Projekt Hurtowni Danych - Analiza wypadków drogowych

Tymoteusz Urban, Mikołaj Mróz

1. Koncepcja i cel biznesowy

- Dane z wypadków samochodowych/rowerowych/z udziałem pieszych
- Dane techniczne samochodów
- Dane pogodowe

Korzyści/cele:

- Rząd i policja może korzystać z tych danych np. w celu lokalizacji niebezpiecznych miejsc, czy częstszych kontroli konkretnych modeli samochodów powodujących kolizje pod wpływem.
- Firmy ubezpieczeniowe mogą wykorzystać takie dane do oceny ryzyka wypadków w określonych warunkach pogodowych. Pozwala to na lepsze ustalanie stawek ubezpieczeniowych w danych obszarach.
- Dane te mogą pomóc w lepszym planowaniu dróg, sygnalizacji świetlnej i innych elementów infrastruktury drogowej.
- Firmy transportowe i logistyczne mogą korzystać z tych danych do optymalizacji tras, uwzględniając warunki pogodowe, aby zminimalizować ryzyko opóźnień i wypadków.
- Użytkownicy końcowi (np. kierowcy, mieszkańcy miast) mogą uzyskać dostęp do informacji o warunkach pogodowych i związanego z nimi ryzyka wypadków. Może to pomóc w podejmowaniu świadomych decyzji dotyczących podróży i zachowania na drodze.

2. Architektura systemu

Schemat architektury i przepływu danych w naszym systemie:

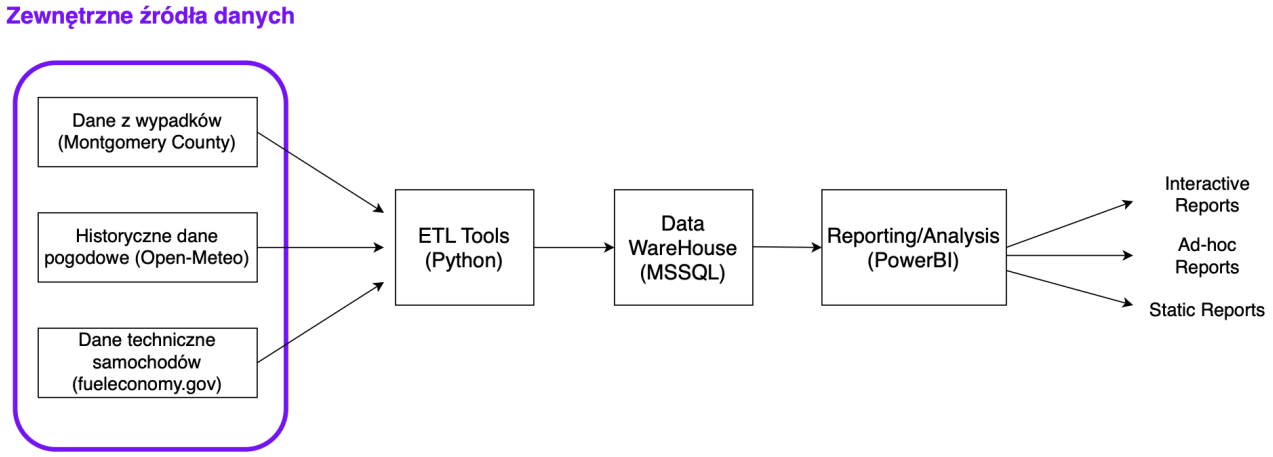


Diagram przedstawia przepływ danych, który wykorzystuje zewnętrzne źródła danych do analizy i raportowania w środowisku hurtowni danych. Oto proces przedstawiony na diagramie:

1. Zewnętrzne źródła danych:
    - Dane z wypadków (Montgomery County): Zawiera dane dotyczące zdarzeń drogowych w hrabstwie Montgomery.
    - Historyczne dane pogodowe (Open-Meteo): Dane pogodowe pochodzące z Open-Meteo, które dostarczają informacji o historycznych warunkach pogodowych.
    - Dane techniczne samochodów (fueleconomy.gov): Informacje z fueleconomy.gov, które obejmują specyfikacje i dane o zużyciu paliwa różnych pojazdów.
2. Narzędzia ETL (Python):
    - Dane z zewnętrznych źródeł są przetwarzane przy użyciu narzędzi ETL (Ekstrakcja, Transformacja, Załadunek) opracowanych w Pythonie. Ten etap obejmuje ekstrakcję danych z oryginalnych źródeł i transformację, aby dane były w odpowiednim formacie, oraz załadowanie do centralnego repozytorium.
3. Hurtownia danych (MSSQL):
    - Przetworzone dane są przechowywane w hurtowni danych zbudowanej przy użyciu Microsoft SQL Server (MSSQL). To centralne repozytorium zapewnia, że dane są zorganizowane i łatwo dostępne do dalszej analizy.
4. Narzędzia do raportowania/analizy (PowerBI):
    - Dane z hurtowni są wykorzystywane do tworzenia raportów i analiz za pomocą PowerBI, narzędzia do inteligencji biznesowej. Obsługuje ono tworzenie różnych rodzajów raportów:
        - Raporty interaktywne: Umożliwiają użytkownikom interaktywną pracę z danymi w formie wizualnej.
        - Raporty ad-hoc: Spersonalizowane raporty generowane na bieżąco, aby spełniać konkretne wymagania użytkowników.
        - Raporty statyczne: Regularne raporty, które są generowane okresowo, aby dostarczać stałe aktualizacje.

3. Opis zbiorów danych

Źródło danych	Dostęp	Format	Częstotliwość odświeżania	Częstotliwość zaciągania do hurtowni	Link do danych
Dane z wypadków (Montgomery County)	Początkowo plik z linku, potem API	JSON	Co tydzień (do końca 2023), symulacyjnie co miesiąc	Co miesiąc	Crash Reporting - Incidents Data
Historyczne dane pogodowe (Open-Meteo)	API	JSON	Co godzinę	Tak często jak wypadki	Historical Weather API Open-Meteo
Dane techniczne samochodów (fueleconomy.gov)	Pobranie pliku z linku	CSV	Nieregularnie, co kilka dni	Tak często jak wypadki	FuelEconomy.gov Web Services
Dane geograficzne (Montgomery County)	Pobranie pliku z linku	CSV	Nigdy	Raz	ZIPCODES in Montgomery County

Dane z wypadków składają się z trzech zbiorów danych. Wypadki, samochody biorące udział w wypadkach, piesi i rowerzyści biorący udział w wypadkach. Dane zawieają szczegółowe informacje o wszystkich wypadkach drogowych w obrębie hrabstwa Montgomery zgłoszonych przez różne oddziały policji. Jak wskazują autorzy zbioru danych, informacje mogą zawierać błędy mechaniczne, ludzkie i braki danych.

Historyczne dane pogodowe, jak sama nazwa wskazuje, zawierają dane atmosferyczne i pochodzą z serwisu Open Meteo, które oferuje bezpłatne, otwarte api i dostęp do danych 80 lat wstecz.

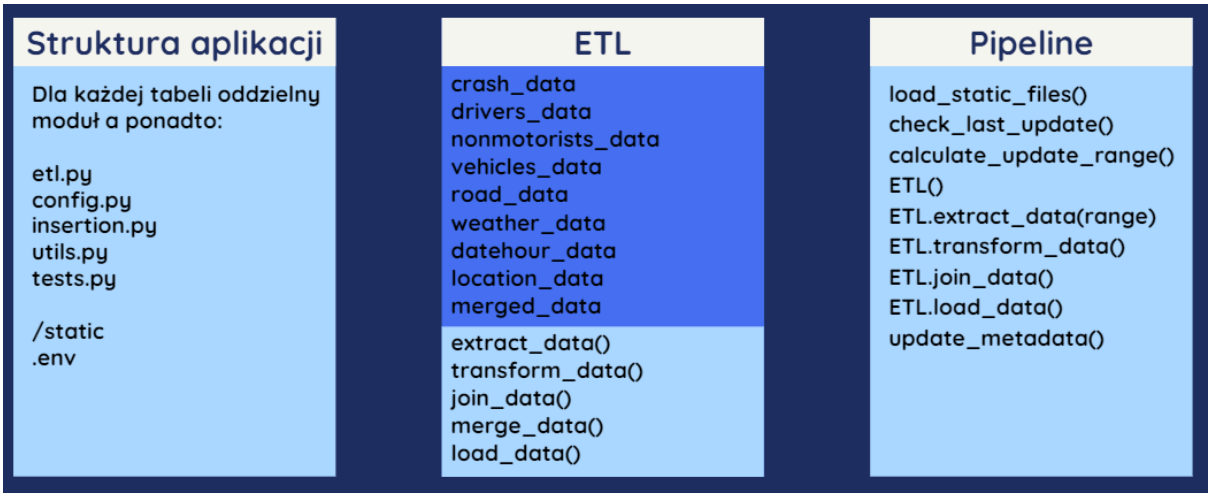
Dane techiczne zaciągane są ze strony fueleconomy.gov, oficjalnej strony rządowej USA będącej źródłem informacji o zużyciu paliwa. Baza zawiera jednak także podstawowe dane techniczne dla wszystkich modeli samochodów.

Dane geograficzne to zbiór danych, który umożliwia identyfikację obszaru na podstawie współrzędnych geograficznych. Zaciągnięcie tych danych tylko raz podczas tworzenia hurtowni jest wystarczające.

4. Model przetwarzania ETL

Jak już wcześniej zostało wspomniane, proces ETL został zaimplementowany w Pythonie. Wszystkie dane są zaciągane co miesiąc w tym samym czasie, przekształcane na format *pandas.DataFrame*, następnie transformowane, łączone poprzez dodanie kluczy obcych w odpowiednich tabelkach i załadowywane do hurtowni.

Na etapie ekstrakcji zaciągane są dane pogodowe i wypadkowe z API tylko dla okresu od ostatniego zaciągnięcia. Dzięki temu unika się pobierania nadmiarowych danych. Dane dotyczące samochodów są zaciągane w całości, ale są filtrowane tak, by nie przetwarzać za każdym razem tych wszystkich rekordów.



Rysunek 1. Od lewej: moduły w aplikacji, diagram klasy ETL, pseudokod procesu ETL

Dla niektórych tabel niezbędne jest trzymanie plików statycznych, które służą między innymi do mapowania marek i modeli, mapowania lokalizacji na obszary i zaciągania danych pogodowych. W tym celu została przygotowana klasa Static, która umożliwia szybki dostęp do statycznych ramek danych i słowników w każdym miejscu aplikacji.

Zostały zaimplementowany także mechanizmy, dzięki którym praktycznie każdy proces ETL kończy się sukcesem i brakiem wyrzucania jakichkolwiek błędów. Jeśli podczas zaciągania z API danych dotyczących wypadków i pogody wystąpiły problemy z połączeniem, zapytania są powtarzane. Został także przygotowany został specjalny kod SQL, który podczas załadowywania danych do hurtowni nie wyrzuca błędów jeśli jakieś rekordy są zduplikowane, a zwyczajnie je pomija.

Tabela	Ekstrakcja	Transformacje
DateHourDim	generowanie rekordów - każdy wiersz to kolejna godzina	• wyciągnięcie atrybutów daty • utworzenie flag • utworzenie klucza głównego

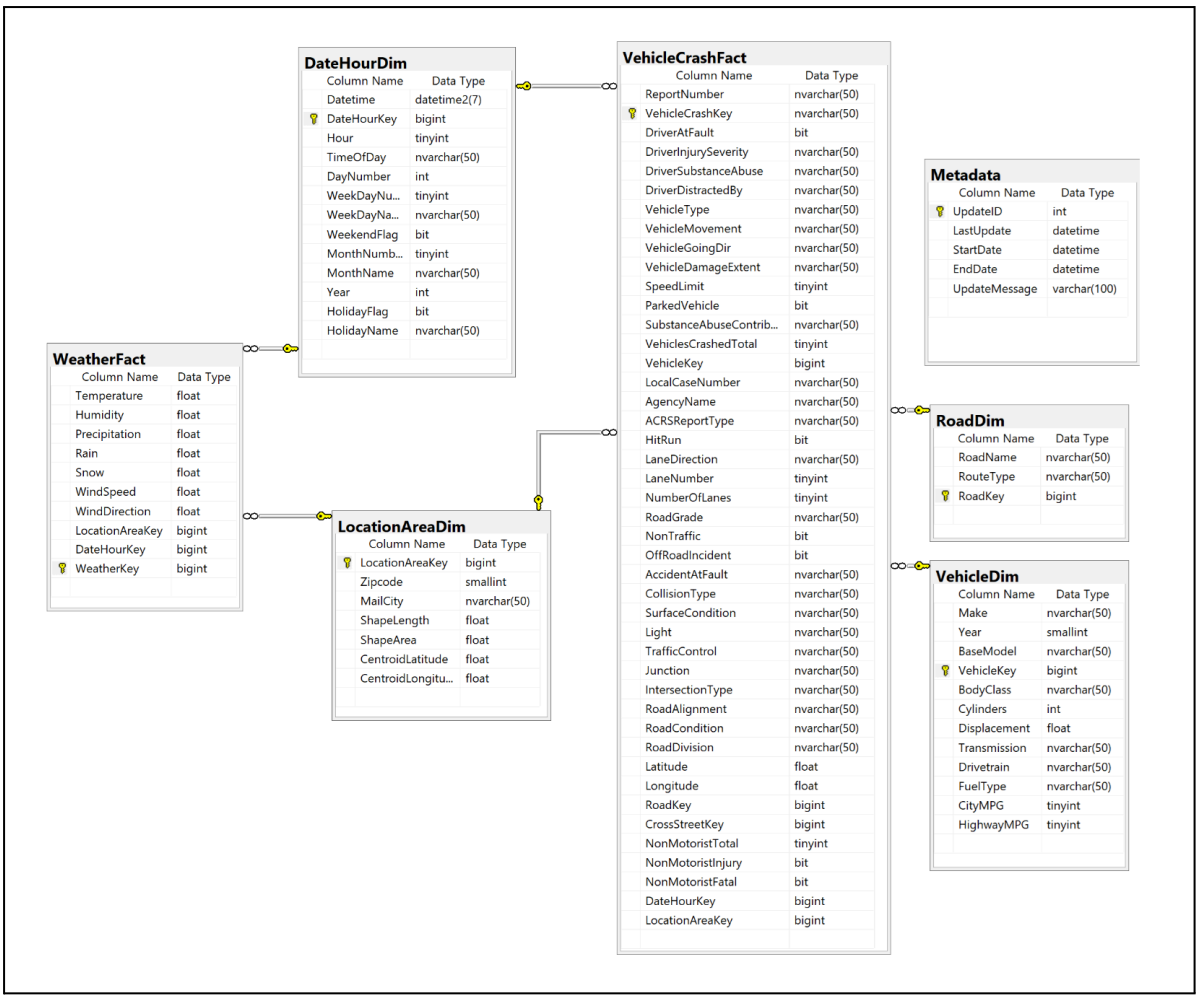
LocationAreaDim	jednorazowe załadowanie z pliku	● wyliczenie punktów centralnych ● dodanie pustej lokalizacji ● utworzenie klucza głównego
WeatherFact	zapytanie do API dla każdej lokalizacji, dane godzinowe	● przesunięcie czasu ● utworzenie klucza głównego i kluczy obych
VehicleDim	odczytanie CSV z URL	● wypełnienie nulli wartością ‘Unknown’ albo zerami ● stworzenie własnych klas dla ‘Transmission’ i ‘Drivetrain’ ● wygenerowanie pustych modeli dla każdej marki ● wygenerowanie kluczy głównych
RoadDim	wyciągnięcie z Incidents Data	● utworzenie wierszy na podstawie wpisów z wypadków ● zastąpienie nulli wartością ‘Unknown’ ● usunięcie duplikatów ● utworzenie klucza głównego
VehicleCrashFact - Drivers Data	zapytanie do API dla konkretnego przedziału czasowego	● zastąpienie nulli wartością ‘Unknown’ i zerami ● wyczyszczenie wartości w ‘DriverSubstanceAbuse’ ● własne kategorie w ‘VehicleType’ ● dodanie flag w ‘SubstanceAbuseContributed’, ‘DriverAtFault’, ‘ParkedVehicle’ ● dodanie zagregowanej wartości ‘CrashedTotal’ ● mapowanie marek, modeli i daty produkcji oraz wygenerowanie klucza obcego dla VehicleDim
VehicleCrashFact - Incidents Data	zapytanie do API dla konkretnego przedziału czasowego	● zastąpienie nulli wartością ‘Unknown’ ● skrócenie wartości w

		'ACRSReportType' ● przekonwertowanie daty ● dodanie flag w 'HitRun', 'NonTraffic' i 'OffRoadIncident' ● mapowanie lokalizacji na regiony ● wygenerowanie kluczy obcych dla RoadDim, VehicleDim, DateHourDim ● scalenie z 'Drivers data'
VehicleCrashFact - Non-motorists Data	zapytanie do API dla konkretnego przedziału czasowego	● własne kategorie dla 'InjurySeverity' ● stworzenie zagregowanych miarek 'NonMotoristTotal', 'NonMotoristInjury', 'NonMotoristFatal' ● scalenie z 'Incidents data' i wypełnienie nulli zerami

5. Model hurtowni danych

W pierwszej tabeli faktów (VehicleCrashFact) faktem jest udział pojedynczego pojazdu w wypadku. Dany wypadek jest identyfikowany poprzez ReportNumber, więc chcąc analizować całe wypadki wystarczy pogrupować dane według ReportNumber. W tabeli faktów znajduje się bardzo dużo atrybutów, ponieważ nie ma możliwości utworzenia z nich oddzielnych wymiarów. Większość atrybutów jest ze sobą niepowiązana tzn. na przykład wartości RoadDivision czy NumberOfLanes zmieniają się na różnych odcinkach drogi, dlatego nie można umieścić tych atrybutów w RoadDim. Wiele atrybutów jest też zwyczajnie związana z konkretnym wypadkiem np. VehicleDamageExtent czy DriverInjurySeverity.

Z tabelą powiązany jest wymiar VehicleDim przedstawiający dane techniczne pojazdu biorącego udział w wypadku. Następnym wymiarem jest RoadDim zawierający nazwę i typ drogi. Używany jest dwa razy, dla głównej drogi na której zdarzył się wypadek i drogi poprzecznej, jeśli wypadek zdarzył się na łączeniu dróg. Stworzony został także wymiar czasu, który oprócz daty i godziny zawiera również różne, sztucznie utworzone atrybuty przydatne do analizy. Ostatnim wymiarem jest LocationAreaDim odpowiadający obszarowi powiązanemu z miejscem, w którym zdażył się wypadek.



Schemat wygenerowany w SQL Server

Ostatnie dwa wymiary służą także do opisania drugiej tabeli faktów zawierającej dane pogodowe. Jeden fakt pogodowy odnosi się do danych pogodowych z danego obszaru i czasu. Atrybutami są podstawowe dane takie jak temperatura, zachmurzenie czy opady.

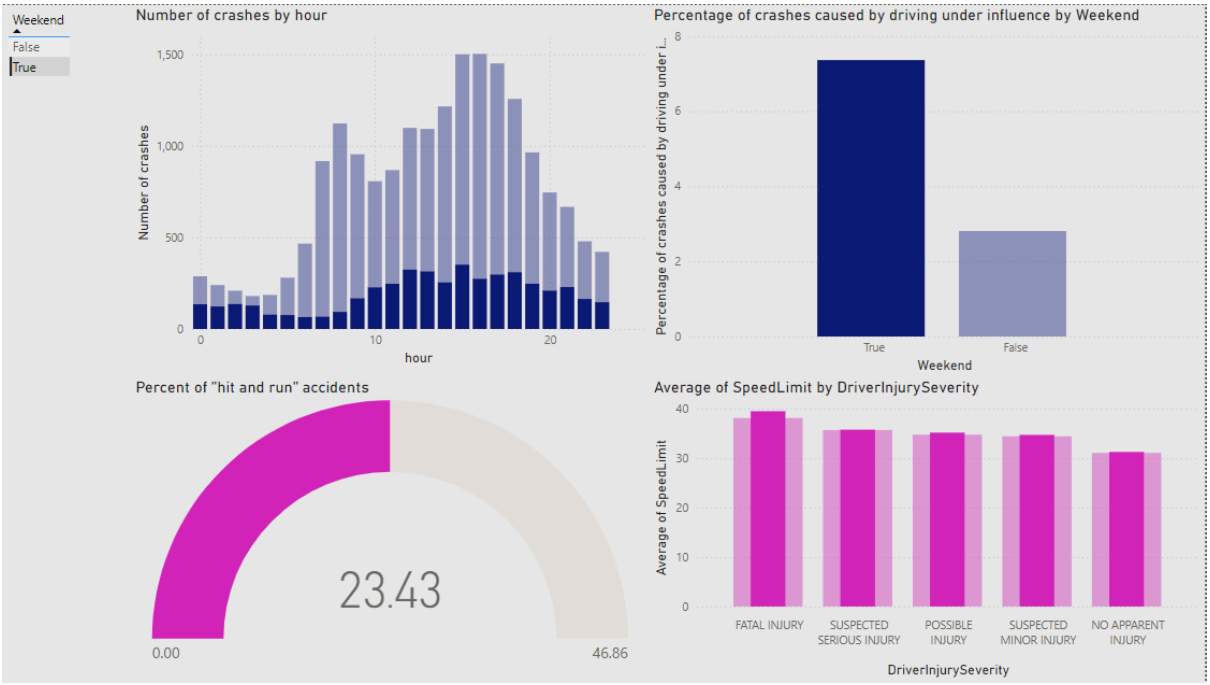
W naszych danych nie ma typowych miar biznesowych, ale jako takie można potraktować liczbę osób poszkodowanych, liczbę osób zabitych, liczbę samochodów biorących udział w wypadku. Podczas raportowania najczęściej będziemy także liczyć sumaryczną liczbę wypadków w określonych warunkach dla różnych wartości atrybutów.

Ostatnią tabelą jest tabela Metadata, w której trzymana jest historia updatów hurtowni. Na tej podstawie wyznaczane są zakresy czasowe dla kolejnych aktualizacji danych.

6. Warstwa raportowa

W ramach opracowanego systemu hurtowni danych, raportowanie jest kluczowym elementem umożliwiającym efektywne wykorzystanie zgromadzonych informacji. Wizualizacje zostały przygotowany przy pomocy Power BI.

Analiza użycia alkoholu i innych używek w zależności od tego, czy jest weekend, czy nie:



Ciekawe wnioski:

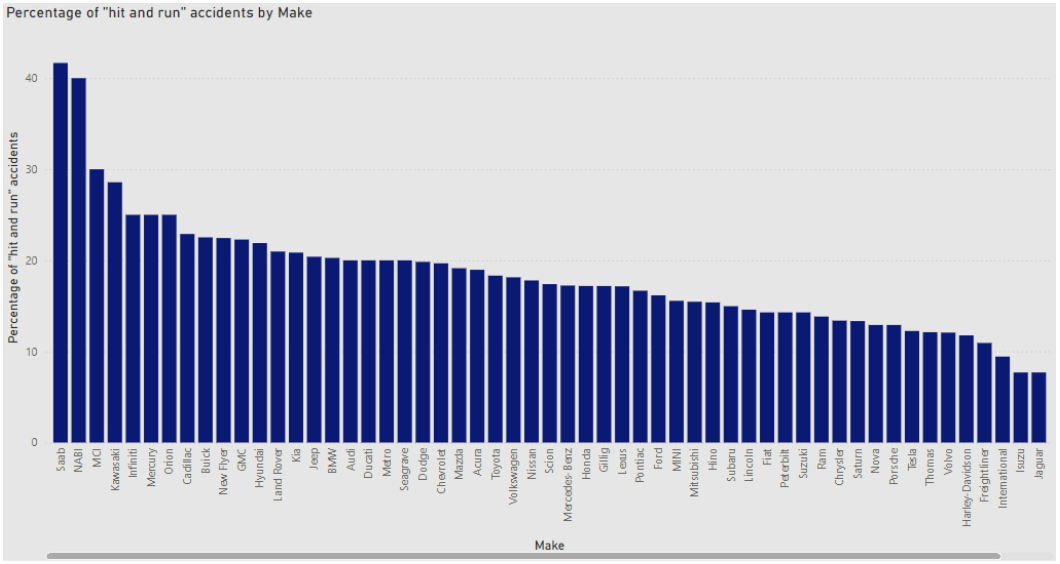
- Widzimy, że w trakcie weekendu wypadki są rozłożone bardziej regularnie podczas całego dnia, w trakcie dni roboczych dużo więcej wypadków jest w godzinach szczytu
- W weekend jest trzykrotnie więcej procent wypadków spowodowanych jazdą pod wpływem alkoholu
- Podczas weekendu aż w 23% wypadków ktoś zbiega z miejsca zdarzenia

Użyte pola wyliczane:

```
PercentCrashesWithDrugAbuse =
VAR TotalCrashes = COUNT(VehicleCrashFact[AccidentAtFault])
VAR CrashesWithDrugAbuse =
    CALCULATE(
        COUNT(VehicleCrashFact[AccidentAtFault]),
        VehicleCrashFact[DriverSubstanceAbuse] IN {"ALCOHOL", "COMBINATION", "ILLEGAL DRUG", "MEDICATION", "OTHER"}
    )
RETURN
DIVIDE(CrashesWithDrugAbuse, TotalCrashes, 0) * 100
```

```
PercentHitRun =
VAR TotalCrashes = COUNT(VehicleCrashFact[AccidentAtFault])
VAR HitRunCrashes =
    CALCULATE(
        COUNT(VehicleCrashFact[AccidentAtFault]),
        VehicleCrashFact[HitRun] = TRUE()
    )
RETURN
DIVIDE(HitRunCrashes, TotalCrashes, 0) * 100
```

Analiza ilości wypadków uwzględniając marki pojazdów:

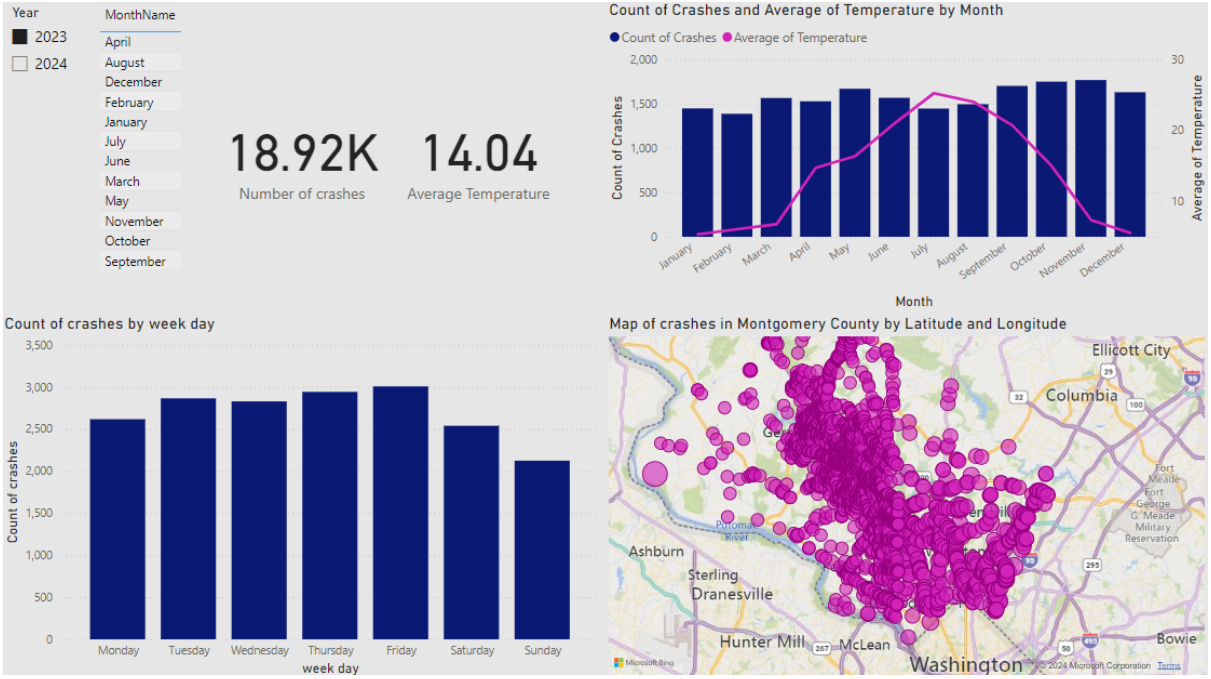


- Możemy zaobserwować, że najczęściej z miejsca wypadku uciekają pojazdy marek, które zajmują się głównie robieniem motorów. Jest to dość logiczny wniosek ponieważ motorom jest najłatwiej szybko odjechać

Użyte pola wyliczane:

```
PercentHitRunByMake =
VAR TotalCrashes = COUNT(VehicleCrashFact[AccidentAtFault])
VAR HitRunCrashesByMake =
    CALCULATE(
        COUNT(VehicleCrashFact[AccidentAtFault]),
        VehicleCrashFact[HitRun] = TRUE()
    )
VAR MakeHitRunCrashes =
    SUMMARIZE(
        VehicleCrashFact,
        VehicleDim[Make],
        "HitRunCount", HitRunCrashesByMake
    )
RETURN
DIVIDE(HitRunCrashesByMake, TotalCrashes, 0) * 100
```

Analiza pogodowa:



- Po wykresie ilości wypadków i średniej temperatury można zauważyć, że gdy zaczyna się lato, kierowcy szarżują na drogach i powodują więcej wypadków. Potem jest spadek przed zimą, a następnie obniżona temperatura, śnieg wpływają na wzrost liczny kolizji
- Na tej stronie raportu możemy wybierać konkretne miesiące i lata, sprawdzać w nich średnią temperaturę oraz konkretną liczbę wypadków

Kroki w celu uzyskania tabeli zawierającej dane o wypadkach i dane o pogodzie podczas konkretnego wypadku:

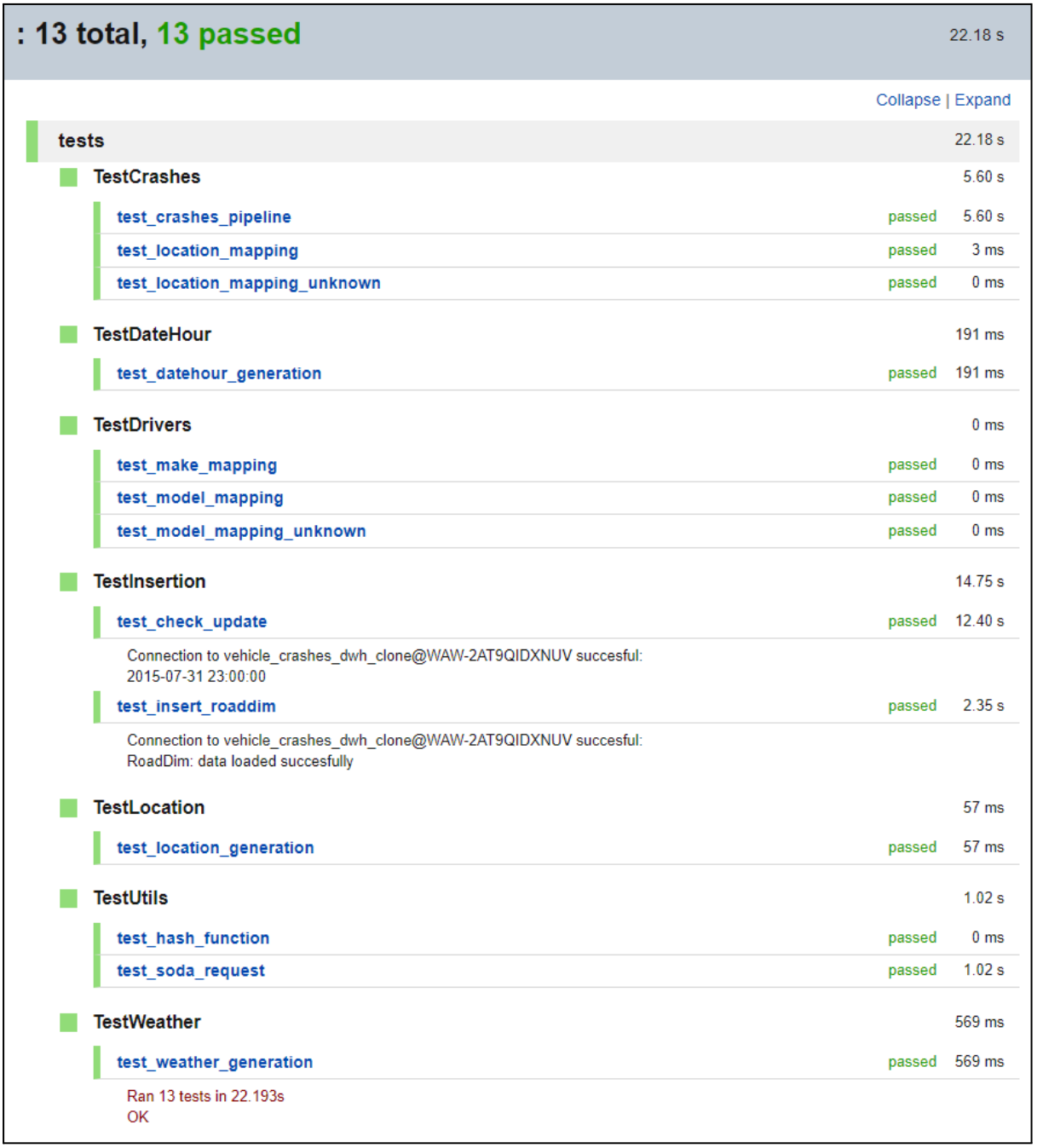
```
= Table.NestedJoin(VehicleCrashFact, {"DateHourKey", "LocationAreaKey"}, WeatherFact, {"DateHourKey", "LocationAreaKey"}, "WeatherFact",
    JoinKind.Inner)
```

```
= Table.ExpandTableColumn(Source, "WeatherFact", {"Temperature", "Humidity", "Precipitation", "Rain", "Snow", "WindSpeed", "WindDirection"}
    , {"Temperature", "Humidity", "Precipitation", "Rain", "Snow", "WindSpeed", "WindDirection"})
```

7. Testy funkcjonalne

Testy jednostkowe

W pierwszej kolejności przeprowadzane były testy jednostkowe tworzone w trakcie implementacji procesu ETL. Kod nie miał 100% pokrycia testami, ale kluczowe elementy zostały dobrze sprawdzone i przeanalizowane. Testy dotyczą m.in. generowania i przetwarzania tabel wymiarów i faktów, mapowania lokalizacji i modeli samochodów czy wykonywania zapytań do bazy danych. Na załączniku 1 widzimy, że wszystkie testy przechodzą bez problemu.



Załącznik 1: wynik przeprowadzonych testów jednostkowych

Test automatyzacji ETL

Przetestowane zostało także działanie automatycznego wgrywania danych. Proces ETL został odpalony 4 razy z domyślnymi ustawieniami, przy czym przy pierwszym odpaleniu flaga *DWH_INITIALIZATION* została ustawiona na wartość *True*. Po skończeniu działania skryptu została sprawdzona zawartość hurtowni.

Na załączniku 2 widzimy rekordy z tabeli *Metadata*. Widzimy, że zakres czasowy transformowanych i wgrywanych danych był dobrze obliczony. Z każdym uaktualnieniem przetwarzany był kolejny miesiąc.

	UpdateID	LastUpdate	StartDate	EndDate	UpdateMessage
1	6	2015-01-10 00:00:00.000	2010-01-01 00:00:00.000	2014-12-31 23:00:00.000	initial
2	7	2024-06-09 21:22:52.510	2015-01-01 00:00:00.000	2015-01-31 23:00:00.000	regular update
3	8	2024-06-09 21:42:03.890	2015-02-01 00:00:00.000	2015-02-28 23:00:00.000	regular update
4	9	2024-06-09 21:47:10.420	2015-03-01 00:00:00.000	2015-03-31 23:00:00.000	regular update
5	10	2024-06-09 21:50:24.520	2015-04-01 00:00:00.000	2015-04-30 23:00:00.000	regular update

Załącznik 2: Wynik zapytania ***select * from Metadata***

W celu zweryfikowania, czy dane zostały odpowiednio przygotowane w kontekście kluczy głównych i kluczy obcych, sprawdzona została liczba wierszy głównej tabeli faktów i tabeli wynikowej po wykonaniu scalenia tabel z całej hurtowni danych.

```
select count(*) from VehicleCrashFact
```

	(No column name)
1	8702

Załącznik 3: Liczba wierszy w głównej tabeli faktów

```
select count(*) from VehicleCrashFact vcf
join VehicleDim vd on vd.VehicleKey = vcf.VehicleKey
join RoadDim rd on rd.RoadKey = vcf.RoadKey
join RoadDim rd2 on rd2.RoadKey = vcf.CrossStreetKey
join DateHourDim dhd on dhd.DateHourKey = vcf.DateHourKey
join LocationAreaDim lad on lad.LocationAreaKey = vcf.LocationAreaKey
```

	(No column name)
1	8702

Załącznik 4: Liczba wierszy po scaleniu wszystkich tabel oprócz tabeli pogodowej

Liczba rekordów jest identyczna. Niestety, dołożenie tabeli faktów z danymi pogodowymi powoduje że nie wszystkie rekordy się łączą.

```
join WeatherFact wf on wf.LocationAreaKey = vcf.LocationAreaKey and
                       wf.DateHourKey = vcf.DateHourKey
```

	(No column name)
1	8670

Załącznik 5: Liczba wierszy po scaleniu wszystkich tabel

Jest to jednak oczekiwane zachowanie. Taka sytuacja może wystąpić, gdy w tabeli faktów znajdują się wypadki, których lokalizacja nie zawiera się w żadnym ze sprecyzowanych regionów. Dane pogodowe są zaciągane jedynie dla konkretnych lokalizacji, zatem nie ma możliwości połączenia takiego wypadku z tabelą faktów pogodowych. Na szczęście jest to zaledwie promil ze wszystkich wypadków.

Test ręcznej konfiguracji ETL

Następnie sprawdzone zostało działanie procesu ETL odpalonego z ręcznie sprecyzowanym zakresem dat. Jak widzimy na zrzucie ekranu, update został wykonany dla 2 miesięcy.

	UpdateID	LastUpdate	StartDate	EndDate	UpdateMessage
1	6	2015-01-10 00:00:00.000	2010-01-01 00:00:00.000	2014-12-31 23:00:00.000	initial
2	7	2024-06-09 21:22:52.510	2015-01-01 00:00:00.000	2015-01-31 23:00:00.000	regular update
3	8	2024-06-09 21:42:03.890	2015-02-01 00:00:00.000	2015-02-28 23:00:00.000	regular update
4	9	2024-06-09 21:47:10.420	2015-03-01 00:00:00.000	2015-03-31 23:00:00.000	regular update
5	10	2024-06-09 21:50:24.520	2015-04-01 00:00:00.000	2015-04-30 23:00:00.000	regular update
6	11	2024-06-10 22:52:07.220	2015-05-01 00:00:00.000	2015-07-31 23:00:00.000	custom update for report

Załącznik 6: Wynik zapytania ***select * from Metadata*** po ponownym odpaleniu procesu ETL

```
select count(*) from VehicleCrashFact
```

	(No column name)
1	13586

Załącznik 7: Uaktualniona liczba rekordów

Dane zostały wgrane bez problemy, widzimy że liczba rekordów zaktualizaowała się z 8702 do 13586. Załącznik 8 przedstawia logi z konsoli podczas wykonywania procesu ETL. Widzimy, że każdy etap został zakończony powodzeniem, warto jednak zwrócić uwagę na linię numer 10. Zadziałał tutaj mechanizm ponownego wysłania zapytania do API. Pomimo pierwszego nieudanego zapytania, drugie już się powiodło i zwróciło wiersze z danymi z wypadków. Gdyby wspomniany mechanizm nie został zaimplementowany, cały proces ETL musiałby zostać przerwany już na samym początku.

```
 9 RUNNING EXTRACTION from 2015-05-01 00:00:00 to 2015-07-31 23:00:00
10 Error ocurred, sending another request HTTPSConnectionPool(host='data.
   montgomerycountymd.gov', port=443): Read timed out. (read timeout=10)
11 crash data rows: 2761
```

Załącznik 8: Fragment logów z konsoli

File - etl (1)

```
C:\Users\tymoteusz.urban\Desktop\moje\repozytoria\hurtownie\.venv\Scripts\python.exe C:\Users\tymoteusz.urban\Desktop\moje\repozytoria\hurtownie\etl\etl.py
Brands mapper loaded
Models mapper loaded
Area mapper loaded
Zipcodes data loaded
-----
RUNNING ETL PIPELINE from 2015-05-01 00:00:00 to 2015-07-31 23:00:00
-----
RUNNING EXTRACTION from 2015-05-01 00:00:00 to 2015-07-31 23:00:00
Error ocurred, sending another request HTTPSConnectionPool(host='data.montgomerycountymd.gov', port=443): Read timed out. (read timeout=10)
crash data rows: 2761
drivers data rows: 4884
non motorists data rows: 170
vehicles data rows: 47523
weather data rows: 214176
datehour data rows: 214176
-----
RUNNING TRANSFORMATIONS
Vehicles data transformed
Models mapper updated
Models mapper loaded
Drivers data transformed
Roads data transformed
Non-motorists data transformed
Crashes data transformed
Weather data transformed
-----
RUNNING JOINING
Drivers data mapped
Crashes data mapped
Vehicle Crashes joined
-----
RUNNING DWH INSERTION
Connection to vehicle_crashes_dwh_clone@WAW-2AT9QIDXNUV succesful: <pyodbc.Connection object at 0x0000018D80E27B80>
RoadDim: data loaded succesfully
Connection to vehicle_crashes_dwh_clone@WAW-2AT9QIDXNUV succesful: <pyodbc.Connection object at 0x0000018D80E27B80>
VehicleDim: data loaded succesfully
Connection to vehicle_crashes_dwh_clone@WAW-2AT9QIDXNUV succesful: <pyodbc.Connection object at 0x0000018D80E27B80>
DateHourDim: data loaded succesfully
Connection to vehicle_crashes_dwh_clone@WAW-2AT9QIDXNUV succesful: <pyodbc.Connection object at 0x0000018D80E27B80>
WeatherFact: data loaded succesfully
Connection to vehicle_crashes_dwh_clone@WAW-2AT9QIDXNUV succesful: <pyodbc.Connection object at 0x0000018D80E27B80>
VehicleCrashFact: data loaded succesfully
Connection to vehicle_crashes_dwh_clone@WAW-2AT9QIDXNUV succesful: <pyodbc.Connection object at 0x0000018D80E27B80>
Metadata: data loaded succesfully
ETL PROCESS FINISHED WITH SUCCESS

Process finished with exit code 0
```


Załącznik 9: Logi z konsoli podczas wykonywania procesu ETL

8. Podsumowanie finalnej wersji rozwiązania

Finalna wersja hurtowni danych skutecznie integruje dane o wypadkach drogowych, dane pogodowe oraz dane techniczne samochodów, umożliwiając ich wspólną analizę. Proces ETL został zoptymalizowany i przetestowany, zapewniając automatyczne przetwarzanie i załadunek danych do hurtowni. Struktura hurtowni danych oraz warstwa raportowa stworzona przy użyciu PowerBI dostarczają wartościowych informacji dla różnych użytkowników końcowych. System oferuje użyteczne narzędzie do analizy i raportowania, dzięki czemu może wspierać decyzje dotyczące bezpieczeństwa drogowego, planowania infrastruktury oraz oceny ryzyka ubezpieczeniowego.

9. Podział pracy

Mikołaj Mróz:

- wykonanie skryptów do przetwarzania danych pogodowych, lokalizacyjnych i czasowych
- stworzenie bazy danych
- wykonanie raportów i wizualizacji

Tymoteusz Urban:

- wykonanie skryptów do przetwarzania danych z wypadków i samochodów
- implementacja procesu ETL i automatyzacja