

CLOUD COMPUTING DATA WAREHOUSING AND VISUALIZATION FOR UK TRAFFIC ACCIDENT ANALYSIS: A MICROSOFT AZURE APPROACH

CHAPTER 1. INTRODUCTION

1.1. Objectives

This project aims to build an end-to-end cloud-based BI system for UK traffic accident analysis using Microsoft Azure. The key objectives include:

- a. Developing a cloud-based data warehouse
 - Centralize accident data for structured analysis.
 - Implement a BI solution on the Azure Functions platform to perform ETL processes across three database layers following the Data Lakehouse architecture.
- b. Implementing event-driven data processing
 - Apply event-driven triggers and scheduled updates to maintain real-time information and automate the data ingestion process.
 - Leverage Azure Functions triggers to optimize the automated ETL workflow across all three layers.
 - Leverage Azure Functions for automated data updates.
 - Apply time-based and event-based triggers to maintain real-time insights.
- c. Building interactive dashboards
 - Design Power BI visualizations aligned with business needs.
 - Provide stakeholders with actionable insights on accident trends and high-risk locations.

By integrating cloud computing, data warehousing, and BI visualization, this project delivers a scalable, automated, and interactive analytics platform, enhancing traffic accident monitoring and supporting data-driven policy decisions in the UK.

1.2. Business context & Requirement

Table 1.1. Traffic Accident Analysis Framework

| Goal | Objective | Business questions |
|--|---|---|
| <i>Understanding Traffic Accidents</i> | <ul style="list-style-type: none">● Identify the role of light conditions in accident numbers and casualties.● Evaluate the impact of road types on accident frequency and severity.● Determine which police forces handle the most accidents to optimize resource allocation. Summarize accident trends by time, location, and severity.● Provide key metrics to help stakeholders track accident patterns. | <ol style="list-style-type: none">1. Is the total number of accidents increasing or decreasing over time?2. What is the distribution of accidents by severity (slight, serious, fatal)?3. How many casualties occur at each severity level?4. How does speed limit correlate with accident severity?5. Which road types have the highest number of accidents? |
| <i>Uncontrollable Factors in</i> | <ul style="list-style-type: none">● Analyze the impact | <ol style="list-style-type: none">1. Which areas have |

| | | |
|-----------------------------------|--|--|
| <p><i>Traffic Accidents</i></p> | <p>of location (urban/rural) on the number of accidents and casualties.</p> <ul style="list-style-type: none"> ● Assess the influence of weather conditions and road surface conditions on accident frequency and severity. | <p>the highest accident rates?</p> <ol style="list-style-type: none"> 2. How do casualty numbers compare between urban and rural areas? 3. What weather conditions contribute to the highest number of accidents? 4. How do different road surface conditions impact accident frequency? 5. Which road surface conditions are associated with the most severe accidents? |
| <p><i>Controllable Causes</i></p> | <ul style="list-style-type: none"> ● Identify the role of light conditions in accident numbers and casualties. ● Evaluate the impact of road types on accident frequency | <ol style="list-style-type: none"> 1. How do light conditions impact the number of accidents? 2. Which light conditions are associated with the |

| | | |
|--|---|---|
| | <p>and severity.</p> <ul style="list-style-type: none"> • Determine which police forces handle the most accidents to optimize resource allocation. | <p>highest number of casualties?</p> <ol style="list-style-type: none"> 3. Which road types have the highest number of accidents? 4. Which road types are associated with the most casualties? 5. Which police forces handle the most accidents? |
|--|---|---|

1.3. KPIs

1.3.1. Total Accidents

- Purpose: This KPI is designed to monitor the frequency of accidents. By tracking the total number of accidents, organizations can identify patterns, pinpoint high-risk times or locations, and assess the need for enhanced safety measures.

1.3.2. Total Fatalities

- Purpose: This KPI focuses on the severity of accidents by tracking fatalities. Monitoring this metric is vital for evaluating the effectiveness of safety protocols and interventions designed to reduce fatal accidents.

1.3.3. Serious Accident Rate (SAR)

- Purpose: SAR is a critical KPI for assessing how often serious accidents occur in relation to the total number of accidents. It helps evaluate the effectiveness of safety measures and can inform decisions regarding speed regulations or infrastructure improvements.

1.3.4. Average Speed Limit (AVGSpeed)

- Purpose: This KPI is used to evaluate the general traffic control and safety measures. It helps assess if speed limits are appropriately set in relation to road safety standards and regulations.

CHAPTER 2. DATA PREPARATION AND DATA MODELING

2.1. Data source

2.1.1. Data collection

The dataset, obtained from the UK Department for Transport and available on platforms like Kaggle and data.gov.uk, constitutes a comprehensive collection of road traffic accident data gathered over an 11-year period from 2005 to 2015. Each year's data is represented in a separate CSV file. Comprising over 876,497 records, this dataset offers a solid foundation for analyzing patterns, trends and contributing factors related to road accidents across the United Kingdom..

2.1.2. Data description

The dataset comprises 32 attributes that encompass a diverse array of variables. By utilizing these attributes, researchers can explore vital questions, such as the relationship between road conditions and the severity of accidents, the effects of speed limits on casualty rates and the distribution of accidents in urban versus rural settings. Within the broader academic landscape, this dataset enhances the evolving field of big data analytics in transportation studies.

Table 2.1. Data summary

| No. | Field Name | Description | Type |
|-----|----------------|---------------------|--------|
| 1 | Accident_Index | Accident identifier | String |

| | | | |
|----|----------------------------|--|----------|
| 2 | Location_Easting_OSGR | Local coordinate in the UK (X) | String |
| 3 | Location_Northing_OSGR | Local coordinate in the UK (Y) | String |
| 4 | Longitude | Longitude | String |
| 5 | Latitude | Latitude | String |
| 6 | Police_Force | Police unit | Int |
| 7 | Accident_Severity | Severity level | Int |
| 8 | Number_of_Vehicles | Number of vehicles damaged in the accident | Int |
| 9 | Number_of_Casualties | Number of casualties | Int |
| 10 | Date | Date of the accident | Date |
| 11 | Day_of_Week | Day of the week | Int |
| 12 | Time | Time of the accident | DateTime |
| 13 | Local_Authority_(District) | Name of the local district where the incident occurred | Int |
| 14 | Local_Authority_(Highway) | Name of the main road where the accident occurred | String |
| 15 | 1st_Road_Class | Road classification | Int |

| | | | |
|----|---|---|-------|
| 16 | 1st_Road_Number | Road number | Int |
| 17 | Road_Type | Road type | Int |
| 18 | Speed_limit | Speed limit | Float |
| 19 | Junction_Detail | Junction details | Int |
| 20 | Junction_Control | Junction control | Int |
| 21 | 2nd_Road_Class | Road classification | Int |
| 22 | 2nd_Road_Number | Road number | Int |
| 23 | Pedestrian_Crossing-Human_Control | Control of pedestrian crossing by humans | Int |
| 24 | Pedestrian_Crossing-Physical_Facilities | Physical facilities for pedestrian crossing | Int |
| 25 | Light_Conditions | Lighting conditions | Int |
| 26 | Weather_Conditions | Weather conditions | Int |
| 27 | Road_Surface_Conditions | Road surface conditions | Int |
| 28 | Special_Conditions_at_Site | Special conditions | Int |
| 29 | Carriageway_Hazards | Carriageway hazards | Int |
| 30 | Urban_or_Rural_Area | Urban or rural area | Int |

| | | | |
|----|---|---|--------|
| 31 | Did_Police_Officer_Attend_Scene_of_Accident | Did a police officer attend the accident scene? | Int |
| 32 | LSOA_of_Accident_Location | Geographic area of the accident location | String |

2.2. Data Transformation

- Handling Missing Data: Implemented binary flags for records with missing spatial coordinates to retain other valuable data and transparently track these quality issues.
- Standardizing Categorical Variables: Addressed inconsistencies in categorical fields (e.g., empty LSOA codes) using binary flags to ensure analytical transparency and prevent skewed results.
- Temporal Aggregation: Transformed raw timestamps into meaningful units (hours, days of the week, months) to facilitate pattern identification, such as discovering accident peaks during commutes.
- Spatial Clustering: Applied clustering techniques to geographic coordinates to identify accident hotspots, converting raw location data into actionable insights on high-risk areas.
- Creating Derived Variables: Analyzed relationships between existing variables (like vehicle counts and casualty numbers) to generate new insights and relationship metrics regarding accident characteristics..

2.3. Data Modeling

2.3.1. Relationship

Table 2.2. Relationship between dimensional and fact tables

| No. | Relationship | Type |
|-----|--------------|------|
|-----|--------------|------|

| | | |
|---|--|-------|
| 1 | Dim_Date → Fact_Accidents | 1 - n |
| 2 | Dim_Time → Fact_Accidents | 1 - n |
| 3 | Dim_LightConditions → Fact_Accidents | 1 - n |
| 4 | Dim_Police → Fact_Accidents | 1 - n |
| 5 | Dim_RoadType → Fact_Accidents | 1 - n |
| 6 | Dim_AccidentSeverity → Fact_Accidents | 1 - n |
| 7 | Dim_RoadSurfaceCondition s → Fact_Accidents | 1 - n |
| 8 | Dim_WeatherConditions → Fact_Accidents | 1 - n |
| 9 | Dim_UrbanorRuralArea → Fact_Accidents | 1 - n |

2.3.2. Dimension tables and Fact tables

Table 2.3. Dim_Date

| Column Name | Data Type | Description |
|-------------|-----------|------------------------|
| Date | DATE | Surrogate key for date |

| | | |
|--------------|----------------|--------------------------------------|
| Year | INT | Numeric value of year |
| Month | INT | Numeric value of month |
| Month Name | NVARCHAR(4000) | Name of the month (Jan, Feb...) |
| Day_Of_Month | INT | Day of the month (1-31) |
| DayName | NVARCHAR(4000) | Name of the day (Monday, Tuesday...) |

Table 2.4. Dim_Time

| Column Name | Data Type | Description |
|-------------|----------------|-------------------------|
| Time | NVARCHAR(4000) | Surrogate key for time |
| Hour | INT | Hour component (0-12) |
| Minute | INT | Minute component (0-59) |

Table 2.5. Dim_LightConditions

| Column | Data Type | Description |
|--------------------|----------------|---|
| LightConditionsKey | INT (PK) | Surrogate key for Light Conditions |
| Light_Conditions | INT | Code representing light conditions |
| Description | NVARCHAR(4000) | 1 = Daylight, 4 = Darkness with street lighting, 5 = Darkness without street lighting, 6 = Darkness with no lighting, 7 = Darkness with unknown lighting status |

| | | |
|------------|-----------|---|
| Start_Date | DATE | Effective start date (SCD Type 2) |
| End_Date | DATE NULL | Effective end date (NULL if current record) |
| Status | BIT | 1 = Active, 0 = Inactive (SCD Type 2) |

Table 2.6. Dim_Police

| Column Name | Data Type | Description |
|----------------|-----------|--|
| PoliceForceKey | INT (PK) | Surrogate key for Police Force |
| Police_Force | INT | Code for the police force recording the accident |
| Start_Date | DATE | Effective start date (SCD Type 2) |
| End_Date | DATE NULL | Effective end date (NULL if current record) |
| Status | BIT | 1 = Active, 0 = Inactive (SCD Type 2) |

Table 2.7. Dim_RoadType

| Column Name | Data Type | Description |
|-------------|-----------|-----------------------------|
| RoadTypeKey | INT (PK) | Surrogate key for Road Type |

| | | |
|-------------|----------------|--|
| Road_Type | INT | Code representing the type of road |
| Description | NVARCHAR(4000) | 1 = Single carriageway, 2 = Dual carriageway, 3 = Other classified road types, 6 = One-way street, 7 = Special/other road types, 9 = Unspecified/other |
| Start_Date | DATE | Effective start date (SCD Type 2) |
| End_Date | DATE NULL | Effective end date (NULL if current record) |
| Status | BIT | 1 = Active, 0 = Inactive (SCD Type 2) |

Table 2.8. Dim_AccidentSeverity

| Column Name | Data Type | Description |
|---------------------|----------------|-------------------------------------|
| AccidentSeverityKey | INT (PK) | Surrogate key for Accident Severity |
| Accident_Severity | INT | Code for accident severity |
| Description | NVARCHAR(4000) | 1 = Fatal, 2 = Serious, 3 = Slight |

| | | |
|------------|-----------|---|
| Start_Date | DATE | Effective start date (SCD Type 2) |
| End_Date | DATE NULL | Effective end date (NULL if current record) |
| Status | BIT | 1 = Active, 0 = Inactive (SCD Type 2) |

Table 2.9. Dim_RoadSurfaceConditions

| Column Name | Data Type | Description |
|--------------------------|--------------------|--|
| RoadSurfaceConditionsKey | INT (PK) | Surrogate key for Road Surface Conditions |
| Road_Surface_Conditions | INT | Code representing surface conditions |
| Description | NVARCHAR(400 0) | -1 = Not recorded, 1 = Dry, 2 = Wet/Damp, 3 = Snow, 4 = Ice, 5 = Flooded |
| Start_Date | DATE | Effective start date (SCD Type 2) |
| End_Date | DATE NULL | Effective end date (NULL if current record) |

| | | |
|--------|-----|---------------------------------------|
| Status | BIT | 1 = Active, 0 = Inactive (SCD Type 2) |
|--------|-----|---------------------------------------|

Table 2.10. Dim_WeatherConditions

| Column Name | Data Type | Description |
|----------------------|----------------|--|
| WeatherConditionsKey | INT (PK) | Surrogate key for Weather Conditions |
| Weather_Conditions | INT | Code representing weather |
| Description | NVARCHAR(4000) | -1 = Not recorded, 1 = Fine (no high winds), 2 = Fine with high winds, 3 = Rain (no high winds), 4 = Rain with high winds, 5 = Snow (no high winds), 6 = Snow with high winds, 7 = Fog/mist, 8 = Other conditions, 9 = Unspecified |
| Start_Date | DATE | Effective start date (SCD Type 2) |
| End_Date | DATE NULL | Effective end date (NULL if current record) |

| | | |
|--------|-----|---------------------------------------|
| Status | BIT | 1 = Active, 0 = Inactive (SCD Type 2) |
|--------|-----|---------------------------------------|

Table 2.11. Dim_UrbanorRuralArea

| Column Name | Data Type | Description |
|---------------------|-----------|---|
| UrbanRuralAreaKey | INT (PK) | Surrogate key for Urban/Rural area |
| Urban_or_Rural_Area | INT | 1 = Urban, 2 = Rural |
| Start_Date | DATE | Effective start date (SCD Type 2) |
| End_Date | DATE NULL | Effective end date (NULL if current record) |
| Status | BIT | 1 = Active, 0 = Inactive (SCD Type 2) |

Table 2.12. Fact_Accidents

| Column Name | Data Type | Description |
|--------------------|---------------------|--|
| AccidentIndex | NVARCHAR(4000) (PK) | Unique accident identifier from source |
| PoliceKey | INT (FK) | Foreign key to Dim_Police |
| LightConditionsKey | INT (PK) | Foreign key to Dim_LightConditions |
| RoadTypeKey | INT (PK) | Foreign key to Dim_RoadType |

| | | |
|--------------------------|----------------|---|
| AccidentSeverityKey | INT (PK) | Foreign key to Dim_AccidentSeverity |
| RoadSurfaceConditionsKey | INT (PK) | Foreign key to Dim_RoadSurfaceConditions |
| WeatherConditionsKey | INT (PK) | Foreign key to Dim_WeatherConditions |
| UrbanRuralAreaKey | INT (PK) | Foreign key to Dim_UrbanRuralArea |
| Longitude | FLOAT | Geographic longitude |
| Latitude | FLOAT | Geographic latitude |
| Local_Authority_District | INT | District authority code |
| Local_Authority_Highway | NVARCHAR(4000) | Highway authority name |
| Date | DATE | Accident date |
| Time | NVARCHAR(4000) | Accident hour and minute |
| Number_of_Vehicles | INT | Number of vehicles involved |
| Number_of_Casualties | INT | Number of casualties |
| Speed_Limit | INT | Speed limit (mph) |

CHAPTER 3. EXPERIMENTING WITH THE ETL PROCESS ON AZURE FUNCTIONS

3.1. ETL process on Azure Functions

The BI solution for the data of traffic accidents in the UK is implemented on the cloud computing platform Microsoft Azure. The execution process applies maximum automation through scheduled events starting from local, from when the data files are stored on the device's hard drive. Overall, the implementation process will include 3 main phases: (1) Ingest from local to blob storage (2) ETL all data into three architectural layers in the SQL database and the final phase (3) Upload data to PowerBI and proceed with visualization.

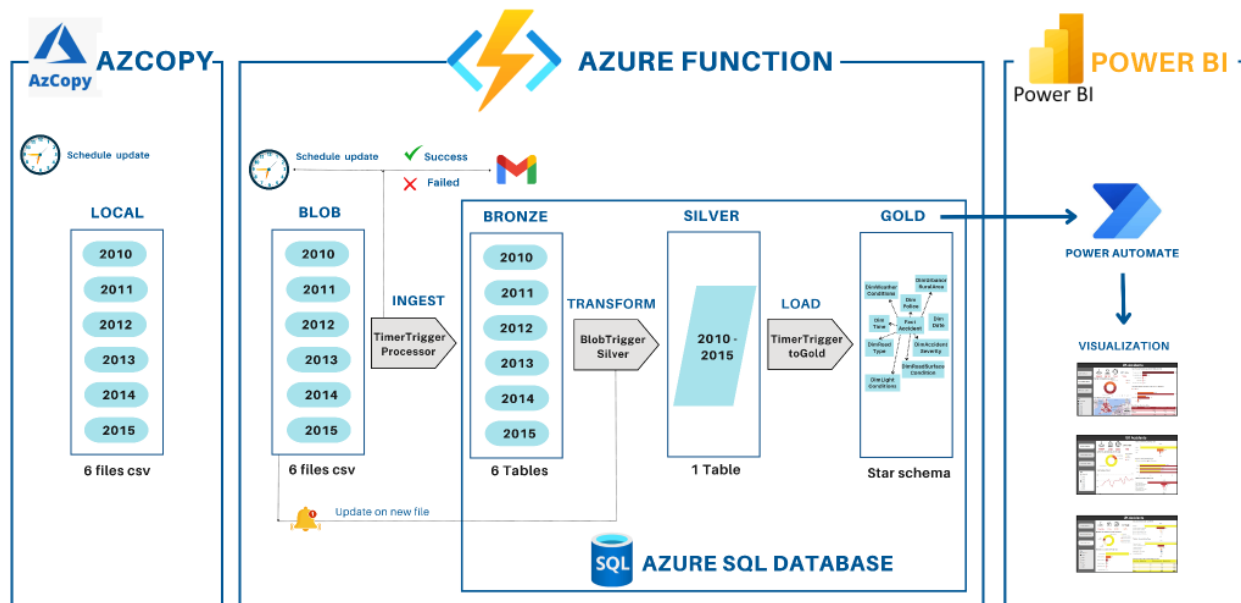


Figure 3.1. Business Intelligence Solutions on the Microsoft Azure Platform

(1) Ingest from local to blob storage: in this phase, the data will be updated based on a fixed date and time set up in advance at local and uploaded to the blob storage account.

(2) ETL all data into three architectural layers in the SQL database and the final phase: this is the most complex phase in the data ingestion process, as it requires processing data through multiple layers, plus setting up some triggers to enhance the ETL process. The ETL process will be fully executed through Azure Functions and conclude with the recording of data into the SQL database. It starts with the bronze layer where raw data is

stored; the data in the bronze layer will be updated according to a pre-scheduled time, and additionally, during the data loading process, email notifications are sent to confirm whether the loading process is completed or if any errors occur. When the data enters the silver layer, it must undergo several transformation techniques to ensure the data is standardized and properly structured; the silver layer is also equipped with trigger operations to automatically load data whenever a new file is updated. Finally, the data when entering the bronze layer will follow the data warehouse structure, which is the star schema.

(3) Upload data to PowerBI and proceed with visualization: this phase focuses on deep analysis and understanding of current and past insights visualized through dashboards. Following that, valuable recommendations and insights are derived, while also addressing the business questions posed from the beginning.

3.1.1. Az Copy to blob storage

The system employs AzCopy, a Microsoft command-line tool, for efficient and secure data transfer to Azure Blob Storage. AzCopy ensures high-speed transfers and robust security through features like encryption, utilizing SAS Tokens for secure, temporary authentication without exposing primary credentials. After initial setup, including PATH configuration for convenient access, AzCopy commands are often embedded in PowerShell scripts to automate the transfer process. These scripts are then scheduled using Windows Task Scheduler for regular, unattended execution, such as a daily transfer at 12:00 AM, which typically completes in 20-30 seconds.

```
Administrator: Windows PowerShell
PS C:\> azcopy -h
AzCopy 10.3.2
Project URL: github.com/Azure/azure-storage-azcopy

AzCopy is a command line tool that moves data into and out of Azure Storage.
To report issues or to learn more about the tool, go to github.com/Azure/azure-storage-azcopy

The general format of the commands is: 'azcopy [command] [arguments] --[flag-name]=[flag-value]'.

Usage:
  azcopy [command]

Available Commands:
  bench      Performs a performance benchmark
  copy       Copies source data to a destination location
  doc        Generates documentation for the tool in Markdown format
  env        Shows the environment variables that you can use to configure the behavior of AzCopy.
  help       Help about any command
  jobs       Sub-commands related to managing jobs
  list       List the entities in a given resource
  login      Log in to Azure Active Directory (AD) to access Azure Storage resources.
  logout     Log out to terminate access to Azure Storage resources.
  make       Create a container or file share.
  remove     Delete blobs or files from an Azure storage account
  sync       Replicate source to the destination location

Flags:
  --cap-mbps uint32  Caps the transfer rate, in megabits per second. Moment-by-moment throughput might vary slightly from the cap. If this option is set to zero, or it is omitted, the throughput isn't capped.
  -h, --help          help for azcopy
  --output-type string Format of the command's output. The choices include: text, json. The default value is 'text'. (default "text")
  --version           version for azcopy

Use "azcopy [command] --help" for more information about a command.
PS C:\>
```

Figure 3.2. Azcopy tool settings screen

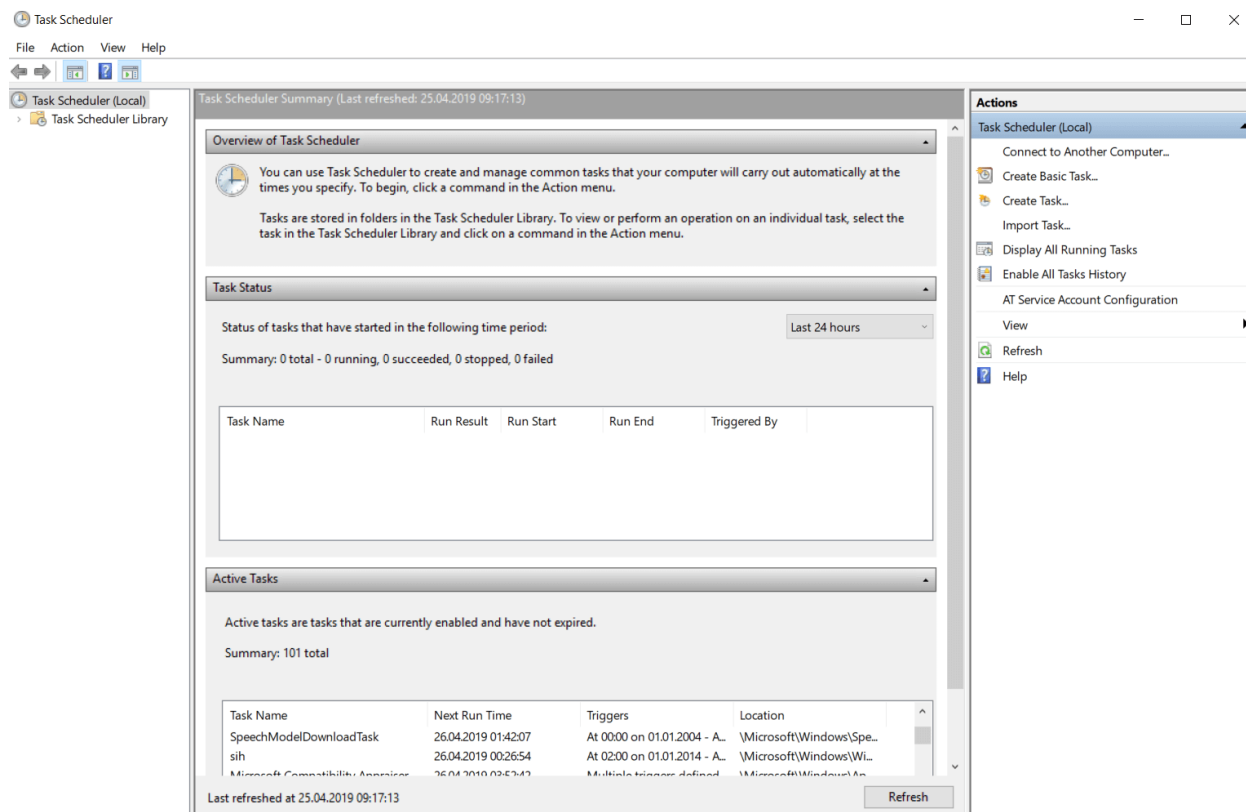


Figure 3.3. Task Scheduler tool settings screen

3.1.2. Azure Functions Deployment on Azure Portal

To start the ETL process using the Azure Function tool, creating a FunctionApp on the Azure Portal is mandatory. The FunctionApp named RawBronzeSilverGoldlayer is created and will link directly to the storage account; here, the author group selects the hosting option for the function app as Consumption. After successfully creating the function app on the Azure Portal environment, the author group uses the local environment, VS Code, to ingest data into the 3 layers in the database. Therefore, three functions—TimerBlobProcessor, BlobTriggerSilver, and TimeTriggertoGold—are created in the local VS Code environment, with each function acting as a pipeline responsible for the data ingestion process into the bronze, silver, and gold layers respectively, corresponding to each listed function.

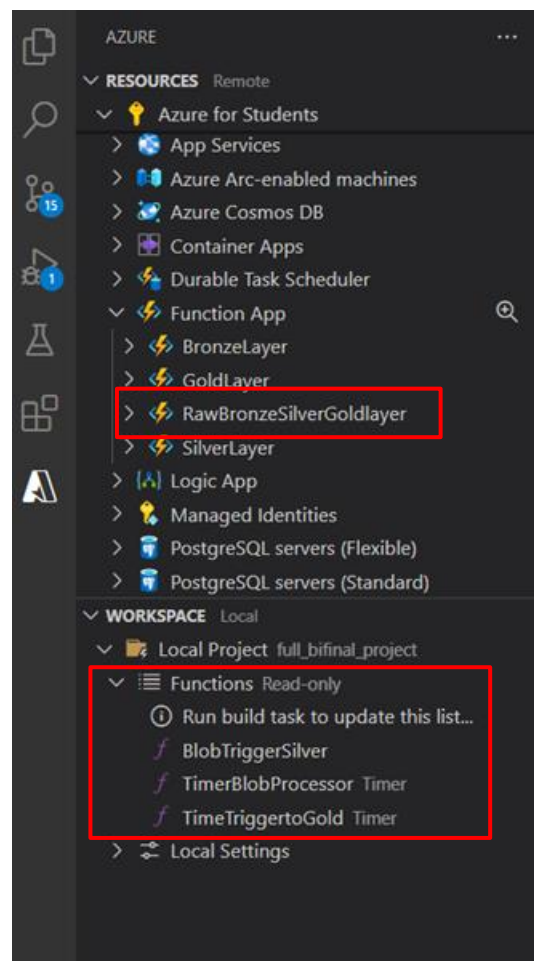


Figure 3.4. Project structure in the VS Code environment

When successfully executed in the local environment, the three functions responsible for each layer will be deployed to the Azure Portal via the initially created Function App, RawBronzeSilverGoldlayer.

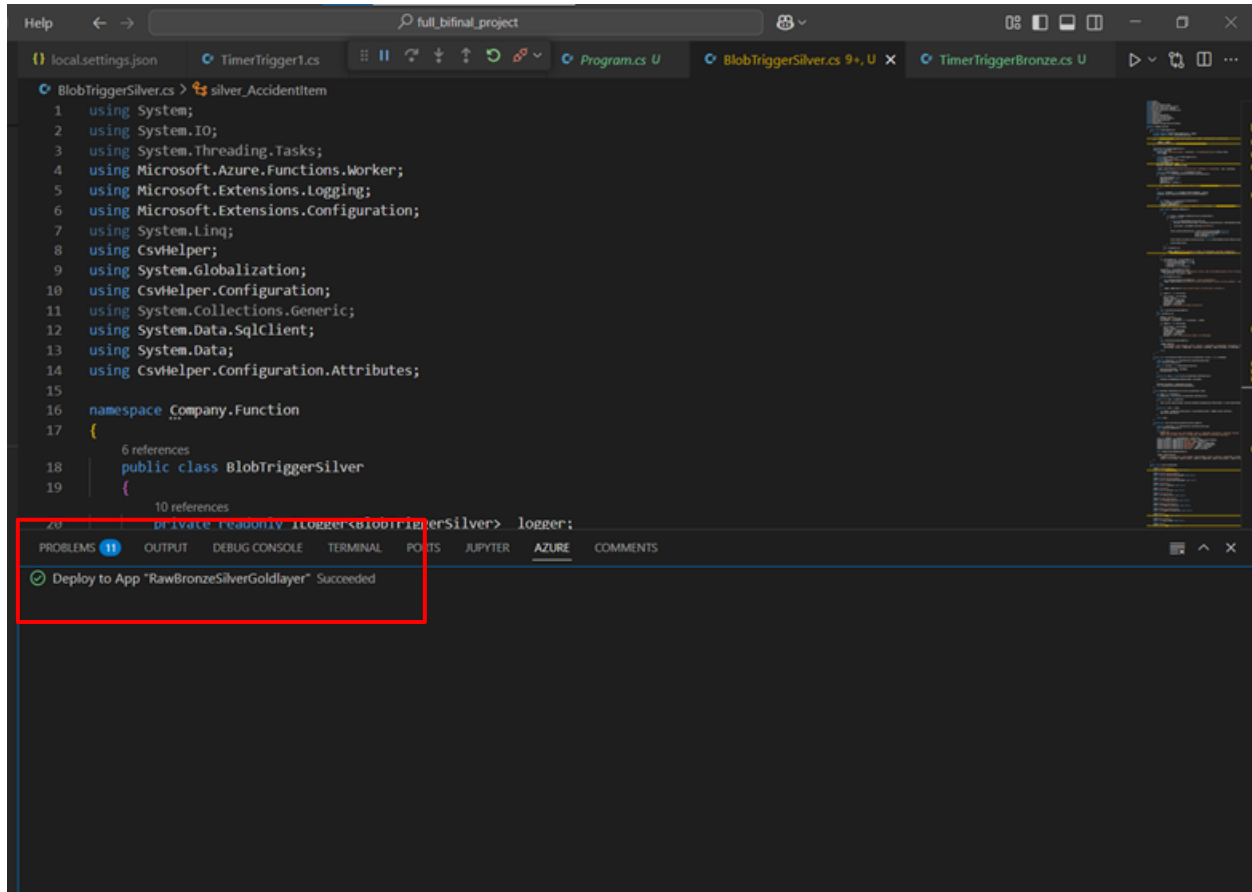


Figure 3.5. Result of deploying functions to Azure Portal

The displayed result shows that the three functions created in the VS Code environment have been successfully deployed to the production environment with an enabled status. The ETL process on the cloud computing platform is carried out automatically thanks to the characteristics established for each function.

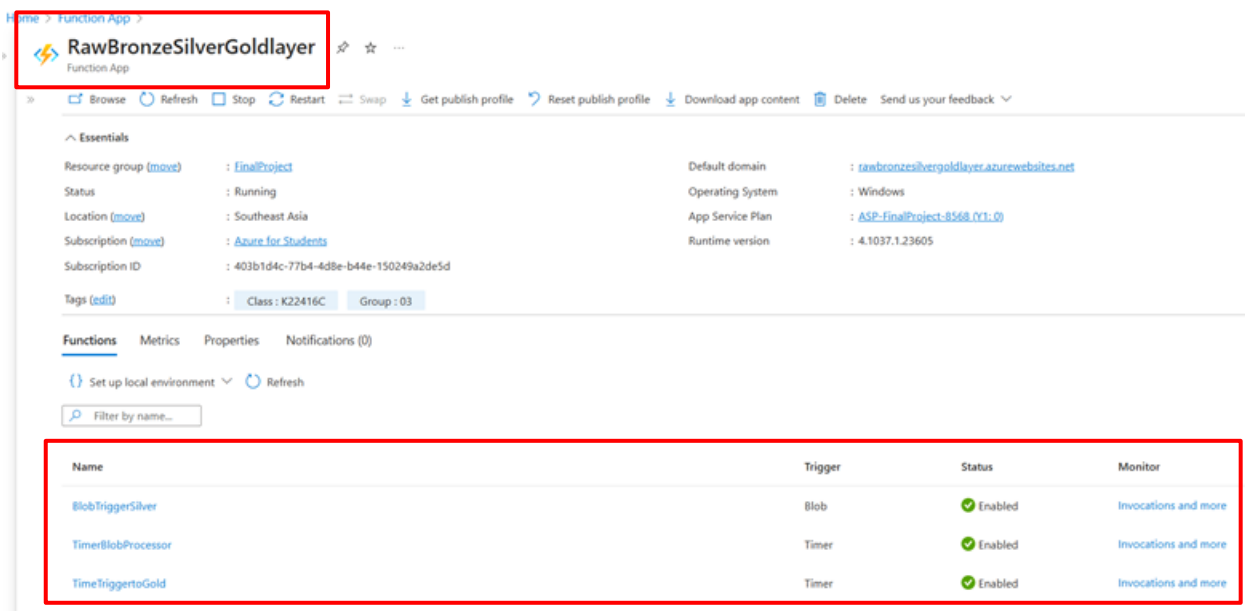


Figure 3.6. List of functions in VS Code deployed via the Function App RawBronzeSilverGoldlayer

Next, the author group will delve into details and specifically explain each function corresponding to the data ingestion process for each layer in the Azure SQL database.

3.2. Bronze layer data ingestion

3.2.1. Raw-to-bronze data pipeline design

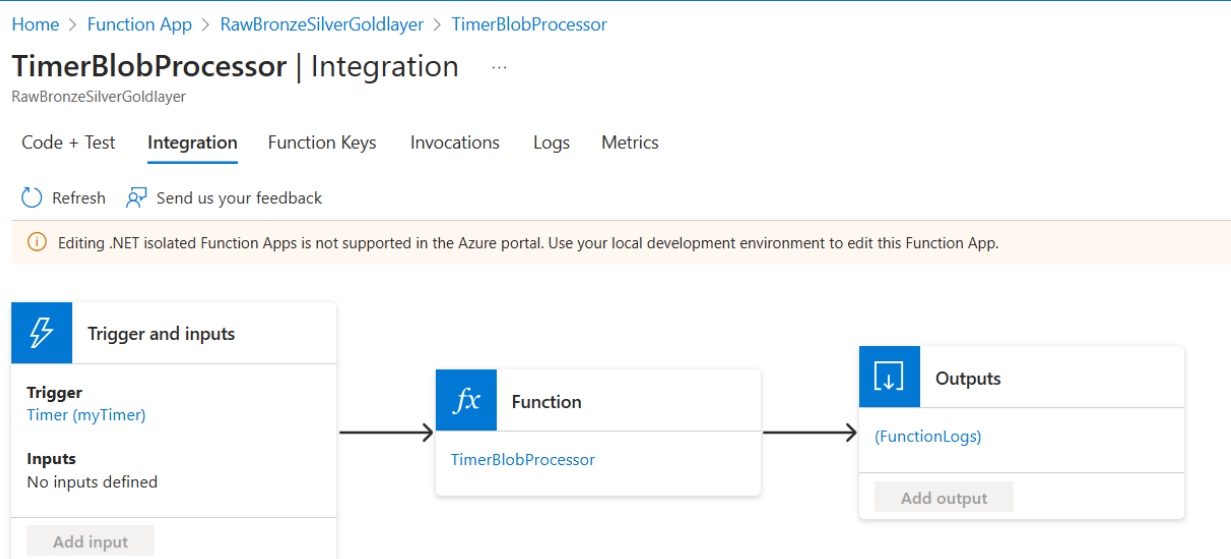


Figure 3.7. Raw to Bronze Pipeline Design

The raw-to-bronze ingestion pipeline, built with C# and deployed on Azure Functions, automates the processing of yearly CSV data (2010-2015) from the bidssfinal Azure Storage container into a SQL Azure Database, establishing the Bronze data layer. Its primary objective is to ensure automatic and accurate data loading, providing a foundational raw database for subsequent analysis. A crucial aspect is the use of blob metadata: once a file (blob) is processed, it's tagged with a "Processed" key and a timestamp. This mechanism prevents duplicate processing on subsequent runs by skipping already tagged blobs, thereby enhancing performance and ensuring the consistency of the Bronze layer.

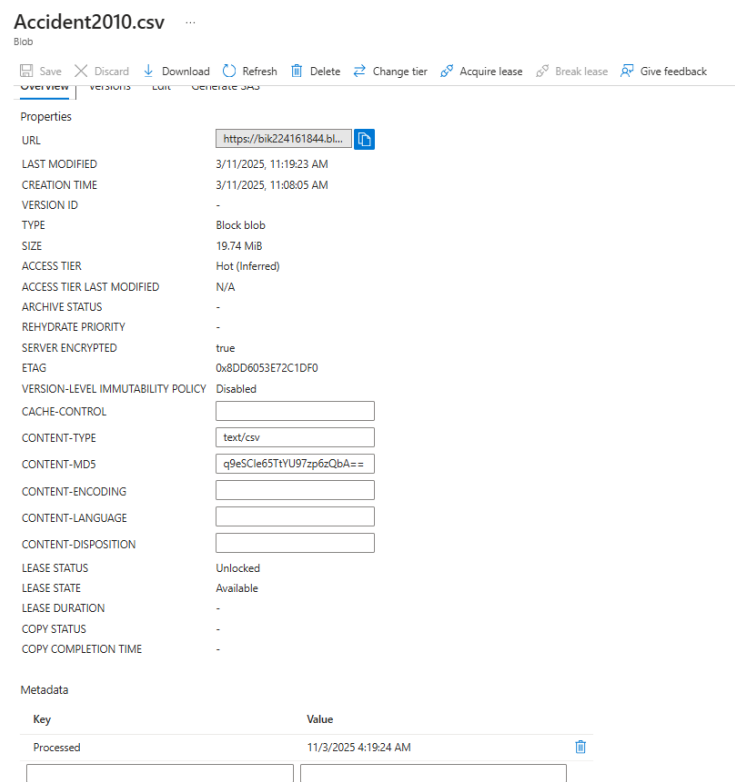


Figure 3.8. Metadata “Processed”

For CSV processing, the authors employ the CsvHelper library to read data from blobs, mapping each row to an AccidentItem object using AccidentItemMap for flexible data type handling and error management. To insert data into the SQL database, SqlBulkCopy is utilized, first converting the list of AccidentItem objects into a DataTable. SqlBulkCopy is configured with a BatchSize of 10,000 records, a 300-second Timeout, and a retry

mechanism (3 attempts with 5-second intervals on timeout) to enhance reliability. The pipeline implements a full-load strategy, deleting and recreating the SQL table in the bronze schema with each run. This approach, optimized by SqlBulkCopy's efficiency, enables data loading in approximately one minute, proving to be a time and resource-efficient method.

3.2.2. Schedule data updates

To automate the periodic processing of continuously updated data, the authors implemented Azure Functions' Timer Trigger. After analyzing business requirements for frequent UK traffic accident data updates and system performance, an optimal schedule was set for the pipeline to run at 1:00 AM every Monday, configured via the CRON expression "00 1 * * 1". The TimerInfo parameter logs activation times, enabling execution history tracking. This setup ensures continuous, unattended data processing from Azure Blob Storage to the Bronze layer, crucial for timely, data-driven decision-making in systems like traffic accident analysis. While Timer Trigger has limitations such as fixed frequency, it provides significant automation, reliability, and scalability, with potential for further optimization through adjustable frequencies and dynamic data handling..

3.2.3. Send an error report email

The pipeline is integrated with SendGrid, a cloud-based email delivery service, to send notifications via email in case of errors or upon successful completion of data processing. SendGrid serves as a monitoring tool, enabling administrators to efficiently track the pipeline's status.

SendGrid is integrated into the pipeline through the SendGrid library in the .NET environment, with its logic embedded in the Run function.

SendGrid Configuration:

- Create an environment variable for SendGridApiKey, which is retrieved from the environment variables defined in the Application Settings of Azure Functions.

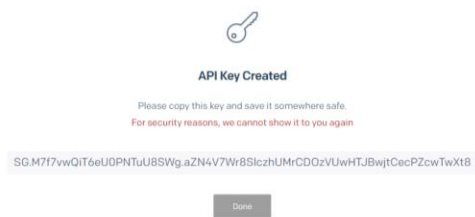


Figure 3.9. Sendgrid API key

- In case of an error (e.g., CSV parsing failure), an email is sent containing detailed error information:

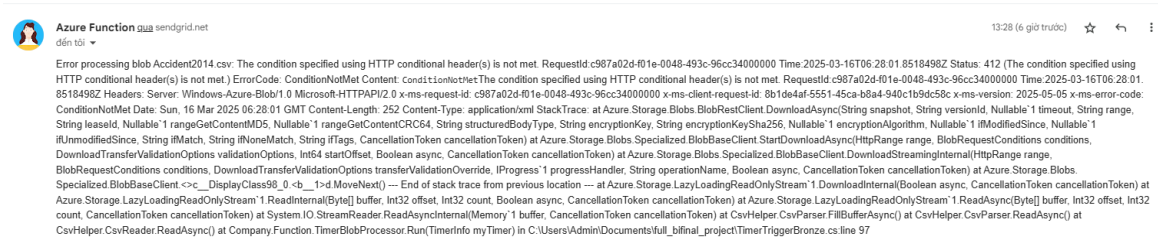


Figure 3.10. Sendgrid error

- Upon successful pipeline execution, a confirmation email is sent with details on the number of records processed.



Figure 3.11. Sendgrid loading successfully

Despite limitations such as network connectivity issues and email quota restrictions, SendGrid remains a reliable and flexible solution. It helps ensure smooth pipeline operation and enables decision-makers to respond promptly to any issues that arise during pipeline execution.

3.3. Silver layer data ingestion

The data in Azure Blob Storage and the data in the bronze layer share similar characteristics and significance. Therefore, during the process of moving data into the silver layer, the source will directly point to Azure Blob Storage—where the raw data is stored. Based on the evaluations and EDA (Exploratory Data Analysis), during the ingestion of data into the silver layer, the authors perform several data transformation operations to ensure that the

data in the silver layer is complete, consistent, properly structured, and clean. The Azure Function responsible for ingesting data into the silver layer is BlobTriggerSilver, and the data ingestion process into the silver layer will consist of three distinct and prominent segments.

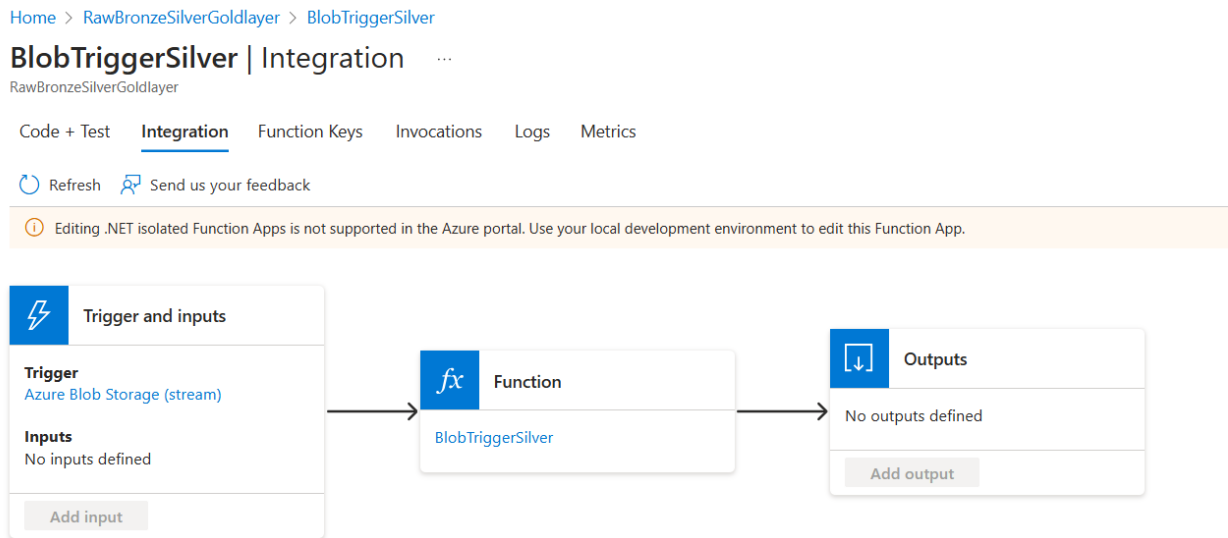


Figure 3.12. Design a pipeline to ingest data from the storage account into the silver layer.

The Azure Blob Storage Trigger is designed to activate the BlobTriggerSilver function. The trigger detects new files uploaded to the container in Blob Storage. Each time a new file is uploaded, the BlobTriggerSilver function initiates the data ingestion process. Here, the trigger acts equivalently to an input: whenever a CSV file is uploaded to the “bidssfinal” container, it serves as the input for the function.

Once activated by the trigger, the BlobTriggerSilver function identifies the parameter findings—the CSV file name—and the connection, which is the connection string to Blob Storage. Instead of passing the entire content of the CSV file as a string or byte array, the BlobTriggerSilver function passes the data as a Stream object. Similar to the data ingestion process for the bronze layer, CsvReader is used to read data from the CSV file, which is then mapped to the predefined AccidentItem class.

Valid data is recorded into the silver.accident1015 table within the silver layer. Unlike the storage method in the bronze layer, data in the silver layer is ingested into a single table using BulkInsertToSql. This is a method for bulk inserting data into an Azure SQL database. A connection is established between SqlBulkCopy and the silver.accident1015 table, followed by mapping columns from the AccidentItem class to the target table. This produces a list of records containing all objects from AccidentItem after being read from the CSV. Finally, this list of records is converted into a DataTable. Converting it into a DataTable allows SQL to recognize the format and successfully copy the data into the Azure SQL database.

>>

Query 1 × Query 2 ×

Run Cancel query Save query Export data as Show all Open Copilot

Results Messages

Search to filter items...

| Accident_Index | Location_Easting_OSGR | Location_Northing_OSGR | Longitude | Latitude | Police_Force |
|----------------|-----------------------|------------------------|-----------|-----------|--------------|
| 200501BS00001 | 525680 | 178240 | -0.19117 | 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 | 51.51554 | 1 |
| 200501BS00007 | 524220 | 180830 | -0.211277 | 51.512695 | 1 |
| 200501BS00009 | 525890 | 179710 | -0.187623 | 51.50226 | 1 |
| 200501BS00010 | 527350 | 177650 | -0.167342 | 51.48342 | 1 |
| 200501BS00011 | 524550 | 180810 | -0.206531 | 51.512443 | 1 |
| 200501BS00012 | 526240 | 178900 | -0.182872 | 51.494902 | 1 |
| 200501BS00014 | 526170 | 177690 | -0.184312 | 51.484044 | 1 |

Figure 3.13. The data in the silver layer within the SQL database

The data, after being cleaned and transformed, has been recorded in the SQL database. Compared to the original data, the data in the silver layer has undergone certain changes. Specifically:

- The "Date" column has been standardized to the format yyyy-MM-dd.
- Two new columns have been added: "LSOA_of_Accident_Location_missing" to flag missing values in the LSOA_of_Accident_Location column, and "Location_Data_Missing" to indicate if any of the four location-related columns—

Location_Easting_OSGR, Location_Northing_OSGR, Longitude, or Latitude—contain null values.

- Rows where all four columns—Location_Easting_OSGR, Location_Northing_OSGR, Longitude, and Latitude—are simultaneously missing have been removed. These are identified as accidents with undetermined locations, which are deemed meaningless for long-term analysis, prompting the team to eliminate them.

3.4. Gold layer data ingestion

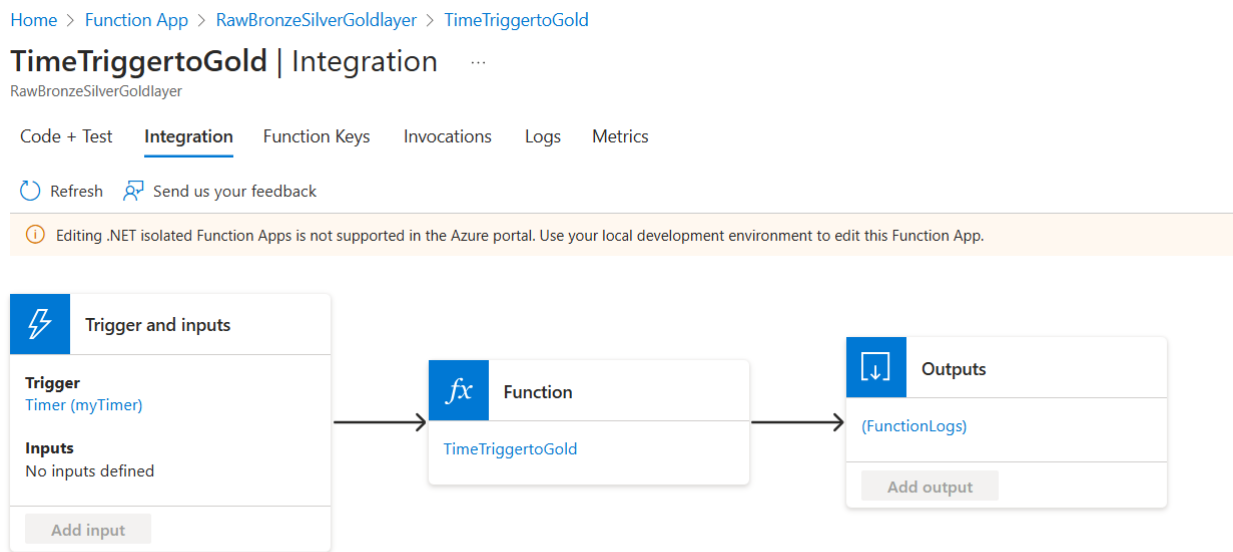


Figure 3.14. Design a pipeline to ingest data from the silver layer into gold layer

The Silver-to-Gold layer ingestion process transforms and structures data into a Star Schema for optimized analysis and reporting, automated by an Azure Function (TimeTriggertoGold) with a Timer Trigger. Data from the Silver layer's accident1015 table is categorized into multiple Dimension tables (e.g., Dim_AccidentSeverity, Dim_LightConditions) and a central Fact_Accidents table. A key technique is SCD Type 2), which preserves historical changes in Dimension tables by inactivating old records (setting End_Date, Status=0) and inserting new active ones (Start_Date, Status=1, End_Date=NULL). The Azure Function, scheduled via a CRON expression, performs

ETL: extracting data from Silver, updating Dimensions using SCD Type 2, removing outdated records from the Fact table, and loading new, linked data into the Fact table. Performance is enhanced by bulk inserts, and error logging facilitates monitoring. This successfully creates a Gold layer with historically accurate Dimension tables and a current Fact table, ready for efficient querying..

Fact_Accidents Table

The Fact_Accidents table has been refreshed to include:

- The latest accident data, ensuring reports and dashboards reflect up-to-date statistics.
- Accurate references to Dimension tables, preserving the integrity of relationships within the Star Schema.
- A structured format optimized for Power BI visualization, enhancing analytical efficiency.

3.5. Data Governance

The logging system in all three Azure Functions is designed to track the entire execution process, from start to finish, including both successful and failed cases. All three functions utilize a basic FunctionLog structure, but the implementation and log details vary depending on the purpose of each function.

- Real-time logging: The ILogger is used to record logs instantly (info, warning, error) during processing, enabling progress tracking and immediate issue detection.
- Structured log storage: FunctionLog entries are created and stored in the `function_logs` table in the Azure SQL database, ensuring long-term retention of activity history for later analysis.
- Error handling: When exceptions occur, the log captures detailed error information.
- Log formatting and return: Logs are serialized in JSON format and returned in the function's output, facilitating integration with other systems if needed.

The logging system in the three Azure Functions—BlobTriggerSilver, TimeTriggertoGold, and TimerBlobProcessor—employs the FunctionLog class with a consistent structure, including columns such as FunctionName, Status, TriggeredBy, RecordCount, Timestamp, and Message, ensuring uniformly formatted log information. The ILogger feature logs in real time with different levels, such as LogInformation for general information, LogWarning for warnings, and LogError for errors, allowing for immediate progress monitoring and issue detection. All logs from the three functions are recorded in the function_logs table.

A key difference lies in the logging approach: BlobTriggerSilver records logs by directly inserting them via InsertFunctionLog, whereas TimeTriggertoGold and TimerBlobProcessor perform indirect insertion through SqlOutput. Consequently, BlobTriggerSilver does not return an output, while TimeTriggertoGold and TimerBlobProcessor return outputs in the form of function logs via gold_OutputType and OutputType, respectively. Effective log management across all three functions includes detailed error-handling mechanisms, capturing exception details and failure statuses in the logs, which supports efficient troubleshooting and analysis. These features establish a robust and consistent logging foundation throughout the data processing stages.

| Id | FunctionName | Status | TriggeredBy | RecordCount | Timestamp | Message |
|-----|--------------------|---------|--------------|-------------|-----------------------------|---|
| 340 | TimerBlobProcessor | Failed | TimerTrigger | 0 | 2025-03-16T05:52:04.0000000 | Error processing blob Accident2012.csv: The condition specified using HTTP conditional header(s) is r |
| 341 | TimerBlobProcessor | Failed | TimerTrigger | 0 | 2025-03-16T05:52:02.0000000 | Error processing blob Accident2009.csv: The condition specified using HTTP conditional header(s) is r |
| 342 | BlobTriggerSilver | Failed | BlobTrigger | 145571 | 2025-03-16T05:52:00.7330000 | Error processing blob Accident2012.csv: Violation of PRIMARY KEY constraint 'PK_accident1015': Cani |
| 339 | TimerBlobProcessor | Success | TimerTrigger | 0 | 2025-03-16T05:52:00.0000000 | Processed 0 blobs successfully |
| 338 | BlobTriggerSilver | Failed | BlobTrigger | 145571 | 2025-03-16T05:51:53.5430000 | Error processing blob Accident2012.csv: Violation of PRIMARY KEY constraint 'PK_accident1015': Cani |
| 337 | BlobTriggerSilver | Failed | BlobTrigger | 145571 | 2025-03-16T05:51:38.0630000 | Error processing blob Accident2012.csv: Violation of PRIMARY KEY constraint 'PK_accident1015': Cani |
| 336 | BlobTriggerSilver | Success | BlobTrigger | 163554 | 2025-03-16T05:51:28.2700000 | Processed blob Accident2009.csv successfully |
| 335 | TimerBlobProcessor | Success | TimerTrigger | 0 | 2025-03-16T05:50:00.0000000 | Processed 0 blobs successfully |
| 334 | TimeTriggertoGold | Success | TimerTrigger | 1075136 | 2025-03-16T05:48:00.0000000 | Processed 1075136 records successfully |
| 332 | TimerBlobProcessor | Success | TimerTrigger | 0 | 2025-03-16T05:48:00.0000000 | Processed 0 blobs successfully |
| 333 | TimeTriggertoGold | Success | TimerTrigger | 1075136 | 2025-03-16T05:47:33.0000000 | Processed 1075136 records successfully |
| 331 | TimerBlobProcessor | Success | TimerTrigger | 0 | 2025-03-16T05:47:14.0000000 | Processed 0 blobs successfully |

Figure 3.15. Table function_logs recording logs in SQL database

The recorded logs include both successful and failed cases, along with related information such as the number of records, execution time, and returned messages from all three functions, stored in the `function_logs` table in the SQL database.

3.6. Load data using Power Automate

The authors utilize a pre-built template provided by Power Automate. This template supports adding rows to the dataset used in Power BI whenever a new entry is created in SQL Server.

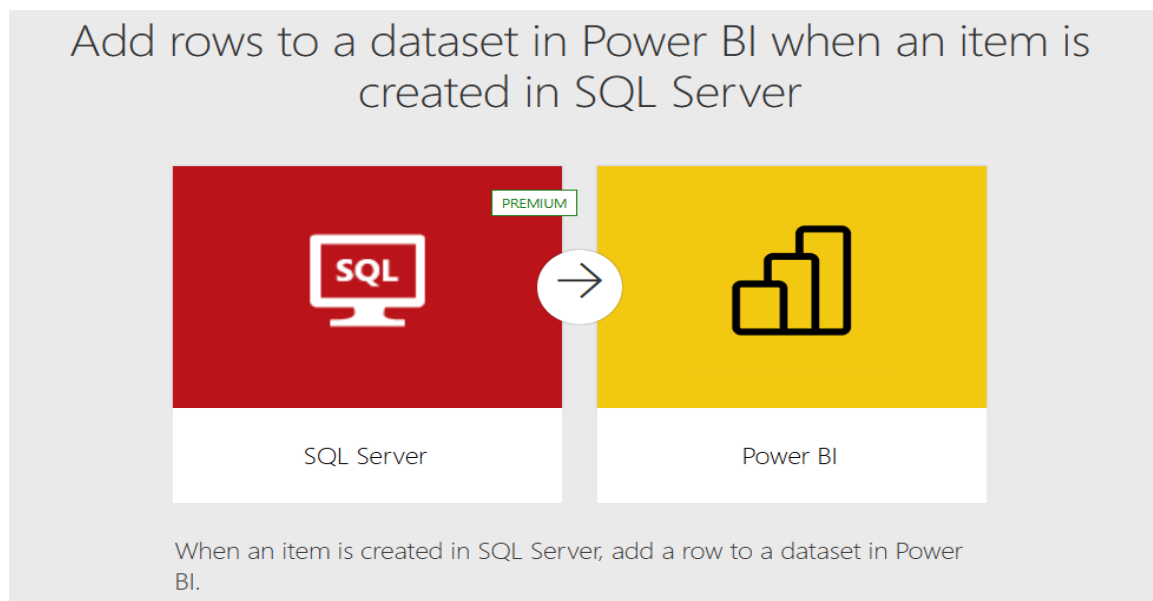


Figure 3.16. Add rows to a dataset in PowerBI when an item is created in SQL Server

In this setup:

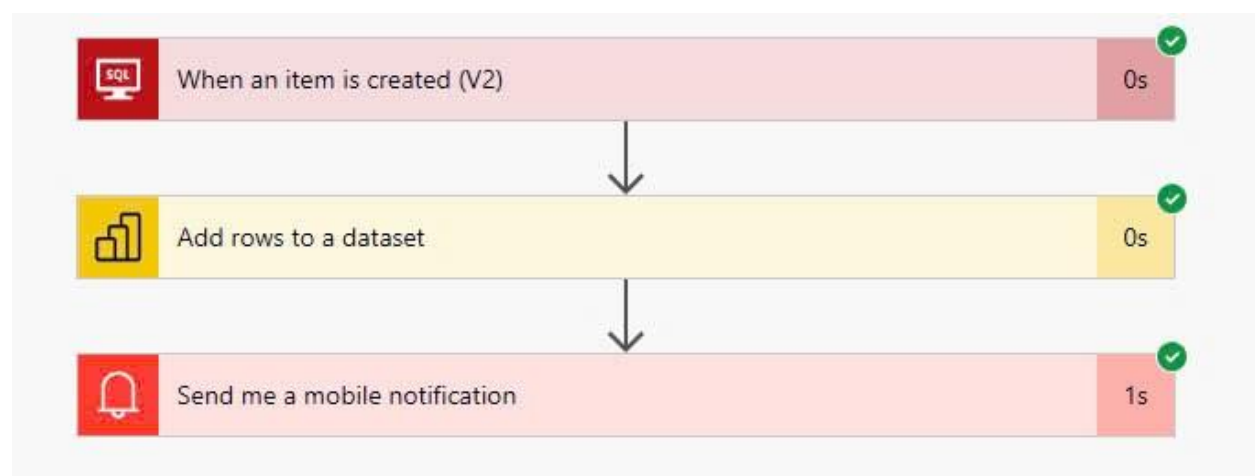


Figure 3. 1: Power Automate flow

- The "When an item is created in SQL Server" component connects directly to the data lake and links to tables in the Gold layer, which includes Dim and Fact tables. Whenever a new record is detected in the Gold layer, the pipeline is triggered.
- The "Add rows to a dataset in Power BI" component directly connects to the dataset used for dashboard visualization in Power BI. When a change is detected in the SQL Server data source, this component is activated by mapping the corresponding columns from the SQL Database to the Power BI dataset.



Figure 3.17. Upload Dataset Successfully in Power Automate platform

- Finally, a mobile notification feature is integrated to provide real-time alerts whenever the pipeline runs, ensuring prompt awareness of updates or potential issues.

CHAPTER 4. DATA VISUALIZATION

4.1. Data Modeling

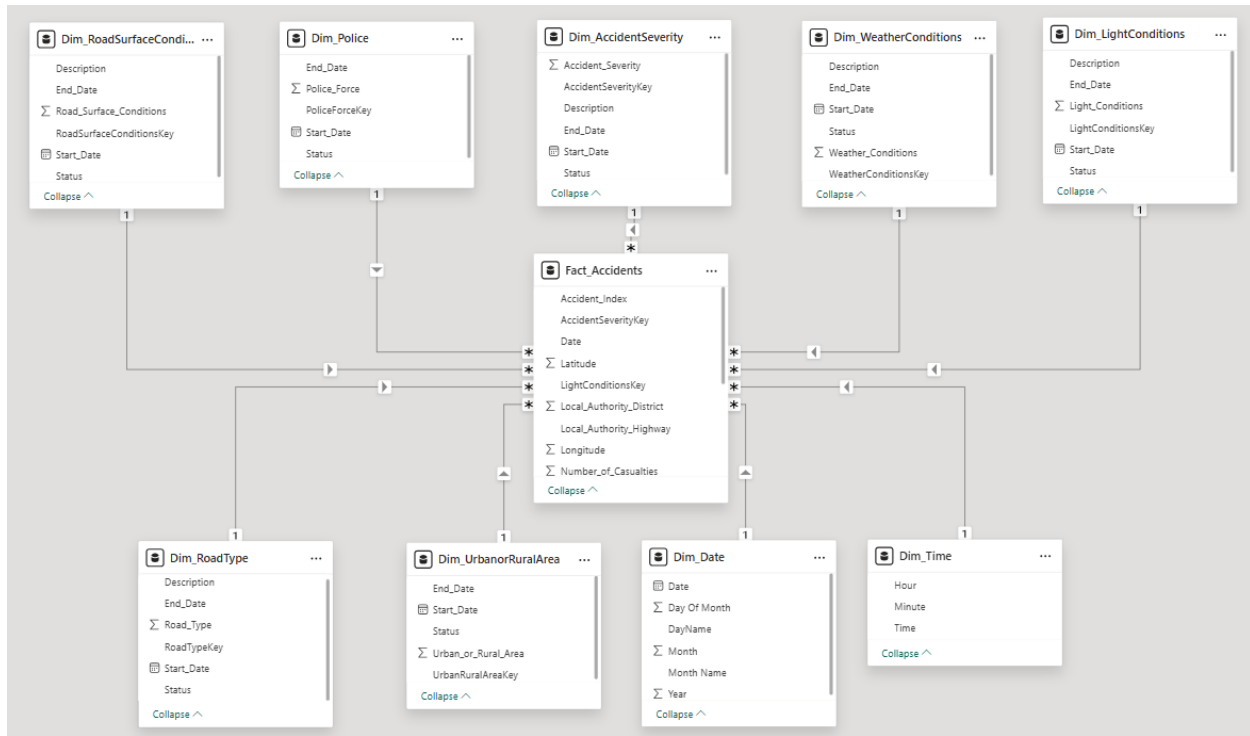























Figure 4.1. Data model

Based on the information from section 3.3.5 on Gold layer data ingestion, which defines the relationship between dimensional and fact tables, the data modeling in Power BI has been identified and structured accordingly. Using a star schema architecture, the **Fact_Accidents** table serves as the central table storing events and linking to dimension tables. All dimension tables have a one-to-many relationship with the fact table. In addition, our team also created **Dim_Date** and **Dim_Time** by M language in Power Query tables to enhance time-based analysis and visualization. **Dim_Date** includes time - related data such as **Date**, **Day of Month**, **DateName**, **Month**, **MonthName**, **Year**, covering the period from January 1 to December 31, 2015. **Dim_Time** provides more granular time details, including hours, minutes, and seconds within a day, enabling precise tracking of accident occurrences throughout the day.

4.2. Dashboard Visualization

4.2.1. KPIs

Table 4.1. KPIs trend through time

| Year | KPIs | | | |
|------|---|---|---|-----------------|
| All |  876470 |  18194 |  38.48 | SAR Rate 15% |
| 2010 |  154414 |  3256 |  38.88 | SAR Rate 14% |
| 2011 |  151474 |  3314 |  38.54 | SAR Rate 15% |
| 2012 |  145571 |  2938 |  38.50 | SAR Rate 15% |
| 2013 |  138660 |  2948 |  38.53 | SAR Rate 15% |
| 2014 |  146322 |  2898 |  38.24 | SAR Rate 15% |
| 2015 |  140029 |  2840 |  38.18 | SAR Rate 15% |

4.2.2. Page 1 - Accident Overview

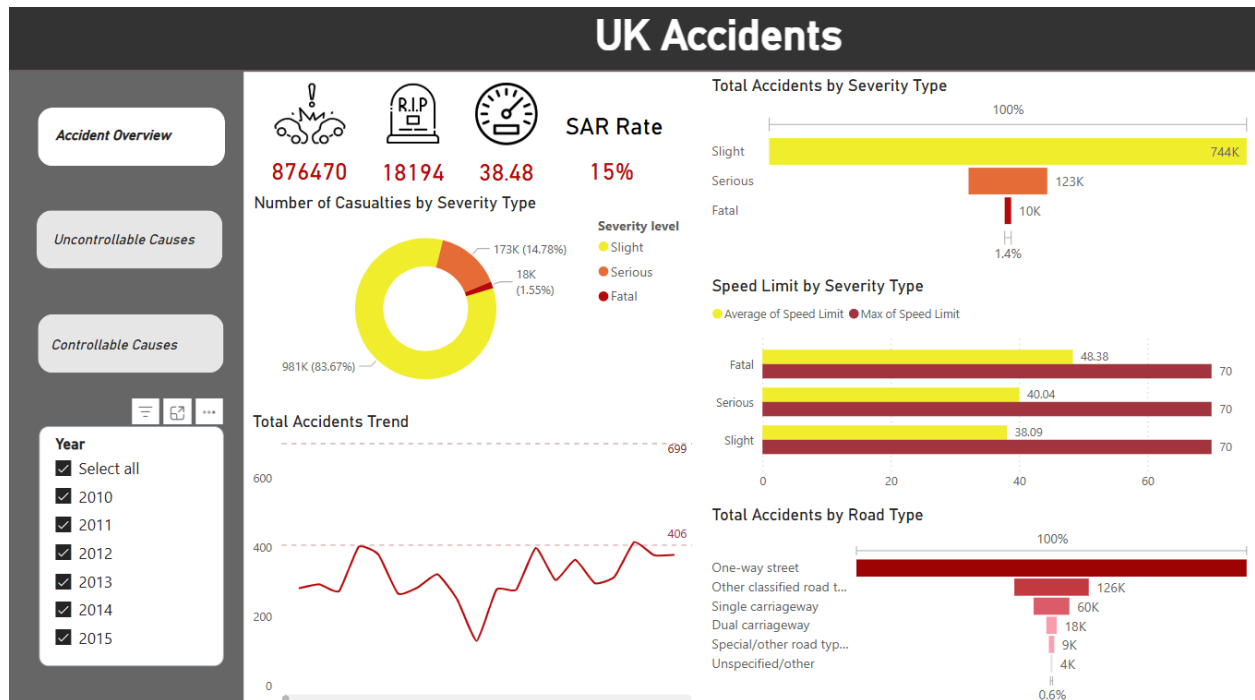


Figure 4.2. Accident Overview Dashboard

a) Is the total number of accidents increasing or decreasing over time? .

- Total Accident Trend (2010 - 2015)

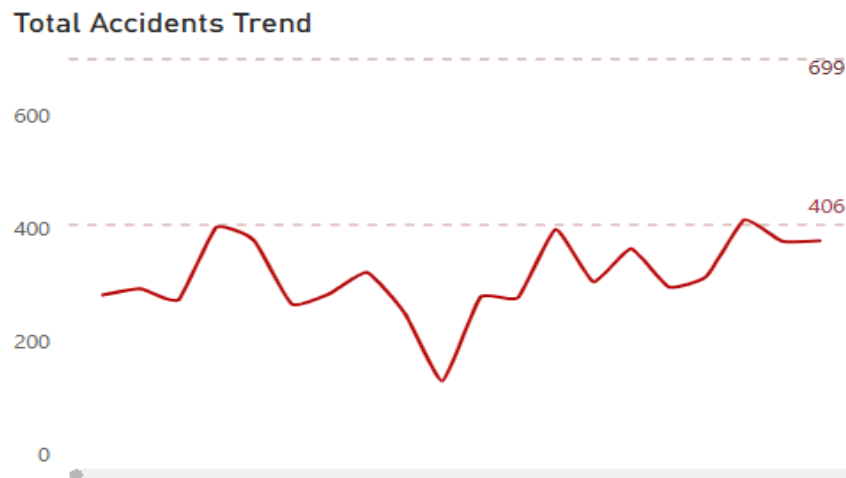


Figure 4.3. Total Accident Trend (2010 - 2015)

- Total Accident Trend (2010)

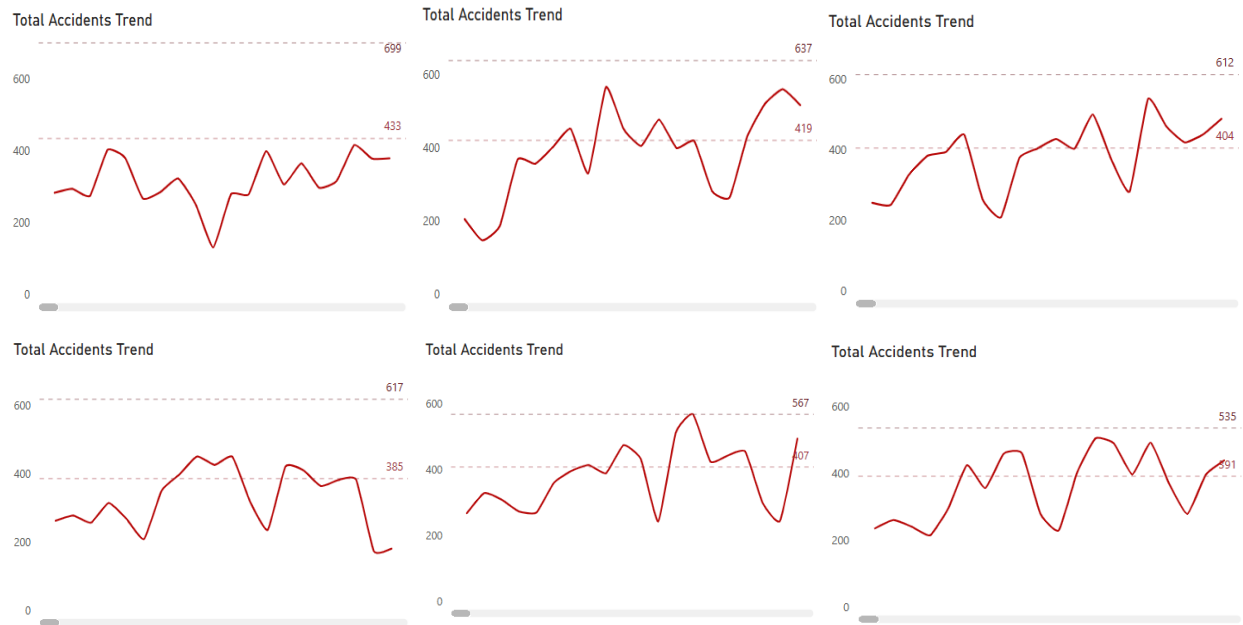


Figure 4.4. Detail total accident trend over years

b) What is the distribution of accidents by severity (slight, serious, fatal)?

Total Accidents by Severity Type

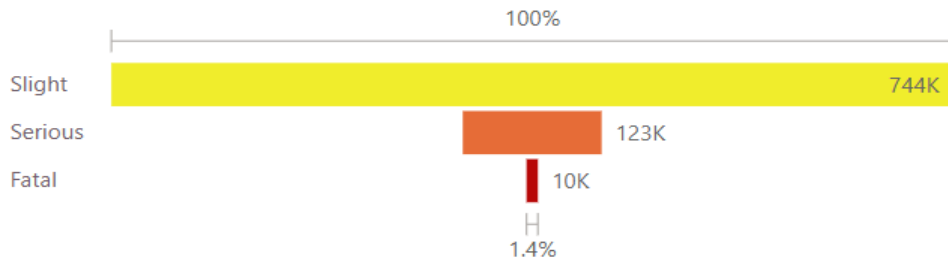


Figure 4.5. Total Accidents by Severity Type

c) How many casualties occur at each severity level?

Number of Casualties by Severity Type

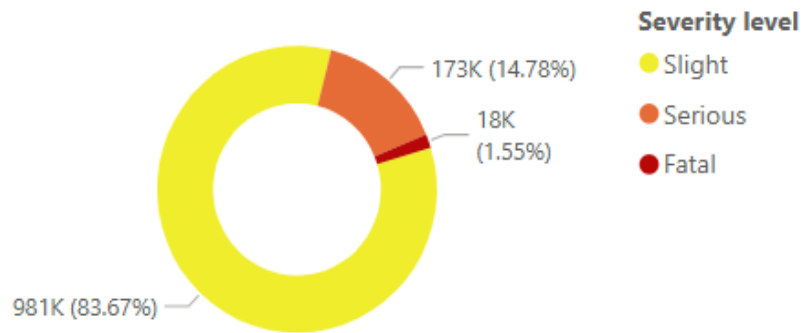


Figure 4.6. Number of Casualties by Severity Type

d) How does speed limit correlate with accident severity ?

Speed Limit by Severity Type

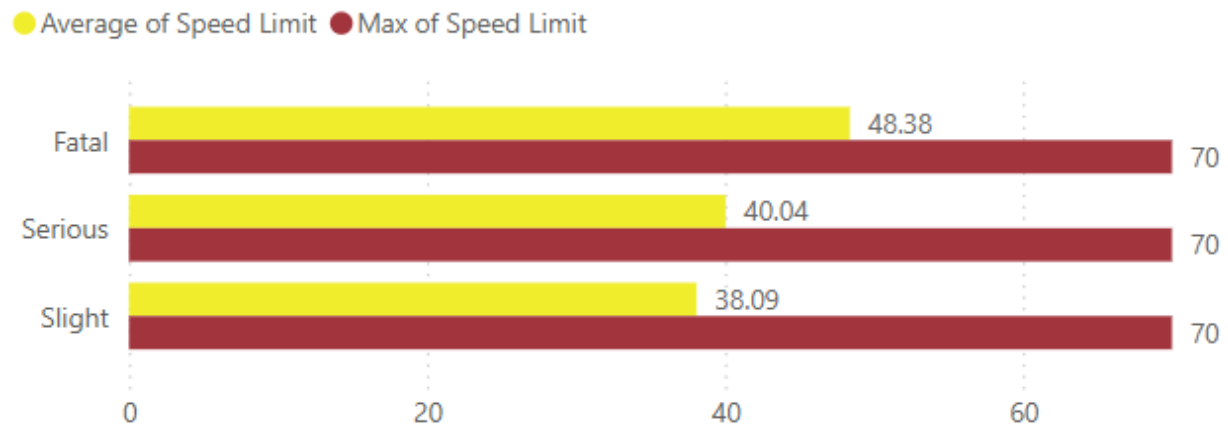


Figure 4.7. Speed Limit by Severity Type

e) Which road types have the highest number of accidents?

Total Accidents by Road Type

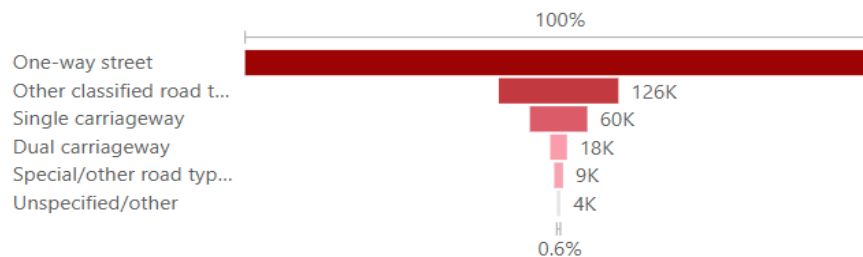


Figure 4.8. Total accident by road type

4.2.3. Page 2 - Uncontrollable Causes

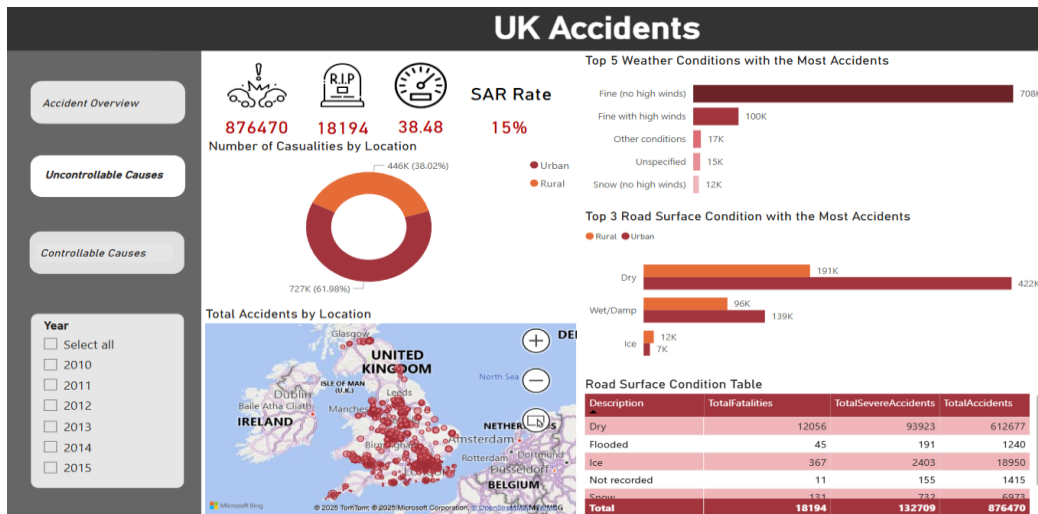


Figure 4.9. Uncontrollable Causes

a) Which areas have the highest accident rates?



Figure 4.10. Map displaying the total number of accidents by location in the United Kingdom

b) How do casualty numbers compare between urban and rural areas?

Number of Casualties by Location

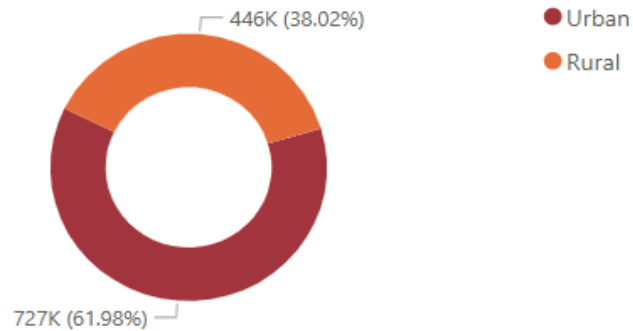


Figure 4.11. Number of Casualties by Location

c) What weather conditions contribute to the highest number of accidents?

Top 5 Weather Conditions with the Most Accidents

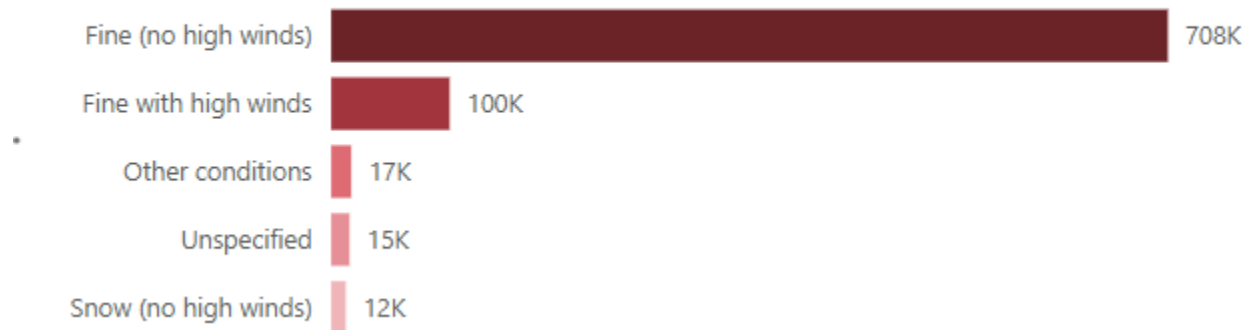


Figure 4.12. Top 5 Weather Conditions with the Most Accidents

d) How do different road surface conditions impact accident frequency?

| < Back to report | | ROAD SURFACE CONDITION TABLE | |
|-------------------------------------|-----------------|------------------------------|----------------|
| Description | TotalFatalities | TotalSevereAccidents | TotalAccidents |
| Dry | 12056 | 93923 | 612677 |
| Flooded | 45 | 191 | 1240 |
| Ice | 367 | 2403 | 18950 |
| Not recorded | 11 | 155 | 1415 |
| Snow | 131 | 732 | 6973 |
| Wet/Damp | 5584 | 35305 | 235215 |
| Total | 18194 | 132709 | 876470 |

Figure 4.13. Road Surface Condition Table

e) Which road surface conditions are associated with the most severe accidents?

Top 3 Road Surface Condition with the Most Accidents

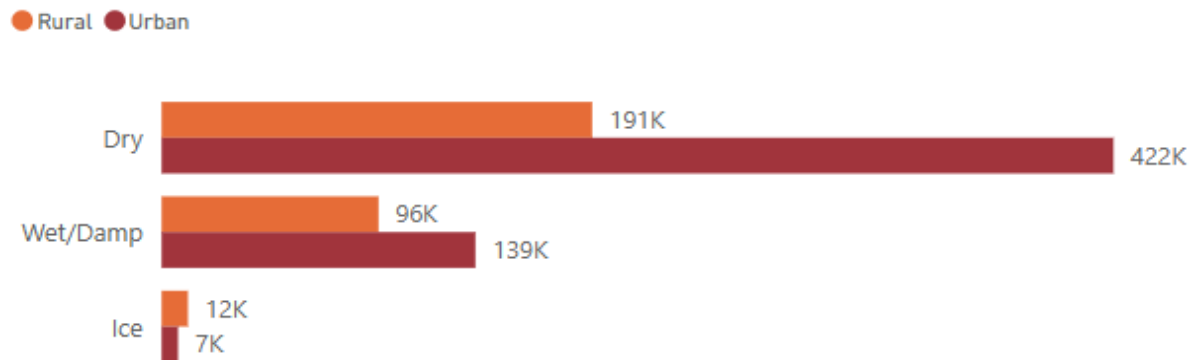


Figure 4.14. Top 3 Road Surface Conditions with the Most Accidents

4.2.4. Page 3 - Controllable Causes

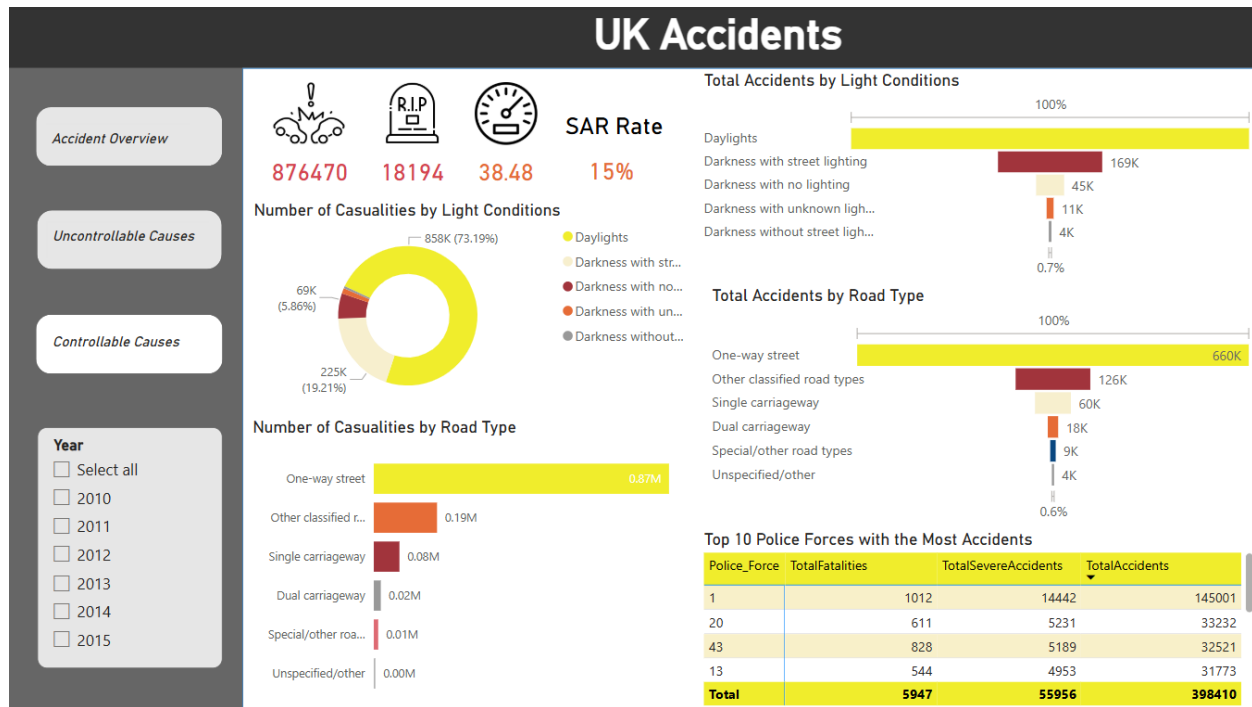


Figure 4.15. Controllable Causes Dashboard

a) How do light conditions impact the number of accidents?

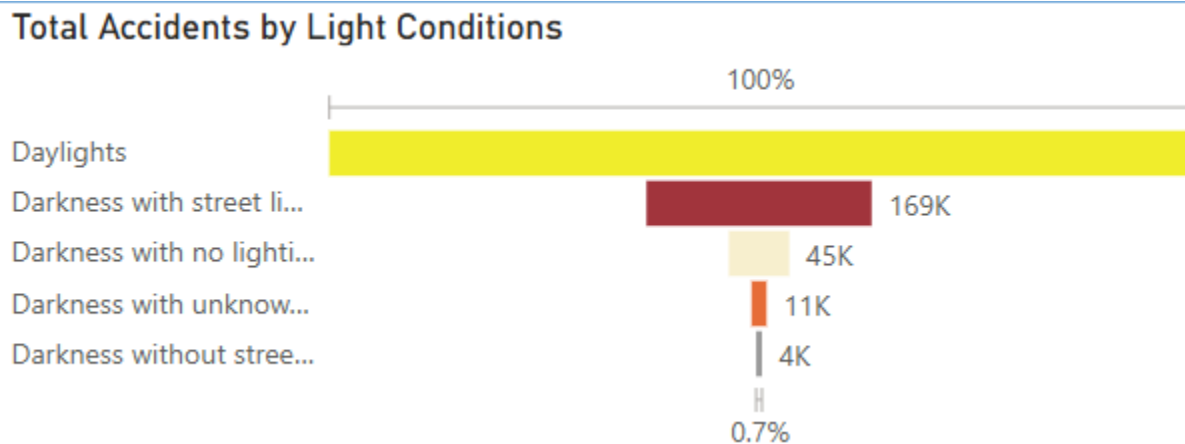


Figure 4.16. Total Accidents by Light Conditions

b) Which light conditions are associated with the highest number of casualties?

Number of Casualties by Light Conditions

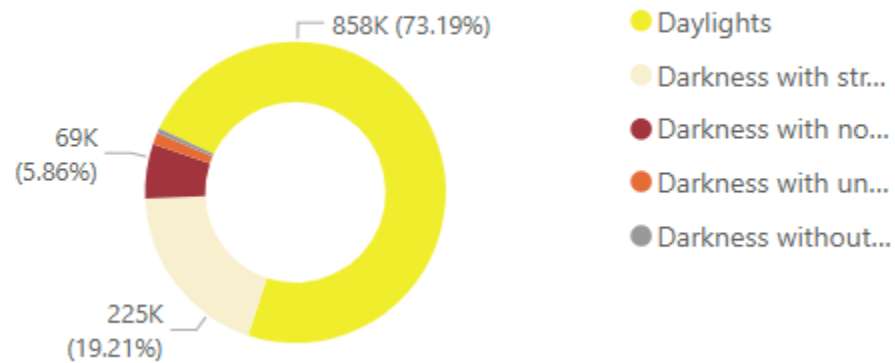


Figure 4.17. Number of Casualties by Light Conditions

c) Which road types have the highest number of accidents?

Total Accidents by Road Type

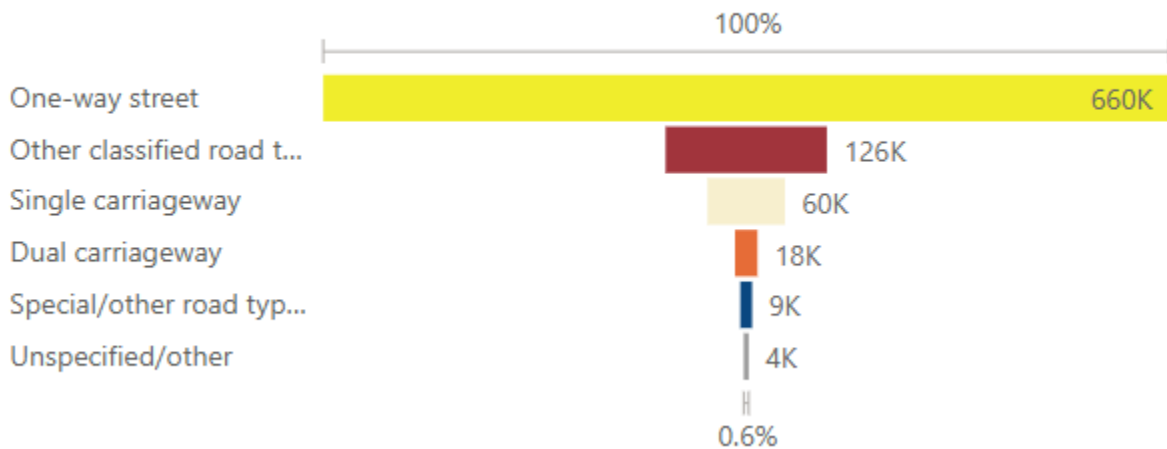


Figure 4.18. Total Accidents by Road Type

d) Which road types are associated with the most casualties?

Number of Casualties by Road Type

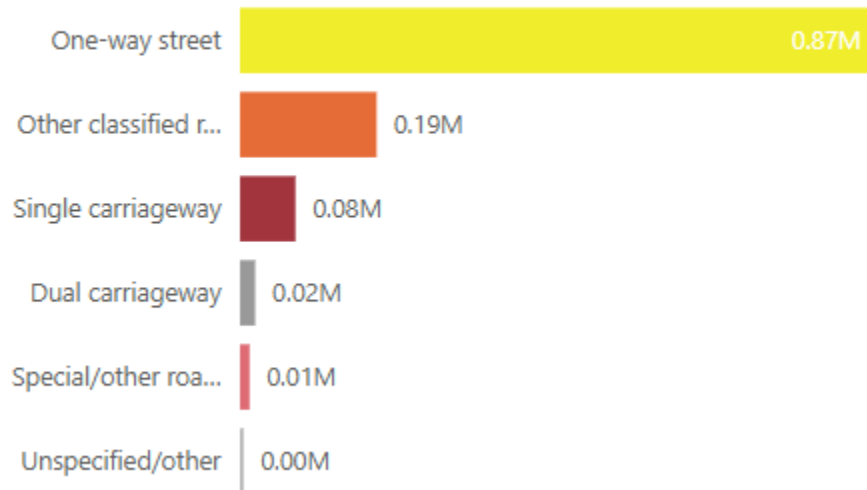


Figure 4.19. Number of Casualties by Road Type

e) Which police forces handle the most accidents?

| Back to report | | TOP 10 POLICE FORCES WITH THE MOST ACCIDENTS | | |
|--------------------------------|-----------------|--|----------------|--|
| Police_Force | TotalFatalities | TotalSevereAccidents | TotalAccidents | |
| 1 | 1012 | 14442 | 145001 | |
| 20 | 611 | 5231 | 33232 | |
| 43 | 828 | 5189 | 32521 | |
| 13 | 544 | 4953 | 31773 | |
| 46 | 548 | 3448 | 29511 | |
| 44 | 428 | 5862 | 27917 | |
| 6 | 470 | 3963 | 25887 | |
| 47 | 552 | 5044 | 24482 | |
| 50 | 540 | 3427 | 24292 | |
| 4 | 414 | 4397 | 23794 | |
| Total | 5947 | 55956 | 398410 | |

Figure 4.20. Top 10 Police Forces with the most accidents

