# DAMG 7370

## Designing Advance Data Architecture for Business Intelligence

### End Term Report

Chinmay Deshpande 002859266

Shreya Thakur 002818444

Sushil Girish 002817666

# Part 1

**Data profiling y-data profile, Data staging (Staging tables), Talend for ETL jobs, MySQL validations**

**Dimensional model (Target tables), Facts and Dimensions, Mapping document explaining the source column name and where it finally maps to target column, Stage to Target, Document all transformations if any**

## Y-data profiling

### Austin Crash Dataset Statistics

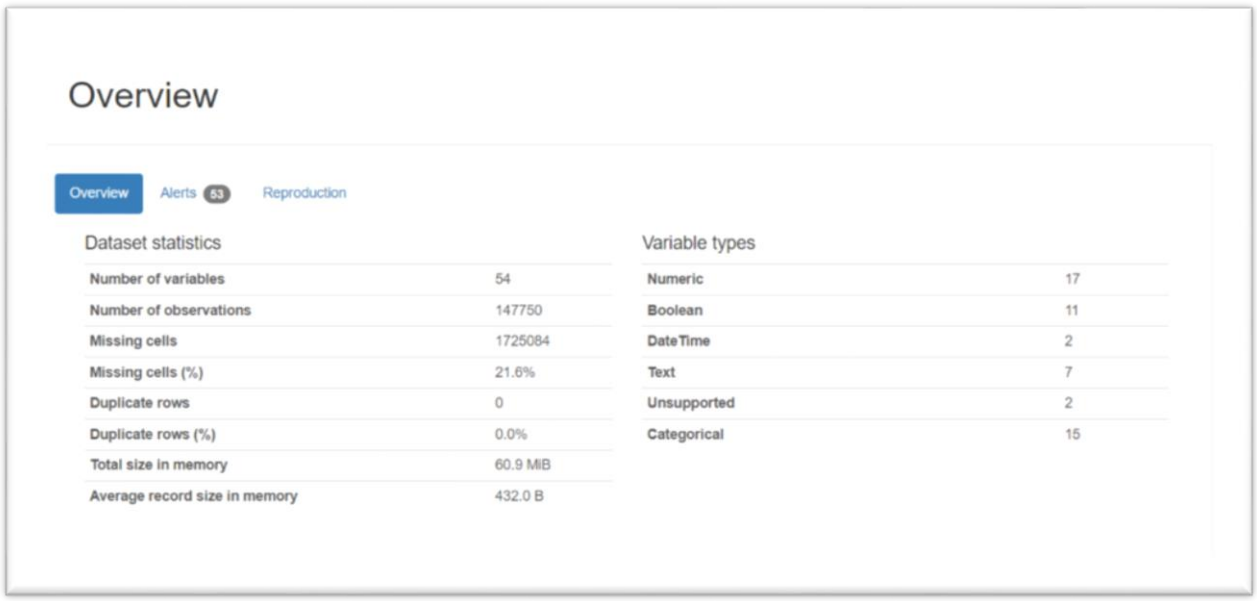
The dataset under analysis refers to traffic collisions in Austin and includes a wide range of variables that provide detailed information about each occurrence, hence facilitating a comprehensive analysis for possible insights. The dataset's significant size and diversity 54 variables distributed over 147,750 observations—provide a strong basis for in-depth study.

**Quality of Data:** One noteworthy feature of this dataset is that 1,725,084 cell or 21.6% of the total are missing. This degree of incompleteness creates difficulties and demands cautious management in order to preserve the accuracy of studies made using the dataset. In spite of this, the dataset is notable for having no duplicate rows, guaranteeing the uniqueness of every record and streamlining the preliminary stages of data preparation.

**Usage of Memory:** The dataset has an average record size of 432.0 B and a memory footprint of 60.9 MiB. These numbers imply that even if the dataset is large, it is still manageable for examination on modern computer systems without requiring specialised data processing tools. With 17 numeric, 11 boolean, 2 datetime, 7 text, 15 categorical, and 2 unsupported types among its variables, the dataset is broad and will require a variety of preprocessing methods to be used in the best possible way for analysis or modelling.

**Potential Repercussions:** The substantial percentage of missing data have an effect on how well machine learning models or statistical studies utilising this dataset perform. It is essential to deal with these missing data via imputation or exclusion, depending on the needs of the study. One benefit is that there aren't any duplicate rows, which means less thorough data cleansing is required. It would be wise to look into the nature of the missing data to see if it is missing randomly or if it is a sign of underlying systemic problems that need to be fixed before moving on with any in-depth research.

**Conclusion:** In conclusion, the dataset's comprehensiveness and lack of duplicates confirm its excellent quality and suitability for additional research, even with the significant quantity of missing data providing obstacles. Once the missing values have been addressed, the dataset may be used for in-depth exploratory analysis, predictive modelling, and other statistical investigations. Because of its small size, it can be easily analysed on basic computing systems, which makes it an invaluable tool for understanding traffic collision trends and developing safety improvement plans.



Overview

Overview | Alerts 53 | Reproduction

Dataset statistics

| | |
|---|---|
| Number of variables | 54 |
| Number of observations | 147750 |
| Missing cells | 1725084 |
| Missing cells (%) | 21.6% |
| Duplicate rows | 0 |
| Duplicate rows (%) | 0.0% |
| Total size in memory | 60.9 MiB |
| Average record size in memory | 432.0 B |

Variable types

| | |
|---|---|
| Numeric | 17 |
| Boolean | 11 |
| DateTime | 2 |
| Text | 7 |
| Unsupported | 2 |
| Categorical | 15 |

## Chicago Crash Dataset Statistics

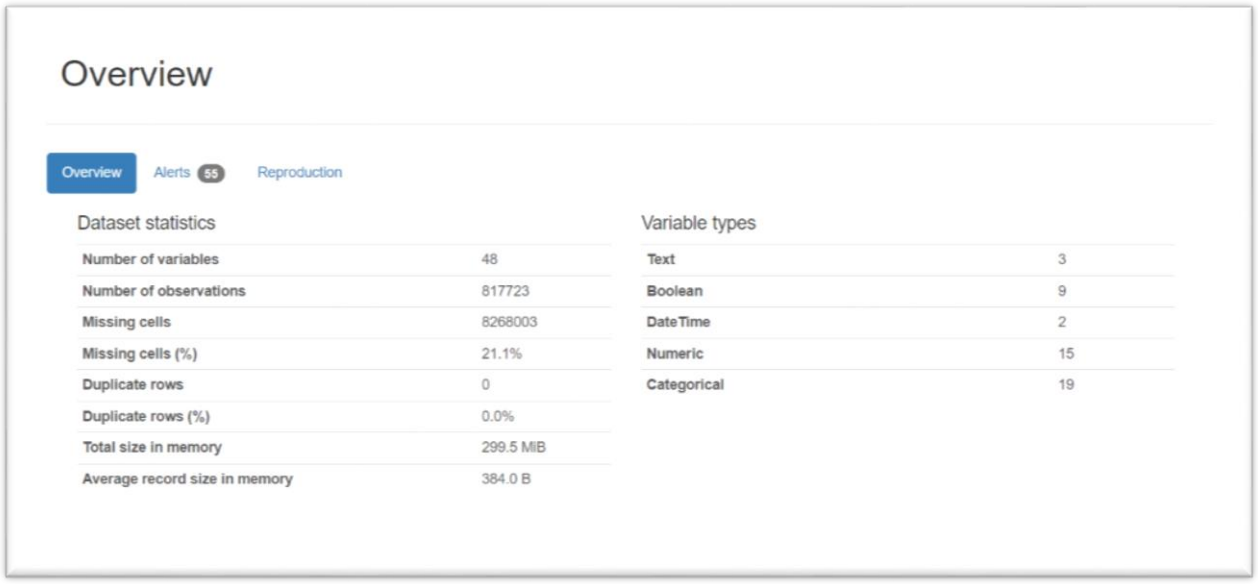
**Quality of Data:** One notable feature of this dataset is the presence of 8,268,003 missing cells, accounting for 21.1% of the total. This level of incompleteness creates issues and demands cautious treatment to ensure the trustworthiness of any studies performed on this dataset. Despite this, the dataset stands out for having zero duplicate rows, which confirms the uniqueness of each item and simplifies the early stages of data processing.

**Usage of Memory:** The dataset uses 299.5 MiB of memory, with an average record size of 384.0 B. These numbers indicate that, despite its size, the dataset is manageable for study on modern computing systems without the need of specialised data processing techniques. The dataset, which includes 3 text, 9 boolean, 2 datetime, 15 numeric, and 19 categorical variables, is varied and will require a variety of preprocessing approaches to be properly exploited for analysis or modelling.

**Potential Repercussions:** The large amount of missing data reduces the efficacy of machine learning models or statistical studies that use this dataset. It is critical to resolve these gaps using imputation or exclusion strategies appropriate to the study aims. One advantage is the lack of duplicate rows, which eliminates the need for extensive data cleansing. Prior to doing any in-depth research, it is recommended to investigate the nature of the missing data to establish if it is absent at random or symptomatic of underlying systemic issues that must be addressed.

**Conclusion**: In summary, the dataset's completeness and lack of duplicates testifies to its excellent quality and preparedness for future investigation, however the substantial amount of missing data creates challenges. After the missing values have been corrected, the dataset may be used for extensive exploratory analysis, predictive modelling, and other

statistical investigations. Because of its manageable size, it can be easily analysed on common computing systems, making it a useful tool for interpreting traffic collision patterns and developing safety improvements.




## NewYork Crash Dataset Statistics

With 29 variables spread over a sizable 2,075,427 observations, the dataset offers a sizable amount of data points appropriate for in-depth examination and model training.

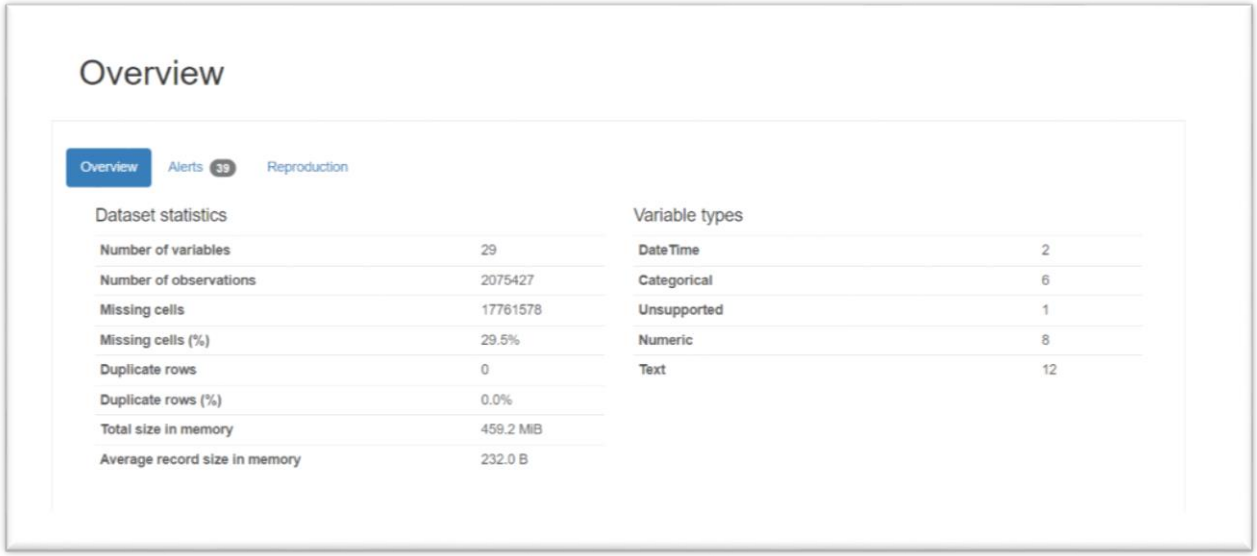
**Quality of Data:** There are 17,761,578 missing cells, which account for 29.5% of the data, posing a significant problem in terms of completeness. However, the dataset has no duplicate rows, which confirms each observation's uniqueness.
The dataset has a combination of DateTime, Categorical, Numeric, and Text variables. There is a significant amount of missing data in numerous essential sections, notably those including geographical identifiers such as borough and zip code, as well as detailed vehicle details.

**Usage of Memory:** Resource requirements: With a total size of 459.2 MiB and an average record size of 232.0 B, the dataset is rather huge and may necessitate effective data handling approaches to handle, particularly on ordinary computer systems with constrained resources.

**Potential Repercussions:**

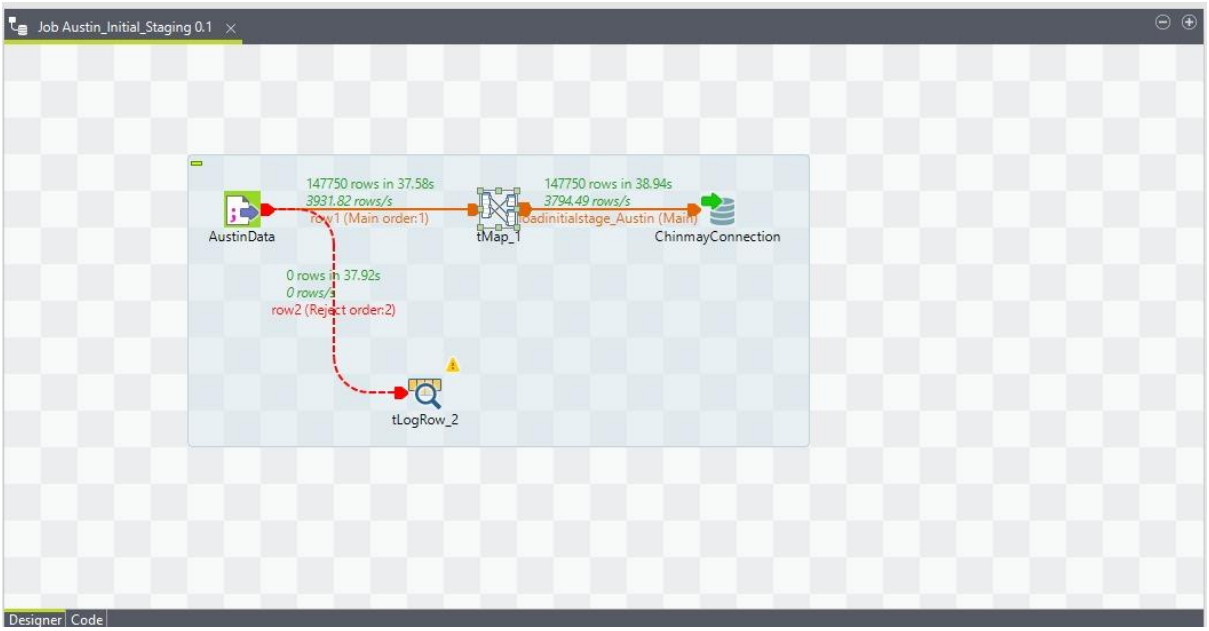
Analysis Impact: A large proportion of missing data may reduce the dataset's relevance for some types of analysis, such as machine learning models that require full instances. The missingness requires careful consideration, and suitable solutions for filling these gaps—such as imputation or omission—must be used.
Data Preparation: Due to the variety of data types and the availability of large amounts of missing and skewed data, data pretreatment will be required. This should entail looking into missing data to see if the patterns of absence are random or systematic.

**Conclusion:** Finally, the dataset's large quantity and diversity of data points make it ideal for detailed study. While the large missing data is a barrier, there are no duplicate entries, which makes the first data cleaning procedure easier. The preparation of the dataset for analysis or usage in machine learning models will involve careful management of missing information. Once handled, the dataset has the potential to be a great tool in providing insights about traffic incident patterns and contributing variables, as well as guiding preventative measures and policy choices.



# Talend Initial Staging

## Staging Austin Crash Dataset




Input Component (AustinData): The data processing pipeline's input component, Austin Data, is where data is ingested from the dataset as an external source. Metrics show that the intake phase is effective since 147,750 rows of data were read and processed in 37.58 seconds.

Transformation Component (tMap_1): Data transformation is carried out via the tMap component. In this stage, specified business rules are applied, data from various sources is merged, rows are filtered according to certain criteria, and data types are converted as needed. To make sure the data is in the right format and structure for the next steps, the transformation is essential.

Output Component (tLoadinitialstage_Austin): Data is loaded into a staging table following processing. In a database, staging tables are usually used as a holding region in between to help with additional processing before the data is sent to its ultimate location. The operation loads the same amount of rows as were initially read in 38.94 seconds, demonstrating that no data was lost during the transformation step.

Logging Component (tLogRow_2): This component is logged in order to record any rows that are rejected from the transformation component because of errors or data that does not match the specified requirements. It processes 0 rows in 37.92 seconds, which means that no data was turned down. This logging is crucial for preserving data integrity and debugging reasons.

# Austin Dataset SQL Query Validations

```
use finalproject;
select * from Initial_Austin_Staging
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

| crash_id | crash_fatal_fl | crash_date | crash_time | case_id | rpt_latitude | rpt_longitude | rpt_block_num | rpt_street_pfx | rpt_street_name | rpt_street_sfx | crash_speed_limit | road_constr_zone_fl | latitude | long |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 13762420 | N | 03/30/2014 10:58:00 AM | 10:58:00 | 140890874 | | | | | 3707 MANCHACA | RD | 10 | N | | |
| 13777334 | N | 03/27/2014 01:07:00 PM | 13:07:00 | 140860852 | | | 3400 | | PALM WAY TO MOPAC NB RAMP | | 50 | N | 30.404 | -97.7 |
| 13777441 | N | 03/28/2014 03:42:00 PM | 15:42:00 | 140871196 | | | 8704 | | BALCONES CLUB DR | DR | -1 | N | 30.43798421 | -97.7 |
| 13797332 | N | 04/09/2014 02:09:00 PM | 14:09:00 | 140991015 | | | 8000 | | E US 290 HWY SVRD EB | | 60 | N | 30.3276207970184 | -97.6 |
| 13795604 | N | 04/07/2014 06:00:00 PM | 18:00:00 | 140971248 | | | 200 | W | BEN WHITE | BLVD | -1 | N | 30.22515795 | -97.7 |
| 13765070 | N | 03/31/2014 03:26:00 AM | 03:26:00 | 140900191 | | | 8700 | S | IH 35 SVRD | | 50 | N | 30.16984339 | -97.7 |
| 13790426 | N | 04/04/2014 02:34:00 PM | 14:34:00 | 140941160 | | | 4000 | | S FM 973 RD | RD | -1 | N | 30.19320516 | -97.6 |
| 13795213 | N | 04/18/2014 02:06:00 AM | 02:06:00 | 141080164 | | | 1500 | E | ANDERSON | LN | 65 | N | 30.33279476 | -97.6 |
| 13786430 | N | 04/04/2014 12:11:00 AM | 00:11:00 | 140940020 | | | 3400 | N | IH 35 NB | | 55 | N | 30.29613303 | -97.7 |
| 13792606 | N | 04/18/2014 10:05:00 AM | 10:05:00 | 141080445 | | | 700 | S | MOPAC | EXPY | 65 | N | 30.2657795 | -97.7 |
| 13803033 | N | 04/13/2014 01:06:00 PM | 13:06:00 | 141030908 | | | 2100 | | N IH 35 SVRD SB | | 55 | N | 30.2785745242044 | -97.7 |
| 13765043 | N | 03/28/2014 04:04:00 PM | 16:04:00 | 140871241 | | | | | MIDDLE FISKVILLE | RD | 40 | N | 30.37492476 | -97.6 |
| 13800672 | N | 04/11/2014 12:13:00 PM | 12:13:00 | 141010879 | | | 8300 | | E BEN WHITE BLVD EB | BLVD | 50 | N | 30.2221866 | -97.6 |
| 13792523 | N | 04/08/2014 10:00:00 AM | 10:00:00 | 140980602 | | | 2900 | S | 1ST | ST | 35 | N | 30.23599898 | -97.7 |
| 13781584 | N | 03/31/2014 08:59:00 AM | 08:59:00 | 140900474 | | | 400 | E | E ST JOHNS AVE | AVE | 35 | N | 30.3342295 | -97.7 |
| 13773523 | N | 04/04/2014 09:14:00 AM | 09:14:00 | 140940563 | | | 600 | W | SLAUGHTER | LN | 40 | N | 30.1724548 | -97.7 |
| 13797628 | N | 04/10/2014 12:30:00 PM | 12:30:00 | 141000856 | | | 2200 | S | S IH 35 SB | HWY | 65 | N | 30.23571 | -97.7 |
| 13767418 | N | 04/01/2014 01:10:00 AM | 01:10:00 | 140910083 | | | 6300 | W | WILLIAM CANNON | DR | 40 | N | 30.23373062 | -97.6 |
| 13788264 | N | 04/15/2014 08:30:00 AM | 08:30:00 | 141050418 | | | 11400 | N | IH 35 | FWY | 70 | N | 30.37819578 | -97.6 |
| 13781499 | N | 03/29/2014 09:25:00 AM | 09:25:00 | 140880700 | | | | | QUAIL PARK DR | DR | 30 | N | 30.36387476 | -97.7 |
| 13802545 | N | 04/08/2014 08:18:00 PM | 20:18:00 | 140981718 | | | | | LAKELINE | BLVD | 45 | N | 30.47948327 | -97.6 |
| 13803139 | N | 04/14/2014 09:35:00 PM | 21:35:00 | 141041715 | | | | | MANCHACA RD | RD | 35 | N | 30.20283479 | -97.6 |

stin_Staging 1 | Read Only

Output

Action Output

| # | Time | Action | Message |
|---|---|---|---|
| 1 | 15:24:10 | create database finalproject | 1 row(s) affected |
| 2 | 16:29:31 | use finalproject | 0 row(s) affected |
| 3 | 16:29:58 | select * from Initial_Austin_Staging LIMIT 0, 1000 | 1000 row(s) returned |

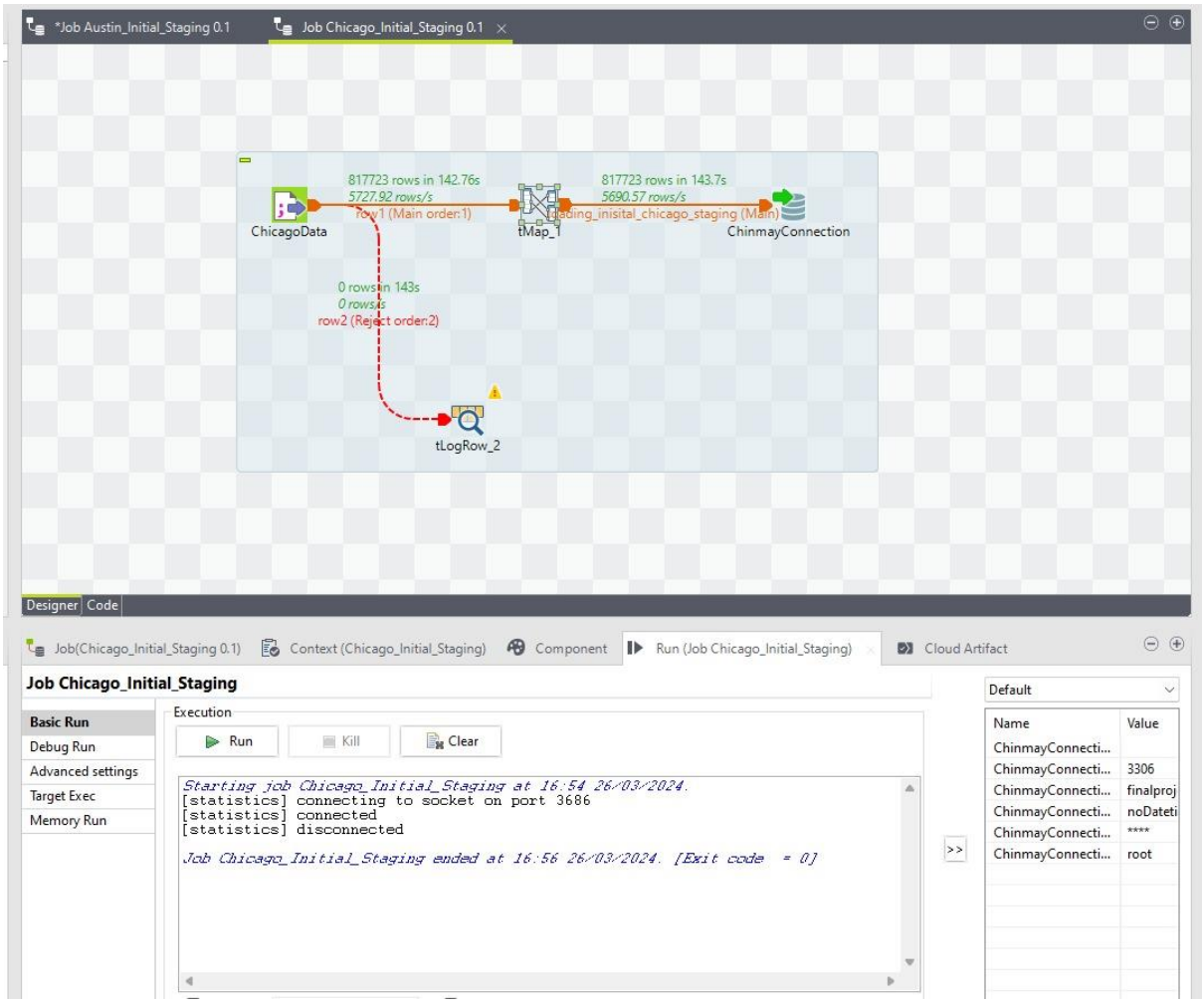
Counts:

```
3 • select * from Initial_Austin_Staging;
4 • select count(*) from Initial_Austin_Staging
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| count(*) |
|---|
| 147750 |

## Staging Chicago Crash Dataset



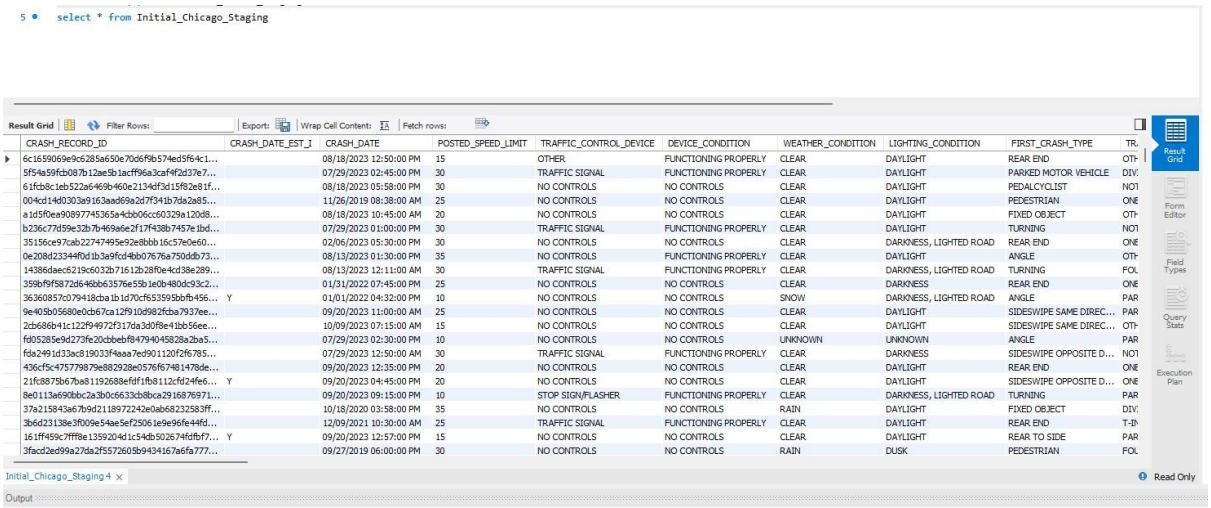

Input Component (ChicagoData): The dataset is read from the source at the beginning of the data flow, which is represented by the input component (ChicagoData). It took 142.76 seconds to process 817,723 rows, indicating that a substantial amount of data was handled rather quickly.

Transformation Component (tMap_1): The input data is transformed using the tMap component. This entails putting business logic into practice, combining data from different sources, and changing data types as needed. In order to ensure that the data fulfils the necessary requirements for storage or analysis, the transformation process is a crucial first step in preparing the data for further operations.

Output Component (tLoadinitialstage_Chicago_staging): The data is routed after transformation. A staging table is a database environment's equivalent of a temporary holding space. In this stage, the converted data is imported into the staging table; it takes 143.7 seconds to load 817,723 rows. Staging areas are frequently used as a first step in the preparation of data transfer to the destination or for additional data modification operations.

Logging Component (tLogRow_2): This component is put up to handle rejected rows that don't match the requirements. No rows were rejected, according to the statistics (0 rows in 143 seconds), indicating the quality of the data source and the success of the data transformation procedures.

## Chicago Dataset SQL Query Validations



```
5 •  select * from Initial_Chicago_Staging
```

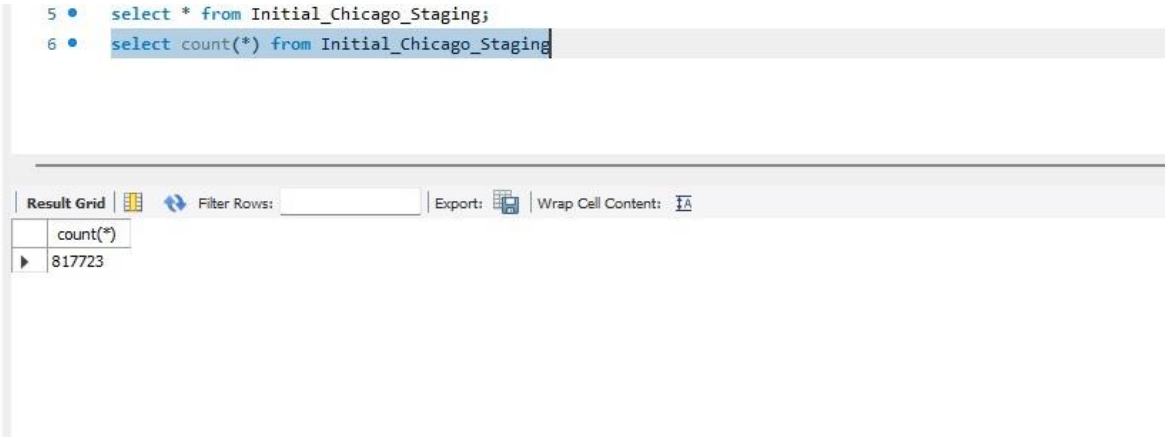
Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

| CRASH_RECORD_ID | CRASH_DATE_EST_I | CRASH_DATE | POSTED_SPEED_LIMIT | TRAFFIC_CONTROL_DEVICE | DEVICE_CONDITION | WEATHER_CONDITION | LIGHTING_CONDITION | FIRST_CRASH_TYPE | TR. |
|---|---|---|---|---|---|---|---|---|---|
| 6c1659069e9c6285a650e70d6f9b574ed5f64c1... | | 08/18/2023 12:50:00 PM | 15 | OTHER | FUNCTIONING PROPERLY | CLEAR | DAYLIGHT | REAR END | OTH |
| 5f54a59fcb087b12ae5b1acff96a3caf4f2d37e7... | | 07/29/2023 02:45:00 PM | 30 | TRAFFIC SIGNAL | FUNCTIONING PROPERLY | CLEAR | DAYLIGHT | PARKED MOTOR VEHICLE | DIV |
| 61fcb8c1eb522a6469b460e2134df3d15f82e81f... | | 08/18/2023 05:58:00 PM | 30 | NO CONTROLS | NO CONTROLS | CLEAR | DAYLIGHT | PEDALCYCLIST | NOT |
| 004cd14d0303a9163aad69a2d7f341b7da2a85... | | 11/26/2019 08:38:00 AM | 25 | NO CONTROLS | NO CONTROLS | CLEAR | DAYLIGHT | PEDESTRIAN | ONE |
| a1d5f0ea90897745365a4cbb06cc60329a120d8... | | 08/18/2023 10:45:00 AM | 20 | NO CONTROLS | NO CONTROLS | CLEAR | DAYLIGHT | FIXED OBJECT | OTH |
| b236c77d59e32b7b469a6e2f17f438b7457e1bd... | | 07/29/2023 01:00:00 PM | 30 | TRAFFIC SIGNAL | FUNCTIONING PROPERLY | CLEAR | DAYLIGHT | TURNING | NOT |
| 35156ce97cab22747495e92e8bbb16c57e0e60... | | 02/06/2023 05:30:00 PM | 30 | NO CONTROLS | NO CONTROLS | CLEAR | DARKNESS, LIGHTED ROAD | REAR END | ONE |
| 0e208d23344f0d1b3a9fcd4bb07676a750ddb73... | | 08/13/2023 01:30:00 PM | 35 | NO CONTROLS | FUNCTIONING PROPERLY | CLEAR | DAYLIGHT | ANGLE | OTH |
| 14386daec6219c6032b71612b28f0e4cd38e289... | | 08/13/2023 12:11:00 AM | 30 | TRAFFIC SIGNAL | FUNCTIONING PROPERLY | CLEAR | DARKNESS, LIGHTED ROAD | TURNING | FOL |
| 359bf9f5872d646bb63576e55b1e0b480dc93c2... | | 01/31/2022 07:45:00 PM | 25 | NO CONTROLS | NO CONTROLS | CLEAR | DARKNESS | REAR END | ONE |
| 36360857c079418cba1b1d70cf653595bbfb456... | Y | 01/01/2022 04:32:00 PM | 10 | NO CONTROLS | NO CONTROLS | SNOW | DARKNESS, LIGHTED ROAD | ANGLE | PAR |
| 9e405b05680e0cb67ca12f910d982fcba7937ee... | | 09/20/2023 11:00:00 AM | 25 | NO CONTROLS | NO CONTROLS | CLEAR | DAYLIGHT | SIDESWIPE SAME DIREC... | PAR |
| 2cb686b41c122f94972f317da3d0f8e41bb56ee... | | 10/09/2023 07:15:00 AM | 15 | NO CONTROLS | NO CONTROLS | CLEAR | DAYLIGHT | SIDESWIPE SAME DIREC... | OTH |
| fd05285e9d273fe20cbbebf8479405828a2ba5... | | 07/29/2023 02:30:00 PM | 10 | NO CONTROLS | NO CONTROLS | UNKNOWN | UNKNOWN | ANGLE | PAR |
| fda2491d33ac819033f4aaa7ed901120f2f6785... | | 07/29/2023 12:50:00 AM | 30 | TRAFFIC SIGNAL | FUNCTIONING PROPERLY | CLEAR | DARKNESS | SIDESWIPE OPPOSITE D... | NOT |
| 436cf5c475779879e882928e0576f67481478de... | | 09/20/2023 12:35:00 PM | 20 | NO CONTROLS | NO CONTROLS | CLEAR | DAYLIGHT | REAR END | ONE |
| 21fc8875b67ba81192688efdf1fb8112cfd24fe6... | Y | 09/20/2023 04:45:00 PM | 20 | NO CONTROLS | NO CONTROLS | CLEAR | DAYLIGHT | SIDESWIPE OPPOSITE D... | ONE |
| 8e0113a690bbc2a3b0c6633cb8bca2916876971... | | 09/20/2023 09:15:00 PM | 10 | STOP SIGN/FLASHER | FUNCTIONING PROPERLY | CLEAR | DARKNESS, LIGHTED ROAD | TURNING | PAR |
| 37a215843a67b9d2118972242e0ab68232583ff... | | 10/18/2020 03:58:00 PM | 35 | NO CONTROLS | NO CONTROLS | RAIN | DAYLIGHT | FIXED OBJECT | DIV |
| 3b6d23138e3f009e54ae5ef25061e9e96fe44fd... | | 12/09/2021 10:30:00 AM | 25 | TRAFFIC SIGNAL | FUNCTIONING PROPERLY | CLEAR | DAYLIGHT | REAR END | T-IN |
| 161ff459c7fff8e1359204d1c54db502674fdfbf7... | Y | 09/20/2023 12:57:00 PM | 15 | NO CONTROLS | NO CONTROLS | CLEAR | DAYLIGHT | REAR TO SIDE | PAR |
| 3facd2ed99a27da2f557260 5b9434167a6fa777... | | 09/27/2019 06:00:00 PM | 30 | NO CONTROLS | NO CONTROLS | RAIN | DUSK | PEDESTRIAN | FOL |

Initial_Chicago_Staging 4 | Read Only

Output

Counts:

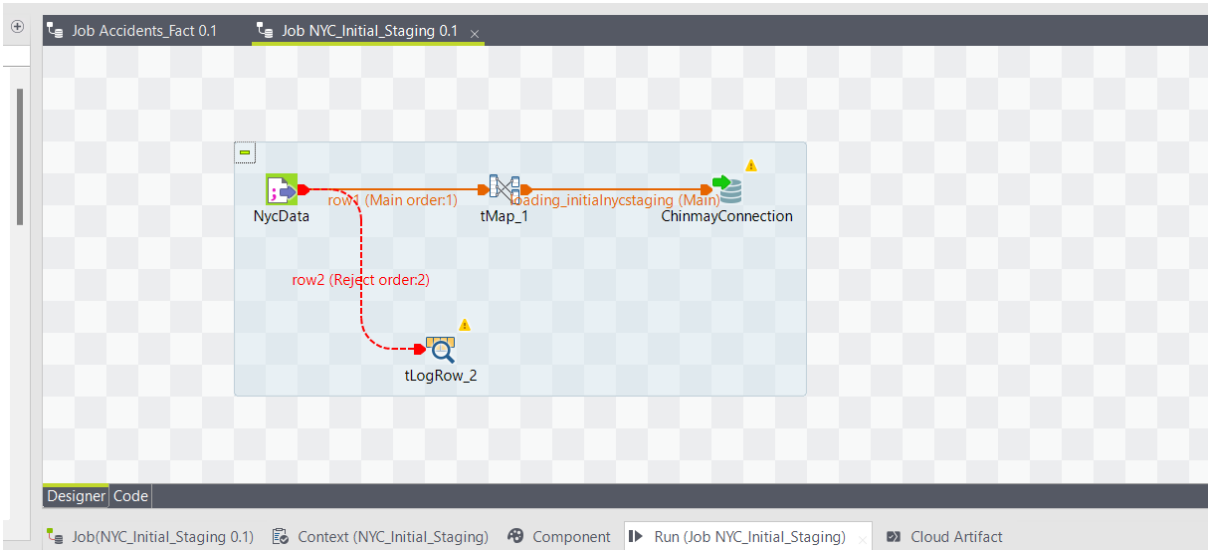


```
5 •  select * from Initial_Chicago_Staging;
6 •  select count(*) from Initial_Chicago_Staging
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| count(*) |
|---|
| 817723 |

## Staging NYC Crash Dataset



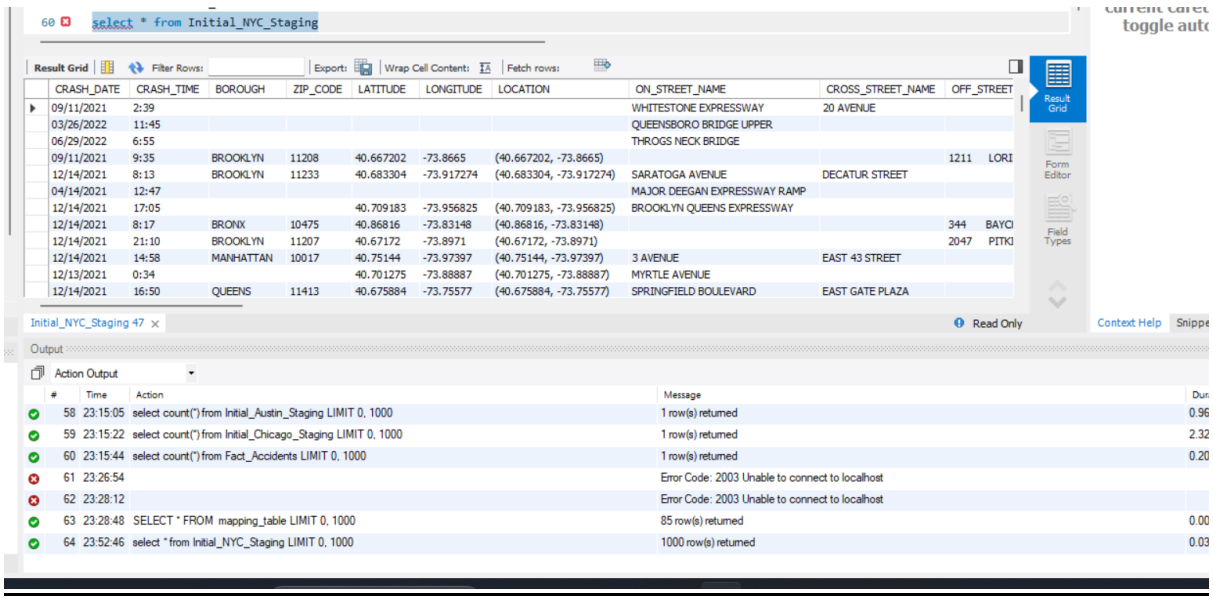

Input Component (NYCData): This is the first step in the data flow, where data is read from a NYC Datasource. The exact row count for this component is based on the typical function, it would extract data from a given place, such as a database or a flat file, for use in the subsequent stages of the ETL process.

Transformation Component (tMap_1): Following initial extraction by the NycData component, the data is transformed using the tMap component. This stage is critical for enforcing business standards, combining fields from various sources, transforming data types, and ensuring that the data is properly formatted and structured for its intended use. The transformation step is frequently where the majority of data modification happens, laying the groundwork for precise and efficient loading into the target system.

Output Component (loading_initialnycstaging): After transformation, the data is sent to the output component, labelled 'loading_initialnycstaging', which most likely corresponds to a staging table in a database system. Staging areas are utilised as intermediate storage locations for altered data before it is sent to its final destination. It's an important phase for carrying out extra quality checks or modifications as needed.

Logging Component (tLogRow_2): This component is set up to gather and log any rows of data that were rejected during the transformation. These rejections might be for a variety of reasons, including failure to fulfil validation requirements specified in the business logic or transformation failures. According to the statistics in your workflow, there were no rejected rows, indicating a smooth transformation process free of data quality concerns or mistakes that would prohibit the rows from being put into the staging area.

# Nyc Dataset SQL Query Validations



```
60  select * from Initial_NYC_Staging
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

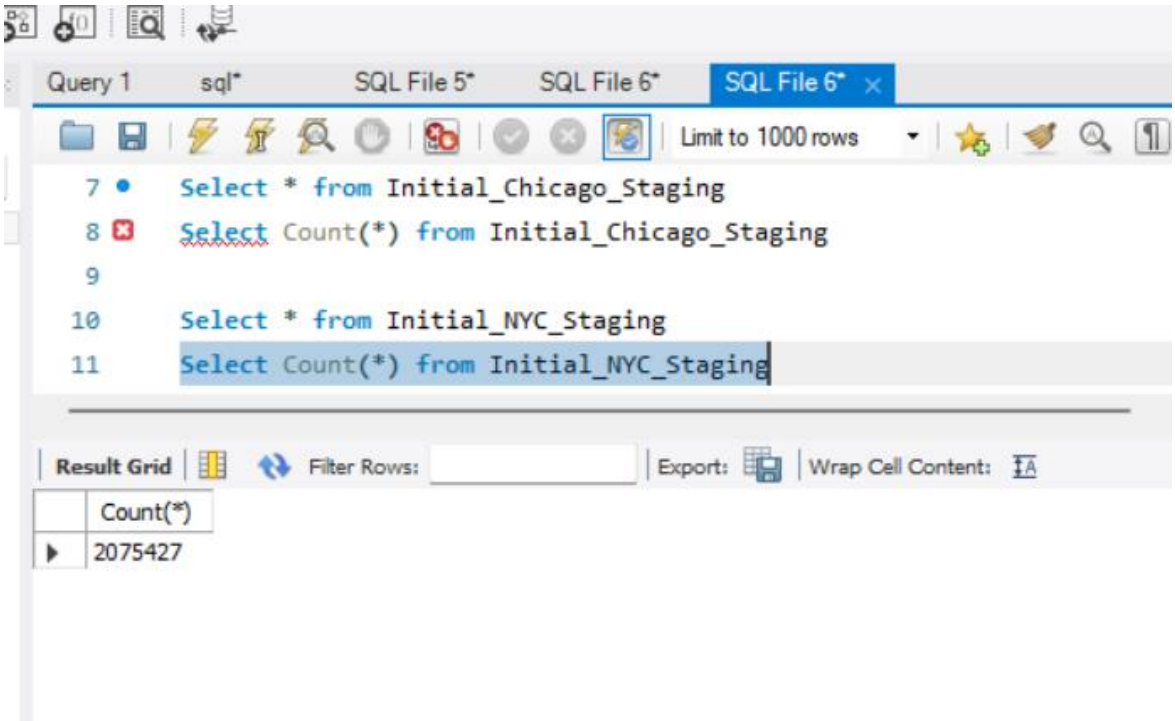
| CRASH_DATE | CRASH_TIME | BOROUGH | ZIP_CODE | LATITUDE | LONGITUDE | LOCATION | ON_STREET_NAME | CROSS_STREET_NAME | OFF_STREET |
|---|---|---|---|---|---|---|---|---|---|
| 09/11/2021 | 2:39 | | | | | | WHITESTONE EXPRESSWAY | 20 AVENUE | |
| 03/26/2022 | 11:45 | | | | | | QUEENSBORO BRIDGE UPPER | | |
| 06/29/2022 | 6:55 | | | | | | THROGS NECK BRIDGE | | |
| 09/11/2021 | 9:35 | BROOKLYN | 11208 | 40.667202 | -73.8665 | (40.667202, -73.8665) | | | 1211 LORI |
| 12/14/2021 | 8:13 | BROOKLYN | 11233 | 40.683304 | -73.917274 | (40.683304, -73.917274) | SARATOGA AVENUE | DECATUR STREET | |
| 04/14/2021 | 12:47 | | | | | | MAJOR DEEGAN EXPRESSWAY RAMP | | |
| 12/14/2021 | 17:05 | | | 40.709183 | -73.956825 | (40.709183, -73.956825) | BROOKLYN QUEENS EXPRESSWAY | | |
| 12/14/2021 | 8:17 | BRONX | 10475 | 40.86816 | -73.83148 | (40.86816, -73.83148) | | | 344 BAYC |
| 12/14/2021 | 21:10 | BROOKLYN | 11207 | 40.67172 | -73.8971 | (40.67172, -73.8971) | | | 2047 PITK |
| 12/14/2021 | 14:58 | MANHATTAN | 10017 | 40.75144 | -73.97397 | (40.75144, -73.97397) | 3 AVENUE | EAST 43 STREET | |
| 12/13/2021 | 0:34 | | | 40.701275 | -73.88887 | (40.701275, -73.88887) | MYRTLE AVENUE | | |
| 12/14/2021 | 16:50 | QUEENS | 11413 | 40.675884 | -73.75577 | (40.675884, -73.75577) | SPRINGFIELD BOULEVARD | EAST GATE PLAZA | |

Initial_NYC_Staging 47 | Read Only | Context Help

Output

Action Output

| # | Time | Action | Message | Dur |
|---|---|---|---|---|
| 58 | 23:15:05 | select count(*) from Initial_Austin_Staging LIMIT 0, 1000 | 1 row(s) returned | 0.96 |
| 59 | 23:15:22 | select count(*) from Initial_Chicago_Staging LIMIT 0, 1000 | 1 row(s) returned | 2.32 |
| 60 | 23:15:44 | select count(*) from Fact_Accidents LIMIT 0, 1000 | 1 row(s) returned | 0.20 |
| 61 | 23:26:54 | | Error Code: 2003 Unable to connect to localhost | |
| 62 | 23:28:12 | | Error Code: 2003 Unable to connect to localhost | |
| 63 | 23:28:48 | SELECT * FROM mapping_table LIMIT 0, 1000 | 85 row(s) returned | 0.00 |
| 64 | 23:52:46 | select * from Initial_NYC_Staging LIMIT 0, 1000 | 1000 row(s) returned | 0.03 |



Query 1 | sql* | SQL File 5* | SQL File 6* | SQL File 6*

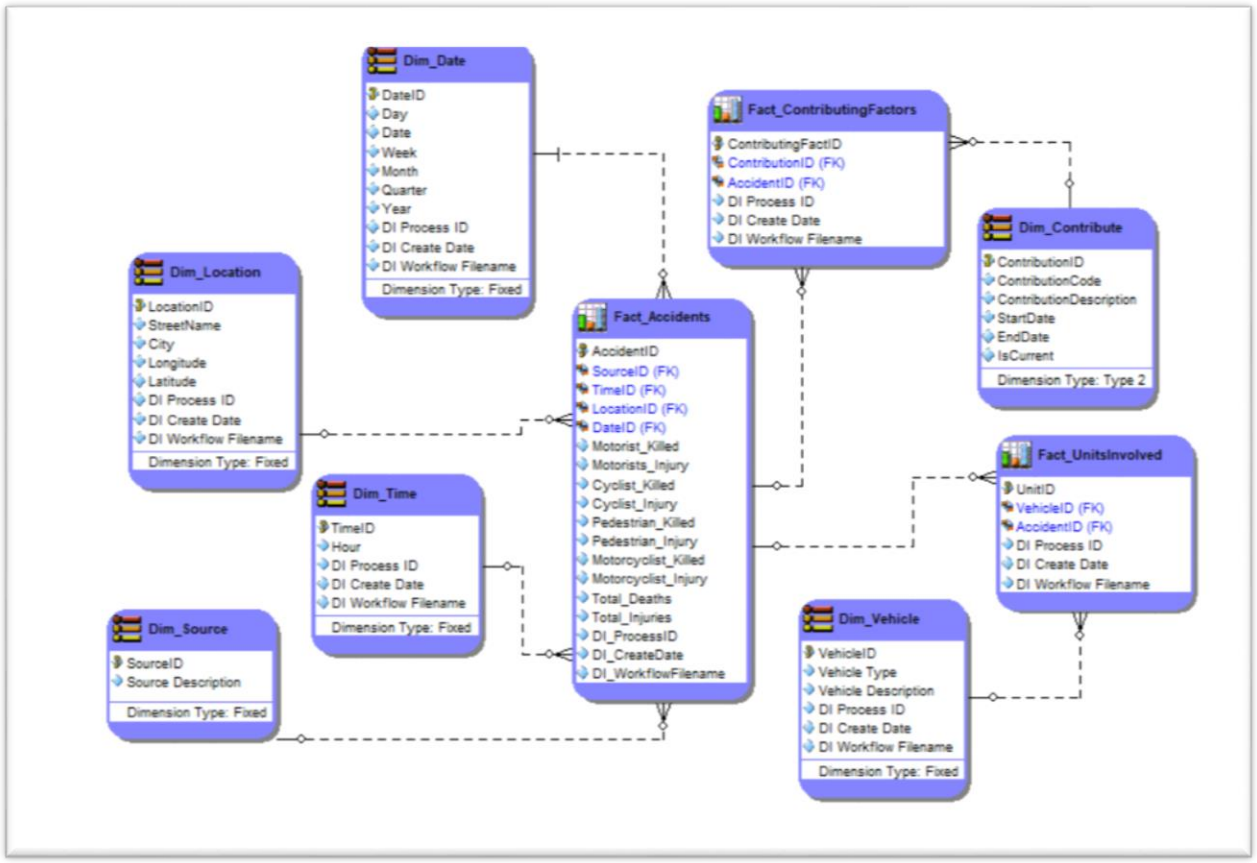
Limit to 1000 rows

```
7  Select * from Initial_Chicago_Staging
8  Select Count(*) from Initial_Chicago_Staging
9
10 Select * from Initial_NYC_Staging
11 Select Count(*) from Initial_NYC_Staging
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| Count(*) |
|---|
| 2075427 |

## Dimension Table




Our team has successfully deployed this dimensional data model to improve traffic accident analysis reporting for a city's traffic management system. A detailed perspective of each occurrence is provided by the interconnected set of dimension and fact tables that comprise the database structure.

Among the primary dimension tables are:
Dim_Date: This table, which has fields for day, week, month, quarter, and year, allows for analysis by date and hence supports temporal trend analysis.
Dim_Location: The street names and coordinates for each accident are included in this table, which offers spatial analysis capabilities.
Dim_Time: Dim_Time provides more detailed information on the hours during which accidents are most likely to happen, complementing the Dim_Date database.

Dim_Source: It lists the data's sources and describes each one in detail to provide data traceability.
Dim_Contribute: It uses a Slowly Changing Dimension Type 2 technique to retain historical changes while capturing the contributing elements to accidents, their codes, descriptions, and the length of their validity.
Dim_Vehicle: The sorts of vehicles involved in accidents are listed in this table, which serves

as a foundation for analysis pertaining to automobiles.

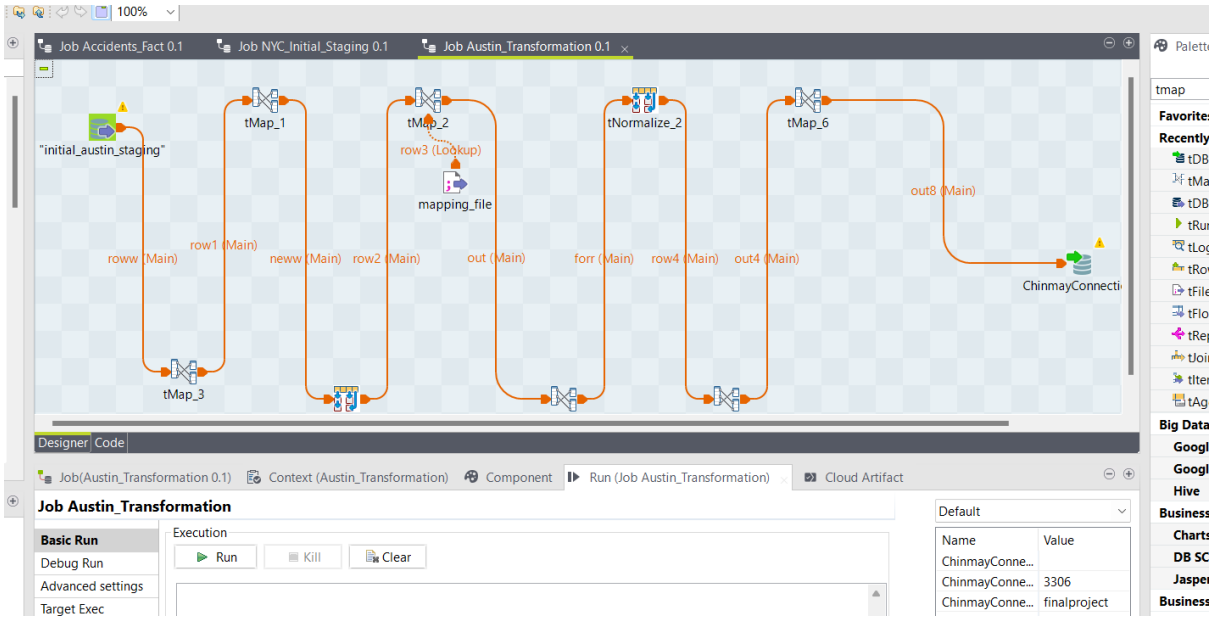
The fact tables in the model include:
Fact_Mishaps: Located in the centre of the model, this fact table logs accident metrics and associates them with time, location, and vehicle dimensions as well as important incident information such as injuries and fatalities.
Factors that Contribute: This fact table acknowledges that several causes may contribute to a single accident and vice versa, bridging the gap between incidents and their contributing elements.
Dim_Source: It lists the data's sources and describes each one in detail to provide data traceability.
Dim_Contribute: It uses a Slowly Changing Dimension Type 2 technique to retain historical changes while capturing the contributing elements to accidents, their codes, descriptions, and the length of their validity.
Dim_Vehicle: The sorts of vehicles involved in accidents are listed in this table, which serves as a foundation for analysis pertaining to automobiles.

The fact tables in the model include:
Fact_Mishaps: Located in the centre of the model, this fact table logs accident metrics and associates them with time, location, and vehicle dimensions as well as important incident information such as injuries and fatalities.
Factors that Contribute: This fact table acknowledges that several causes may contribute to a single accident and vice versa, bridging the gap between incidents and their contributing elements.
Fact_UnitsInvolved: This table, which is devoted to ongoing investigations, provides a unique avenue for the analysis of unresolved occurrences by containing data on unsolved instances.

# Part 2

## Talend Transformations

### Austin Transformations




Input Component (Initial_austin_staging):Data extraction begins with the Input Component (Initial_austin_staging), which pulls data from an initial staging area. The staging table contains raw data that was previously retrieved from the Austin dataset. This data has been initially cleaned and validated, but it has yet to be changed into the dimension tables. Transformation Components This ETL method employs several transformation components, each of which is likely to execute a unique transformation logic.

tMap_1: In data processing, determine if the 'ContributingFactor' property has a value or not. If the 'ContributingFactor' is determined to be null, use the value -1 to indicate its absence. Otherwise, keep the current non-null value of 'ContributingFactor'.

tMap_2: In the ETL procedure, we use an expression to generate a'mergedcontribution' field. This field is formed by concatenating all of the 'contribution' field values. The expression verifies each 'contribution' value; if they are all null,'mergedcontribution' stays empty, with no concatenation. If at least one 'contribution' value is not null, the expression concatenates it with the'mergedcontribution' field, resulting in a combined string of all non-null contributions.

tMap_3: When a null value is encountered in the mergedContriID field during the data transformation procedure, it is replaced with -1 as the default value. This replacement guarantees that all entries have a proper identifier, which maintains the dataset's integrity.

The non-null mergedContriID (whether initially present or substituted with -1) is then used to link the dataset to a mapping document. This mapping document provides the necessary codes for generating descriptive labels for each entry. The resulting description provides relevant context for the data, making it easier to interpret and analyse.

tMap_4: If the fields'mapping' and 'ContributionDesc' are empty or null, the value 'N/A' should be used to indicate that no relevant information is available for these fields.

tMap_5: If contributionDescription is null, print 'N/A'; else, output the value of contributionDescription as given.

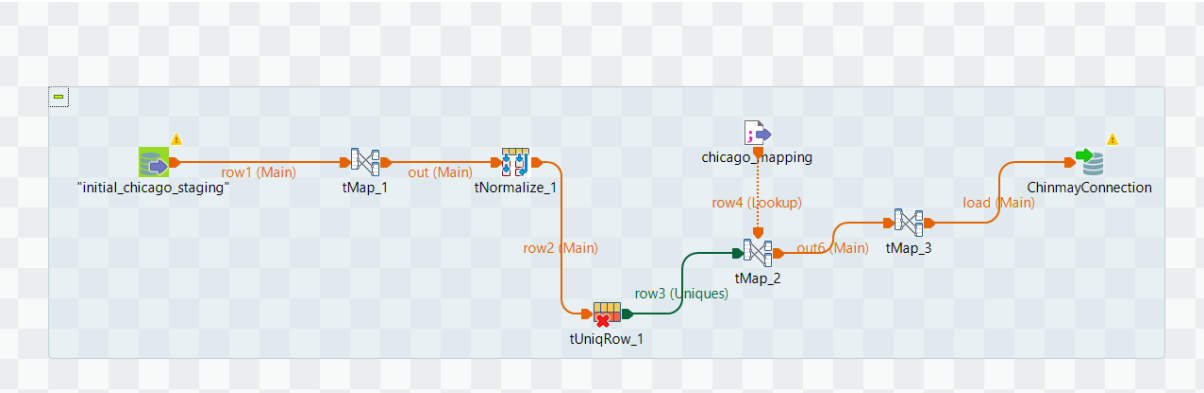
tMap_6: During the ETL process, units were trimmed for consistency, the staging table was assigned a primary key named 'AustinID' with an added sequence number for uniqueness, 'contri_id' was converted to an integer data type, dates were formatted to 'MM/DD/YYYY', and all null values were addressed by populating them with 'N/A'.

Normalisation Component : This component is commonly used to normalise data, such as breaking down repetitive groupings into a standard format or pivoting data to improve alignment with the goal dimension structure. It assures data consistency and readiness for analysis. The contributorID have been normalised and combined. The `unitsinvolved` column is normalized to separate entries that contain an ampersand ("&"), indicating the presence of multiple units within a single field.

Output Component (ChinmayaConnect): After the transformations, the data is available for loading into the target dimension table. The name implies that ChinmayaConnect is the output connection to the dimension table, which will contain the final, modified data. This is an important step since dimension tables are part of the star schema used in data warehousing for analytics.

Logging and Control Flow: The task design has a variety of flows between components, such as main, lookup, and reject connections. These manage the flow of data through the task, ensuring that each record is handled as intended. The procedure involves managing rejections and mistakes that may arise during transformation processes, although the graphic does not provide precise details on logging or error handling.

## Chicago Transformations




Input Component: "initial_chicago_staging" The procedure begins with this component, which reads data from the first Chicago staging table. This is raw data, which hau ndergone early cleaning and validation.

Transformation Component (tMap_1): The 'tMap_1' component handles initial data transformations. It performs activities like as checking the presence of values in certain columns, changing data types, and applying business logic to prepare data for future processing. An expression is used to construct the field 'Merged'. This 'Merged' field is created by adding all values from the 'contribution' field. The expression examines each 'contribution' value; if they are all null, 'Merged' is left blank without any concatenation. However, if there is at least one non-null 'contribution' value, the expression adds it to the 'Merged' column. This yields a cumulative string containing all non-null 'contribution' values. All null values are systematically controlled by using preset rules or expressions to guarantee that each field has a correct value or placeholder, preserving the dataset's consistency and integrity for subsequent actions.

Normalisation Component (tNormalize_1): This component normalises the data by dividing combined fields into separate fields or formatting the data to suit the target schema criteria. The 'Merged' column is normalised by dividing semicolon-separated values (';') into individual rows, and any nulls in 'Merged' are addressed carefully to avoid issues later.

tUniqRow_1: This component guarantees that records are unique based on the 'CrashRecordID' property, screening out duplicates and perhaps rejecting records with a null 'CrashRecordID' to maintain data integrity.

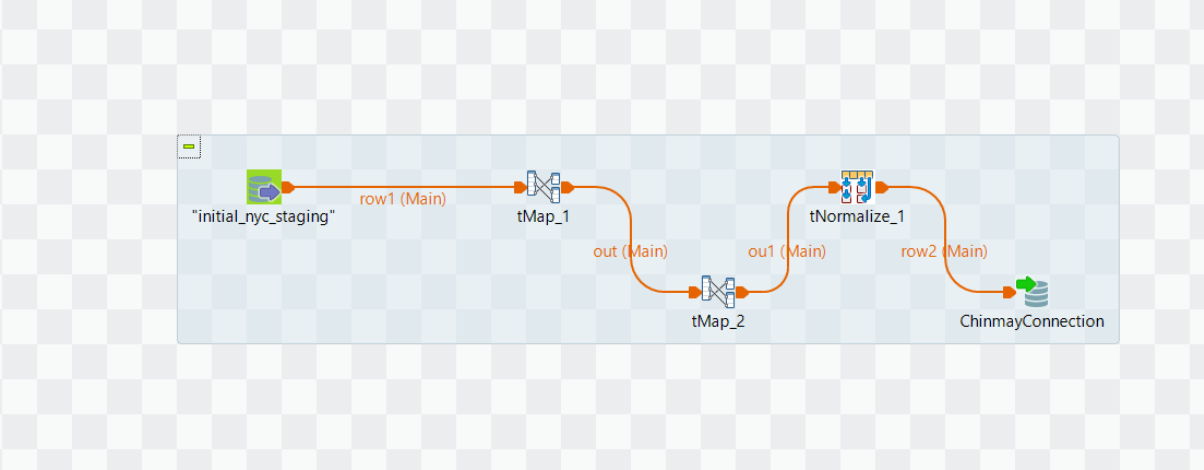
tMap_2: This transformation phase is used to match records with a mapping document and populate the 'ContributingFactorDescription' with the appropriate codes. It guarantees that any null values in 'ContributingFactor' are addressed, either by using a default description or excluding them from the matching procedure.

tMap_3: Final transformations, including checks on the Chicago-specific 'SourceName' field, are performed to guarantee that no null values are loaded into the destination system's non-nullable fields.

Logging and Control Flow: These control the flow of data through the job, ensuring that each record is treated appropriately. The approach entails managing rejections and errors that may occur throughout transformation operations, yet the image does not include specifics on logging or error management.

## New York Transformations

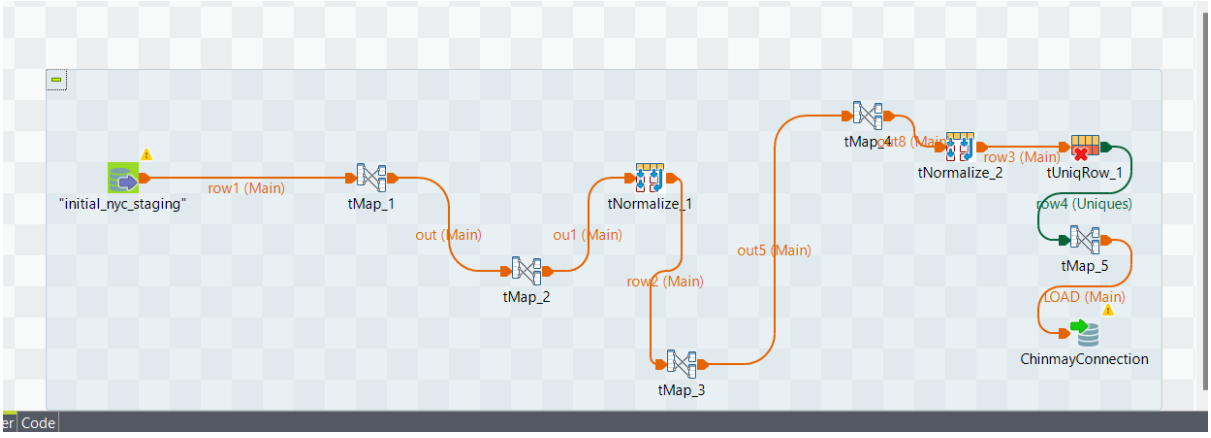
### Part01




This ETL process's tMap1 and tMap2 components integrate several 'ContriFactor' fields. If the fields contain values, they are concatenated; if a field has no value, it is left unmodified. If all 'ContriFactor' values are null, the merged field is set to 'N/A'.

The normalisation component processes the 'MergedContri' field by breaking any semicolon (';') separated values into distinct rows, hence normalising the data for consistent and systematic analysis.

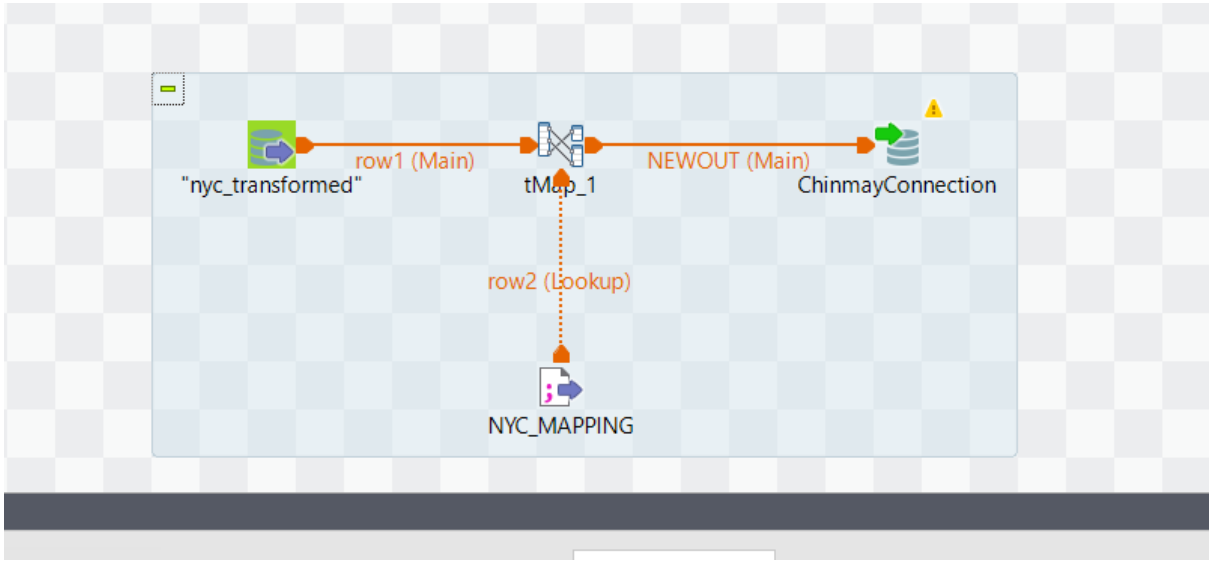
### Part02



The tMap component of this ETL method consolidates numerous 'VehicleCodeType' fields. If the fields have values, they are concatenated; if a field is empty, the value remains unchanged. If all 'VehicleCodeType' values are null, the consolidated field is assigned the value 'N/A'.

The normalisation component handles the 'MergedVehicleCode' field by splitting entries into independent rows using semicolons (';'), therefore normalising the dataset for consistent and systematic examination.

Part03

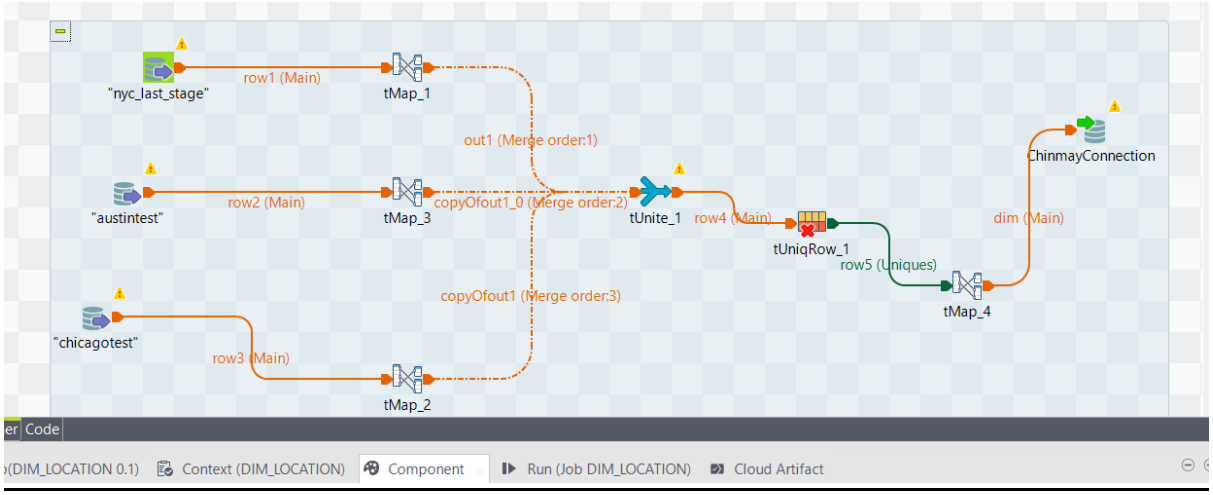



This transformation step uses a mapping document to match data to their matching 'VehicleCodeType' descriptions. It guarantees that any null values in 'VehicleCodeType' are handled appropriately, either by providing a default description or excluding them from the matching process.

# Data Warehouse Loading

## Dimension Loading

### Location Dimension




The stage tables from the databases for Austin, New York City, and Chicago are utilised as inputs in the shown ETL procedure. Each one contributes to the process, where they experience the modifications required for dimension mapping. This step is necessary to guarantee that the data from all three cities is standardised and aligned before it is combined and placed into the dimensional model for analysis.

The ETL operation extracts and transforms the'street_name', 'longitude', and 'latitude' variables from the Austin, New York City, and Chicago stage tables. The tMap components are used to guarantee that the 'longitude' and 'latitude' fields have a double data type, which ensures consistency across all three sources. These altered fields are then passed to the tUnite component, which merges the data streams into a single flow. This unified data is then utilised to create the 'location_dim' dimension, which provides a consistent view of location-related information across the multiple city stage tables, allowing for extensive geographical analysis. Also the updation attributes are added as ‘DI_ProcessID’, ‘DI_CreateDate’ and ‘DI_WorkflowFilename’.

The tUnite component combines , 'city', 'street_name', 'longitude', and 'latitude' columns from the Austin, NYC, and Chicago stage tables, guaranteeing that these columns from various tMaps are synchronised for the 'location_dim' dimension.

The synchronised attributes 'city', 'street_name', 'longitude', and 'latitude' are assigned to the 'Location_Dim' dimension database, with 'LocationID' acting as the key attribute for identification and indexing.

## Dim Location SQL Query Validations

```
1  Select * from [dbo].[DIM_LOCATION]
2  Select count(*) from [dbo].[DIM_LOCATION]
```

Results Messages

Search to filter items...

| LocationID | LONGITUDE | LATITUDE | Street_Name | City |
|---|---|---|---|---|
| 100 | 0 | 0 | 3707 manchaca rd | Austin |
| 101 | 30.404 | -97.7245 | palm way to mopac nb ramp | Austin |
| 102 | 30.438 | -97.7886 | balcones club dr | Austin |
| 103 | 30.3276 | -97.6627 | us0290 | Austin |

---

Counts:

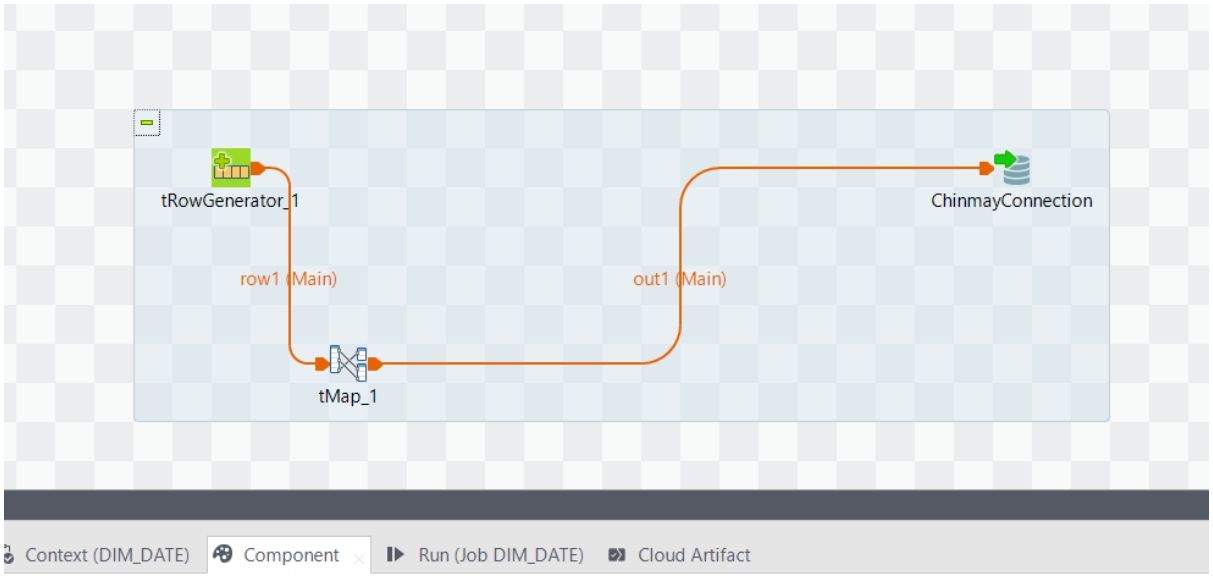
```
1  Select * from [dbo].[DIM_LOCATION]
2  Select count(*) from [dbo].[DIM_LOCATION]
```

Results Messages

Search to filter items...

766249

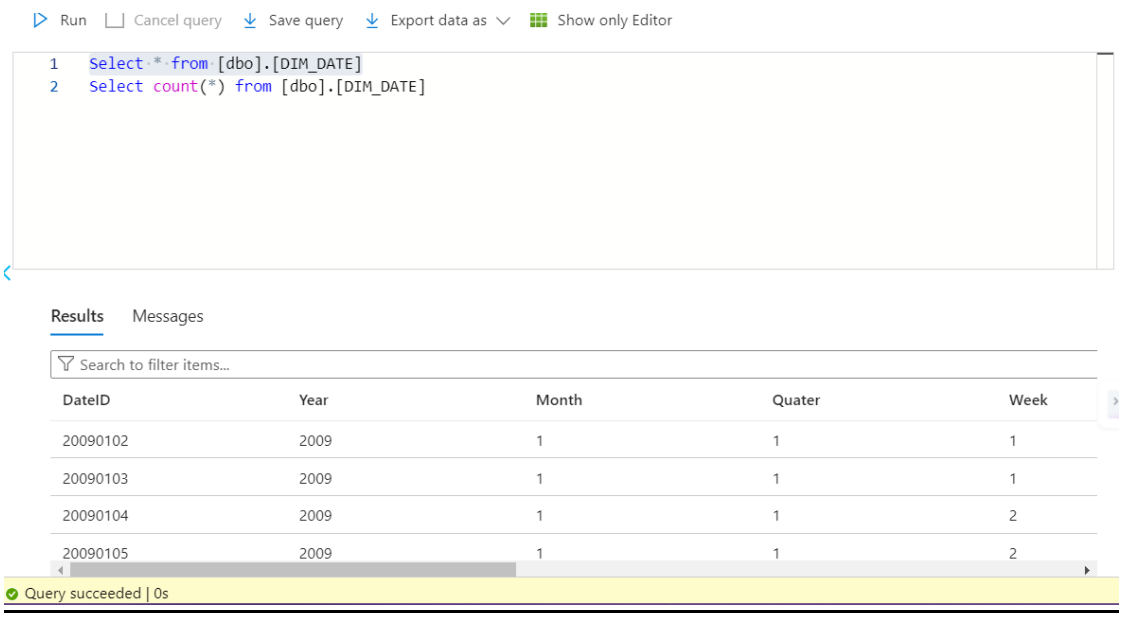
# Date Dimension



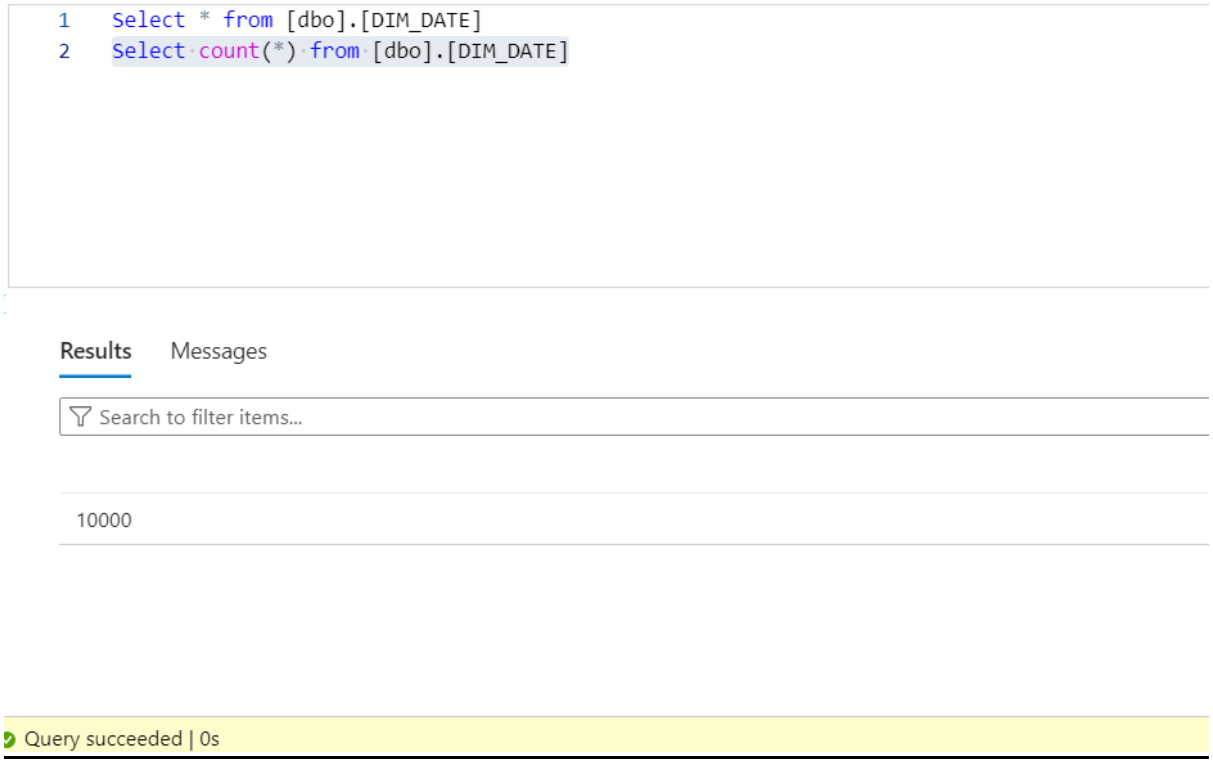

A tRowGenerator component is used in an ETL job design to produce rows of data, maybe to mimic date dimensions for a data warehouse. This can entail compiling a list of dates with associated details like day, month, and year.

After that, the produced data enters a tMap component, where transformation logic is used. This might involve adding more time-related columns, formatting dates, or even adding calculated fields like DateID, Day, Date, Week, Month, Quarters, and Year to enhance the data.

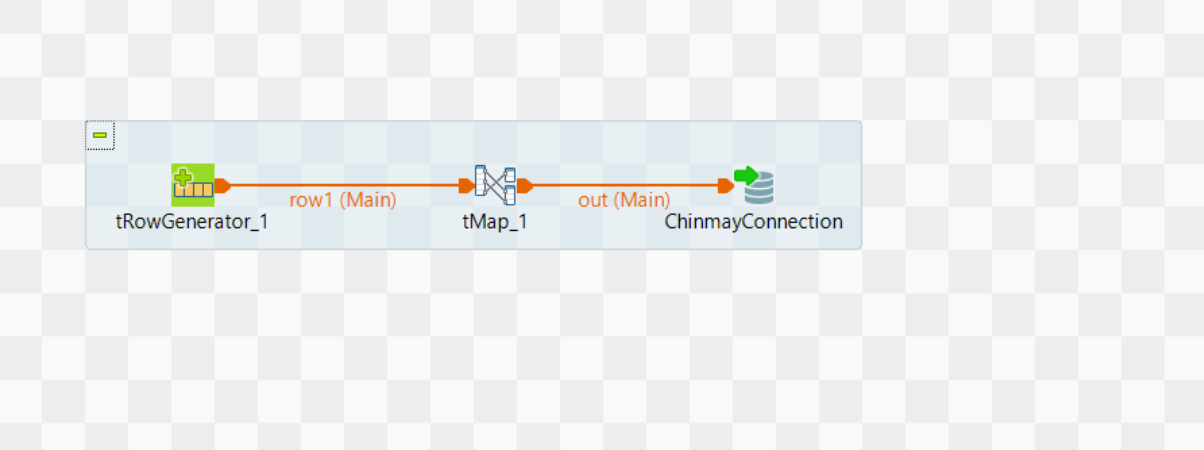
## Dim Date SQL Query Validations




## Counts:



# Time Dimension




tRowGenerator_1: This part generates sample rows according to a specified schema. It is helpful for evaluating the remaining tasks since it may provide random data based on the given patterns.
The {tMap_1} component of a Talend job intended for loading a time dimension is usually set up to map and transform the incoming data into the format needed by the time dimension table. Given that `TimeID` is to be utilised as a key attribute and that the input data from `tRowGenerator_1} contains time and hour fields, the following may be the configuration statement for the `tMap_1} component:
"The TimeID field is designated as the primary attribute in the tMap_1 component, guaranteeing that every record in the time dimension table may be uniquely recognised. The data may be accurately entered into the time table's dimensions because the fields for time and hour from the {tRowGenerator_1} component are mapped to the relevant columns in the output schema."

# Dim Time Query Validations

```
Select * from [dbo].[DIM_TIME]
Select count(*) from [dbo].[DIM_TIME]
```

Results Messages

Search to filter items...

| TimeID | Hour | DI_CreateDate | DI_ProcessID | DI_WorkflowFileName |
|---|---|---|---|---|
| 1 | 0 | 2024-04-07T13:36:21.0000... | 8GZZOJ | DIM_TIME |
| 2 | 1 | 2024-04-07T13:36:21.0000... | 8GZZOJ | DIM_TIME |
| 3 | 2 | 2024-04-07T13:36:21.0000... | 8GZZOJ | DIM_TIME |
| 4 | 3 | 2024-04-07T13:36:21.0000... | 8GZZOJ | DIM_TIME |

Query succeeded | 0s

## Counts:

Run Cancel query Save query Export data as Sho

```
Select * from [dbo].[DIM_TIME]
Select count(*) from [dbo].[DIM_TIME]
```

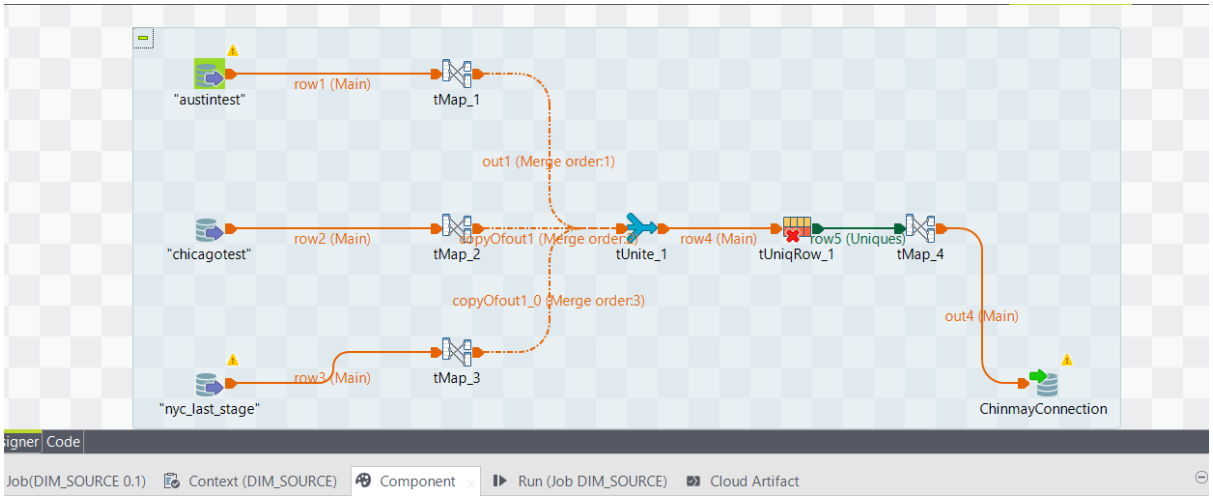
Results Messages

Search to filter items...

24

Query succeeded | 0s

## Source Dimension




"Austintest", "Chicagotest", "nyc_last_stage" are examples of input components. These are source components that are set up to read information from three distinct databases or datasets, each of which corresponds to a distinct phase of the data processing workflow.
To enhance the output schema for each tMap (tMap_1, tMap_2, tMap_3), you would include a new column called'sourcename' or something similar. This is often done in the tMap output table in the component's editor. Next, you would specify a static value for this column that would match to the source name, such as "nyc_last_stage," "chicagotest," or "austintest."
The outputs from {tMap_1}, {tMap_2}, and {tMap_3} are combined into a single stream by the Talend job's `tUnite_1} component, which also synchronises the columns across all sources to guarantee schema compatibility. This merges rows from 'chicagotest', 'nyc_last_stage', and 'austintest' into a single, unified dataset in advance of additional processing stages.
By filtering the data stream such that only unique entries based on the'source description' column are passed through, the `tUniqueRow_1} component successfully removes duplicate descriptions from the combined dataset.
The'source description' from the input data is mapped by the tMap component to the 'Source Dimension' in the output schema, therefore adding descriptive metadata about the data's origin to the dataset for use in the subsequent operations.

## Dim Source SQL Query Validations

Run | Cancel query | Save query | Export data as | Show only Editor

```
1  Select * from [dbo].[DIM_Source]
2  Select count(*) from [dbo].[DIM_Source]
```

Results | Messages

Search to filter items...

| SourceDescription | SourceID |
|---|---|
| AustinData | 8545 |
| ChicagoData | 8546 |
| NYCDataset | 8547 |

Query succeeded | 0s

## Counts:

Run | Cancel query | Save query | Export data as

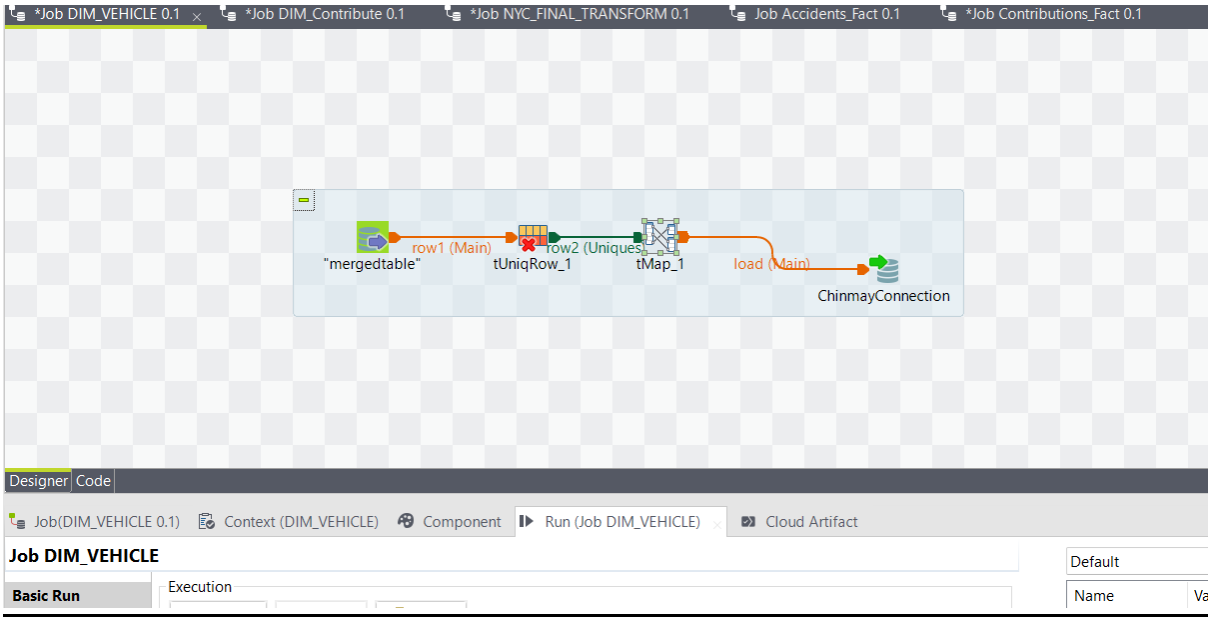
```
1  Select * from [dbo].[DIM_Source]
2  Select count(*) from [dbo].[DIM_Source]
```

Results | Messages

Search to filter items...

| |
|---|
| 3 |

## Vehicle Dimension

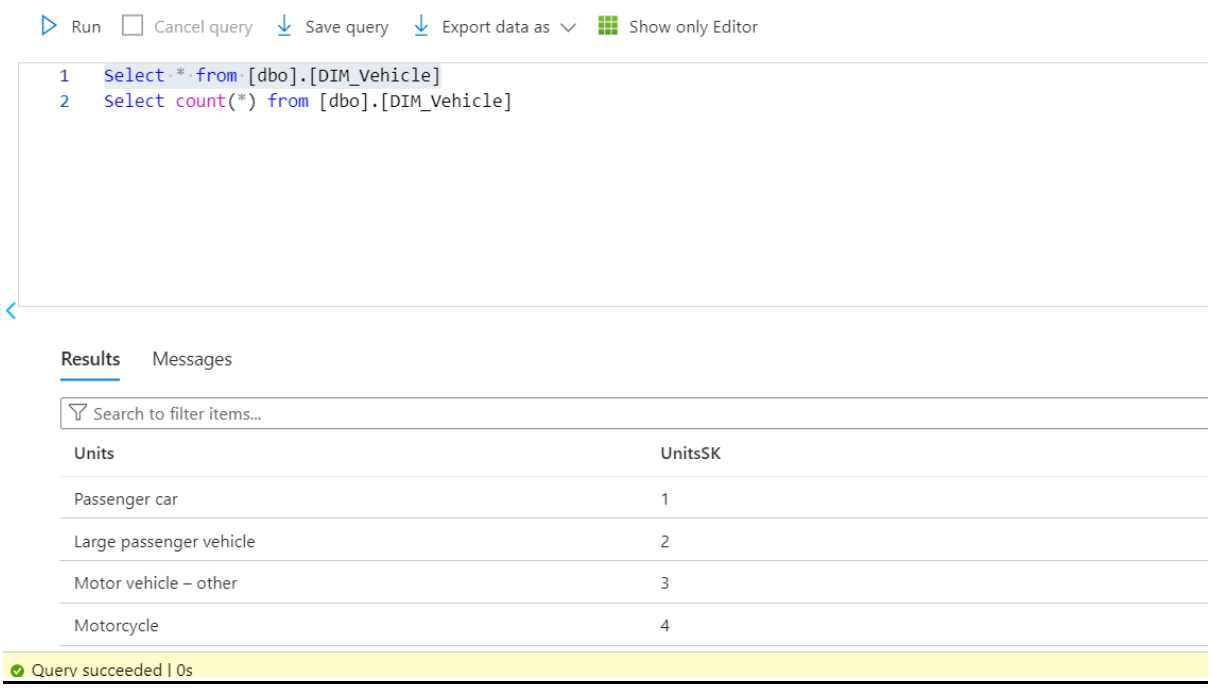



We have a simple yet efficient data processing workflow in the supplied Talend job, Job_DIM_VEHICLE, that is meant to fill or update the vehicle dimension database in a data warehouse.
The task starts with a "mergedtable" input, indicating that information has been combined into a single format from several sources or tables. This is a crucial step in guaranteeing the alignment and consistency of vehicle data that may originate from several databases or systems.
Next, a tUniqRow component (UniqRow_1)—a critical component for data quality—is sent through in the flow. By removing duplicate data, this step probably makes sure that every vehicle entry in the dimension table is distinct. Maintaining a data warehouse's integrity requires that each record be unique in order to prevent redundant data.
Once the data's uniqueness has been established, the tMap component comes into play. In order to match the source data structure with the target dimension table, data is mapped and perhaps changed in this instance. The "DIM_VEHICLE" table may require the inclusion of new fields, which might be calculated or added using the tMap to enhance the data.

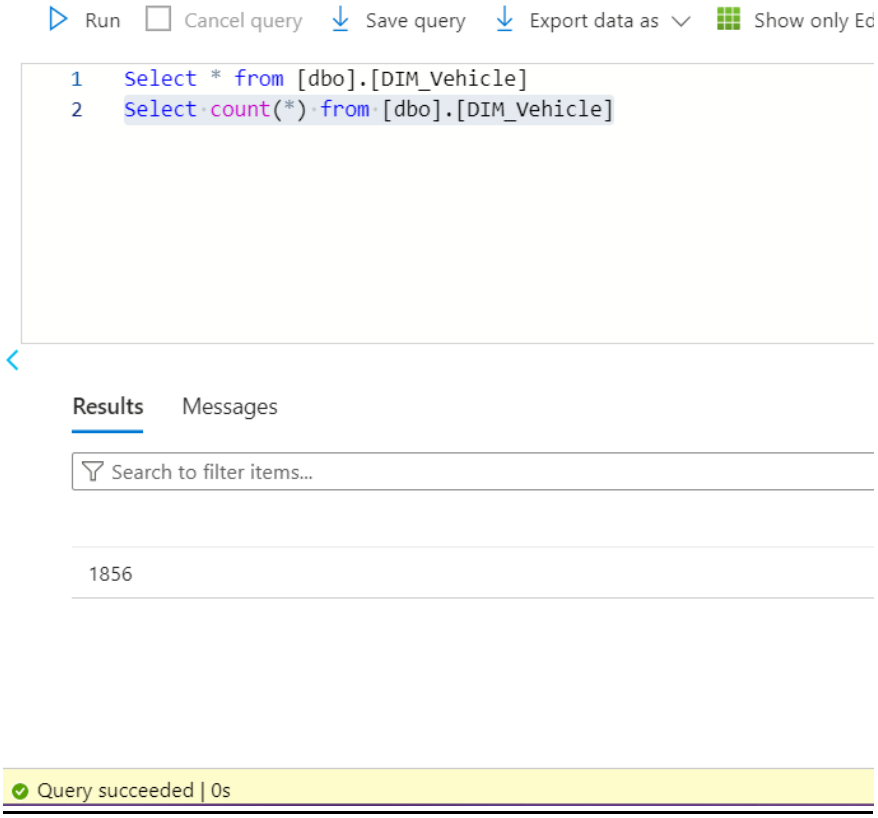
The load component, which is the last stage of the task, loads the mapped and processed data into the target table. The corrected data is pushed to the persistent store in this output, which is designated as "Main" and has a green connection line that ends with a database symbol.

The ChinmayConnection indicates the existence of a named or personalised database connection, most likely set up with the settings and credentials required to connect to the target database containing the "DIM_VEHICLE" table.

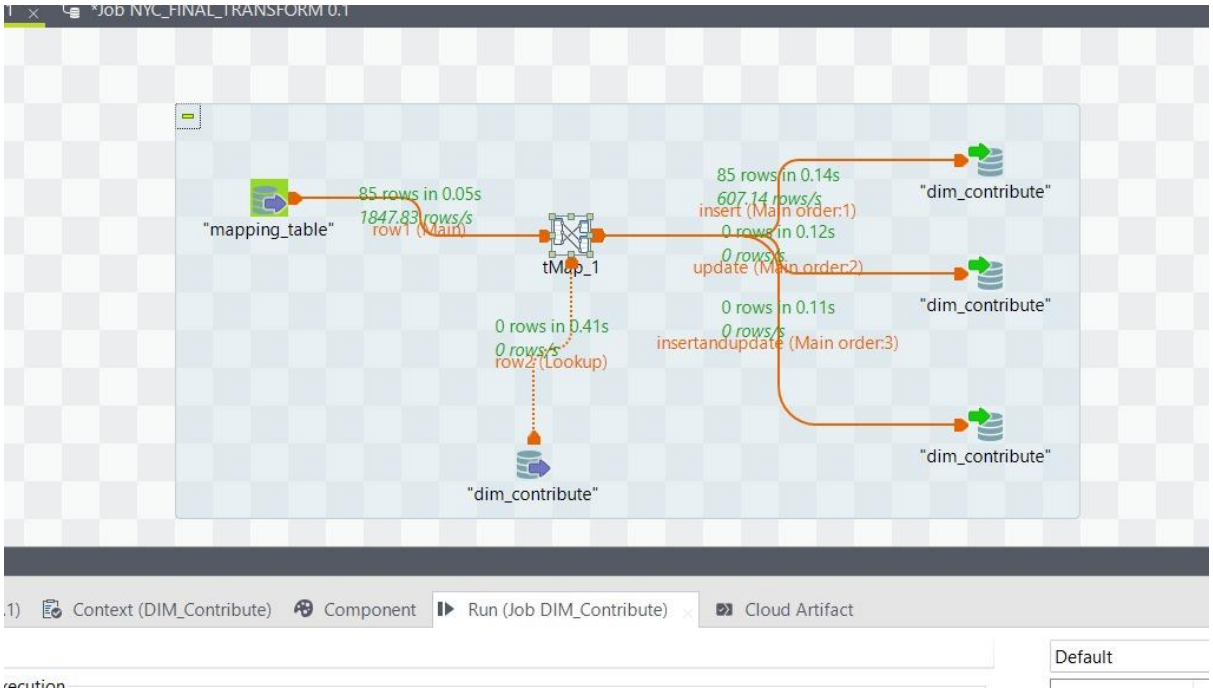
## Dim Vehicle SQL Query Validations




## Counts:



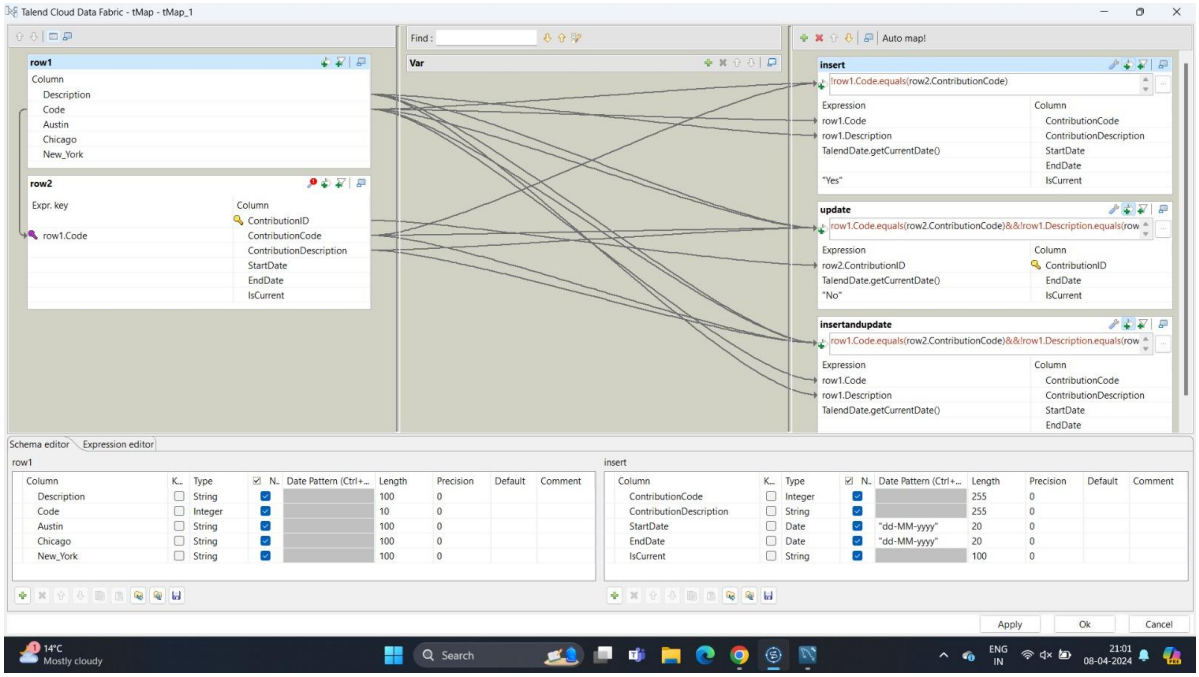
## Contribute Dimension




First Input Component (Mapping_table): As our first input component, we created a "mapping_table". Mapping document codes to individual contribution IDs and providing comprehensive explanations for each was made possible thanks to this table. Our contributions were correctly recognised and documented throughout the transformation process thanks to this mapping.
tMap: A tMap component was used in the transformation stage. tMap played two distinct roles. It started by carrying out the required data transformations to make sure the incoming data complied with our data warehouse model. Second, it oversaw the SCD2 activities, giving us the ability to update records, add new ones, and combine the two operations as required. In order to handle the temporal nature of our contribution data, this was an essential step.
Dim_Contribute: Following transformation, the data was sent to our "dim_contribute" dimension table. The purpose of this table was to record the characteristics of our contributions, as well as any modifications made over time. Supporting intricate queries for temporal trend analysis that our business analysts would execute was essential.

Results: Updates, Inserts, and Inserts with Updates
Three separate flows were carefully created out of our outputs:
Insert Output: The processing of fresh contribution records that had not yet been entered into our dimension table was the sole purpose of this stream.
Update Output: Records that required updates owing to modifications in the donation details were handled by this stream.
Insert and Update Output: Lastly, a smooth record integration was ensured by this combined flow, which handled fresh contributions that also contained changes.

## SCD2:

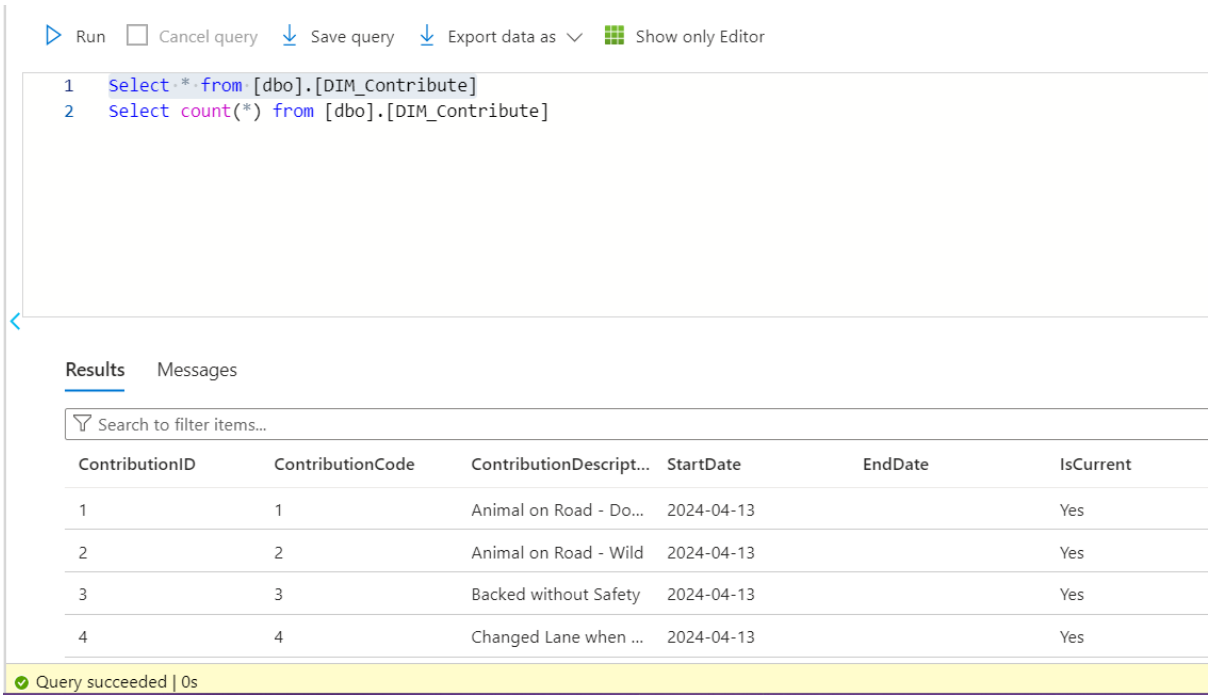



Insert: This action adds new records to the data warehouse. When a new entry is discovered that does not already exist in the warehouse (identified by unique keys), a new record is created with a start date and a "IsCurrent" flag that is normally set to "Yes". This signifies that this version of the record is the current one.

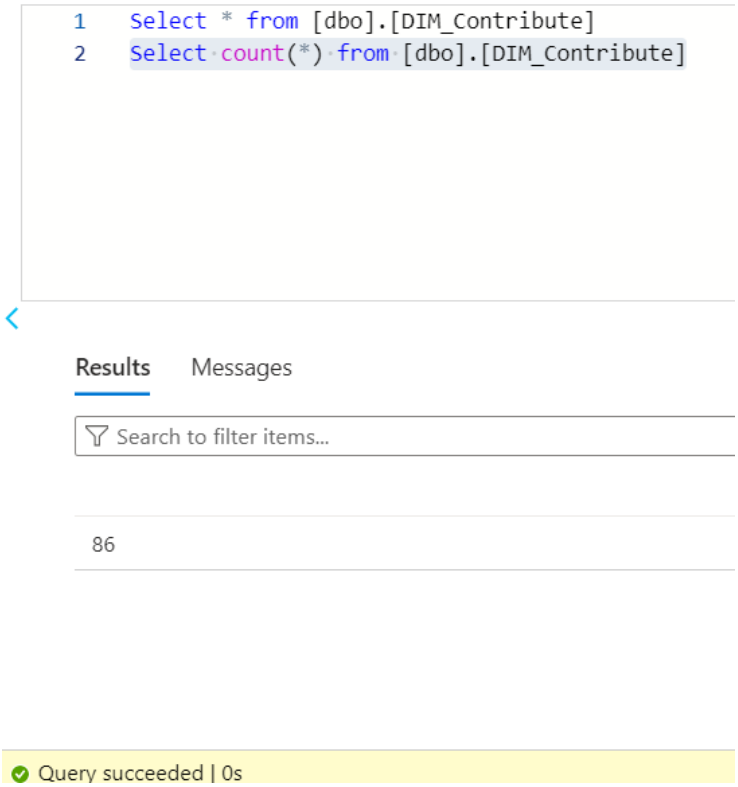
Update: When a change is detected in an existing record, instead of overwriting it, SCD Type 2 requires that the current record's "IsCurrent" flag be set to "No" and the end date be populated to indicate the record's time validity.

Insert and Update: A combo of the previous two actions. When an existing record needs to be updated, the current record is closed (updated) by setting "IsCurrent" to "No" and specifying an end date, and a new record is created with the revised values, a new start date, and "IsCurrent" set to "Yes". This ensures that the history data is retained while the most recent data is available as the current record.

# Dim Contribute SQL Query Validations

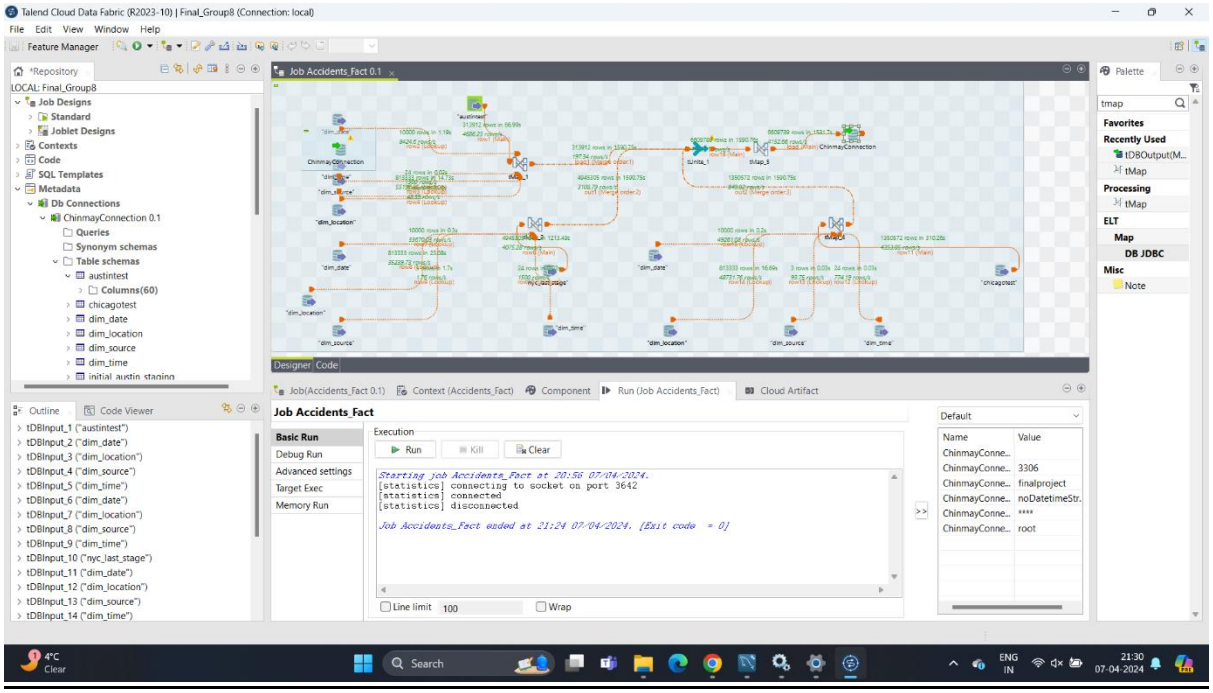



## Counts:



# Fact Loading

## Accident Fact



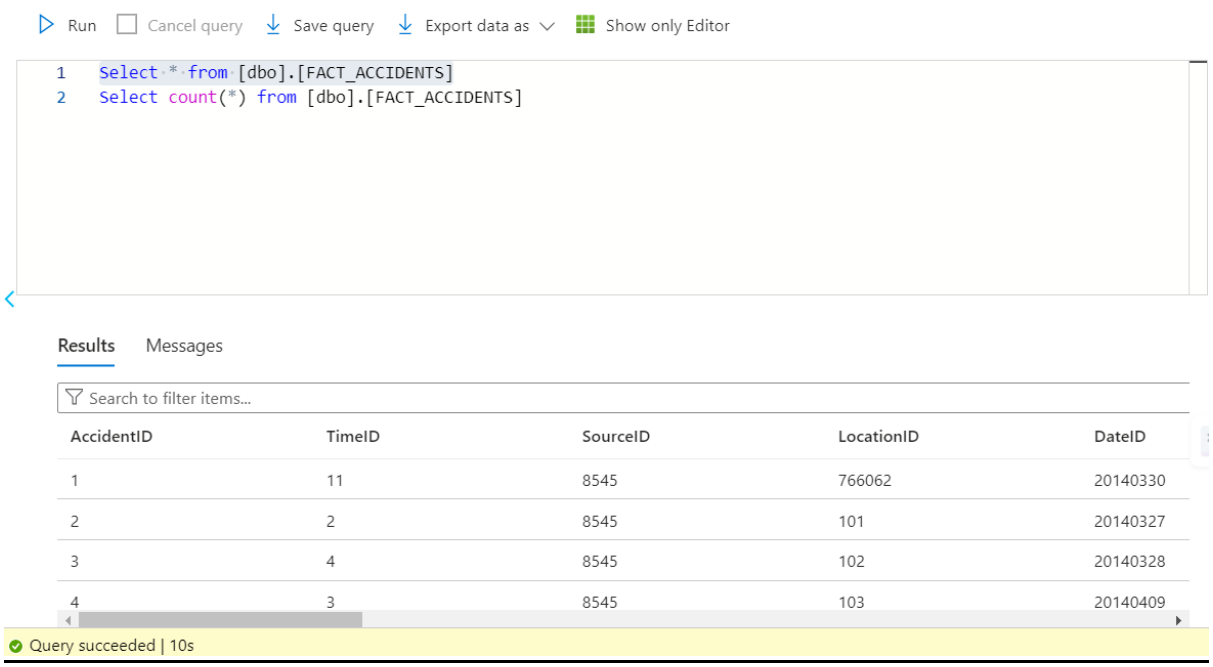

Numerous dimension tables as inputs for current Talend job, {Job_Accidents_Fact}. These are essential for giving the fact table I'm creating context and specifics. Each dimension is represented by a DbInput component, such as time, location, and jurisdiction. The goal of the approach is to combine these many data points to provide a complete picture of the accident data.

It's alignment and modification of the data from each dimension table as I go through the task using different tMap components. Careful mapping is required throughout this procedure to guarantee that the dimensions' keys and attributes correctly match the fact table's fields. This mapping needs to be accurate since it establishes the framework for the multi-dimensional study. It is successfully converting unstructured data into a format that can be effectively queried for meaningful information by doing this.

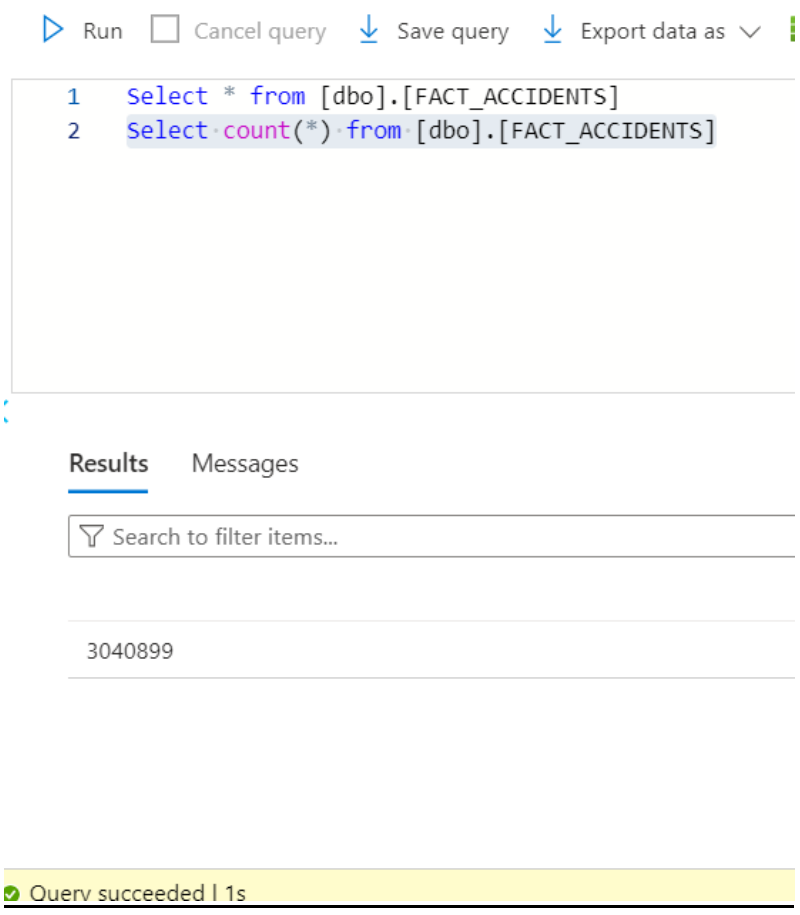
Finally added all of the mapped data to the fact table. This is the point at which everything comes together to form a solid dataset that provides a detailed perspective of every occurrence. An in-depth research and produce reports with this fact table that may otherwise be impossible to see and could provide insights or patterns about incidents. In addition to overseeing the technical parts of ETL procedures, it is our responsibility to make sure the data dealt with is converted into information that is useful for making decisions.

All relevant data from the dimension tables, including jurisdiction, time, and place, are exactly transferred into the Accident Fact database. I have made sure that every entry in the fact table has the appropriate descriptive context by carefully mapping it out. This allows for a comprehensive, multidimensional examination of accident data for enlightening reporting and analytics.

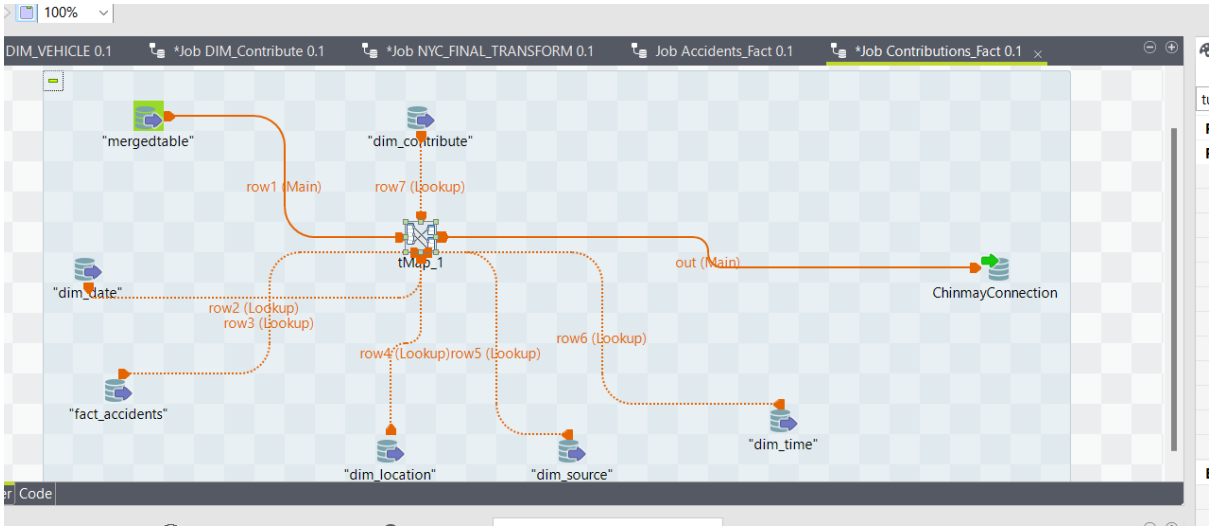
# Fact Accident SQL Query Validations




## Counts:



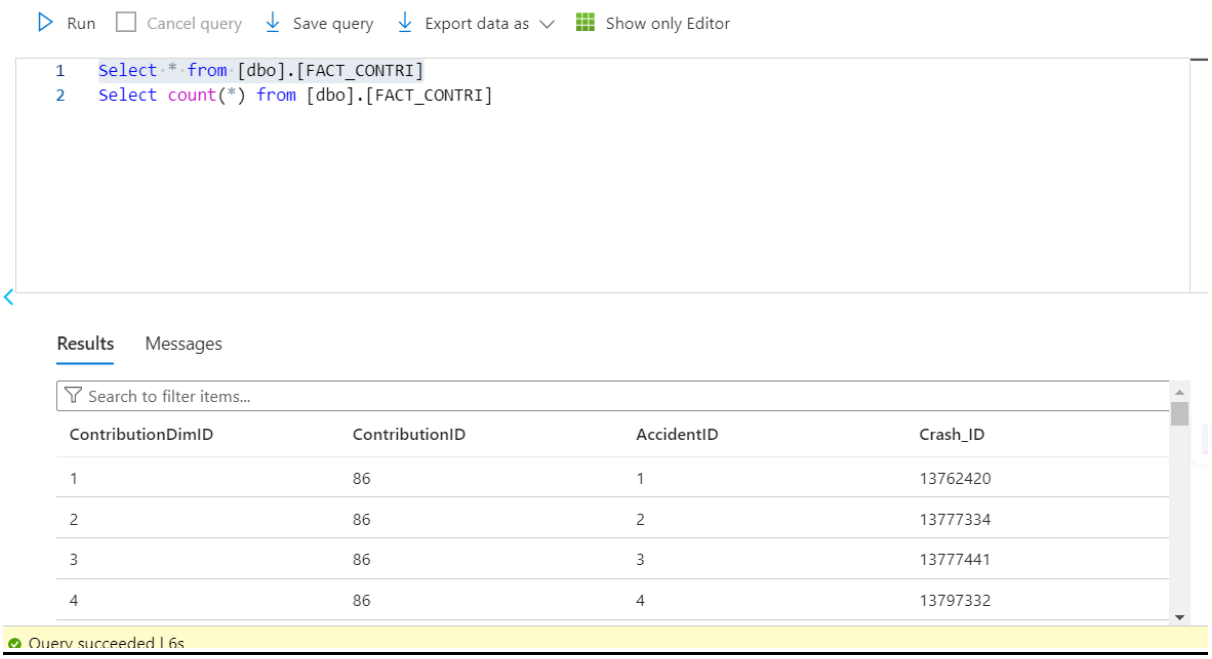
# ContributingFactors Fact



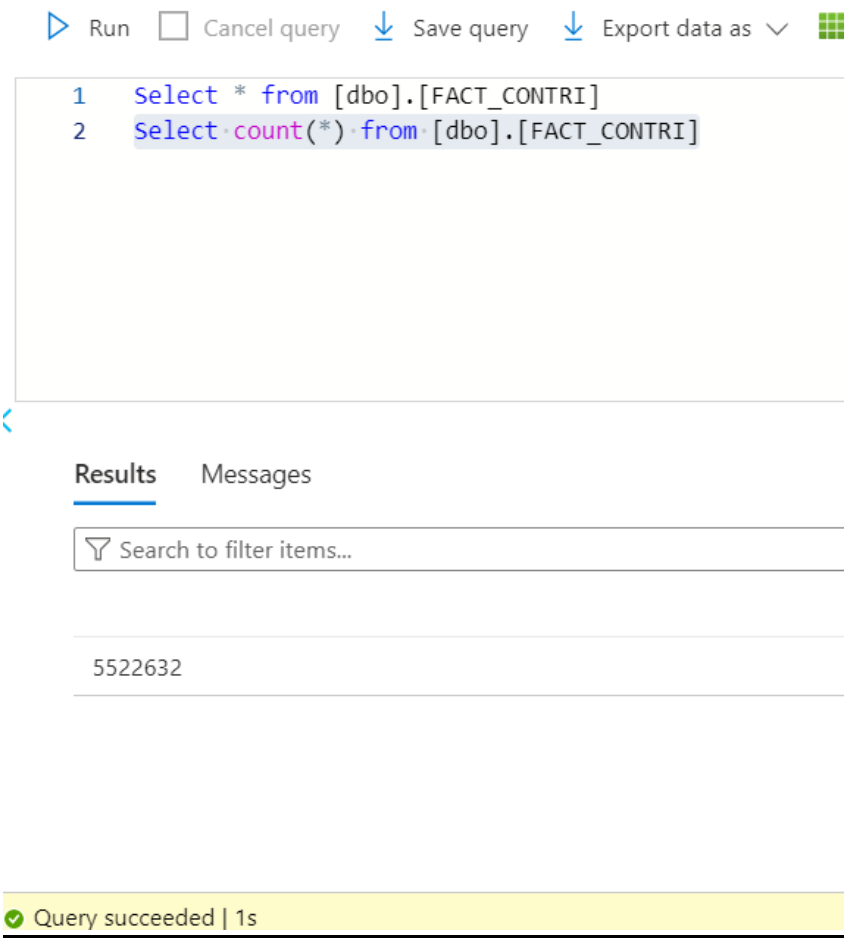

We have arranged a methodical procedure to improve our data warehousing efforts for an extensive traffic accident analysis system in the Talend ETL task that is being shown. The purpose of the task, which is probably called {Job_Accidents_Fact}, is to add rich, multi-dimensional data to a fact database in order to give detailed information on traffic accidents.
Source Dimension Tables: The basis for the context and granularity of the accident data, each dimension table is fed into the task using database input components (DbInput). Among these dimensions are:
dim_date: Gives each accident's date context.
dim_location provides location information, which is essential for spatial analysis.
dim_source: Stores each data record's original source.
dim_time: Provides exact timing details for the events.
These dimensions are crucial to include since they provide the necessary descriptive information that, when combined with the accident details, will allow for the creation of intricate analytical questions.

Transformation and Mapping: The tMap component, which handles the alignment and transformation of data from each dimension table, is essential to this ETL task. Careful mapping is required during the tMap process to guarantee that the keys and attributes from the dimension tables precisely match the anticipated fields in the fact table.

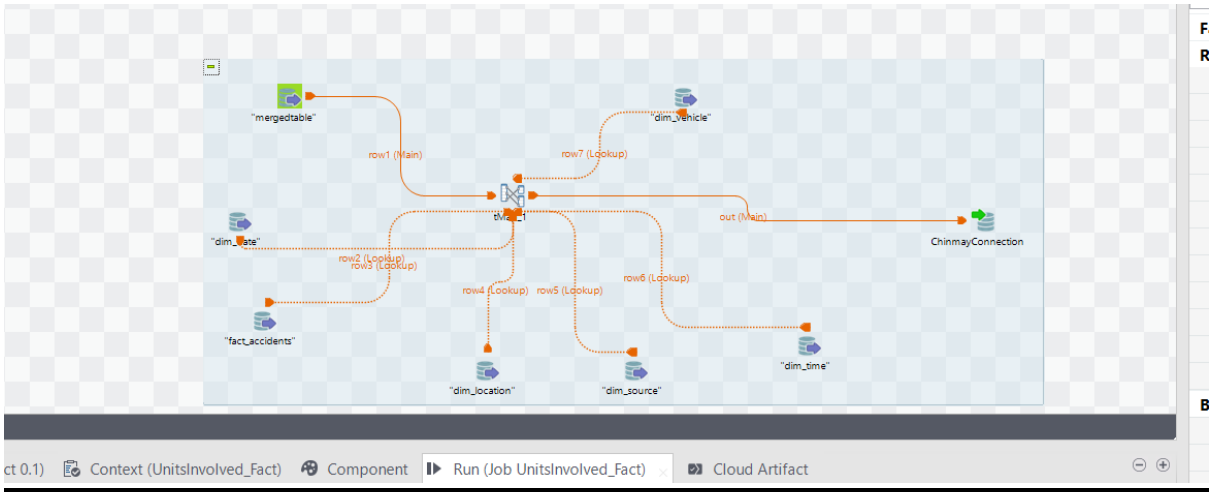
# Fact ContributingFactors SQL Query Validations




## Counts:



# UnitsInvolved Fact



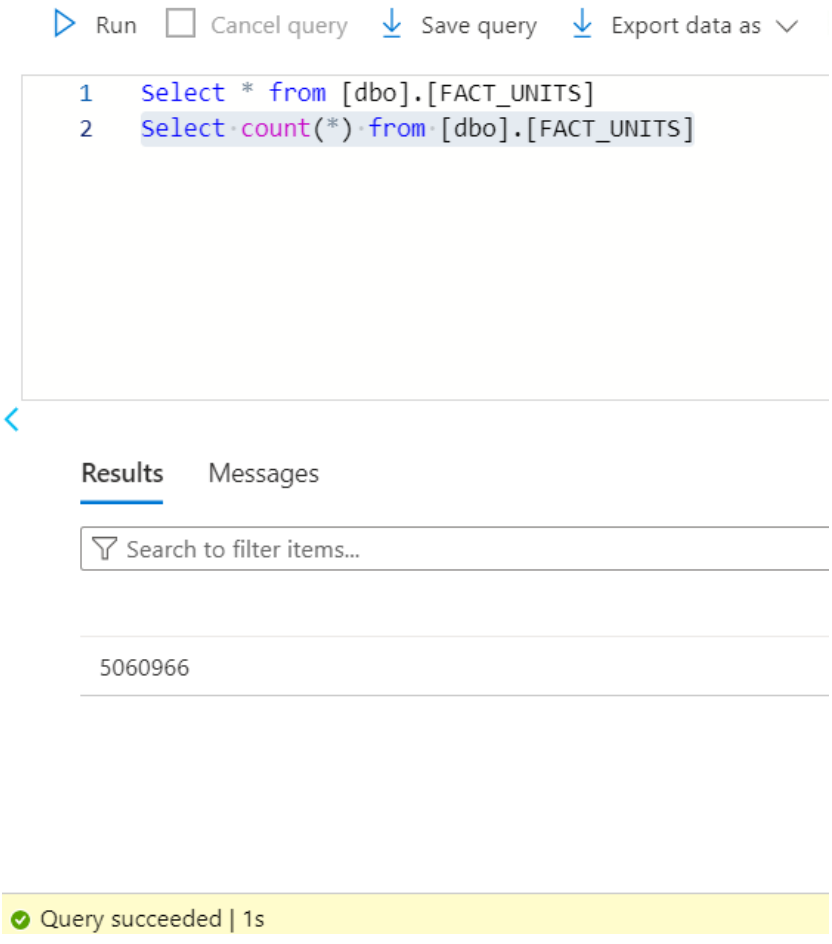

It proposes integrating several data streams into a single, cohesive dataset by beginning with a "mergedtable" component. This combined data probably includes crucial facts regarding a number of different aspects of traffic incidents, including location information, vehicle characteristics, accident details, and time references.

The combined table's flow then splits into several different directions, some of which enter the tMap component directly and others of which serve as lookup flows. In order to provide more context to the primary data stream, certain search flows are essential:
dim_vehicle: Information on the cars involved in the collisions is obtained by this search.
dim_location: Location data is incorporated here, which is necessary for geographical analysis.
dim_source: Provides more details on the data source, such as where and how the accident data was gathered.
dim_time: Provides information about time that is useful for temporal investigation of incidents.
The information from the different lookups and the "mergedtable" is probably being mapped and changed in the tMap component to match the target fact table's schema. The operation's central component, the tMap, is where intricate transformations and data alignments take place.

The ChinmayConnection, which seems to be a named database connection, receives the output of tMap and is prepared to load the processed data into a target table.

## Fact UnitsInvolved SQL Query Validations


## Counts:

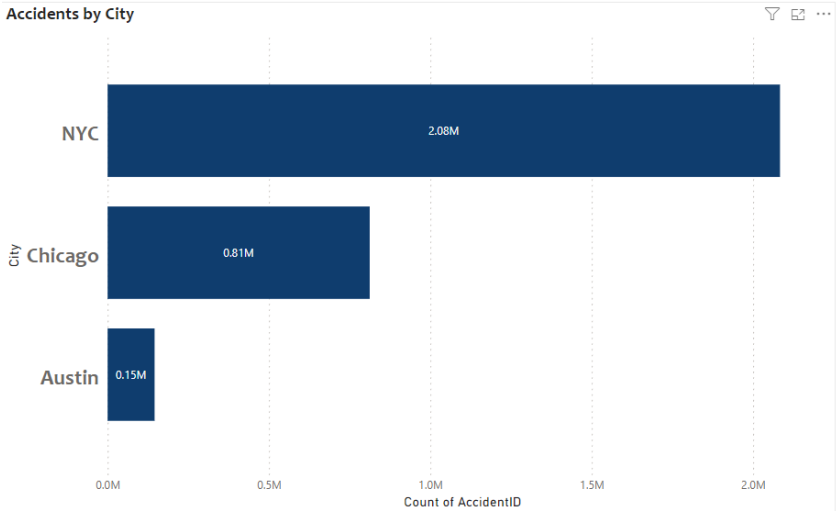


## Part 3

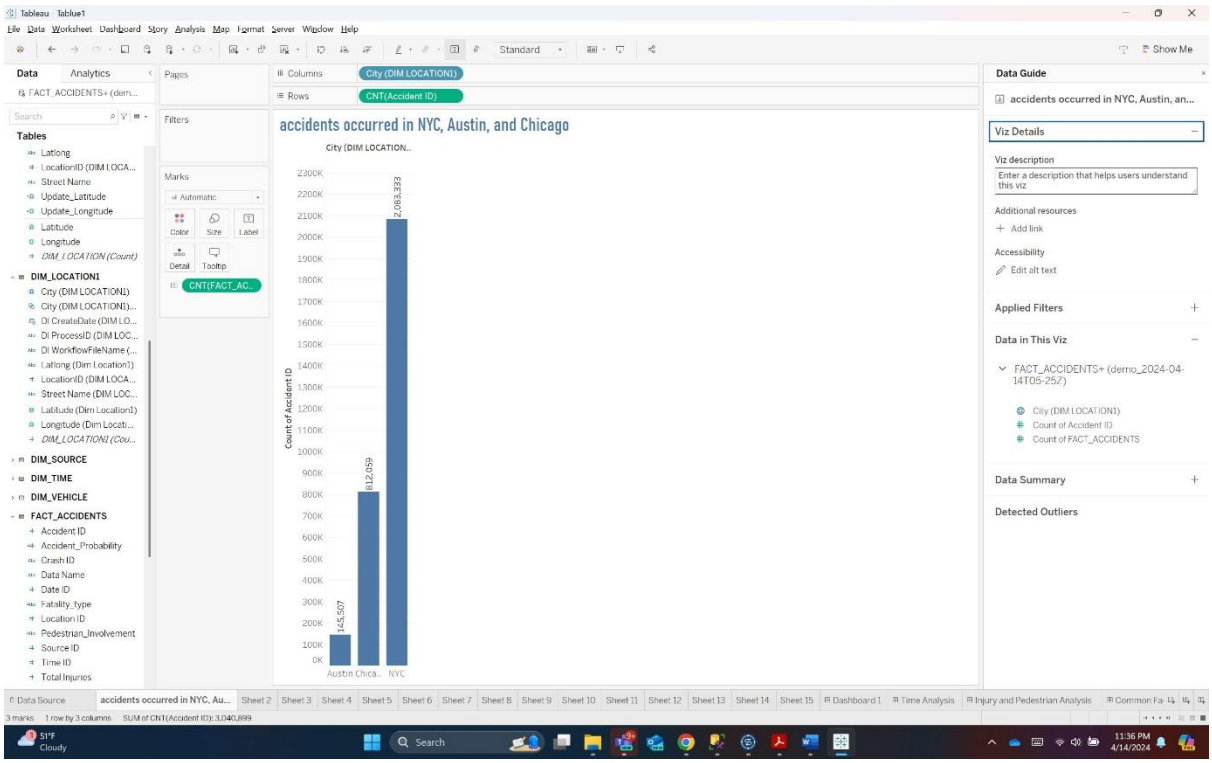
### Business Questions with their Data Visualisations and Validations

1. How many accidents occurred in NYC, Austin, and Chicago?

**ACCIDENTS IN NYC, CHICAGO AND AUSTIN**




3.04M



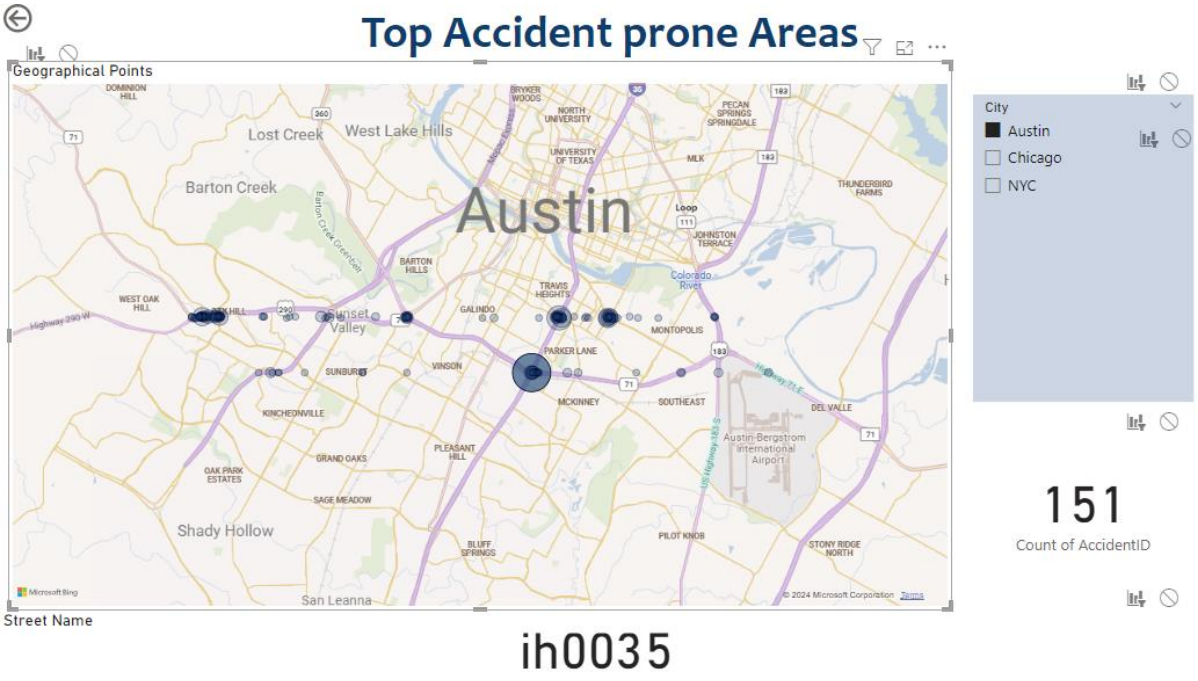
```sql
SELECT City, COUNT(AccidentID) AS NumberOfAccidents
FROM Fact_Accidents
JOIN Dim_Location ON Fact_Accidents.LocationID = Dim_Location.LocationID
WHERE City IN ('NYC', 'Austin', 'Chicago')
GROUP BY City;
```

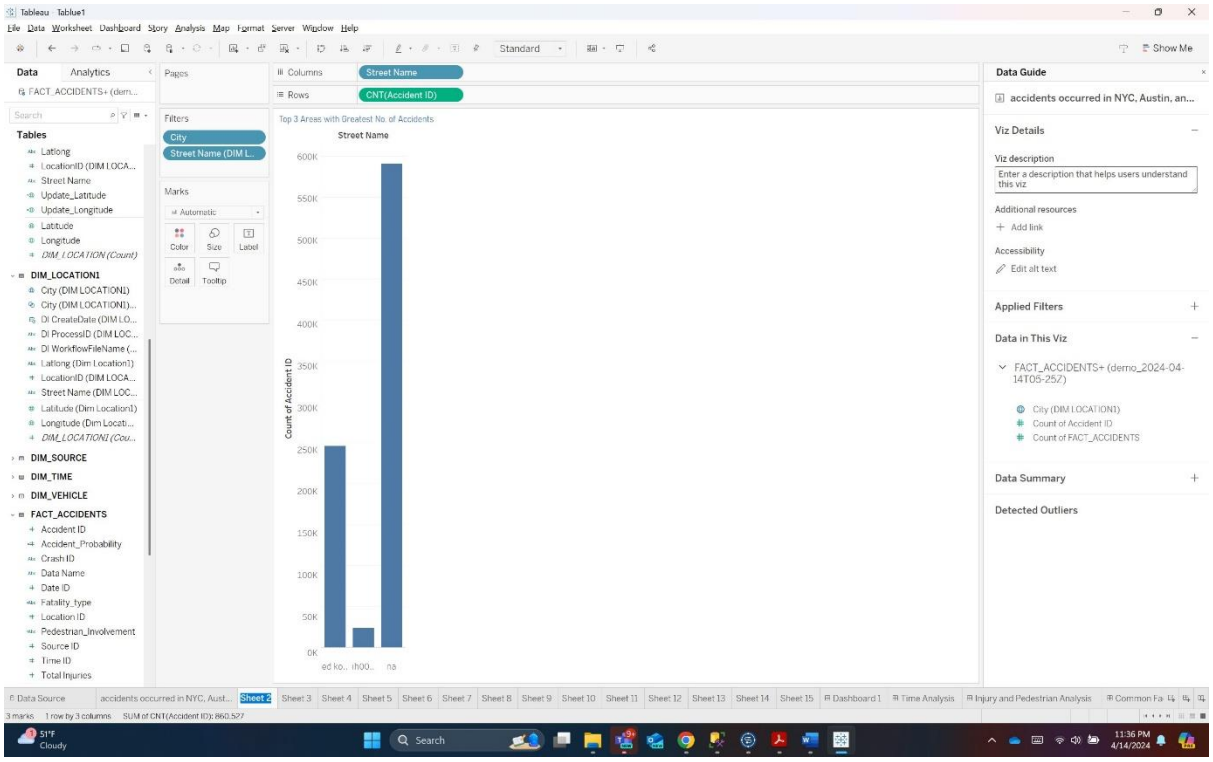
Results | Messages

Search to filter items...

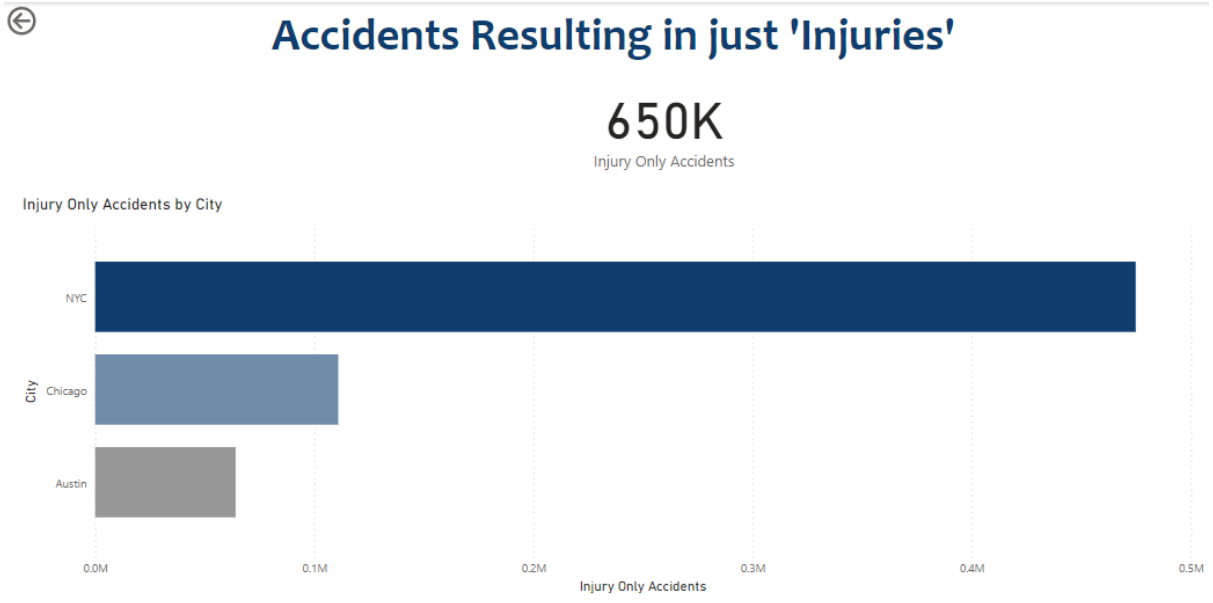
| City | NumberOfAccidents |
|---|---|
| Chicago | 812059 |
| Austin | 145507 |
| NYC | 2083333 |

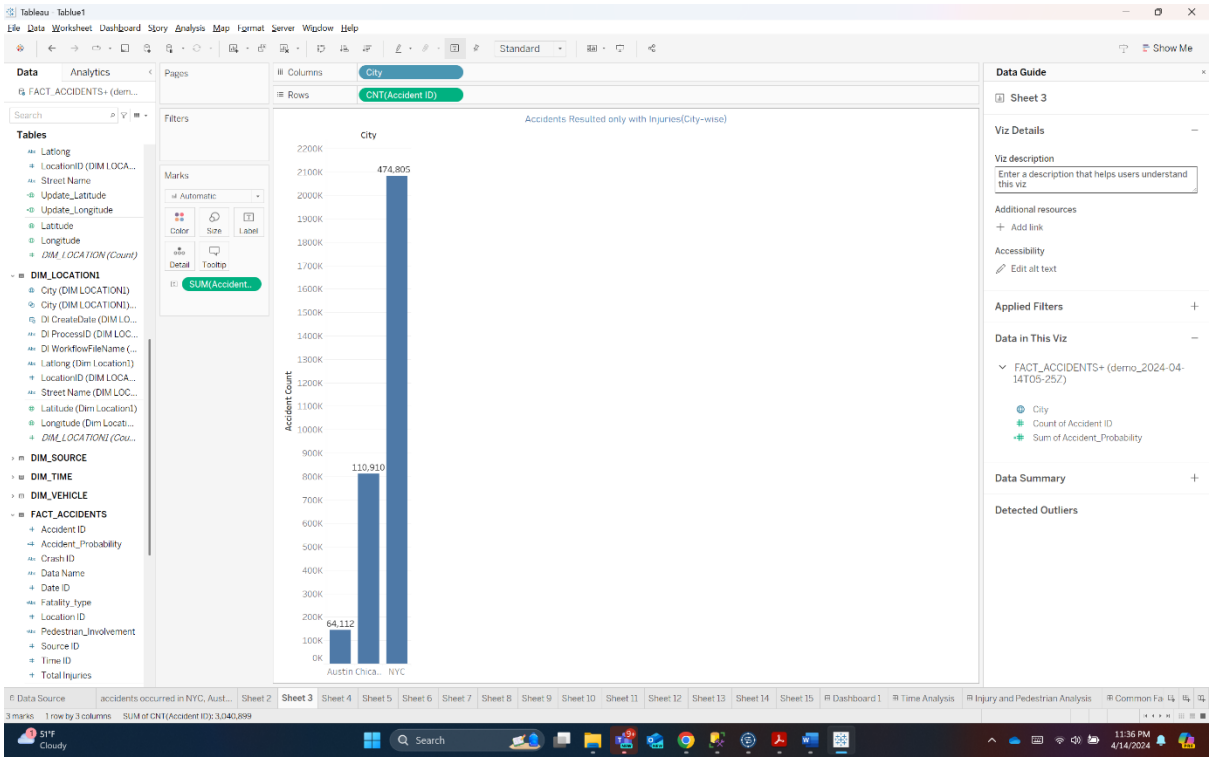
2. Which areas in the three cities experienced the most accidents? (Top three areas of each city)






3. How many accidents resulted just in injuries? (Overall and by city).





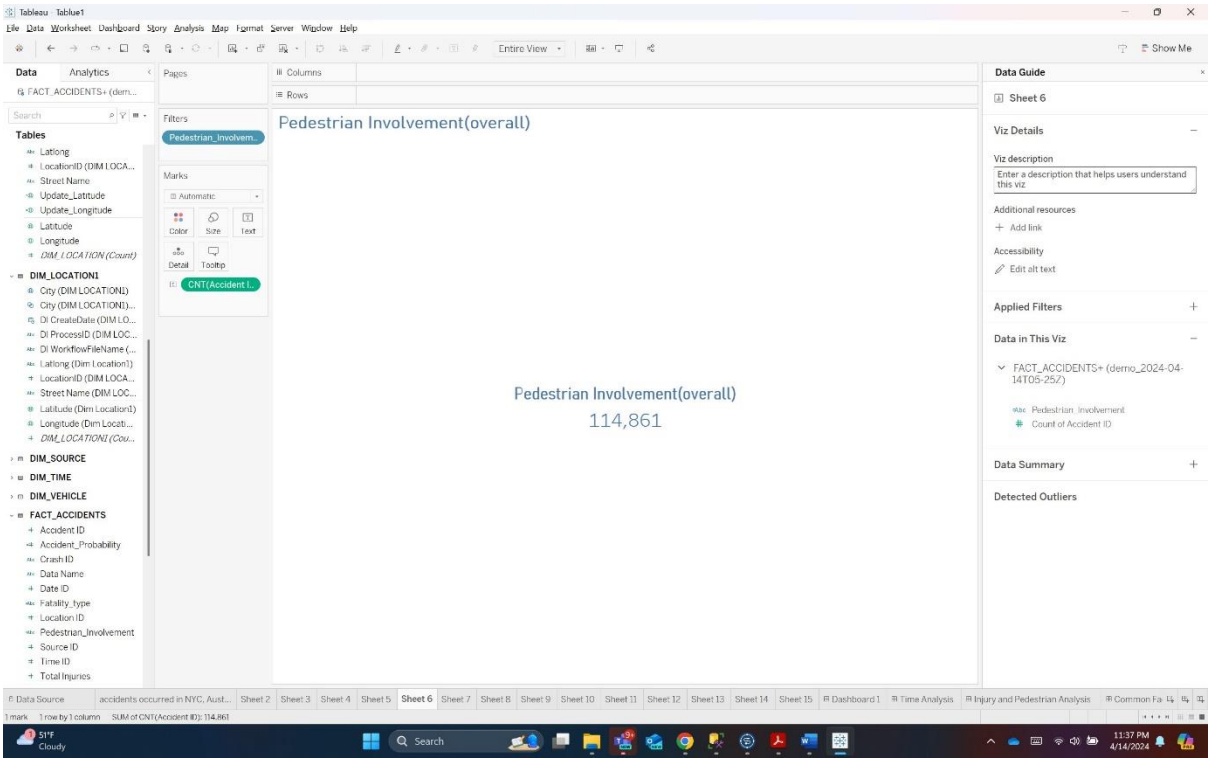

```sql
SELECT Dim_Location.City, COUNT(Fact_Accidents.AccidentID) AS NumberOfInjuryOnlyAccidents
FROM Fact_Accidents
JOIN Dim_Location ON Fact_Accidents.LocationID = Dim_Location.LocationID
WHERE Fact_Accidents.Total_Injuries > 0 AND Fact_Accidents.Total_Deaths = 0
GROUP BY Dim_Location.City;
```

Results | Messages

Search to filter items...

| City | NumberOfInjuryOnlyAccidents |
|---|---|
| Chicago | 110910 |
| Austin | 64112 |
| NYC | 474805 |

4. How often are pedestrians involved in accidents? (Overall and by city)

Overall:

Run | Cancel query | Save query | Export data as | Show only Editor

```
1  SELECT
2   COUNT(*) AS PedistrianInvolvedAccidents
3  FROM dbo.FACT_ACCIDENTS
4  WHERE Pedistrian_Injury > 0 OR Pedestrian_Killed > 0;
5
```

Results | Messages

Search to filter items...

| PedistrianInvolvedAccidents |
|---|
| 114861 |

By city:

Run | Cancel query | Save query | Export data as | Show only Editor

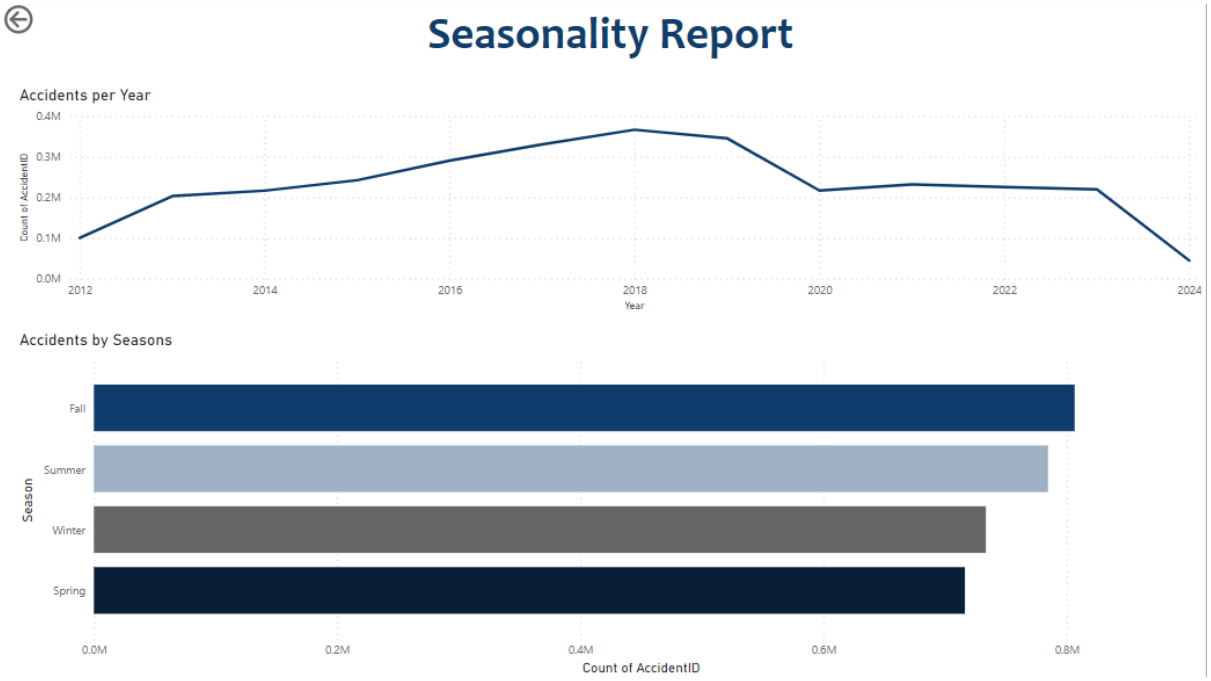
```
1  SELECT
2   dl.City,
3   COUNT(*) AS PedestrianInvolvedAccidents
4  FROM dbo.FACT_ACCIDENTS fa
5  JOIN dbo.DIM_LOCATION dl ON fa.LocationID = dl.LocationID
6  WHERE fa.Pedistrian_Injury > 0 OR fa.Pedestrian_Killed > 0
7  GROUP BY dl.City;
8
```

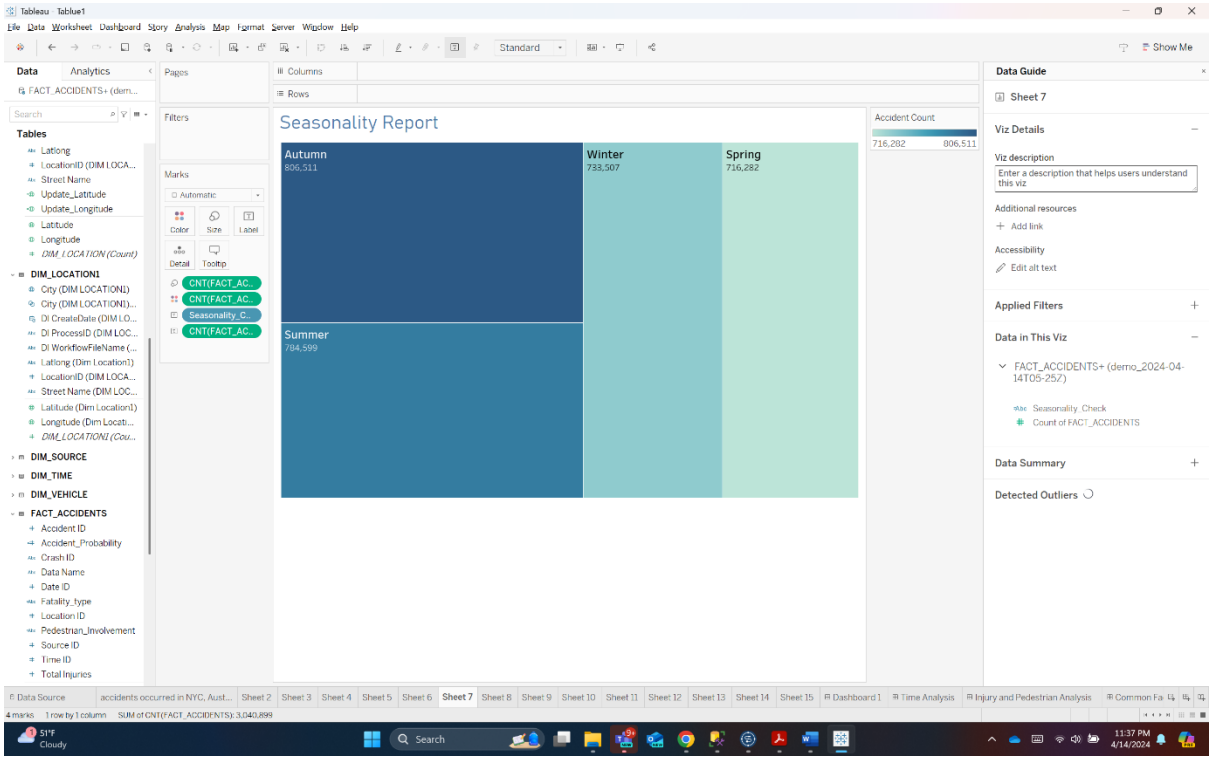
Results | Messages

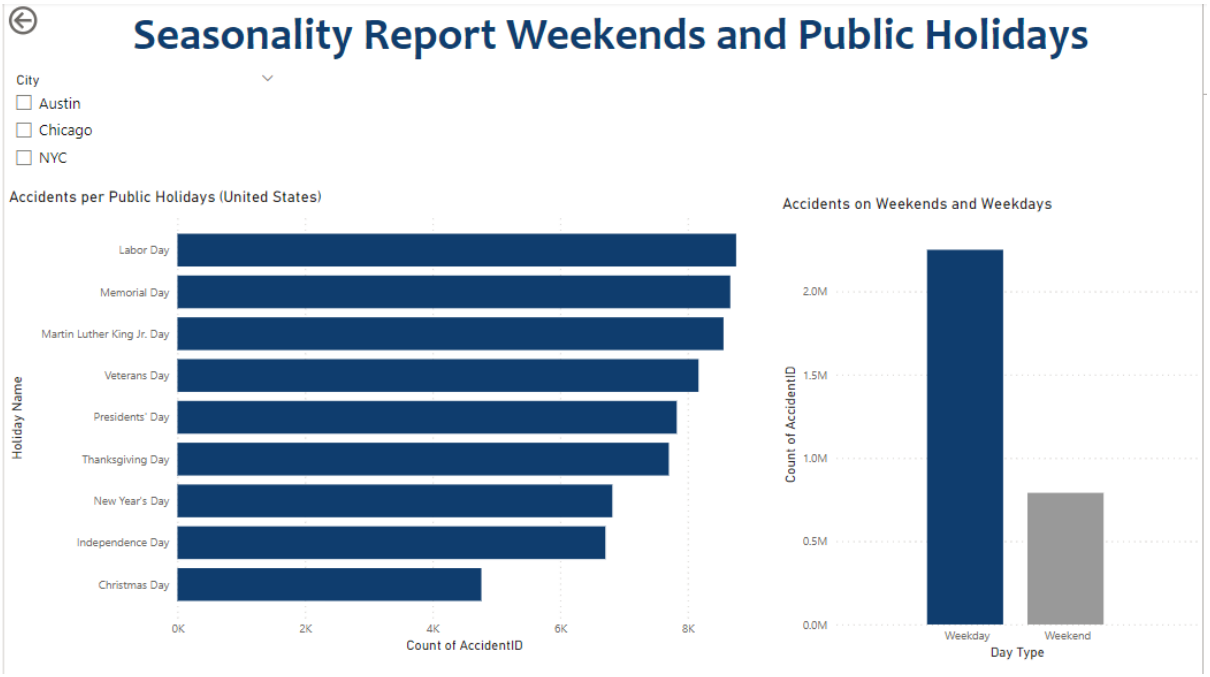
Search to filter items...

| City | PedestrianInvolvedAccidents |
|---|---|
| Austin | 934 |
| NYC | 113927 |

5. When do most accidents happen? (Seasonality report)




Seasonality Report Weekends and Public Holidays
City
Austin
Chicago
NYC
Accidents per Public Holidays (United States)
Holiday Name
Labor Day
Memorial Day
Martin Luther King Jr. Day
Veterans Day
Presidents' Day
Thanksgiving Day
New Year's Day
Independence Day
Christmas Day
0K
2K
4K
6K
8K
Count of AccidentID
Accidents on Weekends and Weekdays
Count of AccidentID
2.0M
1.5M
1.0M
0.5M
0.0M
Weekday
Weekend
Day Type

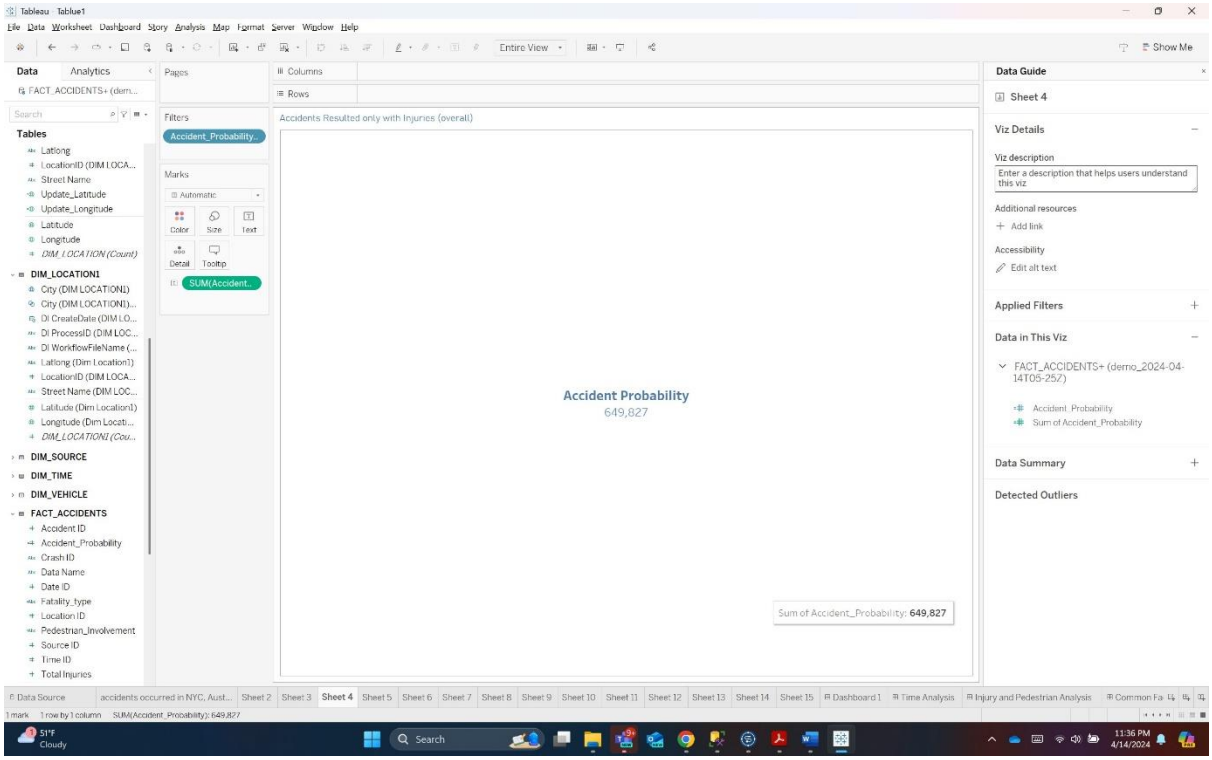


Tableau Tablue1
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Entire View
Show Me
Data
Analytics
FACT_ACCIDENTS+ (dem...
Tables
Latlong
LocationID (DIM LOCA...
Street Name
Update_Latitude
Update_Longitude
Latitude
Longitude
DIM_LOCATION (Count)
DIM_LOCATION1
City (DIM LOCATION1)
City (DIM LOCATION1)...
DI CreateDate (DIM LO...
DI ProcessID (DIM LOC...
DI WorkflowFileName (...
Latlong (Dim Location1)
LocationID (DIM LOCA...
Street Name (DIM LOC...
Latitude (Dim Location1)
Longitude (Dim Locati...
DIM_LOCATION1 (Cou...
DIM_SOURCE
DIM_TIME
DIM_VEHICLE
FACT_ACCIDENTS
Accident ID
Accident_Probability
Crash ID
Data Name
Date ID
Fatality_type
Location ID
Pedestrian_Involvement
Source ID
Time ID
Total Injuries
Pages
Columns
Rows
Filters
Accident_Probability...
Marks
Automatic
Color
Size
Text
Detail
Tooltip
SUM(Accident...
Accidents Resulted only with Injuries (overall)
Accident Probability
649,827
Sum of Accident_Probability: 649,827
Data Guide
Sheet 4
Viz Details
Viz description
Enter a description that helps users understand this viz
Additional resources
Add link
Accessibility
Edit alt text
Applied Filters
Data in This Viz
FACT_ACCIDENTS+ (demo_2024-04-14T05-25Z)
Accident_Probability
Sum of Accident_Probability
Data Summary
Detected Outliers
Data Source
accidents occurred in NYC, Aust...
Sheet 2
Sheet 3
Sheet 4
Sheet 5
Sheet 6
Sheet 7
Sheet 8
Sheet 9
Sheet 10
Sheet 11
Sheet 12
Sheet 13
Sheet 14
Sheet 15
Dashboard 1
Time Analysis
Injury and Pedestrian Analysis
1 mark
1 row by 1 column
SUM(Accident_Probability): 649,827
51°F
Cloudy
Search
11:36 PM
4/14/2024

Run | Cancel query | Save query | Export data as | Show only Editor

```
SELECT
 dd.Quater,
 dd.Month,
 COUNT(*) AS AccidentCount
FROM dbo.FACT_ACCIDENTS fa
JOIN dbo.DIM_DATE dd ON fa.DateID = dd.DateID
GROUP BY dd.Quater, dd.Month
ORDER BY dd.Quater, dd.Month
OFFSET 0 ROWS FETCH NEXT 1000 ROWS ONLY;
```

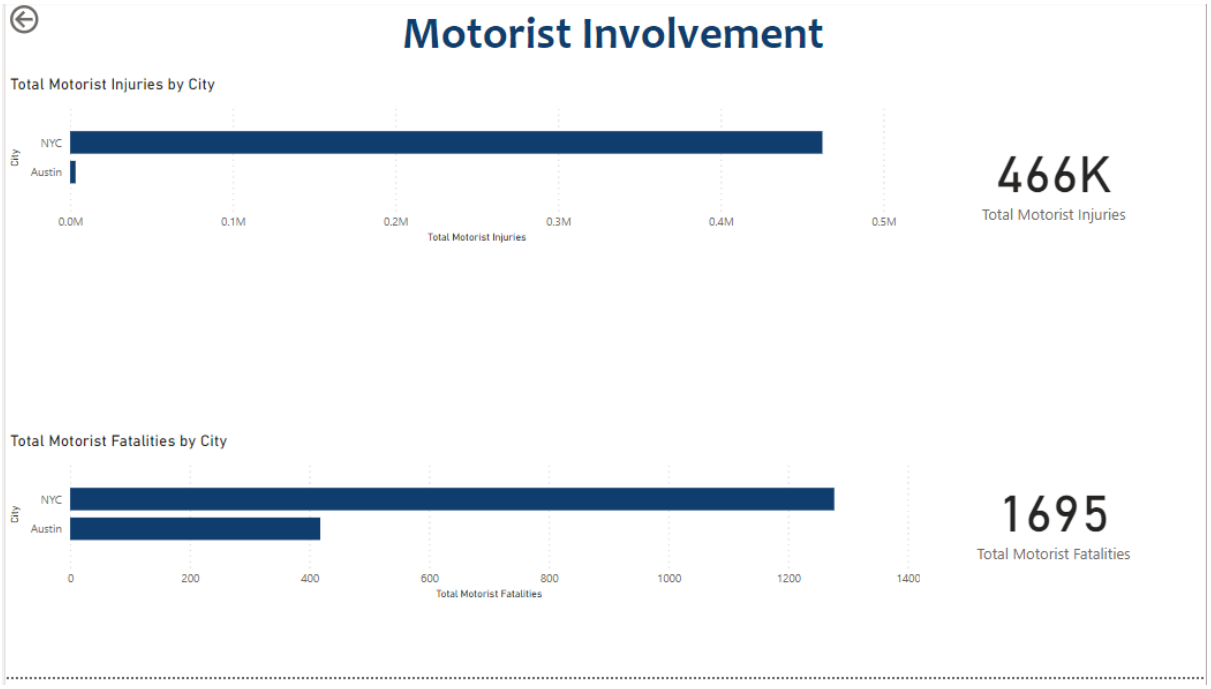
Results | Messages

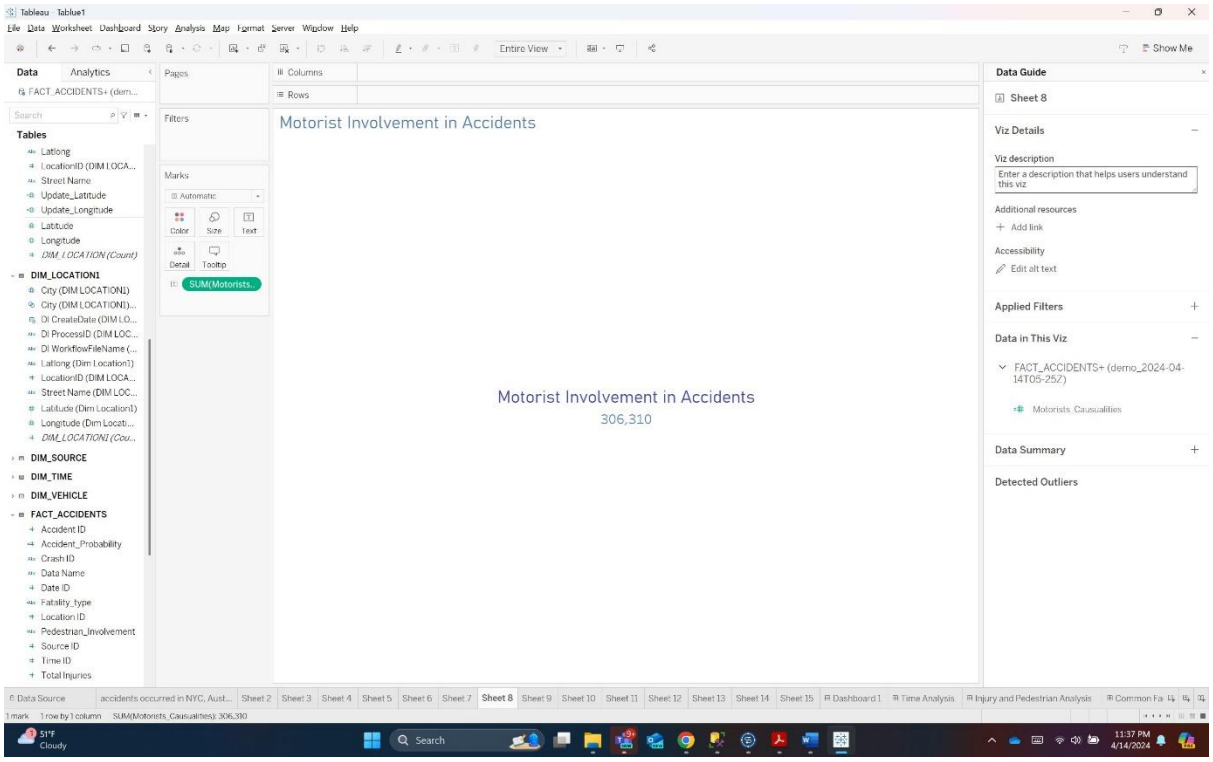
Search to filter items...

| Quater | Month | AccidentCount |
|---|---|---|
| 1 | 1 | 242206 |
| 1 | 2 | 228363 |
| 1 | 3 | 245370 |
| 2 | 4 | 219393 |

Query succeeded | 1s

## 6. How many motorists are injured or killed in accidents? (Overall and by city)






# Overall

```
1  SELECT
2  | SUM(Motorist_Injury) AS TotalMotoristInjured,
3  | SUM(Motorist_Killed) AS TotalMotoristKilled
4  FROM dbo.FACT_ACCIDENTS
5  WHERE Motorist_Injury <> -999 AND Motorist_Killed <> -999;
6
```

Results | Messages

Search to filter items...

| TotalMotoristInjured | TotalMotoristKilled |
|---|---|
| 465787 | 1695 |

Query succeeded | 1s

# By city

Run | Cancel query | Save query | Export data as | Show only Editor

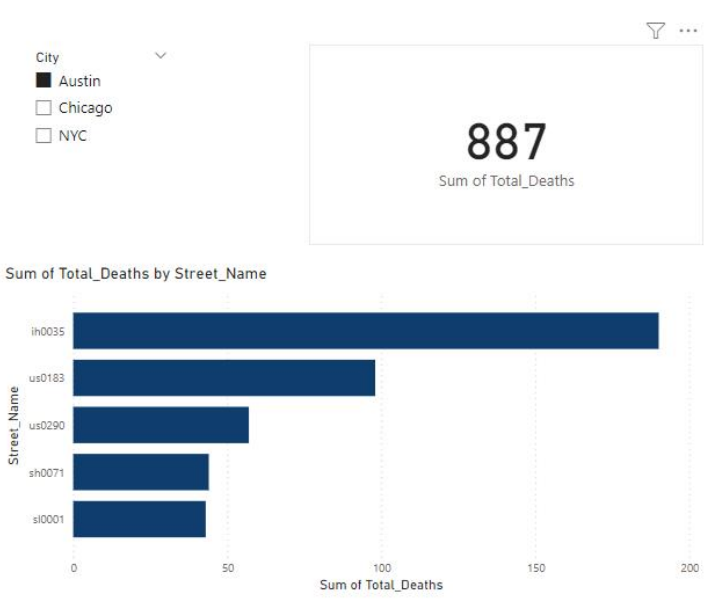
```sql
SELECT
 l.City,
 SUM(CASE WHEN fa.Motorist_Injury <> -999 THEN fa.Motorist_Injury ELSE 0 END) AS TotalMotoristInjured,
 SUM(CASE WHEN fa.Motorist_Killed <> -999 THEN fa.Motorist_Killed ELSE 0 END) AS TotalMotoristKilled
FROM dbo.FACT_ACCIDENTS fa
JOIN dbo.DIM_LOCATION l ON fa.LocationID = l.LocationID
GROUP BY l.City;
```

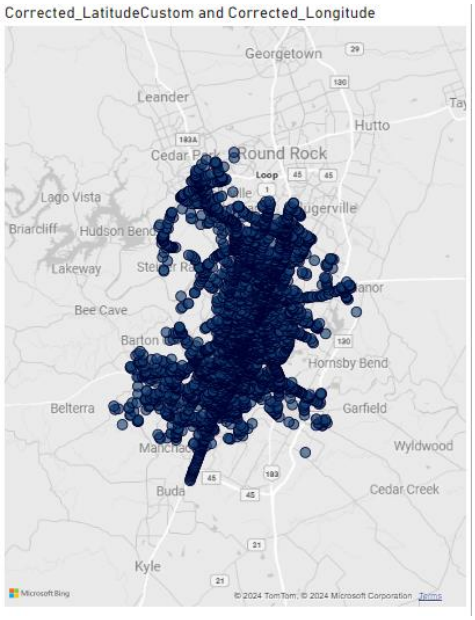
Results | Messages

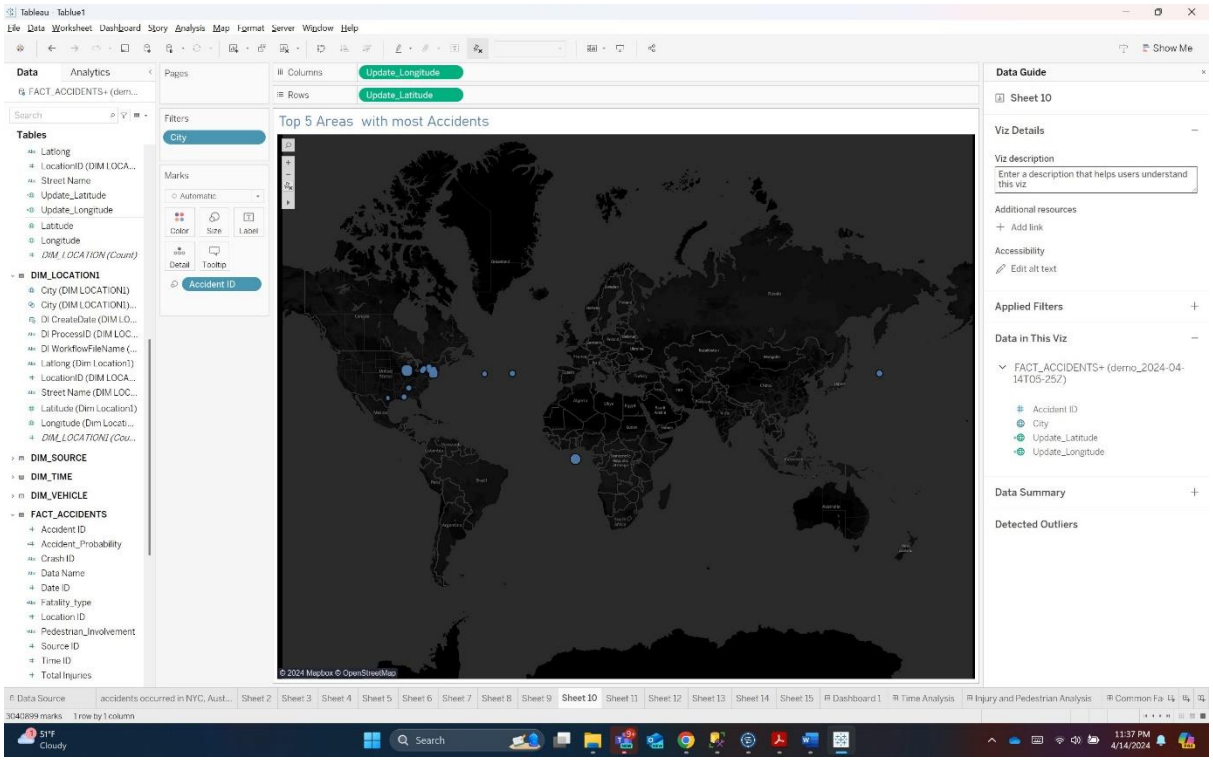
Search to filter items...

| City | TotalMotoristInjured | TotalMotoristKilled |
|---|---|---|
| Chicago | 0 | 0 |
| Austin | 3332 | 418 |
| NYC | 462455 | 1277 |

## 7. Which top 5 areas in 3 cities have the most fatal number of accidents?





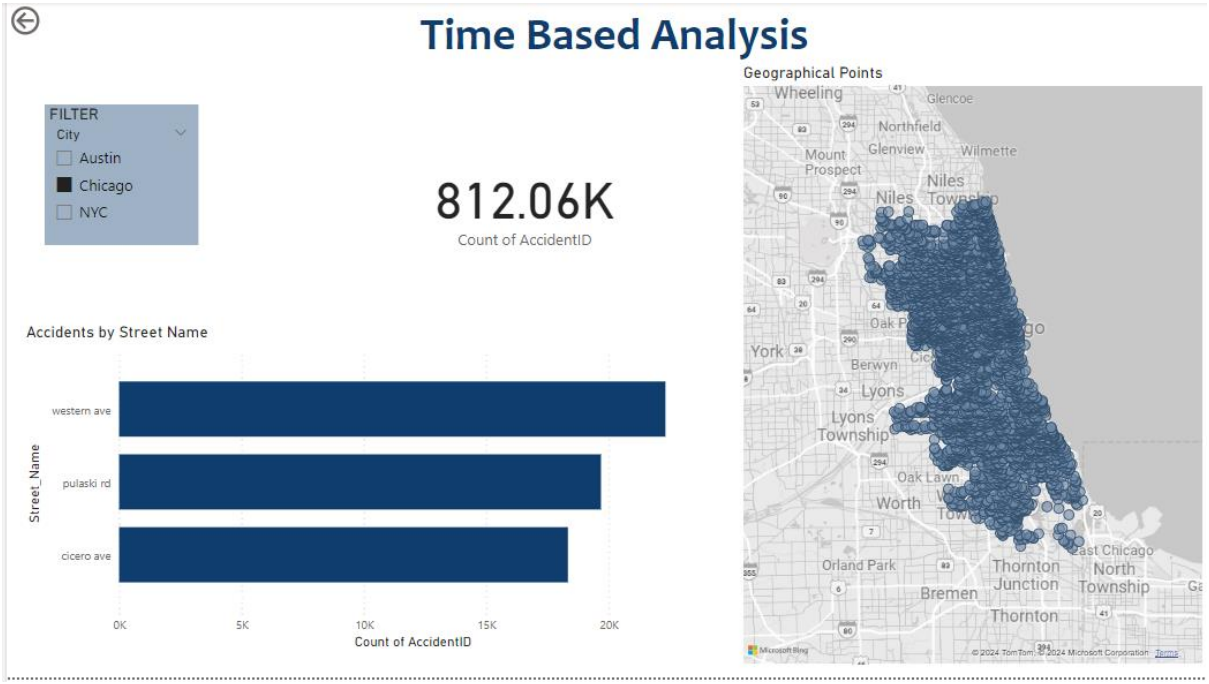

```sql
SELECT
  l.City,
  l.Street_Name AS Area,
  SUM(fa.Total_Deaths) AS FatalAccidents
FROM dbo.FACT_ACCIDENTS fa
JOIN dbo.DIM_LOCATION l ON fa.LocationID = l.LocationID
GROUP BY l.City, l.Street_Name
ORDER BY l.City, FatalAccidents DESC
OFFSET 0 ROWS FETCH NEXT 5 ROWS ONLY;
```

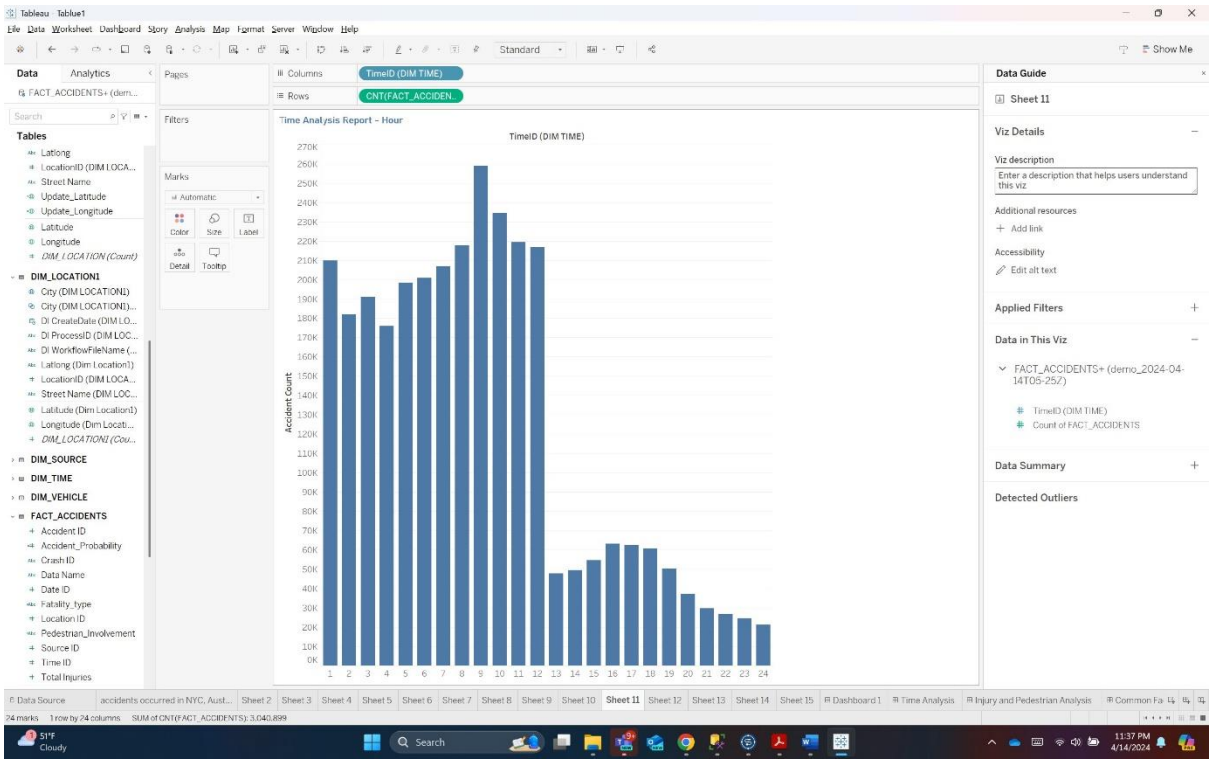
Results Messages

| City | Area | FatalAccidents |
|---|---|---|
| Austin | ih0035 | 190 |
| Austin | us0183 | 98 |
| Austin | us0290 | 57 |
| Austin | sh0071 | 44 |
| Austin | sl0001 | 43 |

Query succeeded | 2s

8. Time-based analysis of accidents (Time of the day, day of the week, weekdays or weekends)

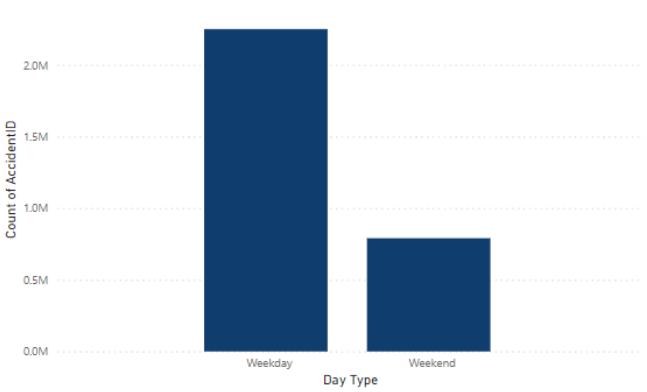


# Time Based Analysis

| Day Time | Friday | Monday | Saturday | Sunday | Thursday | Tuesday | Wednesday | Total |
|---|---|---|---|---|---|---|---|---|
| 8 AM | 42571 | 40175 | 26501 | 21302 | 42808 | 43142 | 42444 | 258943 |
| 9 AM | 38260 | 34830 | 28306 | 22210 | 37443 | 37272 | 36208 | 234529 |
| 10 AM | 35265 | 30570 | 31249 | 24316 | 33297 | 32604 | 32191 | 219492 |
| 7 AM | 35724 | 33014 | 23361 | 19995 | 34985 | 35289 | 35433 | 217801 |
| 11 AM | 35566 | 29378 | 33030 | 25333 | 31937 | 31319 | 30168 | 216731 |
| 12 AM | 31594 | 28992 | 32992 | 31649 | 28938 | 27840 | 27852 | 209857 |
| 6 AM | 33034 | 30559 | 24745 | 22315 | 31811 | 32130 | 32277 | 206871 |
| 5 AM | 31719 | 28958 | 25973 | 25086 | 29902 | 29727 | 29501 | 200866 |
| 4 AM | 30417 | 27893 | 28963 | 28623 | 27796 | 27218 | 27417 | 198329 |
| 2 AM | 28925 | 25671 | 30371 | 29980 | 25796 | 24922 | 25370 | 191037 |
| 1 AM | 26904 | 24637 | 30066 | 29352 | 24128 | 23485 | 23384 | 181956 |
| 3 AM | 27077 | 24005 | 26480 | 27164 | 24073 | 23355 | 23861 | 176015 |
| 3 PM | 10825 | 9227 | 8050 | 6375 | 9703 | 9562 | 9359 | 63101 |
| 4 PM | 10774 | 9143 | 7633 | 6098 | 9753 | 9730 | 9321 | 62452 |
| 5 PM | 10330 | 8572 | 7387 | 5918 | 9487 | 9741 | 9423 | 60858 |
| 2 PM | 9056 | 7580 | 8250 | 6586 | 7643 | 7709 | 7932 | 54756 |
| 6 PM | 8694 | 6618 | 6652 | 5501 | 7719 | 7611 | 7415 | 50210 |
| 1 PM | 8030 | 6752 | 8002 | 6209 | 6779 | 6898 | 6934 | 49604 |
| 12 PM | 7525 | 6634 | 7860 | 5942 | 6633 | 6678 | 6670 | 47942 |
| 7 PM | 6528 | 4562 | 5820 | 4702 | 5344 | 5048 | 5086 | 37090 |
| **Total** | **487835** | **429002** | **421939** | **368737** | **449692** | **443249** | **440445** | **3040899** |

Count of AccidentID by Day Type

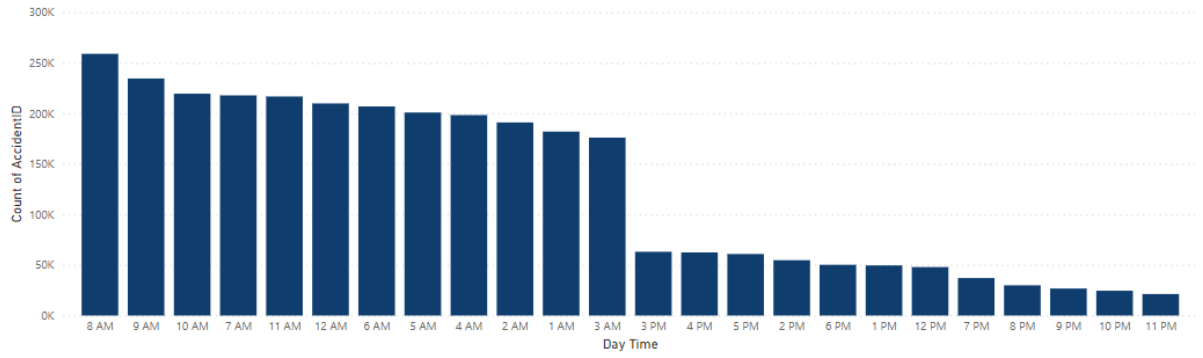



Count of AccidentID by Day Time

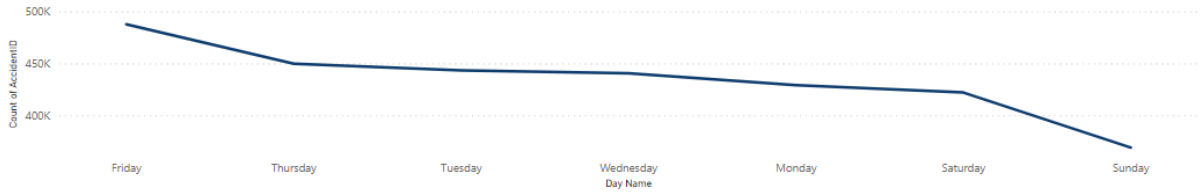


# Time Based Analysis


Count of AccidentID by Day Time




Count of AccidentID by Day Name



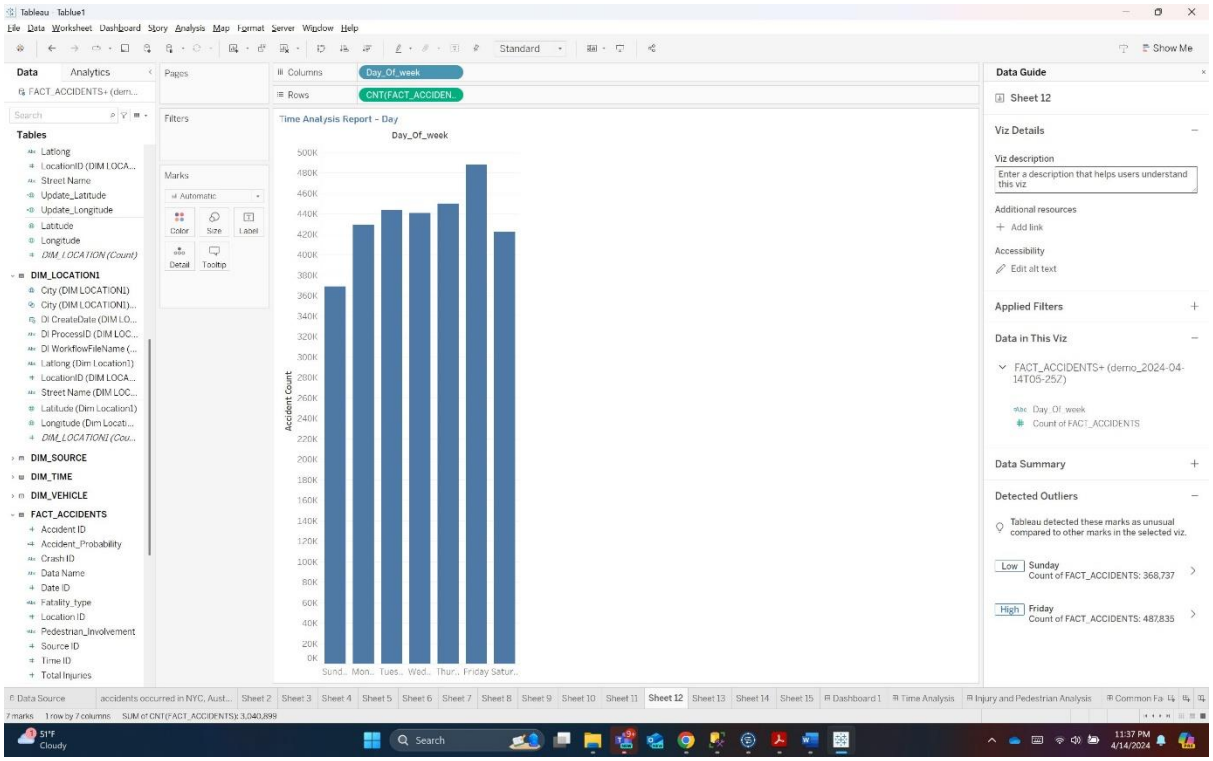


Tableau - Tablue1
Time Analysis Report - Day
Day_Of_week
Accident Count
Sund.. Mon.. Tues.. Wed.. Thur.. Friday Satur..
Data Guide
Sheet 12
Viz Details
Viz description
Enter a description that helps users understand this viz
Additional resources
Add link
Accessibility
Edit alt text
Applied Filters
Data in This Viz
FACT_ACCIDENTS+ (demo_2024-04-14T05-25Z)
Day_Of_week
Count of FACT_ACCIDENTS
Data Summary
Detected Outliers
Tableau detected these marks as unusual compared to other marks in the selected viz.
Low Sunday
Count of FACT_ACCIDENTS: 368,737
High Friday
Count of FACT_ACCIDENTS: 487,835

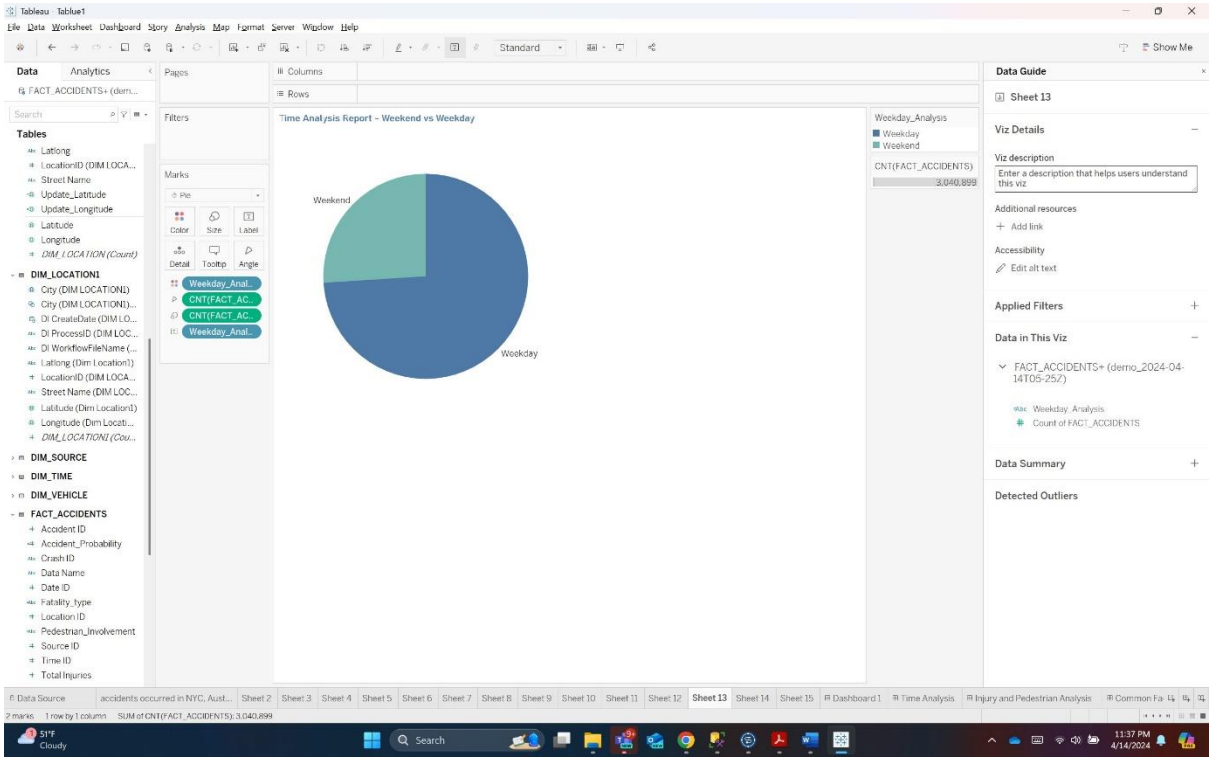


Tableau - Tablue1
Time Analysis Report - Weekend vs Weekday
Weekend
Weekday
Weekday_Analysis
Weekday
Weekend
CNT(FACT_ACCIDENTS)
3,040,899
Data Guide
Sheet 13
Viz Details
Viz description
Enter a description that helps users understand this viz
Additional resources
Add link
Accessibility
Edit alt text
Applied Filters
Data in This Viz
FACT_ACCIDENTS+ (demo_2024-04-14T05-25Z)
Weekday_Analysis
Count of FACT_ACCIDENTS
Data Summary
Detected Outliers

## Time of the day:

```
SELECT
 dt.Hour AS TimeOfTheDay,
 COUNT(*) AS AccidentCount
FROM dbo.FACT_ACCIDENTS fa
JOIN dbo.DIM_TIME dt ON fa.TimeID = dt.TimeID
GROUP BY dt.Hour
ORDER BY AccidentCount DESC;
```

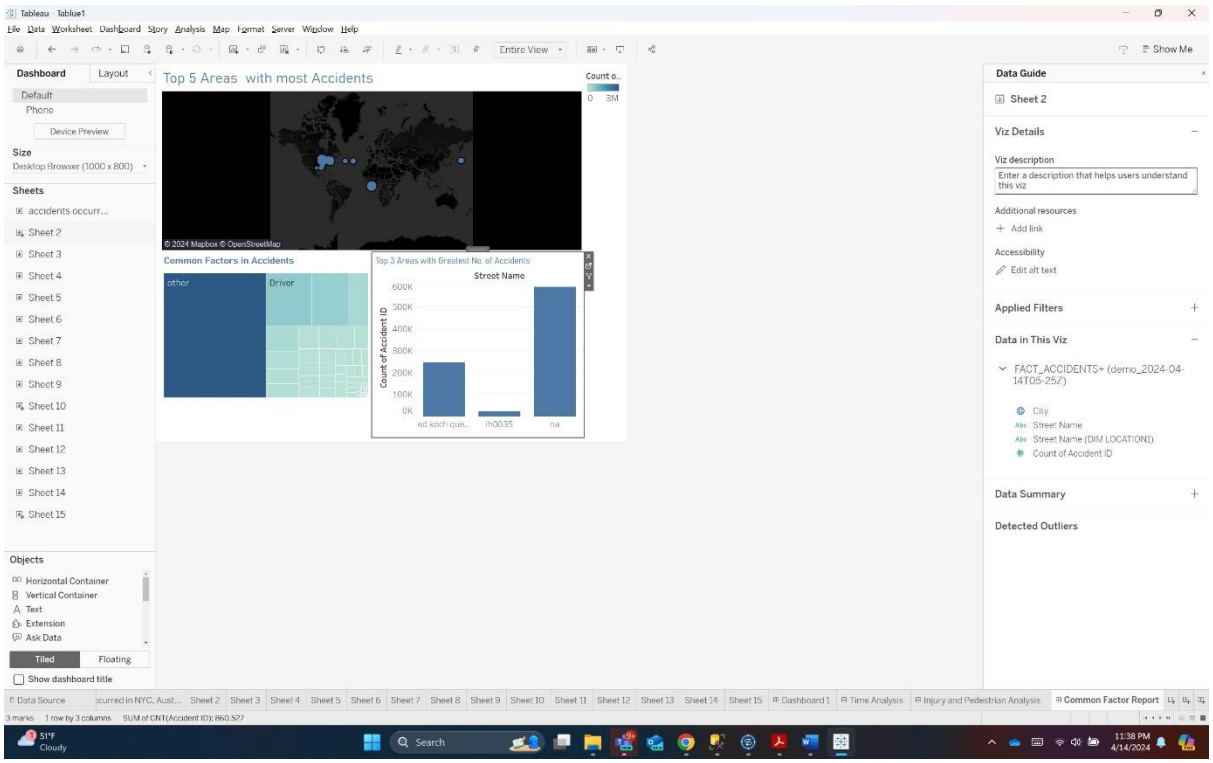
Results | Messages

Search to filter items...

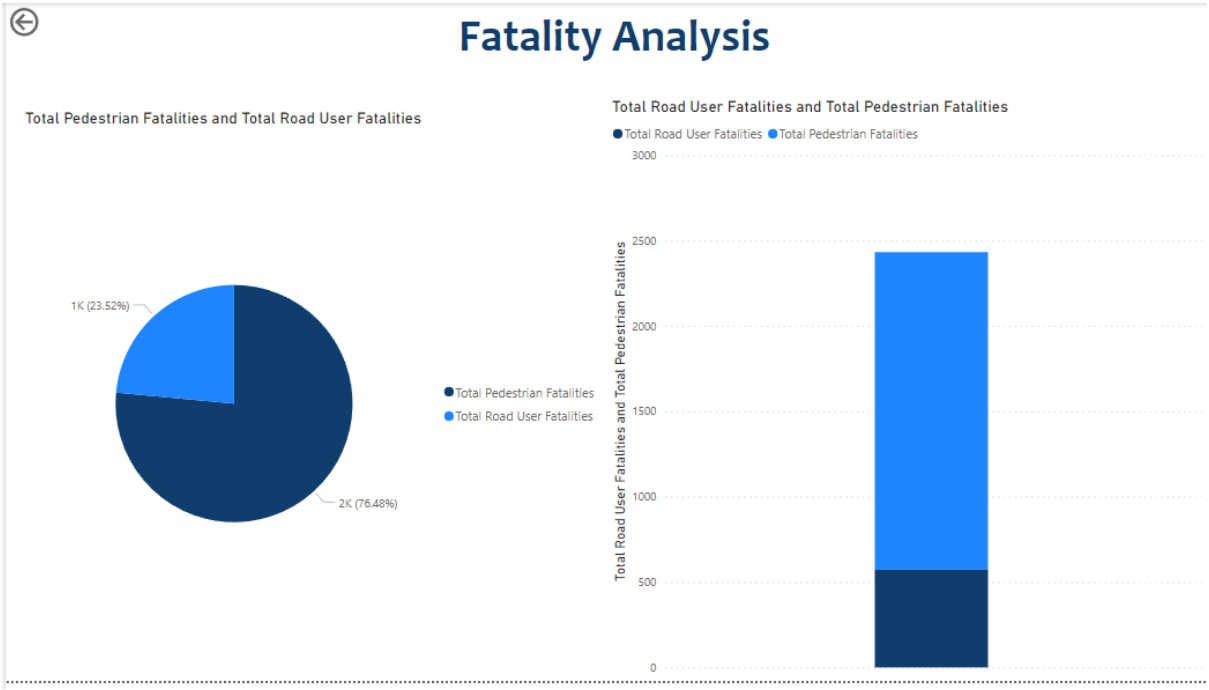
| TimeOfTheDay | AccidentCount |
|---|---|
| 8 | 258943 |
| 9 | 234529 |
| 10 | 219492 |
| 7 | 217801 |

Query succeeded | 0s

## Rest all is calculated by Data Analysis Expressions Functions.



# 9. Fatality analysis




Run | Cancel query | Save query | Export data as | Show only Editor

```
1  SELECT
2    'Pedestrians' AS VictimType,
3    SUM(CASE WHEN Pedestrian_Killed <> -999 THEN Pedestrian_Killed ELSE 0 END) AS Deaths
4  FROM dbo.FACT_ACCIDENTS
5
6  UNION ALL
7
8  SELECT
9    'Motorists' AS VictimType,
```

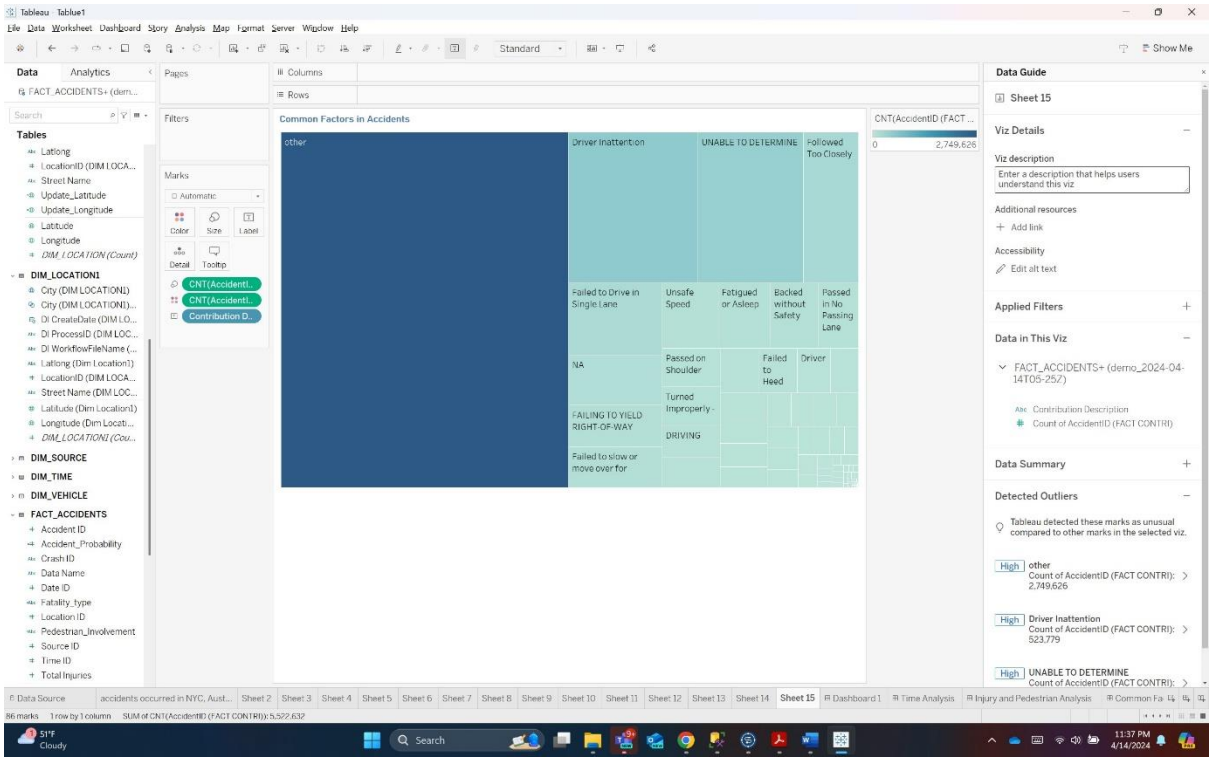
Results | Messages

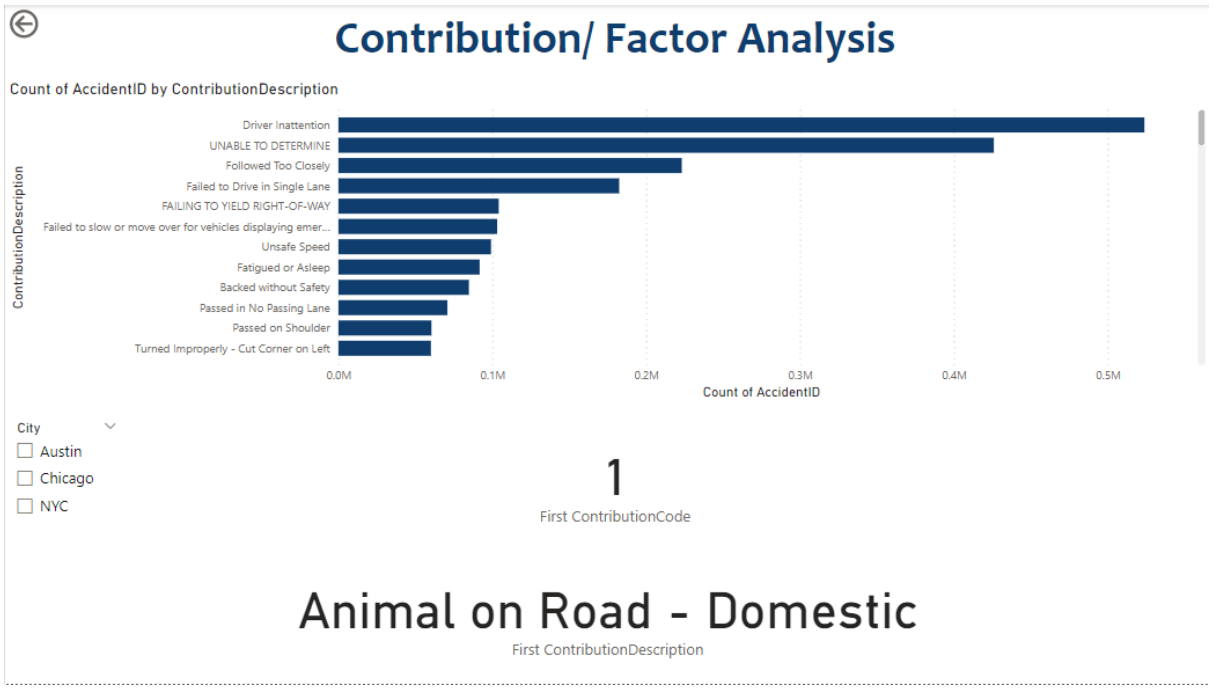
Search to filter items...

| VictimType | Deaths |
|---|---|
| Pedestrians | 1860 |
| Motorists | 1695 |
| Cyclists | 268 |

Query succeeded | 2s

# 10. What are the most common factors involved in accidents?



Run | Cancel query | Save query | Export data as | Show only Editor

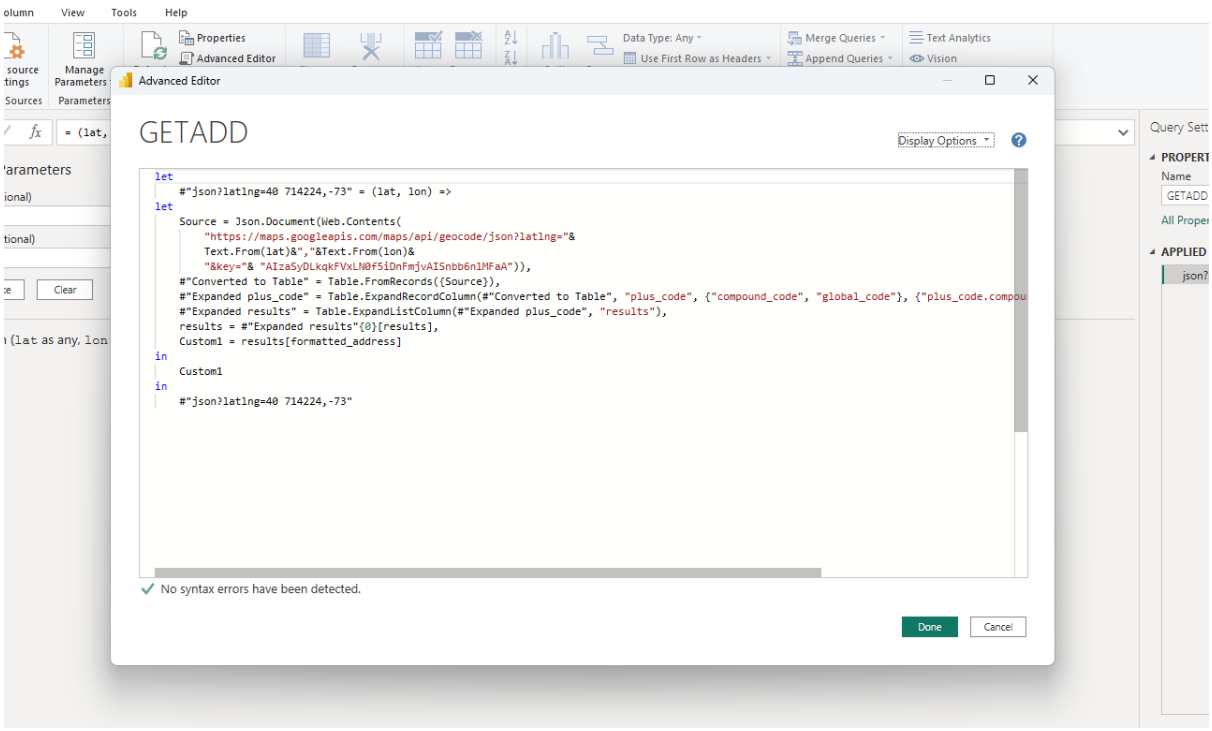
```sql
SELECT
 df.ContributionDescription AS FactorDescription,
 COUNT(*) AS Occurrences
FROM dbo.FACT_CONTRi fcf
JOIN dbo.DIM_Contribute df ON fcf.ContributionID = df.ContributionID
GROUP BY df.ContributionDescription
ORDER BY Occurrences DESC
OFFSET 0 ROWS FETCH NEXT 10 ROWS ONLY;
```

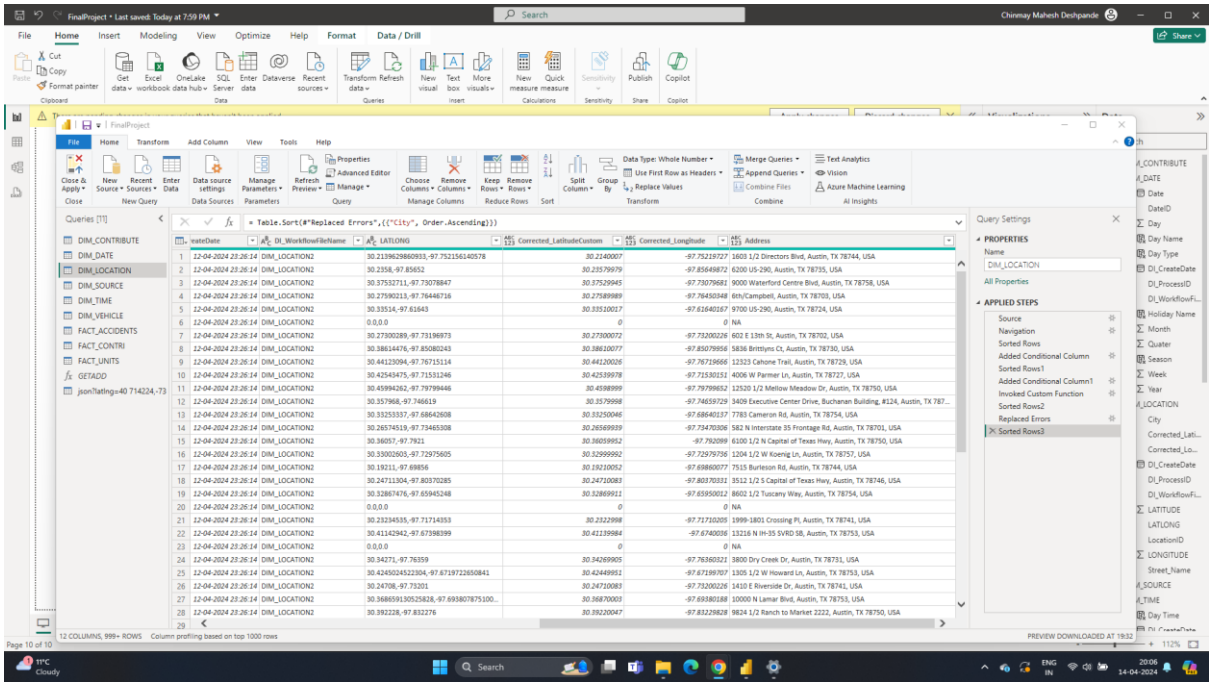
Results | Messages

| | |
|---|---|
| Driver Inattention | 523779 |
| UNABLE TO DETERMINE | 425938 |
| Followed Too Closely | 223139 |
| Failed to Drive in Single Lane | 182462 |
| NA | 127610 |
| FAILING TO YIELD RIGHT-OF-WAY | 104173 |

Query succeeded | 1s

# Note:





During a recent endeavour to improve our geospatial data capabilities, we looked into numerous options for integrating geocoding services that would translate longitude and latitude coordinates into detailed address information. Our initial method was using Google's Geocoder API. However, this solution encountered serious challenges:

System Limitations: Integration issues developed that were not fully compatible with our existing infrastructure, resulting in operational inefficiencies.

Budget Constraints: The financial cost associated with Google's Geocoder API, particularly its free credit constraints, was deemed unsustainable given the extent of our data processing requirements.

Given these problems, we decided to try a different method with Microsoft Power Query. This solution proved to be really effective and was effectively integrated into our current frameworks. Here are the major accomplishments with the

Power Query-based geocoding process:

Address Extraction: We were able to reliably extract extensive address details, such as zip codes, street names, and block numbers, from geographical coordinates.

Integration and Scalability: Power Query integrated seamlessly with our systems, allowing us to scale our geocoding jobs without incurring additional costs.

Data enrichment: Successfully extracting and adding address data dramatically improved the quality and utility of our geographic analytics.

This strategic adjustment not only aligned with our operational capabilities, but also improved resource allocation and budgeting for geospatial data initiatives.