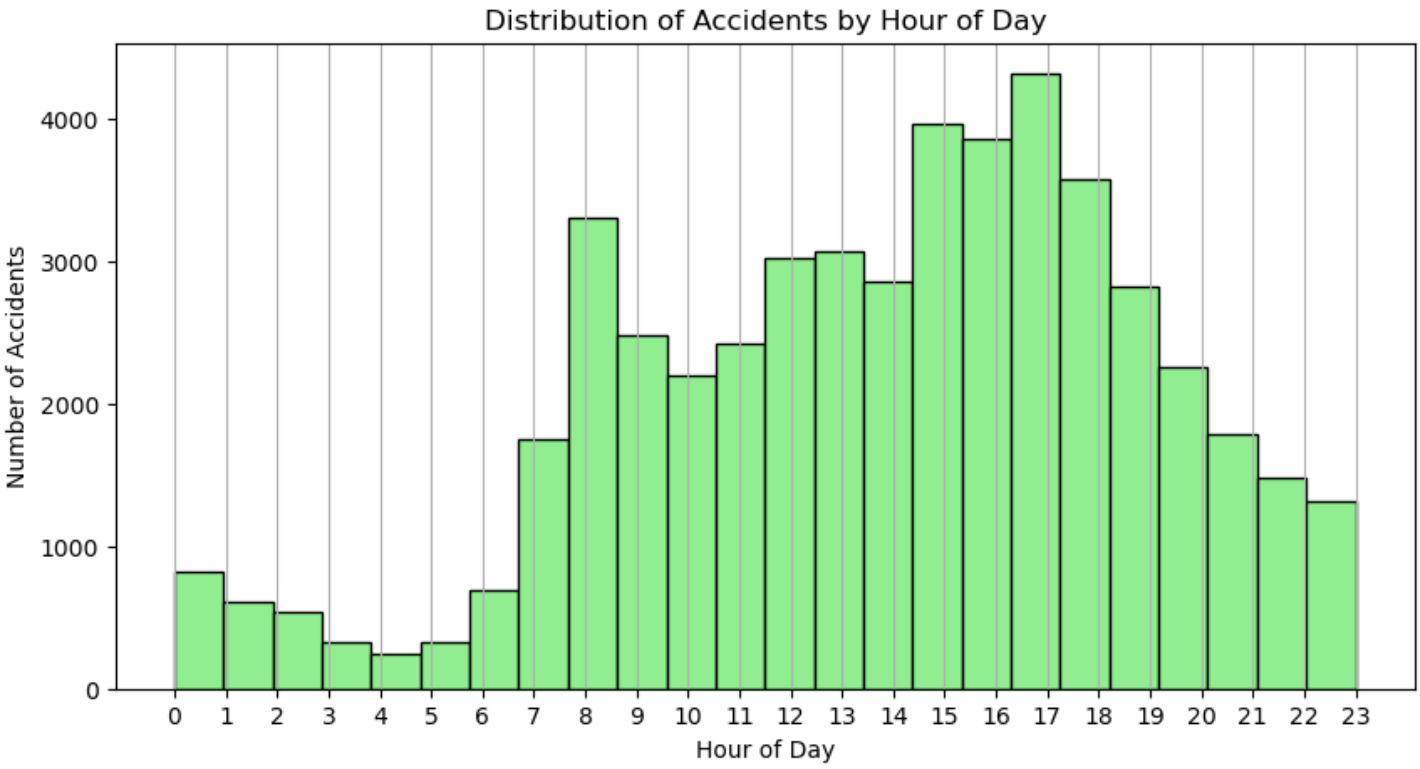

Distribution of Local Authority Highways
Number of Records
0
1000
2000
3000
4000
Local Authority Highway
E10000017
E08000003
E10000012
E10000006
E09000033
E09000022
E09000008
E09000003
E09000009
E08000001
E08000015
E09000010
E09000005
E09000007
E08000010
E09000023
E09000012
E09000017
E09000032
E09000006
E09000025
E08000030
E08000014
E08000005
E09000018
E09000026
E09000011
E08000031
E08000006
E09000020
E08000007
E09000004
E09000019
E09000013
E09000016
E08000002
E06000049
E08000008
E09000002
E08000013
E09000004
E06000050
E09000029
E06000015
E06000008
E08000011
E06000009
E09000027
E09000024
E09000021
E06000007
E09000001
E06000006
EHEATHROW

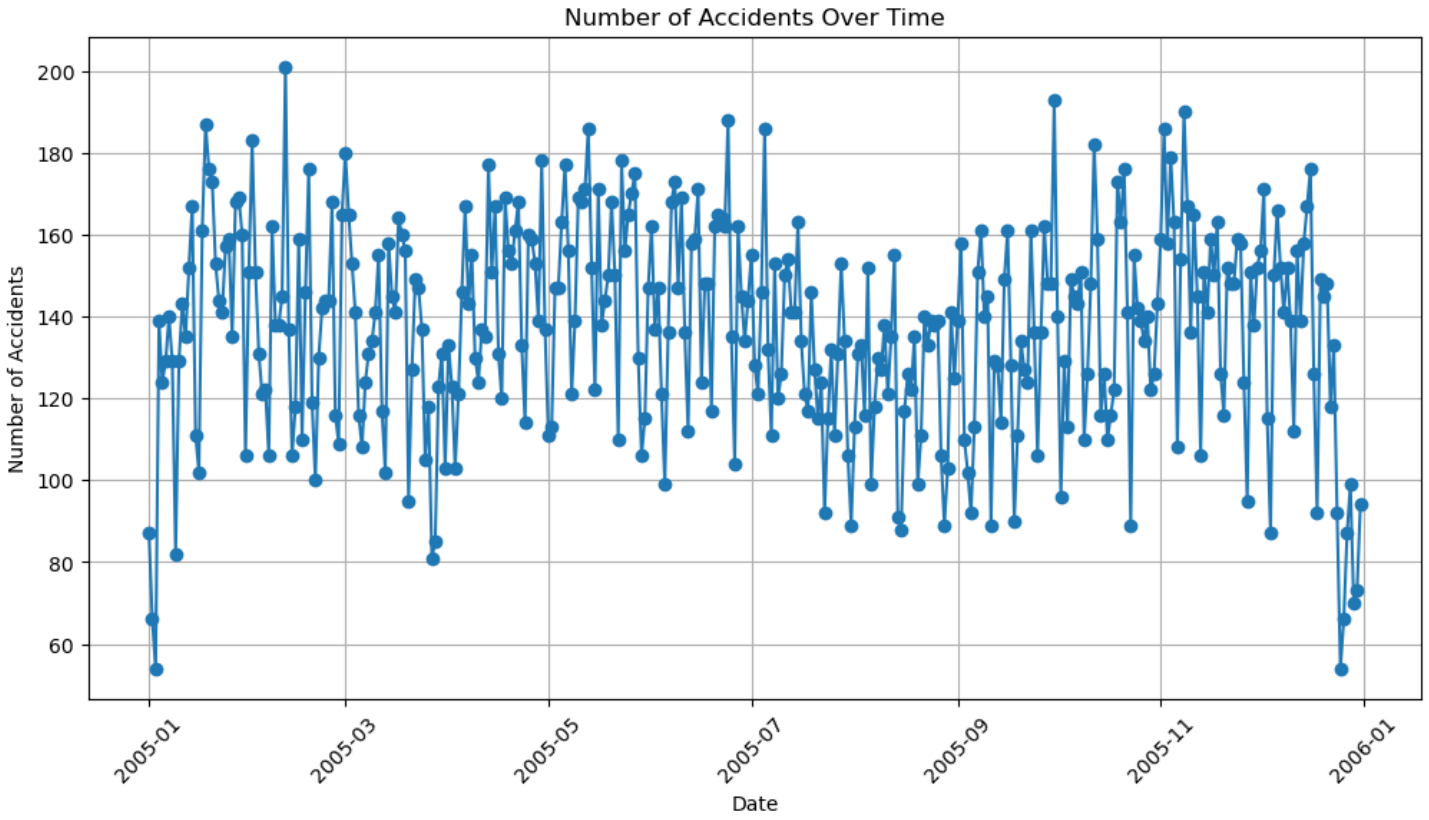

Distribution of Local Authority Districts
Number of Records
0
500
1000
1500
2000
Local Authority District

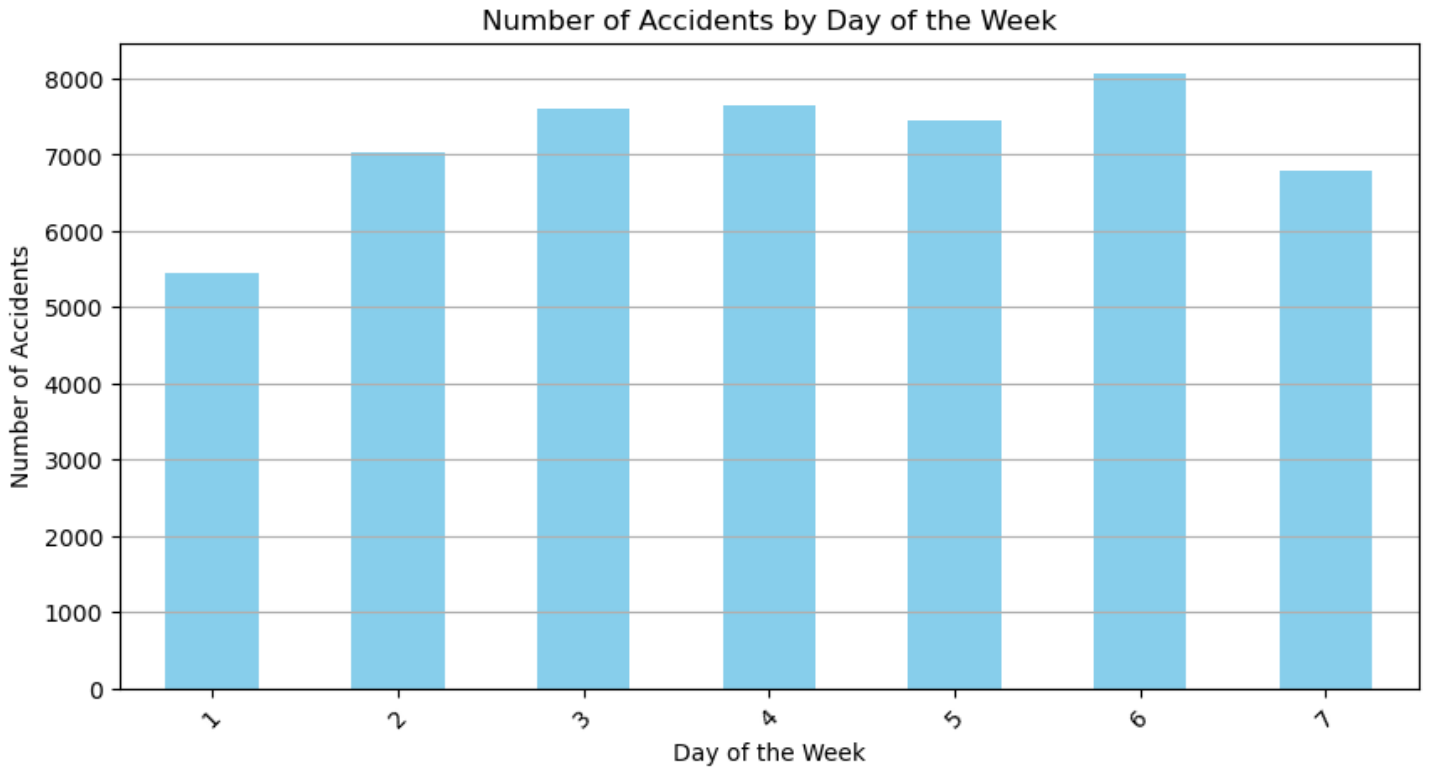

Distribution of Road Types
Number of Roads
25000
20000
15000
10000
5000
0
Single carriageway
Dual carriageway
Roundabout
One way street
Unknown
Slip road


Distribution of Junction Control
Number of Roads
25000
20000
15000
10000
5000
0
Giveway or uncontrolled
Automatic traffic signal
Stop Sign
Authorised person
Junction Control Type

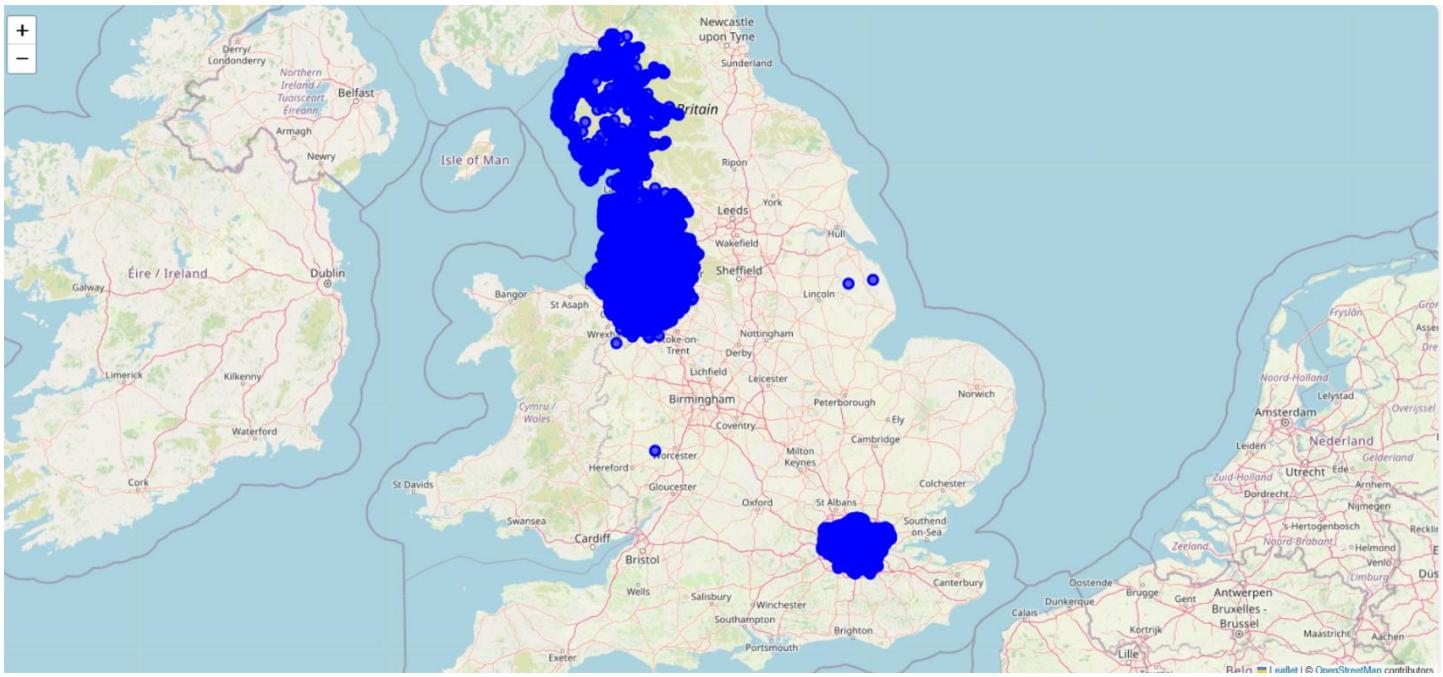

Distribution of First Road Class
Number of Roads
17500
15000
12500
10000
7500
5000
2500
0
3
6
5
4
1
2
First Road Class


Distribution of Accidents by Hour of Day
Number of Accidents
4000
3000
2000
1000
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour of Day


Number of Accidents Over Time
Number of Accidents
200
180
160
140
120
100
80
60
2005-01
2005-03
2005-05
2005-07
2005-09
2005-11
2006-01
Date


Number of Accidents by Day of the Week
Number of Accidents
8000
7000
6000
5000
4000
3000
2000
1000
0
1
2
3
4
5
6
7
Day of the Week

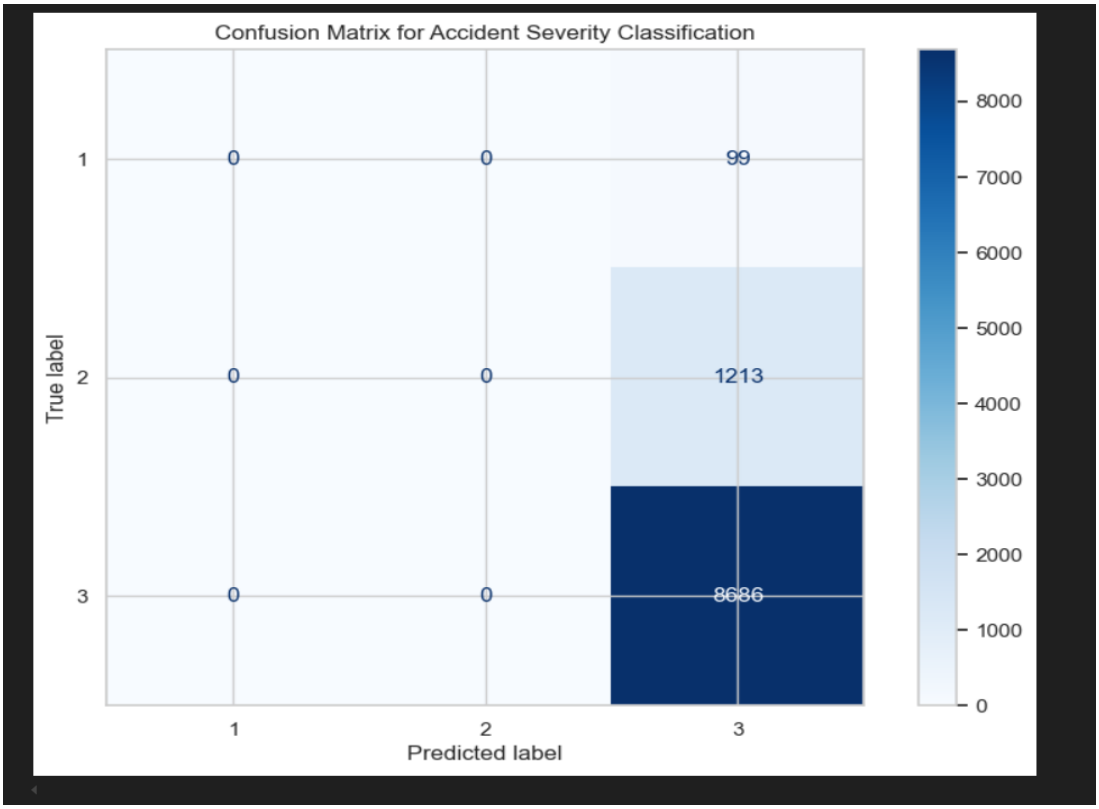
- Accident distribution across UK:



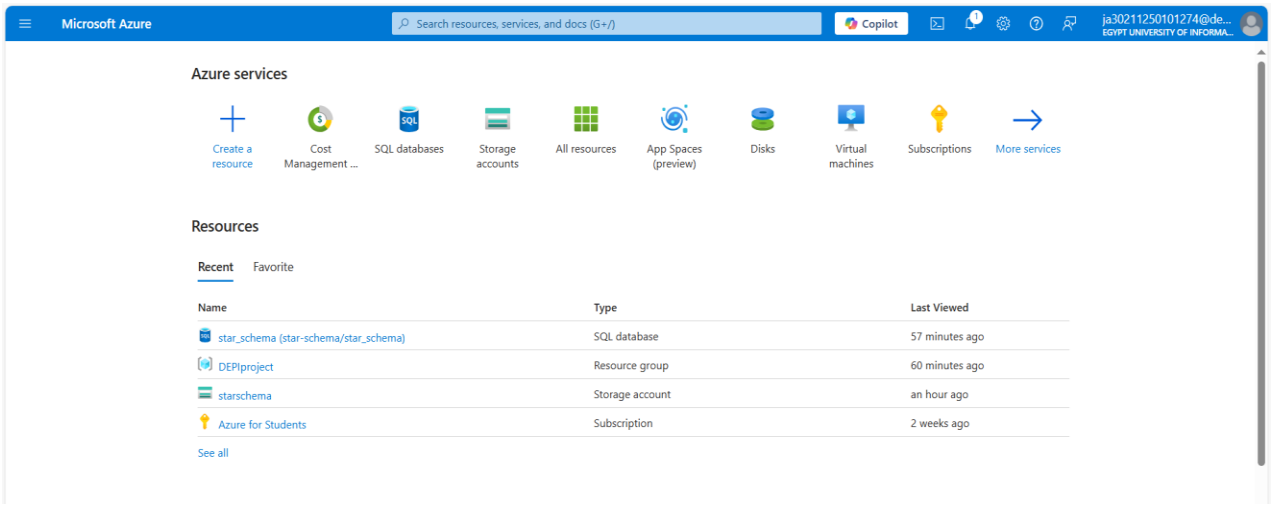
✓ Model Development:

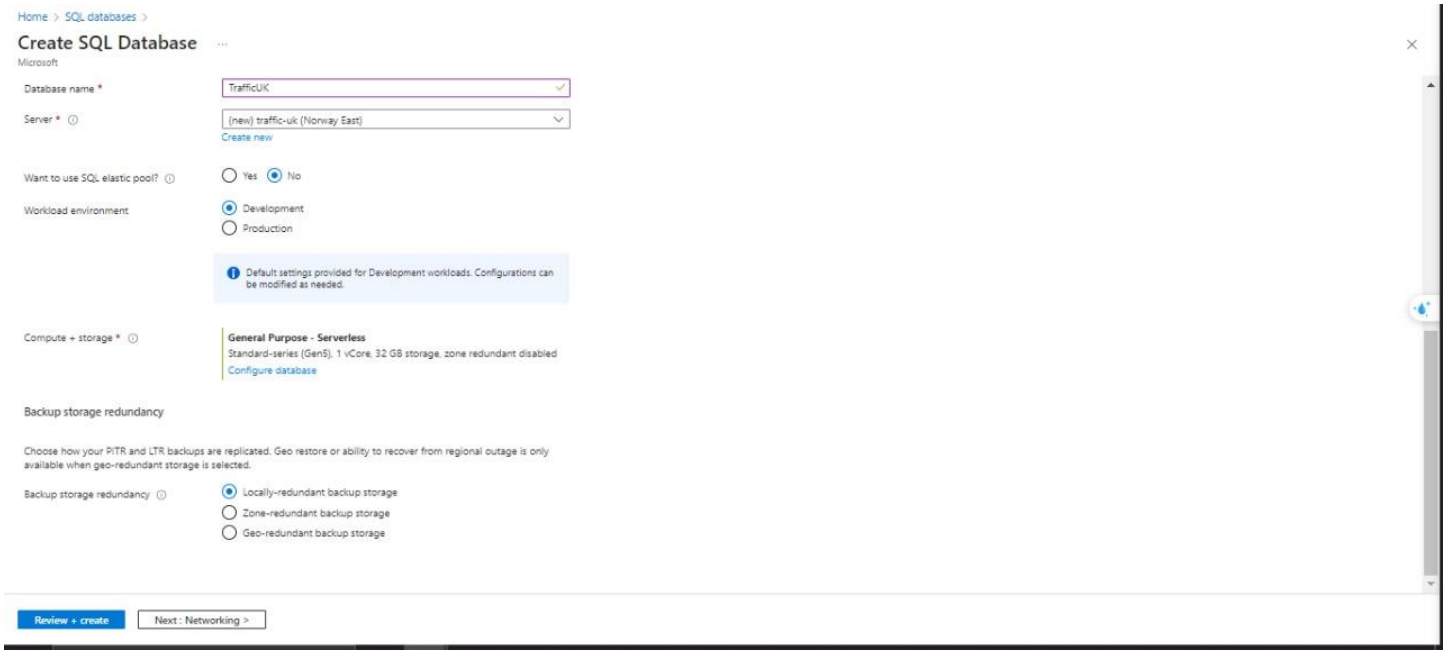
- Used random forest classification and logistic regression classification for classifying accidents severity with accuracy 87% each:

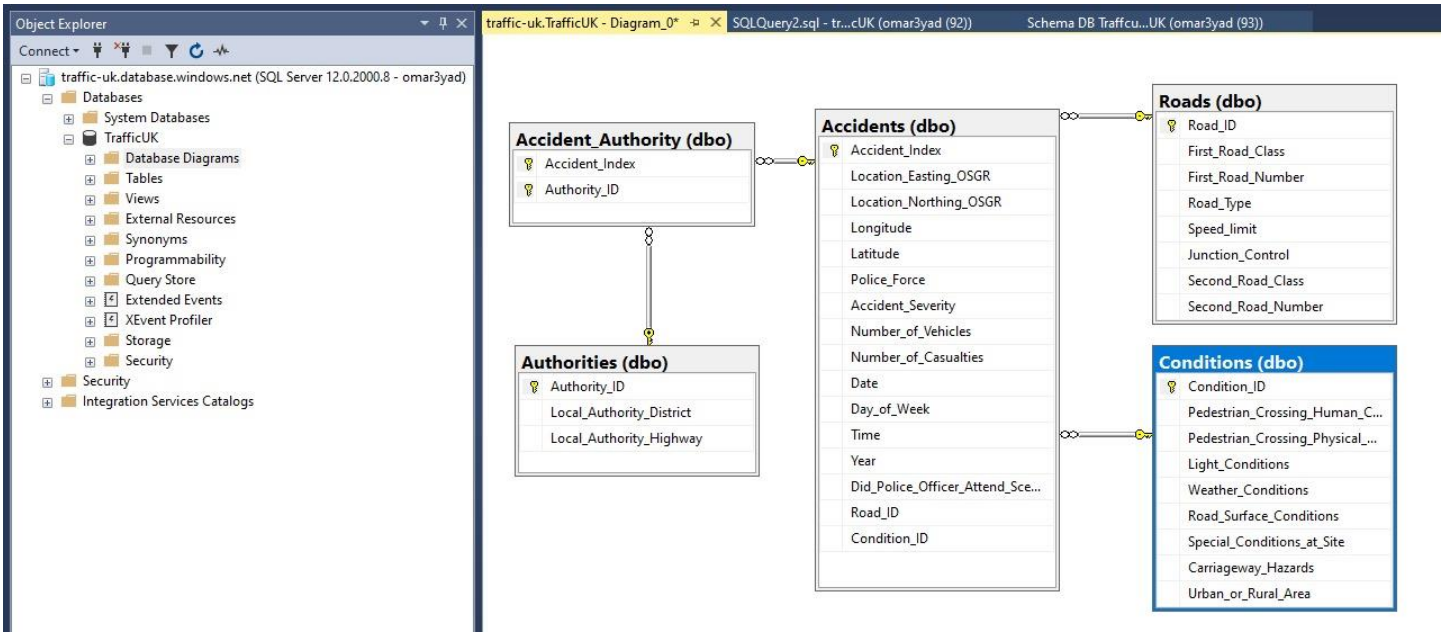
```
Classification Report:
              precision    recall  f1-score   support

           0       0.50      0.01      0.02        99
           1       0.33      0.03      0.06      1213
           2       0.87      0.99      0.93      8686

    accuracy                           0.87      9998
   macro avg       0.57      0.35      0.34      9998
weighted avg       0.80      0.87      0.81      9998

Confusion Matrix:
 [[   1    6   92]
 [   0   42 1171]
 [   1   79 8606]]
Count of Each Accident Severity Type:
 0       2
1     127
2    9869
dtype: int64
```



✓ Azure Services:





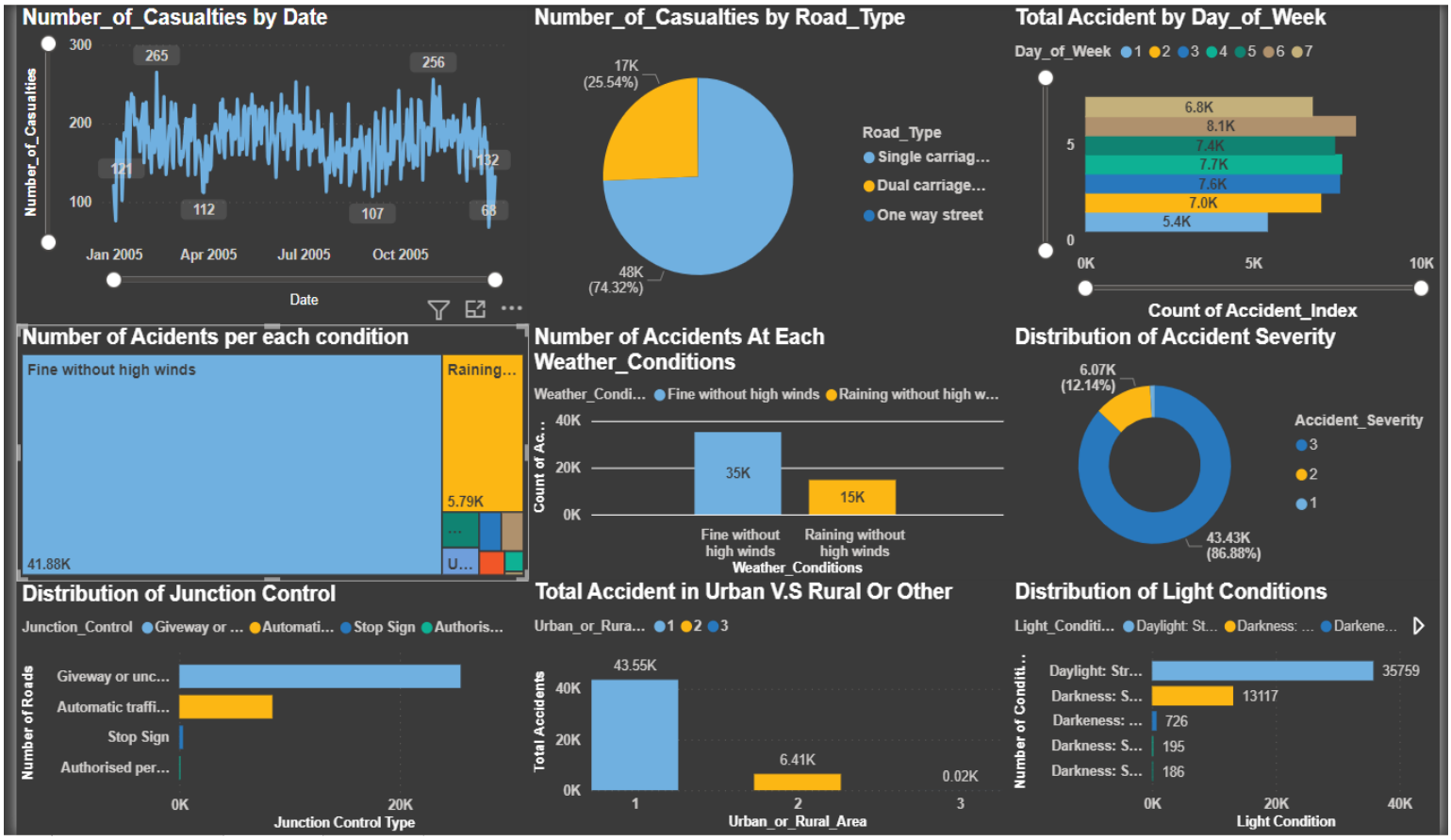


Tools used for week 3:

Azure different services
Machine learning algorithms

4) Week 4: and Deployment

- ✓ Deployment: Dashboard



Tools used for week 4:

Power bi for displaying the dashboard