Testy

Niniejszy dokument zawiera opis testów wykonanych w celu zweryfikowania poprawności każdego etapu procesu ETL. Niektóre kwerendy testowe zostały napisane przy pomocy Chata GPT. Takie testy są oznaczone w poniższym sprawozdaniu jako (GPT-4) dla celów informacyjnych.

Testy ładowania do Staging DB

Pierwsza część procesu ETL to ładowanie danych z plików csv i xlsx do tzw. Staging Database. Poniżej znajdują się testy dotyczące ładowania tabel takich jak EMS, NYC Population, Social Help, Crashes, Speed Limits do pośredniej bazy danych. Testy mają na celu zweryfikowanie poprawności przeprowadzonego w procesie ETL:

1. Usuwania wierszy z brakującymi danymi w wybranych kolumnach
2. Określania typów danych na typy docelowe w hurtowni
3. Obliczania nowych kolumn numerycznych
4. Imputowania braków danych, tam gdzie uznano to za konieczne

Tabela EMS

1. **Testy Dat**

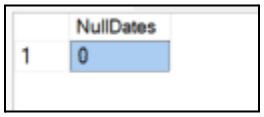
a) Usunięcie rekordów z brakującym którymkolwiek potem daty.
Oczekiwany wynik :

0 rekordów z brakami w kolumnach IncidentDateTime, FirstActivationDateTime, FirstOnSceneDateTime, FirstToHospitalDateTime, IncidentClosingDateTime

Implementacja testu:

```
select count(*) as NullDates from EMS
where IncidentDateTime is null or FirstActivationDateTime is
null or FirstOnSceneDateTime is null or FirstToHospitalDateTime
is null or IncidentClosingDateTime is null
```

Wynik:



Zgodny z oczekiwaniami ✅

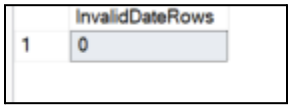
a) Usunięcie rekordów z niespójnościami w datach
Oczekiwany wynik:

Brak rekordów gdzie daty, które powinny być chronologicznie później są wcześniejsze, np. FirstToHospitalDateTime wcześniejsze niż FirstActivationDateTime itd.

Implementacja testu:

```
select count(*) as InvalidDateRows from EMS
where FirstActivationDateTime<IncidentDateTime or
FirstOnSceneDateTime<FirstActivationDateTime or
FirstToHospitalDateTime<FirstOnSceneDateTime or
IncidentClosingDateTime<FirstToHospitalDateTime
```

Wynik:



	InvalidDateRows
1	0

Zgodny z oczekiwaniami ✅

2. Testy typów danych
Oczekiwany wynik:

IncidentDateTime, FirstActivationDateTime, FirstOnSceneDateTime, FirstToHospitalDateTime, IncidentClosingDateTime → datetime

Initial/Final SeverityLevel → int

Initial/Final CallType, Borough → nvarchar

Wszystkie kolumny typu duration → decimal

Implementacja testu:

```
SELECT COLUMN_NAME, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'EMS';
```

Wynik:

	COLUMN_NAME	DATA_TYPE
1	IncidentDateTime	datetime
2	InitialCallType	nvarchar
3	InitialSeverityLevel	int
4	FinalCallType	nvarchar
5	FinalSeverityLevel	int
6	FirstActivationDateTime	datetime
7	FirstOnSceneDateTime	datetime
8	FirstToHospitalDateTime	datetime
9	IncidentClosingDateTime	datetime
10	Borough	nvarchar
11	ActivationDuration	decimal
12	OnSceneArrivalDuration	decimal
13	ToHospitalArrivalDuration	decimal
14	InterventionDuration	decimal

Zgodny z oczekiwaniami ✅

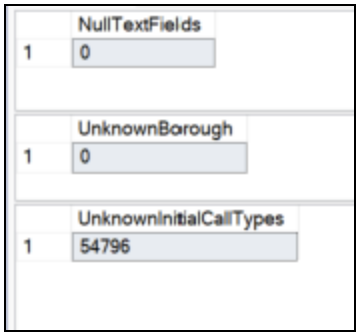
3. **Testy imputacji braków danych**
   Oczekiwany wynik:

   Nie powinno być wierszy zawierających NULL lub puste pole “” w kolumnach Call Type, Severity Level i Borough. Braki danych kolumn tekstowych powinny być zastąpione “UNKNOWN”, a numerycznych wartością -1.

   Implementacja testu:

```
select count(*) as NullTextFields from EMS
where TRIM(InitialCallType) = '' or InitialCallType is null or
FinalCallType is null or TRIM(FinalCallType)='' or Borough is
null or TRIM(Borough) = ''

select count(*) as UnknownBorough from EMS
where Borough = 'UNKNOW'

select count(*) as UnknownInitialCallTypes from EMS
where InitialCallType = 'UNKNOW'
```

   Wynik:



Zgodny z oczekiwaniami ✅

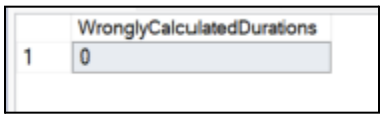
4. **Testy nowych Calculated Columns**
   Oczekiwany wynik:

   Nie powinno być wierszy gdzie obliczone kolumny typu duration różnią się od wyniku odejmowania dat o więcej niż 0.01 minuty (epsilon błędu wynikający z różnic numerycznych).

   Implementacja testu:

```
select count(*) as WronglyCalculatedDurations from EMS
WHERE ABS(ActivationDuration - (DATEDIFF(SECOND,
IncidentDateTime, FirstActivationDateTime) * 1.0 / 60)) > 0.01
or
ABS(OnSceneArrivalDuration - (DATEDIFF(SECOND,
FirstActivationDateTime, FirstOnSceneDateTime) * 1.0 / 60)) >
0.01 or
ABS(ToHospitalArrivalDuration - (DATEDIFF(SECOND,
FirstOnSceneDateTime, FirstToHospitalDateTime) * 1.0 / 60)) >
0.01 or
ABS(InterventionDuration - (DATEDIFF(SECOND, IncidentDateTime,
IncidentClosingDateTime) * 1.0 / 60)) > 0.01
```

   Wynik:



Zgodny z oczekiwaniami ✅

Tabele Social Help i NYC Population

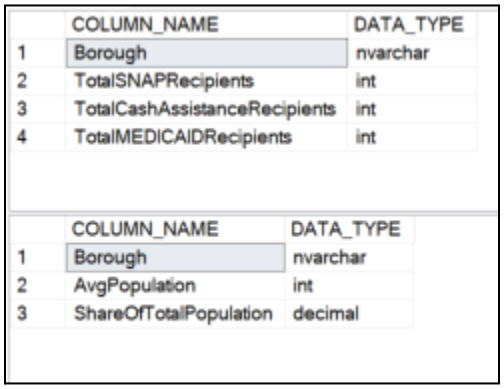
1. **Test poprawnej konwersji typów danych**
   Oczekiwany wynik:

   Kolumny w Social Help typu Recipients zmapowane na int, kolumna Borough jako nvarchar
   Kolumny w NYC Population : Borough → nvarchar, AvgPopulation → int, ShareOfTotalPopulation → decimal

   Implementacja testu:

```
SELECT COLUMN_NAME, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'SocialHelp';

SELECT COLUMN_NAME, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'NYCPopulation';
```

   Wynik:



	COLUMN_NAME	DATA_TYPE
1	Borough	nvarchar
2	TotalSNAPRecipients	int
3	TotalCashAssistanceRecipients	int
4	TotalMEDICAIDRecipients	int

	COLUMN_NAME	DATA_TYPE
1	Borough	nvarchar
2	AvgPopulation	int
3	ShareOfTotalPopulation	decimal

   Zgodny z oczekiwaniami ✅

Tabela Crashes

1. **Testy dat**
   Oczekiwany wynik:

   Brak wierszy gdzie pole CrashDate lub CrashTime ma brakujące wartości lub wartości wypełnione domyślną datą np. 1753-01-01.

Implementacja testu:

```
select count(*) as NullDateOrTime from Crashes
where CrashDate is null or CrashTime is null or year(CrashDate)
!=2020
```

Wynik:

	NullDateOrTime
1	0

Zgodny z oczekiwaniami ✅

2. **Test metryk**

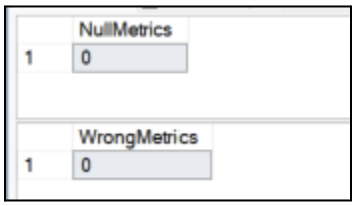
Oczekiwany wynik:

Brak wierszy gdzie pola przechowujące liczby poszkodowanych typu PeopleInjured są brakujące lub mniejsze od 0.

Implementacja testu:

```
select count(*) as NullMetrics from Crashes
where PeopleInjured is null or PeopleKilled is null or
PedestriansInjured is null or PedestriansKilled is null
or CyclistsInjured is null or CyclistsKilled is null or
MotoristInjured is null or MotoristsKilled is null

select count(*) as WrongMetrics from Crashes
where PeopleInjured <0 or PeopleKilled<0 or CyclistsInjured<0
or CyclistsKilled<0 or PedestriansInjured<0 or
PedestriansKilled<0 or MotoristInjured<0 or MotoristsKilled<0
```

Wynik:



	NullMetrics
1	0

	WrongMetrics
1	0

Zgodny z oczekiwaniami ✅

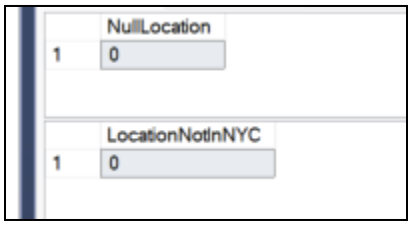
3. **Test lokalizacji**
   Oczekiwany wynik:

   Brak wierszy gdzie pola Latitude, Longitude, Borough i StreetName są brakujące oraz kolumny Latitude i Longitude powinny odpowiadać wartościom geograficznym lokalizacji Nowego Jorku. Nie powinno być wierszy, gdzie Latitude jest poza przedziałem ok [40,41] oraz Longitude poza przedziałem [-75, -73].

   Implementacja testu:

```
select count(*) as NullLocation from Crashes
where Latitude is null or Longitude is null or trim(Borough)
='' or Borough is null or trim(StreetName) = '' or StreetName
is null

select count(*) as LocationNotInNYC
from Crashes where Latitude <40 or Latitude>41 or Longitude<-75
or Longitude>-73
```

   Wynik:



   Zgodny z oczekiwaniami ✅

4. **Test typów danych**
   Oczekiwany wynik:

   Kolumna CrashDate → Date
   Kolumna CrashTime → time
   Kolumny Latitude, Longitude → float
   Kolumny informujące o poszkodowanych np. PeopleKilled → int
   Kolumny Borough, StreetName, Factors, Vehicle Types → nvarchar

   Implementacja testu:

```
SELECT COLUMN_NAME, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'Crashes';
```

Wynik:

	COLUMN_NAME	DATA_TYPE
1	Crash_ID	int
2	CrashDate	date
3	CrashTime	time
4	Borough	nvarchar
5	StreetName	nvarchar
6	Latitude	float
7	Longitude	float
8	PeopleInjured	int
9	PeopleKilled	int
10	PedestriansInjured	int
11	PedestriansKilled	int
12	CyclistsInjured	int
13	CyclistsKilled	int
14	MotoristInjured	int
15	MotoristsKilled	int
16	Factor1	nvarchar
17	Factor2	nvarchar
18	Factor3	nvarchar
19	Factor4	nvarchar
20	Factor5	nvarchar
21	Vehicle1	nvarchar
22	Vehicle2	nvarchar
23	Vehicle3	nvarchar
24	Vehicle4	nvarchar
25	Vehicle5	nvarchar
26	VehicleAmt	int

Zgodny z oczekiwaniami ✅

5. **Test NULL w kolumnach Factor i Vehicle Type**
   Oczekiwany wynik:

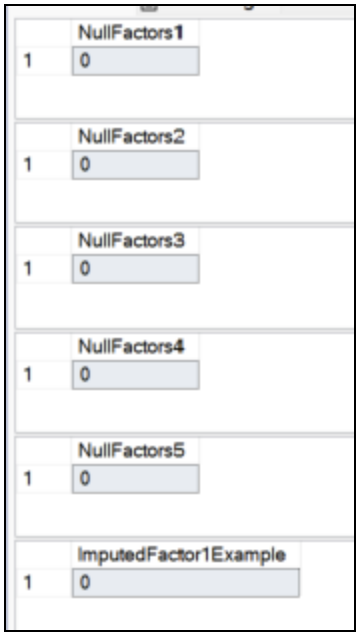
   W tabeli dopuszczone są wartości NULL w kolumnach typu Factor i Vehicle type tylko w przypadku, gdy oba pola odpowiadające za jeden pojazd są brakujące. To znaczy, jeśli Factor3 i Vehicle3 oba są NULL to pojazd nie brał udziału w zdarzeniu, jeśli natomiast tylko jedno z tych pól jest NULL, to zostaje ono zastąpione ‘Unknown’ i jest wliczane jako pojazd biorący udział w wypadku. Zatem kolumna Factor powinna mieć wartość null tylko gdy odpowiadająca kolumna VehicleType również ma wartość NULL, wpp brak NULLi.

   Implementacja testu:

```
-- 5 Null values only if Vehicle + Factor are missing
select count(*) as NullFactors1 from Crashes
where (Factor1 is null or trim(Factor1) = '') and (Vehicle1 is
not null and trim(Vehicle1) != '')
select count(*) as NullFactors2 from Crashes
```

```
where (Factor2 is null or trim(Factor2) = '') and (Vehicle2 is
not null and trim(Vehicle2) != '')
select count(*) as NullFactors3 from Crashes
where (Factor3 is null or trim(Factor3) = '') and (Vehicle3 is
not null and trim(Vehicle3) != '')
select count(*) as NullFactors4 from Crashes
where (Factor4 is null or trim(Factor4) = '') and (Vehicle4 is
not null and trim(Vehicle4) != '')
select count(*) as NullFactors5 from Crashes
where (Factor5 is null or trim(Factor5) = '') and (Vehicle5 is
not null and trim(Vehicle5) != '')

-- 6 Null changed to Unknown when Factor is missing and vehicle
is not
select count(*) as ImputedFactor1Example from Crashes
where Factor1 = 'Unknown' and (Vehicle1 is null or
trim(Vehicle1)='')
```

Wynik:



Zgodny z oczekiwaniami ✅

6. **Test nowej Calculated Column - Vehicles Amount**
   Oczekiwany wynik:

   Kolumna VehiclesAmt zawiera poprawnie obliczoną ilość pojazdów biorących udział w wypadku jako sumę parami nie brakujących kolumn Factor + VehicleType.

Implementacja testu:

```
SELECT COUNT(*) AS IncorrectVehicleAmt
FROM Crashes
WHERE vehicleAmt != (
      (CASE WHEN Factor1 IS NOT NULL AND Vehicle1 IS NOT NULL
THEN 1 ELSE 0 END) +
      (CASE WHEN Factor2 IS NOT NULL AND Vehicle2 IS NOT NULL
THEN 1 ELSE 0 END) +
      (CASE WHEN Factor3 IS NOT NULL AND Vehicle3 IS NOT NULL
THEN 1 ELSE 0 END) +
      (CASE WHEN Factor4 IS NOT NULL AND Vehicle4 IS NOT NULL
THEN 1 ELSE 0 END) +
      (CASE WHEN Factor5 IS NOT NULL AND Vehicle5 IS NOT NULL
THEN 1 ELSE 0 END)
);
```

Wynik:

	IncorrectVehicleAmt
1	0

Zgodny z oczekiwaniami ✅

Tabela SpeedLimits

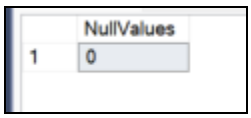
1. **Test brakujących wartości**
   Oczekiwany wynik:

   Żadna z 3 kolumn w tabeli nie ma brakujących wartości.

   Implementacja testu:

   ```
   select count(*) as NullValues from SpeedLimits
   where StreetName is null or trim(StreetName)='' or SpeedLimit
   is null or IsSigned is null or trim(IsSigned)=''
   ```

   Wynik:



	NullValues
1	0

Zgodny z oczekiwaniami ✅

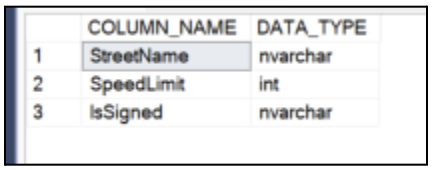
2. **Test typów danych**
   Oczekiwany wynik:

   Kolumny StreetName oraz IsSigned → nvarchar
   Kolumna SpeedLimit → int

   Implementacja testu:

```
SELECT COLUMN_NAME, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'SpeedLimits';
```

   Wynik:



	COLUMN_NAME	DATA_TYPE
1	StreetName	nvarchar
2	SpeedLimit	int
3	IsSigned	nvarchar

   Zgodny z oczekiwaniami ✅

Testy tabel wymiarów

Kolejnym krokiem ETL było stworzenie tabel wymiarów w hurtowni danych. W tym celu korzystano zarówno z wcześniej wypełnionej Staging DB oraz z dodatkowych plików csv. Testy w tej części mają na celu weryfikację procesów takich jak:

1. Załadowanie tabeli wymiarów w określonym schemacie i ustalenie klucza głównego
2. Całkowita eliminacja braków danych poprzez usuwanie rekordów lub imputację braków
3. Mapowania danych z tabel źródłowych w celu wypełnienia wymiaru
4. Czyszczenie i uspójnienie wartości w kolumnach tekstowych

CallTypes Dim

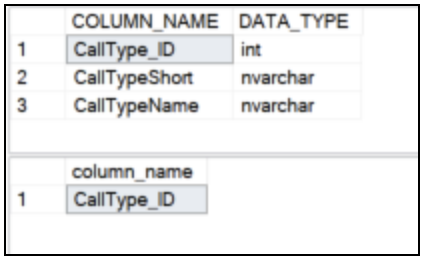
1. **Schemat**
   Oczekiwany wynik:

   Powinna zostać wczytana tabela zawierająca sztuczny klucz główny, kolumnę CallTypeShort będącą skrótem oraz jej wyjaśnienie jako kolumna CallTypeName. Wszystkie kolumny z wyjątkiem klucza głównego powinny być typu nvarchar.

   Implementacja testu:

```
SELECT COLUMN_NAME, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'CallTypesDim';

- Test primary key
SELECT
      k.column_name
FROM
      information_schema.table_constraints t
JOIN
      information_schema.key_column_usage k
  ON t.constraint_name = k.constraint_name
      AND t.table_name = k.table_name
WHERE
      t.constraint_type = 'PRIMARY KEY'
      AND t.table_name = 'CallTypesDim';
```

   Wynik:



	COLUMN_NAME	DATA_TYPE
1	CallType_ID	int
2	CallTypeShort	nvarchar
3	CallTypeName	nvarchar

	column_name
1	CallType_ID

Zgodny z oczekiwaniami ✅

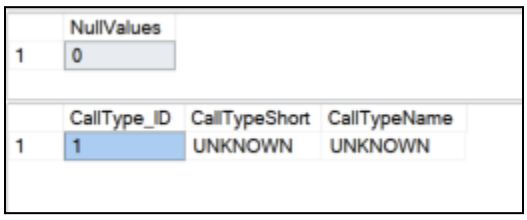
2. **Test braków danych**
   Oczekiwany wynik:

   Żadna kolumna nie powinna zawierać braków danych. Odzwierciedlenie brakującego typu zgłoszenia powinno zostać zaadresowane jako dodatkowy wiersz z wartościami ‘Unknown’ w obu kolumnach.

   Implementacja testu:

```
select count(*) as NullValues from CallTypesDim
where CallType_ID is null or CallTypeShort is null or
trim(CallTypeShort)='' or CallTypeName is null or
trim(CallTypeName)=''

select * from CallTypesDim
where CallTypeShort = 'UNKNOWN'
```

   Wynik:



	NullValues
1	0

	CallType_ID	CallTypeShort	CallTypeName
1	1	UNKNOWN	UNKNOWN

   Zgodny z oczekiwaniami ✅

3. **Test załadowania pełnego udostępnionego słownika**
   Oczekiwany wynik:

   Udostępniony na stronie NYC Data plik Excel zawiera 268 wierszy. W wymiarze powinno znaleźć się zatem 269 rekordów - 268 z tabeli bazowej oraz dodatkowy rekord ‘Unknown’

   Implementacja testu:

```
Select count(*) from CallTypesDim
```

Wynik:

	RowsCount
1	269

Zgodny z oczekiwaniami ✅

SeverityLevels Dim

Jako wymiar stworzony ręcznie, można jedynie przetestować poprawne wypełnienie tabeli zaproponowanymi wartościami oraz poprawne ustawienie klucza głównego.

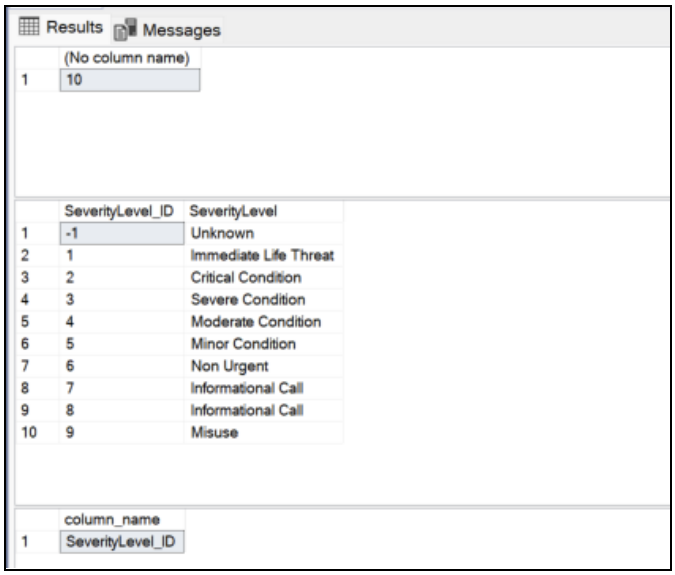
Oczekiwany wynik:

Tabela posiada 10 rekordów, gdzie kody od 1-9 są kodami pojawiającymi się w tabeli faktów, a kod -1 odpowiada wartości ‘Unknown’. Kluczem głównym jest kolumna SeverityLevel_ID.

Implementacja testu:

```
-- 1 row count check
select count(*) from SeverityLevelsDim

-- 2 all values check
select * from SeverityLevelsDim

-- primary key check
SELECT
    k.column_name
FROM
    information_schema.table_constraints t
JOIN
    information_schema.key_column_usage k
  ON t.constraint_name = k.constraint_name
     AND t.table_name = k.table_name
WHERE
    t.constraint_type = 'PRIMARY KEY'
    AND t.table_name = 'SeverityLevelsDim';
```

Wynik:



Results Messages

	(No column name)
1	10

	SeverityLevel_ID	SeverityLevel
1	-1	Unknown
2	1	Immediate Life Threat
3	2	Critical Condition
4	3	Severe Condition
5	4	Moderate Condition
6	5	Minor Condition
7	6	Non Urgent
8	7	Informational Call
9	8	Informational Call
10	9	Misuse

	column_name
1	SeverityLevel_ID

Zgodny z oczekiwaniami ✅

DistrictDetails Dim

Tabela DistrictDetails została stworzona z połączenia informacji z dwóch tabel źródłowych - Social Help oraz NYC Population. Poniżej przedstawione są testy głównych kroków, które zostały wykonane w celu wypełnienia tej tabeli. Dane są docelowo pobierane z tabeli StagingDistrictTable, która zawiera jednak dokładnie ten sam zestaw operacji i wartości co tabela wymiaru z wyjątkiem implementacji SCD2. Posłużymy się tabelą Staging do przetestowania funkcjonalności Slow Changing Dimension w ostatnim podpunkcie testów.

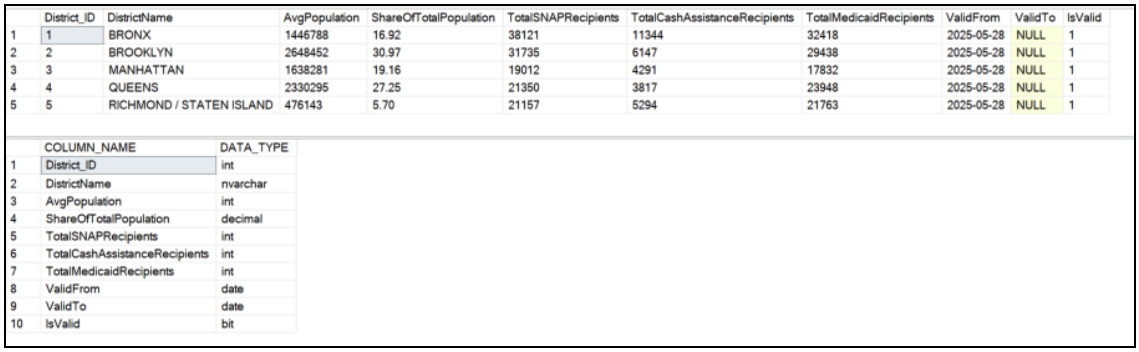
1. **Schemat**
   Oczekiwany wynik:

   Tabela DistrictDetails powinna zawierać sztuczny klucz główny, kolumnę DistrictName zawierającą pełną nazwę dzielnicy, kolumny numeryczne opisujące dzielnicę oraz 3 kolumny związane z mechanizmem SCD2, gdzie początkowo każda data ValidFrom powinna być datą teraźniejszą, ValidTo NULLem a IsValid 1.

   Implementacja testu:

```
select * from DistrictDetailsDim;
SELECT COLUMN_NAME, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'DistrictDetailsDim';
```

   Wynik:



	District_ID	DistrictName	AvgPopulation	ShareOfTotalPopulation	TotalSNAPRecipients	TotalCashAssistanceRecipients	TotalMedicaidRecipients	ValidFrom	ValidTo	IsValid
1	1	BRONX	1446788	16.92	38121	11344	32418	2025-05-28	NULL	1
2	2	BROOKLYN	2648452	30.97	31735	6147	29438	2025-05-28	NULL	1
3	3	MANHATTAN	1638281	19.16	19012	4291	17832	2025-05-28	NULL	1
4	4	QUEENS	2330295	27.25	21350	3817	23948	2025-05-28	NULL	1
5	5	RICHMOND / STATEN ISLAND	476143	5.70	21157	5294	21763	2025-05-28	NULL	1

	COLUMN_NAME	DATA_TYPE
1	District_ID	int
2	DistrictName	nvarchar
3	AvgPopulation	int
4	ShareOfTotalPopulation	decimal
5	TotalSNAPRecipients	int
6	TotalCashAssistanceRecipients	int
7	TotalMedicaidRecipients	int
8	ValidFrom	date
9	ValidTo	date
10	IsValid	bit

Zgodny z oczekiwaniami ✅

2. **Test braków danych i mapowania**
   Oczekiwany wynik:

   Tabela wymiaru nie powinna zawierać brakujących danych w żadnej kolumnie oprócz ValidTo. Nazwy dzielnic powinny być zmapowane na wielkie litery oraz zamiast różnych wariantów Staten Island powinno pojawić się ‘RICHMOND / STATEN ISLAND’

   Wynik:

   Wynik jest widoczny na załączonym zdjęciu dla testu powyżej.
   Zgodny z oczekiwaniami ✅

3. **Test agregacji tabel Social Help, NYCPopulation**

   a) Tyle samo unikalnych nazw dzielnic
   Oczekiwany wynik:

   Ilość unikalnych wartości w kolumnie Borough tabel NYCPopulation, SocialHelp oraz DistrictDetailsDim jest taka sama - nie pomijamy żadnej dzielnicy.

   Implementacja testu:

```
select
(select count(distinct Borough) from
ELT_staging_DB.dbo.NYCPopulation) as Pop_Boroughs,
(select count(distinct Borough) from
ELT_staging_DB.dbo.SocialHelp) as Social_Boroughs,
(select count(distinct DistrictName) from
FinalWarehouse.dbo.DistrictDetailsDim) as Dim_Boroughs;
```

   Wynik:

	Pop_Boroughs	Social_Boroughs	Dim_Boroughs
1	5	5	5

Zgodny z oczekiwaniami ✅

b) Uśrednione wyniki znajdujące się w tabeli DistrictDetails odpowiadają tym z tabeli SocialHelp

Oczekiwany wynik:

Po zgrupowaniu tabeli SocialHelp po kolumnie Borough i obliczeniu uśrednionych metryk numerycznych nie powinny one różnić się od tych załadowanych w tabeli wymiarów o więcej niż 1 (epsilon błędu numerycznego związanego z zaokrągleniami podczas uśredniania i mapowania na integer). W każdej kolumnie reprezentującej to czy wyniki są różne oczekiwana wartość to ‘NO’.

Implementacja testu (GPT-4):

```
WITH aggregated_help AS (
      SELECT
      CASE
            WHEN Borough = 'Staten_Island' THEN 'RICHMOND /
STATEN ISLAND'
            ELSE Borough
      END AS Borough,
      AVG(TotalSNAPRecipients) AS avg_snap,
      AVG(TotalCashAssistanceRecipients) AS avg_cash,
      AVG(TotalMEDICAIDRecipients) AS avg_medicaid
      FROM ELT_staging_DB.dbo.SocialHelp
      GROUP BY
      CASE
            WHEN Borough = 'Staten_Island' THEN 'RICHMOND /
STATEN ISLAND'
            ELSE Borough
      END
)
SELECT
      d.DistrictName AS Borough,
      CASE
      WHEN ABS(h.avg_snap - d.TotalSNAPRecipients) > 1 THEN
'YES'
      ELSE 'NO'
      END AS snap_diff,

      CASE
```

```
      WHEN ABS(h.avg_cash -
d.TotalCashAssistanceRecipients) > 1 THEN 'YES'
      ELSE 'NO'
      END AS cash_diff,

      CASE
      WHEN ABS(h.avg_medicaid - d.TotalMEDICAIDRecipients)
> 1 THEN 'YES'
      ELSE 'NO'
      END AS medicaid_diff
FROM aggregated_help h
JOIN FinalWarehouse.dbo.DistrictDetailsDim d
      ON h.Borough = d.DistrictName;
```

Wynik:

	Borough	snap_diff	cash_diff	medicaid_diff
1	BRONX	NO	NO	NO
2	BROOKLYN	NO	NO	NO
3	MANHATTAN	NO	NO	NO
4	QUEENS	NO	NO	NO
5	RICHMOND / STATEN ISLAND	NO	NO	NO

Zgodny z oczekiwaniami ✅

4. Test mechanizmu SCD2

Oczekiwany wynik:

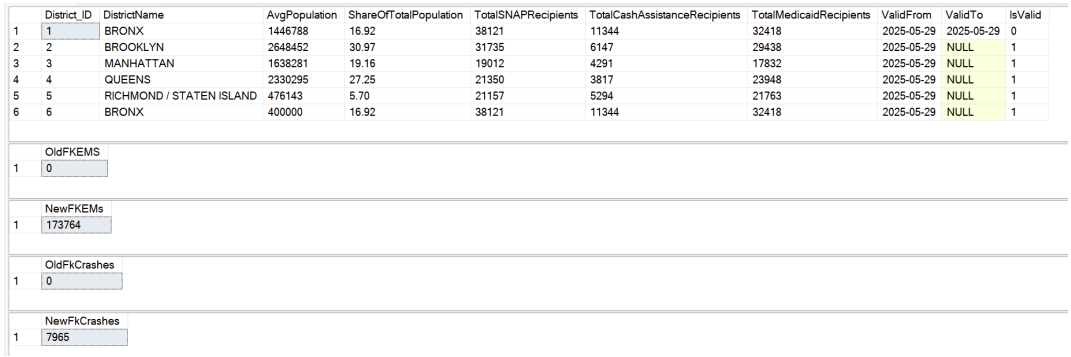
Po zmianie wartości jakiejkolwiek kolumny dotyczącej konkretnej dzielnicy w danych źródłowych i załadowaniu pakietu ponownie, w tabeli wymiarów powinien pokazać się nowy rekord, a stary powinien mieć zmienione wartości kolumn ValidTo i IsValid. Dokładniej, modyfikujemy wartość kolumny AvgPopulation dla dzielnicy Bronx zmieniając ją na 400000. Obecny rekord dla dzielnic Bronx powinien zostać zmodyfikowany ustawiając ValidTo na datę zmiany oraz IsValid na zero, a nowy rekord powinien zostać dodany z wartościami ValidFrom na datę zmiany, ValidTo = NULL oraz IsValid = 1. Równocześnie wszystkie klucze obce z tabel faktów, które wskazywały na dzielnice Bronx powinny zostać uaktualnione, aby wskazywały na jej aktualną wersję.

Implementacja testu:

```
select * from DistrictDetailsDim
select count(*) as OldFKEMS from EMSFacts where BoroughKey = 1
select count(*) as NewFKEMs from EMSFacts where BoroughKey = 6
select count(*) as OldFkCrashes from CrashFacts where
BoroughKey =1
```

```
select count(*) as NewFkCrashes from CrashFacts where
BoroughKey =6
```

Wynik:



	District_ID	DistrictName	AvgPopulation	ShareOfTotalPopulation	TotalSNAPRecipients	TotalCashAssistanceRecipients	TotalMedicaidRecipients	ValidFrom	ValidTo	IsValid
1	1	BRONX	1446788	16.92	38121	11344	32418	2025-05-29	2025-05-29	0
2	2	BROOKLYN	2648452	30.97	31735	6147	29438	2025-05-29	NULL	1
3	3	MANHATTAN	1638281	19.16	19012	4291	17832	2025-05-29	NULL	1
4	4	QUEENS	2330295	27.25	21350	3817	23948	2025-05-29	NULL	1
5	5	RICHMOND / STATEN ISLAND	476143	5.70	21157	5294	21763	2025-05-29	NULL	1
6	6	BRONX	400000	16.92	38121	11344	32418	2025-05-29	NULL	1

	OldFKEMS
1	0

	NewFKEMs
1	173764

	OldFkCrashes
1	0

	NewFkCrashes
1	7965

Jak widać, nie ma już kluczy obcych odnoszących się do wiersza o ID 1, zostały one zamienione na BoroughKey = 6.
Zgodny z oczekiwaniami ✅

SpeedLimits Dim

Wymiar z mechanizmem SCD2, do którego odwołuje się tabela faktów Crashes. Nie wymagał dodatkowych transformacji danych, jednak można przetestować update rekordów oraz ich ilość.

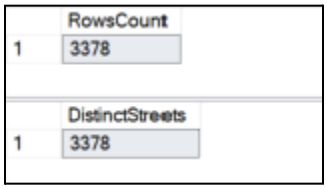
1. **Ilość rekordów w wymiarze po pierwszym ładowaniu** (bez modyfikacji rekordów bazowych)
   Oczekiwany wynik:

   Ilość rekordów w tabeli wymiarów powinna być taka sama jak ilość unikalnych nazw ulic w tabeli Crashes z Staging DB.

   Implementacja testu:

   ```
   select count(*) as RowsCount from SpeedLimitsDim
   select count(distinct StreetName) as DistinctStreets from
   [ELT_staging_DB].dbo.Crashes
   ```

   Wynik:



	RowsCount
1	3378

	DistinctStreets
1	3378

Zgodny z oczekiwaniami ✅

2. **Mechanizm SCD2**
   Oczekiwany wynik:

   Po zmianie wartości SpeedLimit dla ulicy AVENUE H z 25 na 30 zostanie dodany nowy rekord do tabeli wymiarów z ValidFrom ustawionym na datę modyfikacji, ValidTo = NULL oraz IsValid = 1, a poprzedni wiersz dla tej ulicy zostanie zmodyfikowany ustawiając ValidTo oraz IsValid = 0. Ulica AVENUE H początkowo ma ID w wymiarze = 1. Wszystkie rekordy z tabeli CrashFacts, które się do niej odnoszą powinny zostać uaktualnione, aby odnosiły się do aktualnego rekordu.

   Implementacja testu: (przed zmianą rekordu 40 wierszy z tabeli faktów odnosi się do AVENUE H z ID = 1)

```
select count(*) oldFK from CrashFacts where StreetKey = 1
select count(*) NewFK from CrashFacts where StreetKey = 3379

select * from SpeedLimitsDim
where StreetName = 'AVENUE H'
```

   Wynik:

	oldFK
1	0

	NewFK
1	40

	Street_ID	StreetName	SpeedLimit	IsSigned	ValidFrom	ValidTo	IsValid
1	1	AVENUE H	25	YES	2025-05-29	2025-05-29	0
2	3379	AVENUE H	30	YES	2025-05-29	NULL	1

   Jak widać brak już kluczy obcych w tabeli faktów odnoszących się do nieaktualnych informacji o ulicy AVENUE H.
   Zgodny z oczekiwaniami ✅

ContributingFactors Dim

Wymiar służący do przechowywania wszystkich możliwych przyczyn wypadków występujących w tabeli Crashes. Został wypełniony poprzez zebranie wszystkich unikalnych wartości ze wszystkich kolumn Factor1,...Factor5.

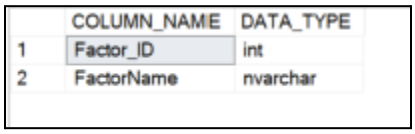
1. **Schemat**
   Oczekiwany wynik:

Wymiar powinien zawierać dwie kolumny - sztuczny klucz główny Factor_ID oraz kolumnę tekstową FactorName

Implementacja testu:

```
SELECT COLUMN_NAME, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'ContributingFactorsDim';
```

Wynik:



	COLUMN_NAME	DATA_TYPE
1	Factor_ID	int
2	FactorName	nvarchar

Zgodny z oczekiwaniami ✅

2. **Braki danych**

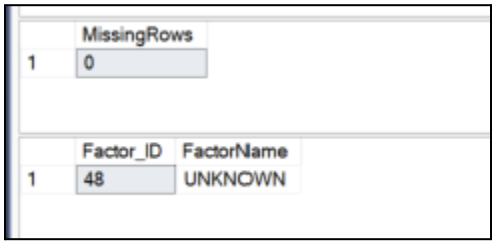
Oczekiwany wynik:

Nie powinno być żadnego rekordu z brakującymi danymi. Dopuszczone są wartości ‘Unknown’.

Implementacja testu:

```
-- 2 missing values
select count(*) as MissingRows from ContributingFactorsDim
where FactorName is null or trim(FactorName)=''

-- 3 Unknown values
select * from ContributingFactorsDim
where FactorName='UNKNOWN'
```

Wynik:



	MissingRows
1	0

	Factor_ID	FactorName
1	48	UNKNOWN

Zgodny z oczekiwaniami ✅

3. **Ilość rekordów**

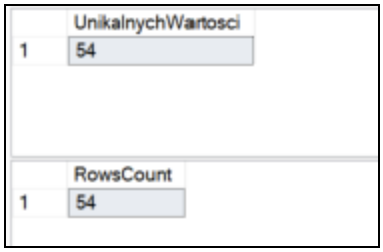
Oczekiwany wynik:

Ilość rekordów w tabeli wymiarów powinna być równa ilości unikalnych wartości we wszystkich kolumnach Factor1,...Factor5 z Staging DB Crashes.

Implementacja testu:

```
SELECT COUNT(DISTINCT value) AS UnikalnychWartosci
FROM (
    SELECT CASE WHEN Factor1 = 'Unspecified' THEN 'Unknown'
ELSE Factor1 END AS value
    FROM [ELT_staging_DB].dbo.Crashes
    WHERE Factor1 IS NOT NULL

    UNION
    SELECT CASE WHEN Factor2 = 'Unspecified' THEN 'Unknown'
ELSE Factor2 END
    FROM [ELT_staging_DB].dbo.Crashes
    WHERE Factor2 IS NOT NULL

    UNION
    SELECT CASE WHEN Factor3 = 'Unspecified' THEN 'Unknown'
ELSE Factor3 END
    FROM [ELT_staging_DB].dbo.Crashes
    WHERE Factor3 IS NOT NULL

    UNION
    SELECT CASE WHEN Factor4 = 'Unspecified' THEN 'Unknown'
ELSE Factor4 END
    FROM [ELT_staging_DB].dbo.Crashes
    WHERE Factor4 IS NOT NULL

    UNION
    SELECT CASE WHEN Factor5 = 'Unspecified' THEN 'Unknown'
ELSE Factor5 END
    FROM [ELT_staging_DB].dbo.Crashes
    WHERE Factor5 IS NOT NULL
) AS AllFactors;


SELECT COUNT(*) AS RowsCount FROM ContributingFactorsDim;
```

Wynik:



	UnikalnychWartosci
1	54

	RowsCount
1	54

Zgodny z oczekiwaniami ✅

VehicleTypes Dim

Wymiar służący do przechowywania wszystkich możliwych typów pojazdów biorących udział w wypadkach z tabeli Crashes. Został wypełniony poprzez zebranie wszystkich unikalnych wartości ze wszystkich kolumn Vehicle1,...Vehicle5.

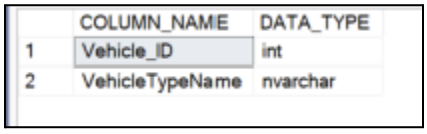
4. **Schemat**
   Oczekiwany wynik:

   Wymiar powinien zawierać dwie kolumny - sztuczny klucz główny Vehicle_ID oraz kolumnę tekstową VehicleTypeName

   Implementacja testu:

   ```
   SELECT COLUMN_NAME, DATA_TYPE
   FROM INFORMATION_SCHEMA.COLUMNS
   WHERE TABLE_NAME = 'VehicleTypesDim';
   ```

   Wynik:



	COLUMN_NAME	DATA_TYPE
1	Vehicle_ID	int
2	VehicleTypeName	nvarchar

   Zgodny z oczekiwaniami ✅

5. **Braki danych**
   Oczekiwany wynik:

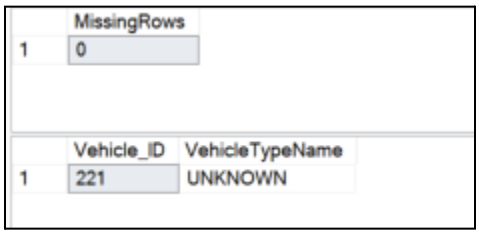
   Nie powinno być żadnego rekordu z brakującymi danymi. Dopuszczone są wartości ‘Unknown’

   Implementacja testu:

   ```
   -- 2 missing values
   ```

```
select count(*) as MissingRows from VehicleTypesDim
where VehicleTypeName is null or trim(VehicleTypeName)=''

-- 3 Unknown values
select * from VehicleTypesDim
where VehicleTypeName='UNKNOWN'
```

Wynik:



	MissingRows
1	0

	Vehicle_ID	VehicleTypeName
1	221	UNKNOWN

Zgodny z oczekiwaniami ✅

6. **Ilość rekordów**

Oczekiwany wynik:

Ilość rekordów w tabeli wymiarów powinna być równa ilości unikalnych wartości we wszystkich kolumnach Vehicle1… Vehicle5 z Staging DB Crashes.

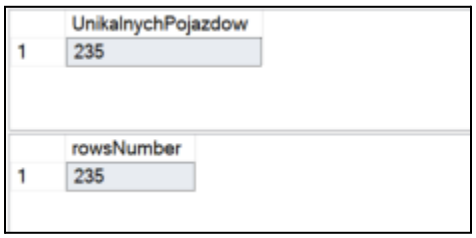
Implementacja testu:

```
SELECT COUNT(DISTINCT value) AS UnikalnychPojazdow
FROM (
    SELECT CASE
        WHEN Vehicle1 IN ('0', 'UKN', 'UNK', 'n/a',
'UNATTACHED') THEN 'UNKNOWN'
        WHEN Vehicle1 = '18 WEELER' THEN '18 WHEELER'
        WHEN Vehicle1 = 'AMBULENCE' THEN 'AMBULANCE'
        WHEN Vehicle1 = 'backh' THEN 'BACK HOE'
        WHEN Vehicle1 = 'comme' THEN 'COMMERCIAL'
        WHEN Vehicle1 = 'CONCRETE M' THEN 'CONCRETE MIXER'
        WHEN Vehicle1 = 'DIRTBIKE' THEN 'DIRT BIKE'
        ELSE Vehicle1
    END AS value
    FROM [ELT_staging_DB].dbo.Crashes
    WHERE Vehicle1 IS NOT NULL

    UNION
    SELECT CASE
```

```
        WHEN Vehicle2 IN ('0', 'UKN', 'UNK', 'n/a',
'UNATTACHED') THEN 'UNKNOWN'
        WHEN Vehicle2 = '18 WEELER' THEN '18 WHEELER'
        WHEN Vehicle2 = 'AMBULENCE' THEN 'AMBULANCE'
        WHEN Vehicle2 = 'backh' THEN 'BACK HOE'
        WHEN Vehicle2 = 'comme' THEN 'COMMERCIAL'
        WHEN Vehicle2 = 'CONCRETE M' THEN 'CONCRETE MIXER'
        WHEN Vehicle2 = 'DIRTBIKE' THEN 'DIRT BIKE'
        ELSE Vehicle2
    END
    FROM [ELT_staging_DB].dbo.Crashes
    WHERE Vehicle2 IS NOT NULL

    UNION
    SELECT CASE
        WHEN Vehicle3 IN ('0', 'UKN', 'UNK', 'n/a',
'UNATTACHED') THEN 'UNKNOWN'
        WHEN Vehicle3 = '18 WEELER' THEN '18 WHEELER'
        WHEN Vehicle3 = 'AMBULENCE' THEN 'AMBULANCE'
        WHEN Vehicle3 = 'backh' THEN 'BACK HOE'
        WHEN Vehicle3 = 'comme' THEN 'COMMERCIAL'
        WHEN Vehicle3 = 'CONCRETE M' THEN 'CONCRETE MIXER'
        WHEN Vehicle3 = 'DIRTBIKE' THEN 'DIRT BIKE'
        ELSE Vehicle3
    END
    FROM [ELT_staging_DB].dbo.Crashes
    WHERE Vehicle3 IS NOT NULL

    UNION
    SELECT CASE
        WHEN Vehicle4 IN ('0', 'UKN', 'UNK', 'n/a',
'UNATTACHED') THEN 'UNKNOWN'
        WHEN Vehicle4 = '18 WEELER' THEN '18 WHEELER'
        WHEN Vehicle4 = 'AMBULENCE' THEN 'AMBULANCE'
        WHEN Vehicle4 = 'backh' THEN 'BACK HOE'
        WHEN Vehicle4 = 'comme' THEN 'COMMERCIAL'
        WHEN Vehicle4 = 'CONCRETE M' THEN 'CONCRETE MIXER'
        WHEN Vehicle4 = 'DIRTBIKE' THEN 'DIRT BIKE'
        ELSE Vehicle4
    END
    FROM [ELT_staging_DB].dbo.Crashes
    WHERE Vehicle4 IS NOT NULL

    UNION
    SELECT CASE
```

```
        WHEN Vehicle5 IN ('0', 'UKN', 'UNK', 'n/a',
'UNATTACHED') THEN 'UNKNOWN'
        WHEN Vehicle5 = '18 WEELER' THEN '18 WHEELER'
        WHEN Vehicle5 = 'AMBULENCE' THEN 'AMBULANCE'
        WHEN Vehicle5 = 'backh' THEN 'BACK HOE'
        WHEN Vehicle5 = 'comme' THEN 'COMMERCIAL'
        WHEN Vehicle5 = 'CONCRETE M' THEN 'CONCRETE MIXER'
        WHEN Vehicle5 = 'DIRTBIKE' THEN 'DIRT BIKE'
        ELSE Vehicle5
    END
    FROM [ELT_staging_DB].dbo.Crashes
    WHERE Vehicle5 IS NOT NULL

) AS AllVehicles;

select count(*) as rowsNumber from VehicleTypesDim
```

Wynik:



	UnikalnychPojazdow
1	235

	rowsNumber
1	235

Zgodny z oczekiwaniami ✅

Testy tabel faktów

Ostatnim elementem hurtowni danych są tabele faktów, które zostały przetestowane w celu weryfikacji:

- Spójności i czystości załadowanych danych
- Poprawności połączeń z tabelami wymiarów
- Spójności mapowania do wymiarów względem oryginalnych tabel

Na tym etapie nie testowano już poprawności wyliczonych kolumn oraz ich logiki, ponieważ zostało to wykonane na etapie Staging DB przetestowanej powyżej, względem której nie było już zmian w późniejszych krokach procesu ETL.

EMS Facts

1. **Schemat danych**
   Oczekiwany wynik:

   Kolumny odpowiadające kluczom obcym powinny mieć typ danych int, kolumny typu durations decimal.

   Implementacja testu:

   ```
   SELECT COLUMN_NAME, DATA_TYPE
   FROM INFORMATION_SCHEMA.COLUMNS
   WHERE TABLE_NAME = 'EMSFacts';
   ```

   Wynik:

	COLUMN_NAME	DATA_TYPE
1	EMS_ID	int
2	IncidentDateKey	int
3	IncidentTimeKey	int
4	ActivationDateKey	int
5	ActivationTimeKey	int
6	OnSceneDateKey	int
7	OnSceneTimeKey	int
8	ToHospitalDateKey	int
9	ToHospitalTimeKey	int
10	ClosingDateKey	int
11	ClosingTimeKey	int
12	ActivationDuration	decimal
13	OnSceneArrivalDuration	decimal
14	ToHospitalArrivalDuration	decimal
15	InterventionDuration	decimal
16	InitialCallTypeKey	int
17	FinalCallTypeKey	int
18	InitialSeverityLevelKey	int
19	FinalSeverityLevelKey	int
20	BoroughKey	int

Zgodny z oczekiwaniami ✅

2. **Braki danych**
   Oczekiwany wynik:

   Tabela faktów nie powinna zawierać braków danych w żadnej z kolumn.

   Implementacja testu:

```
select count(*) as missingVals from EMSFacts
where EMS_ID is null or IncidentDateKey is null or
IncidentTimeKey is null or ActivationDateKey is null or
ActivationTimeKey is null or OnSceneDateKey is null or
OnSceneTimeKey is null or ToHospitalDateKey is null or
ToHospitalTimeKey is null or ClosingDateKey is null or
ClosingTimeKey is null or ActivationDuration is null or
OnSceneArrivalDuration is null or ToHospitalArrivalDuration is
null or InterventionDuration is null
or BoroughKey is null
```

   Wynik:

	missingVals
1	0

   Zgodny z oczekiwaniami ✅

3. **Połączenia z tabelami wymiarów**

a) **Połączenia z wymiarem DateDim**

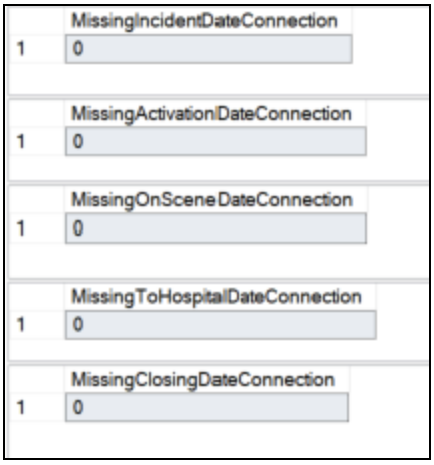
Oczekiwany wynik:

Wszystkie klucze obce IncidentDateKey, … ,ClosingDateKey mają swoje dopasowanie w tabeli wymiarów. Po zastosowaniu left join tabeli faktów i Date Dim nie będzie rekordów, gdzie kolumna np. FullDate jest NULL.

Implementacja testu:

```
select count(*) as MissingIncidentDateConnection
from EMSFacts e
left join DateDim d on e.IncidentDateKey = d.DateKey
where d.FullDate is null;

select count(*) as MissingActivationDateConnection
from EMSFacts e
left join DateDim d on e.ActivationDateKey = d.DateKey
where d.FullDate is null;

select count(*) as MissingOnSceneDateConnection
from EMSFacts e
left join DateDim d on e.OnSceneDateKey = d.DateKey
where d.FullDate is null;

select count(*) as MissingToHospitalDateConnection
from EMSFacts e
left join DateDim d on e.ToHospitalDateKey = d.DateKey
where d.FullDate is null;

select count(*) as MissingClosingDateConnection
from EMSFacts e
left join DateDim d on e.ClosingDateKey = d.DateKey
where d.FullDate is null;
```

Wynik:



	MissingIncidentDateConnection
1	0

	MissingActivationDateConnection
1	0

	MissingOnSceneDateConnection
1	0

	MissingToHospitalDateConnection
1	0

	MissingClosingDateConnection
1	0

Zgodny z oczekiwaniami ✅

b) Połączenia z wymiarem czasu

Oczekiwany wynik:

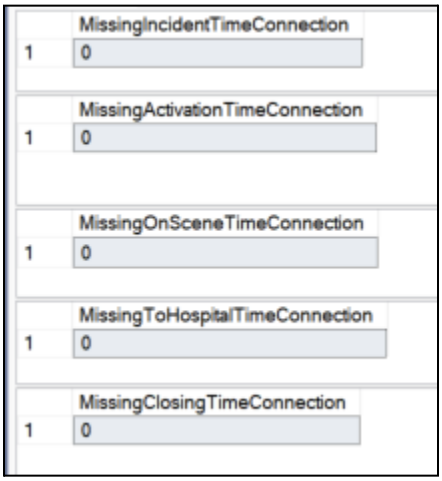
Wszystkie klucze obce IncidentTimeKey, … ,ClosingTimeKey mają swoje dopasowanie w tabeli wymiarów. Po zastosowaniu left join tabeli faktów i Time Dim nie będzie rekordów, gdzie kolumna np. TimeValue jest NULL.

Implementacja testu:

```
select count(*) as MissingIncidentTimeConnection
from EMSFacts e
left join TimeDim t on e.IncidentTimeKey = t.TimeKey
where t.TimeValue is null;

select count(*) as MissingActivationTimeConnection
from EMSFacts e
left join TimeDim t on e.ActivationTimeKey = t.TimeKey
where t.TimeValue is null;

select count(*) as MissingOnSceneTimeConnection
from EMSFacts e
left join TimeDim t on e.OnSceneTimeKey = t.TimeKey
where t.TimeValue is null;

select count(*) as MissingToHospitalTimeConnection
from EMSFacts e
left join TimeDim t on e.ToHospitalTimeKey = t.TimeKey
where t.TimeValue is null;

select count(*) as MissingClosingTimeConnection
```

```
from EMSFacts e
left join TimeDim t on e.ClosingTimeKey = t.TimeKey
where t.TimeValue is null;
```

Wynik:



	MissingIncidentTimeConnection
1	0

	MissingActivationTimeConnection
1	0

	MissingOnSceneTimeConnection
1	0

	MissingToHospitalTimeConnection
1	0

	MissingClosingTimeConnection
1	0

Zgodny z oczekiwaniami ✅

c) Połączenia z wymiarem CallTypes

Oczekiwany wynik:

Każda z kolumn InitialCallTypeKey oraz FinalCallTypeKey ma swoje dopasowanie w tabeli wymiarów. Po zastosowaniu left join tabeli faktów i CallTypes Dim nie będzie rekordów, gdzie pole np. CallTypeName jest brakujące.

Implementacja testu:

```
select count(*) as MissingInitialCallTypeConnection
from EMSFacts e
left join CallTypesDim c on e.InitialCallTypeKey =
c.CallType_ID
where c.CallTypeName is NULL or trim(c.CallTypeName)='';

select count(*) as MissingFinalCallTypeConnection
from EMSFacts e
left join CallTypesDim c on e.FinalCallTypeKey =
c.CallType_ID
where c.CallTypeName is NULL or trim(c.CallTypeName)='';
```

Wynik:

	MissingInitialCallTypeConnection
1	0

	MissingFinalCallTypeConnection
1	0

Zgodny z oczekiwaniami ✅

d) Poprawność mapowania CallTypes

Oczekiwania:

Po sprawdzeniu losowego rekordu z tabeli EMS w Staging DB oraz z tabeli faktów połączonej z wymiarem, Initial i Final Call Types, które zostały dopasowane powinny być takie same jak w oryginalnych danych.

Implementacja testu:

1. Wybór losowego rekordu i odczytanie jego Final i Initial Call Types.

```
select * from [ELT_staging_DB].dbo.EMS where EMS_ID =
200013590
```

	EMS_ID	IncidentDateTime	InitialCallType	InitialSeverityLevel	FinalCallType	FinalSeverityLevel	F
1	200013590	2020-01-01 14:23:10.000	SICK	6	ALTMEN	3	2

Initial Call Type to SICK oraz Final to ALTMEN

2. Left Join tabeli EMS Facts z wymiarami i odczytanie wartości dla zgłoszenia o tym samym ID

```
- initial call type

select e.EMS_ID, e.InitialCallTypeKey, c.CallType_ID,
c.CallTypeShort, c.CallTypeName  from EMSFacts e
left join CallTypesDim c on e.InitialCallTypeKey =
c.CallType_ID
where e.EMS_ID = 200013590

- final call type

select e.EMS_ID, e.FinalCallTypeKey, c.CallType_ID,
c.CallTypeShort, c.CallTypeName  from EMSFacts e
left join CallTypesDim c on e.FinalCallTypeKey =
c.CallType_ID
where e.EMS_ID = 200013590
```

	EMS_ID	InitialCallTypeKey	CallType_ID	CallTypeShort	CallTypeName
1	200013590	236	236	SICK	SICK

	EMS_ID	FinalCallTypeKey	CallType_ID	CallTypeShort	CallTypeName
1	200013590	7	7	ALTMEN	ALTERED MENTAL STATUS

Dołączone zostały poprawne rekordy z tabeli wymiarów - SICK oraz ALTMEN. Zgodny z oczekiwaniami ✅

e) Połączenia z wymiarem SeverityLevels

Oczekiwany wynik:

Każda z kolumn InitialSeverityLevelKey oraz FinalSeverityLevelKey ma swoje dopasowanie w tabeli wymiarów. Po zastosowaniu left join tabeli faktów i SeverityLevels Dim nie będzie rekordów, gdzie pole np. SeverityLevel jest brakujące.

Implementacja testu:

```
select count(*) as MissingInitialSeverityConnection
from EMSFacts e
left join SeverityLevelsDim s on e.InitialSeverityLevelKey
= s.SeverityLevel_ID
where s.SeverityLevel is null or trim(s.SeverityLevel)='';

select count(*) as MissingFinalSeverityConnection
from EMSFacts e
left join SeverityLevelsDim s on e.FinalSeverityLevelKey =
s.SeverityLevel_ID
where s.SeverityLevel is null or trim(s.SeverityLevel)='';
```

Wynik:

	MissingInitialSeverityConnection
1	0

	MissingFinalSeverityConnection
1	0

Zgodny z oczekiwaniami ✅

f) Poprawność mapowania Severity level

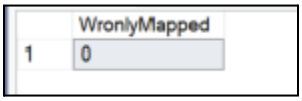
Oczekiwany wynik:

W oryginalnych danych z tabeli EMS dostępny jest tylko Severity Level Code. Można więc jedynie sprawdzić, czy mapowanie nastąpiło zgodnie z ustalonym kluczem opisanym w procesie ETL. Na przykład, czy rekordy mające SeverityLevelKey = 1 mają przypisany tekstowy opis ‘Immediate Life Threat’. Nie powinno być rekordów, które mają kod równy 1, a przypisany Severity Level nie jest równy ‘Immediate Life Threat’.

Implementacja testu:

```
select count(*) as WronglyMapped from EMSFacts e
left join SeverityLevelsDim s on e.InitialSeverityLevelKey
= s.SeverityLevel_ID
where e.InitialSeverityLevelKey = 1 and s.SeverityLevel!=
'Immediate Life Threat'
```

Wynik:



	WronlyMapped
1	0

Zgodny z oczekiwaniami ✅

g) Połączenia z wymiarem DistrictDetails

Oczekiwany wynik:

Kolumna BoroughKey ma swoje dopasowanie w tabeli wymiarów. Po zastosowaniu left join tabeli faktów i DistrictDetails Dim nie będzie rekordów, gdzie pole np. DistrictName jest brakujące.

Implementacja testu:

```
select count(*) as MissingBoroughConnection
from EMSFacts e
left join DistrictDetailsDim d on e.BoroughKey =
d.District_ID
where d.DistrictName is null or trim(d.DistrictName) =''
```

Wynik:

	MissingBoroughConnection
1	0

Zgodny z oczekiwaniami ✅

h) Poprawność mapowania Borough

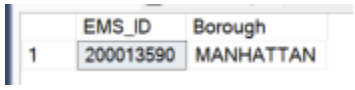
Oczekiwany wynik:

Dzielnica w której doszło do zgłoszenia została poprawnie zmapowana do wymiaru.

Implementacja testu:

1. Wybór losowego rekordu z oryginalnych danych EMS i odczytanie dzielnicy zgłoszenia

```
select EMS_ID, Borough from [ELT_staging_DB].dbo.EMS
where EMS_ID = 200013590
```

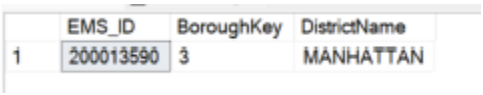


	EMS_ID	Borough
1	200013590	MANHATTAN

Zgłoszenie o takim ID miało miejsce na Manhattanie.

2. Left join tabeli faktów z tabelą DistrictDetails Dim i odczytanie zmapowanej wartości DistrictName

```
select e.EMS_ID, e.BoroughKey, d.DistrictName from
EMSFacts e
left join DistrictDetailsDim d on e.BoroughKey =
d.District_ID
where e.EMS_ID=200013590
```



	EMS_ID	BoroughKey	DistrictName
1	200013590	3	MANHATTAN

Klucz obcy poprawnie dopasowuje dzielnicę zdarzenia jako Manhattan. Zgodny z oczekiwaniami ✅

i) Foreign Key Constraint

Oczekiwany wynik:

W finalnym podsumowaniu aktywnych ograniczeń typu Foreign Key oczekiwane jest zobaczenie wszystkich sprawdzanych wyżej kluczy obcych i ich tabel z kolumną referencyjną.

Implementacja testu (GPT-4):

```
SELECT
    fk.name AS constraint_name,
```

```
    tp.name AS parent_table,
    cp.name AS parent_column,
    tr.name AS referenced_table,
    cr.name AS referenced_column
FROM
    sys.foreign_keys AS fk
INNER JOIN
    sys.foreign_key_columns AS fkc ON fk.object_id =
fkc.constraint_object_id
INNER JOIN
    sys.tables AS tp ON fkc.parent_object_id =
tp.object_id
INNER JOIN
    sys.columns AS cp ON fkc.parent_object_id =
cp.object_id AND fkc.parent_column_id = cp.column_id
INNER JOIN
    sys.tables AS tr ON fkc.referenced_object_id =
tr.object_id
INNER JOIN
    sys.columns AS cr ON fkc.referenced_object_id =
cr.object_id AND fkc.referenced_column_id = cr.column_id
WHERE
    tp.name = 'EMSFacts';
```

Wynik:

	constraint_name	parent_table	parent_column	referenced_table	referenced_column
1	FK__activationtime	EMSFacts	ActivationTimeKey	TimeDim	TimeKey
2	FK__closingtime	EMSFacts	ClosingTimeKey	TimeDim	TimeKey
3	FK__incidenttime	EMSFacts	IncidentTimeKey	TimeDim	TimeKey
4	FK__onscenetime	EMSFacts	OnSceneTimeKey	TimeDim	TimeKey
5	FK__hospitaltime	EMSFacts	ToHospitalTimeKey	TimeDim	TimeKey
6	fk_severityLevel	EMSFacts	InitialSeverityLevelKey	SeverityLevelsDim	SeverityLevel_ID
7	fk_severityLevel2	EMSFacts	FinalSeverityLevelKey	SeverityLevelsDim	SeverityLevel_ID
8	district_fk	EMSFacts	BoroughKey	DistrictDetailsDim	District_ID
9	FK__activationdate	EMSFacts	ActivationDateKey	DateDim	DateKey
10	FK__clodingdate	EMSFacts	ClosingDateKey	DateDim	DateKey
11	FK__incidentdate	EMSFacts	IncidentDateKey	DateDim	DateKey
12	FK__onscenedate	EMSFacts	OnSceneDateKey	DateDim	DateKey
13	FK__hopitaldae	EMSFacts	ToHospitalDateKey	DateDim	DateKey
14	fk_callTypes	EMSFacts	InitialCallTypeKey	CallTypesDim	CallType_ID
15	fk_callTypes2	EMSFacts	FinalCallTypeKey	CallTypesDim	CallType_ID

Zgodny z oczekiwaniami ✅

Crash Facts

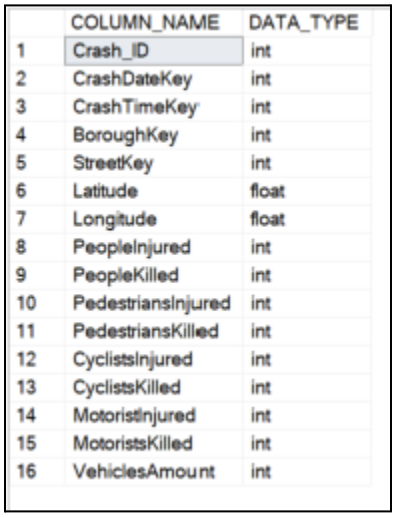
1. **Schemat danych**
   Oczekiwany wynik:

   Tabela faktów powinna zawierać kolumny Crash_ID, CrashDateKey, CrashTimeKey, BoroughKey, StreetKey będące kluczami obcymi typu int. Oprócz tego atrybuty lokalizacyjne Latitude i Longitude typu float i miary numeryczne PeopleInjured…MotoristsKilled, VehiclesAmount typu int.

   Implementacja testu:

```
SELECT COLUMN_NAME, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'CrashFacts';
```

   Wynik:



	COLUMN_NAME	DATA_TYPE
1	Crash_ID	int
2	CrashDateKey	int
3	CrashTimeKey	int
4	BoroughKey	int
5	StreetKey	int
6	Latitude	float
7	Longitude	float
8	PeopleInjured	int
9	PeopleKilled	int
10	PedestriansInjured	int
11	PedestriansKilled	int
12	CyclistsInjured	int
13	CyclistsKilled	int
14	MotoristInjured	int
15	MotoristsKilled	int
16	VehiclesAmount	int

   Zgodny z oczekiwaniami ✅

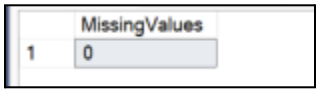
2. **Braki danych**
   Oczekiwany wynik:

   W żadnej z kolumn nie powinno być braków danych.

   Implementacja testu:

```
select count(*) as MissingValues from CrashFacts
where Crash_ID is null or CrashDateKey is null or CrashTimeKey
is null or BoroughKey is null or StreetKey is null or Latitude
is null or Longitude is null
or PeopleInjured is null or PeopleKilled is null or
PedestriansInjured is null or PedestriansKilled is null or
CyclistsInjured is null or CyclistsKilled is null or
MotoristInjured is null or MotoristsKilled is null
or VehiclesAmount is null
```

Wynik:



	MissingValues
1	0

Zgodny z oczekiwaniami ✅

3. **Połączenia z tabelami wymiarów**

   a) **Połączenia z wymiarem dat**

   Oczekiwany wynik:

   Każda wartość kolumny CrashDateKey ma odpowiednika w tabeli wymiarów. Po zastosowaniu left join tabeli faktów z DateDim nie będzie rekordów, w których pole np. FullDate będzie NULL.

   Implementacja testu:

```
select count(*) as MissingDateConnection
from CrashFacts f
left join DateDim d on f.CrashDateKey = d.DateKey
where d.FullDate is null;
```

   Wynik:

	MissingDateConnection
1	0

   Zgodny z oczekiwaniami ✅

   b) **Połączenia z wymiarem czasu**

   Oczekiwany wynik:

Każda wartość kolumny CrashTimeKey ma odpowiednika w tabeli wymiarów. Po zastosowaniu left join tabeli faktów z TimeDim nie będzie rekordów, w których pole np. TimeValue będzie NULL.

Implementacja testu:

```
select count(*) as MissingTimeConnection
from CrashFacts f
left join TimeDim d on f.CrashTimeKey = d.TimeKey
where d.TimeValue is null;
```

Wynik:

	MissingTimeConnection
1	0

Zgodny z oczekiwaniami ✅

c) Połączenia z wymiarem SpeedLimits

Oczekiwany wynik:

Każda wartość kolumny StreetKey ma odpowiednika w tabeli wymiarów. Po zastosowaniu left join tabeli faktów z SpeedLimitsDim nie będzie rekordów, w których pole np. StreetName będzie brakujące.

Implementacja testu:

```
select count(*) as MissingStreetConnection
from CrashFacts f
left join SpeedLimitsDim d on f.StreetKey = d.Street_ID
where d.StreetName = NULL or trim(d.StreetName)=''
```

Wynik:

	MissingStreetConnection
1	0

Zgodny z oczekiwaniami ✅

d) Poprawność mapowania StreetName

Oczekiwany wynik:

Przy losowym wyborze rekordu z tabeli Crashes Staging DB i odczytaniu nazwy ulicy, będzie ona taka sama jak ta, dołączona w wyniku left join do tabeli faktów.

Implementacja testu:

1. Wybór losowego wypadku i odczytanie StreetName z oryginalnych danych

```
select Crash_ID, StreetName from
[ELT_staging_DB].dbo.Crashes where Crash_ID=5
```

	Crash_ID	StreetName
1	5	SNEDIKER AVENUE

Ulica dla wypadku o ID 5 to Snediker avenue.

2. Left join tabeli faktów i wymiaru SpeedLimits oraz odczytanie dopasowanej ulicy dla wypadku o ID 5

```
select f.Crash_ID, f.StreetKey, d.StreetName from
CrashFacts f
left join SpeedLimitsDim d on f.StreetKey =
d.Street_ID
where f.Crash_ID=5
```

	Crash_ID	StreetKey	StreetName
1	5	5	SNEDIKER AVENUE

Poprawnie dołączona ulica.

e) Połączenia z wymiarem DistrictDetails

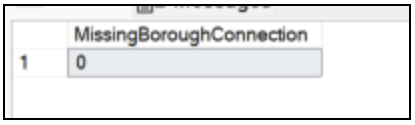
Oczekiwany wynik:

Każda wartość kolumny BoroughKey ma odpowiednika w tabeli wymiarów. Po zastosowaniu left join tabeli faktów z DistrictDetailsDim nie będzie rekordów, w których pole np. DistrictName będzie brakujące.

Implementacja testu:

```
select count(*) as MissingBoroughConnection
from CrashFacts f
left join DistrictDetailsDim d on f.BoroughKey =
d.District_ID
where d.DistrictName is null or trim(d.DistrictName)=''
```

Wynik:



	MissingBoroughConnection
1	0

Zgodny z oczekiwaniami ✅

f) Poprawność mapowania Borough

Oczekiwany wynik:

Przy losowym wyborze rekordu z tabeli Crashes Staging DB i odczytaniu nazwy dzielnicy, będzie ona taka sama jak ta, dołączona w wyniku left join do tabeli faktów.

Implementacja testu:

1. Wybór losowego wypadku i odczytanie Borough z oryginalnych danych

```
select Crash_ID, Borough from
[ELT_staging_DB].dbo.Crashes where Crash_ID=5
```

	Crash_ID	Borough
1	5	BROOKLYN

Dzielnica wypadku o ID 5 to Brooklyn.

2. Left join tabeli faktów i wymiaru DistrictDetails oraz odczytanie dopasowanej dzielnicy dla wypadku o ID 5

```
select f.Crash_ID, f.BoroughKey, d.DistrictName from
CrashFacts f
left join DistrictDetailsDim d on f.BoroughKey =
d.District_ID
where f.Crash_ID=5
```

	Crash_ID	BoroughKey	DistrictName
1	5	2	BROOKLYN

Nazwa dzielnicy została prawidłowo zmapowana.

Zgodny z oczekiwaniami ✅

4. Poprawność połączeń typu Bridge

a) Factors Bridge

Oczekiwany wynik:

Stworzona przez kwerendę SQL tabela Bridge powstała z oryginalnej tabeli Crashes Staging DB nie powinna różnić się od tej załadowanej w hurtowni danych.

Implementacja testu (GPT-4):

1. Widok ExploadedFactors to tabela stworzona przez Unpivot Crashes zawierająca kolumny Crash_ID i Factor. Np. Jeśli wypadek z ID 5 miał dwa nie brakujące czynniki to w tej tabeli powstaną dwa rekordy z Crash_ID 5.

```
WITH ExplodedFactors AS (
    SELECT Crash_ID,
           CASE
               WHEN Factor = 'Unspecified' THEN
'Unknown'
               ELSE Factor
           END AS Factor
    FROM (
        SELECT Crash_ID, Factor1, Factor2, Factor3,
Factor4, Factor5
        FROM [ELT_Staging_DB].dbo.Crashes
    ) AS p
    UNPIVOT (
        Factor FOR FactorCol IN (Factor1, Factor2,
Factor3, Factor4, Factor5)
    ) AS unpvt
    WHERE Factor IS NOT NULL
)
```

2. Widok Mapped Factors zawiera taką samą strukturę jak powyżej, z tą różnicą, że zamiast FactorName w drugiej kolumnie znajduje się Factor_ID przypisane za pomocą Join z tabelą wymiarów

```
MappedFactors AS (
    SELECT ef.Crash_ID, df.Factor_ID
    FROM ExplodedFactors ef
    JOIN ContributingFactorsDim df ON ef.Factor =
df.FactorName
)
```

3. Widok Expected Bridge zawiera strukturę Crash_ID, Factor_ID oraz ExpectedAmt obliczoną za pomocą agregacji Group by powyższego widoku. Actual Bridge to po prostu dane pobrane z wypełnionej w procesie ETL tabeli.

```
ExpectedBridge AS (
    SELECT Crash_ID, Factor_ID, COUNT(*) AS
ExpectedAmt
    FROM MappedFactors
    GROUP BY Crash_ID, Factor_ID
```

```
),
ActualBridge AS (
    SELECT Crash_ID, Factor_ID, FactorAmt AS
ActualAmt
    FROM CrashFactorBridge
)
```

4. Na koniec następuje porównanie Expected i Actual Bridge i na konsole wypisane są różniące się rekordy

```
SELECT
    COALESCE(e.Crash_ID, a.Crash_ID) AS Crash_ID,
    COALESCE(e.Factor_ID, a.Factor_ID) AS Factor_ID,
    e.ExpectedAmt,
    a.ActualAmt
FROM ExpectedBridge e
FULL OUTER JOIN ActualBridge a
    ON e.Crash_ID = a.Crash_ID AND e.Factor_ID =
a.Factor_ID
WHERE ISNULL(e.ExpectedAmt, 0) <> ISNULL(a.ActualAmt,
0)
```

Wynik:
Pusta tabela, czyli brak różnic.

Zgodny z oczekiwaniami ✅

b) Vehicle Types Bridge

Oczekiwany wynik:
Stworzona przez kwerendę SQL tabela Bridge powstała z oryginalnej tabeli Crashes Staging DB nie powinna różnić się od tej załadowanej w hurtowni danych.

Implementacja testu:

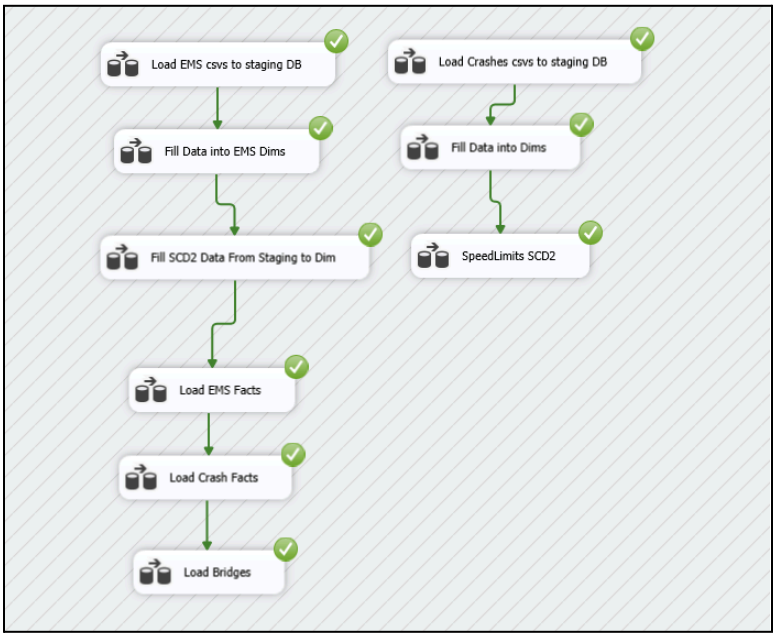
Test zaimplementowany w identyczny sposób jak powyżej, jedynie z uwzględnieniem ręcznych poprawek i mapowań pewnych wartości w tabeli Crashes z pierwszego etapu.

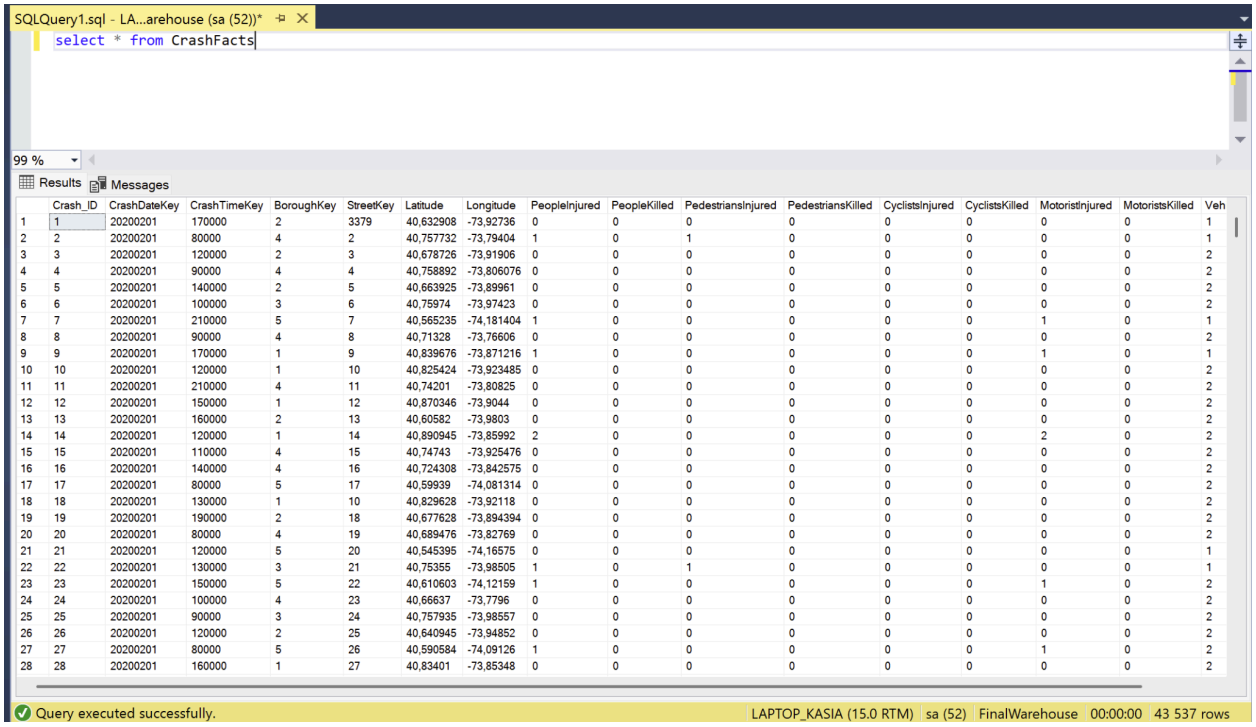
Wynik:
Również pusta tabela, czyli brak różnic.

Zgodny z oczekiwaniami ✅

Testy całościowe ETL i SQL Server

Na koniec zaprezentowano także zrzuty ekranu potwierdzające całościowe działanie procesu ETL w SSIS oraz przykład załadowania danych do hurtowni w SQL server, która została wykorzystana w stworzeniu raportu.



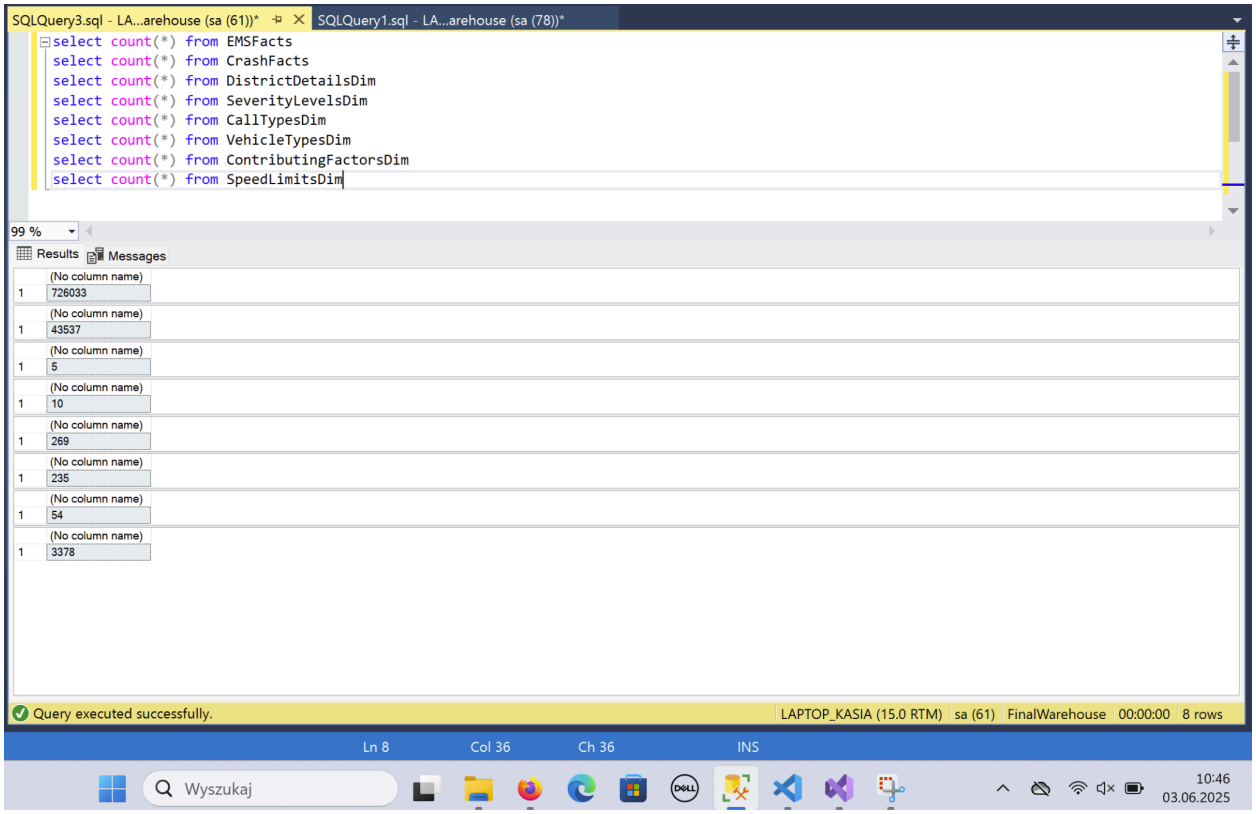


	Crash_ID	CrashDateKey	CrashTimeKey	BoroughKey	StreetKey	Latitude	Longitude	PeopleInjured	PeopleKilled	PedestriansInjured	PedestriansKilled	CyclistsInjured	CyclistsKilled	MotoristInjured	MotoristsKilled	Veh
1	1	20200201	170000	2	3379	40,632908	-73,92736	0	0	0	0	0	0	0	0	1
2	2	20200201	80000	4	2	40,757732	-73,79404	1	0	1	0	0	0	0	0	1
3	3	20200201	120000	2	3	40,678726	-73,91906	0	0	0	0	0	0	0	0	2
4	4	20200201	90000	4	4	40,758892	-73,806076	0	0	0	0	0	0	0	0	2
5	5	20200201	140000	2	5	40,663925	-73,89961	0	0	0	0	0	0	0	0	2
6	6	20200201	100000	3	6	40,75974	-73,97423	0	0	0	0	0	0	0	0	2
7	7	20200201	210000	5	7	40,565235	-74,181404	1	0	0	0	0	0	1	0	1
8	8	20200201	90000	4	8	40,71328	-73,76606	0	0	0	0	0	0	0	0	2
9	9	20200201	170000	1	9	40,839676	-73,871216	1	0	0	0	0	0	1	0	1
10	10	20200201	120000	1	10	40,825424	-73,923485	0	0	0	0	0	0	0	0	2
11	11	20200201	210000	4	11	40,74201	-73,80825	0	0	0	0	0	0	0	0	2
12	12	20200201	150000	1	12	40,870346	-73,9044	0	0	0	0	0	0	0	0	2
13	13	20200201	160000	2	13	40,60582	-73,9803	0	0	0	0	0	0	0	0	2
14	14	20200201	120000	1	14	40,890945	-73,85992	2	0	0	0	0	0	2	0	2
15	15	20200201	110000	4	15	40,74743	-73,925476	0	0	0	0	0	0	0	0	2
16	16	20200201	140000	4	16	40,724308	-73,842575	0	0	0	0	0	0	0	0	2
17	17	20200201	80000	5	17	40,59939	-74,081314	0	0	0	0	0	0	0	0	2
18	18	20200201	130000	1	10	40,829628	-73,92118	0	0	0	0	0	0	0	0	2
19	19	20200201	190000	2	18	40,677628	-73,894394	0	0	0	0	0	0	0	0	2
20	20	20200201	80000	4	19	40,689476	-73,82769	0	0	0	0	0	0	0	0	2
21	21	20200201	120000	5	20	40,545395	-74,16575	0	0	0	0	0	0	0	0	1
22	22	20200201	130000	3	21	40,75355	-73,98505	1	0	1	0	0	0	0	0	1
23	23	20200201	150000	5	22	40,610603	-74,12159	1	0	0	0	0	0	1	0	2
24	24	20200201	100000	4	23	40,66637	-73,7796	0	0	0	0	0	0	0	0	2
25	25	20200201	90000	3	24	40,757935	-73,98557	0	0	0	0	0	0	0	0	2
26	26	20200201	120000	2	25	40,640945	-73,94852	0	0	0	0	0	0	0	0	2
27	27	20200201	80000	5	26	40,590584	-74,09126	1	0	0	0	0	0	1	0	2
28	28	20200201	160000	1	27	40,83401	-73,85348	0	0	0	0	0	0	0	0	2

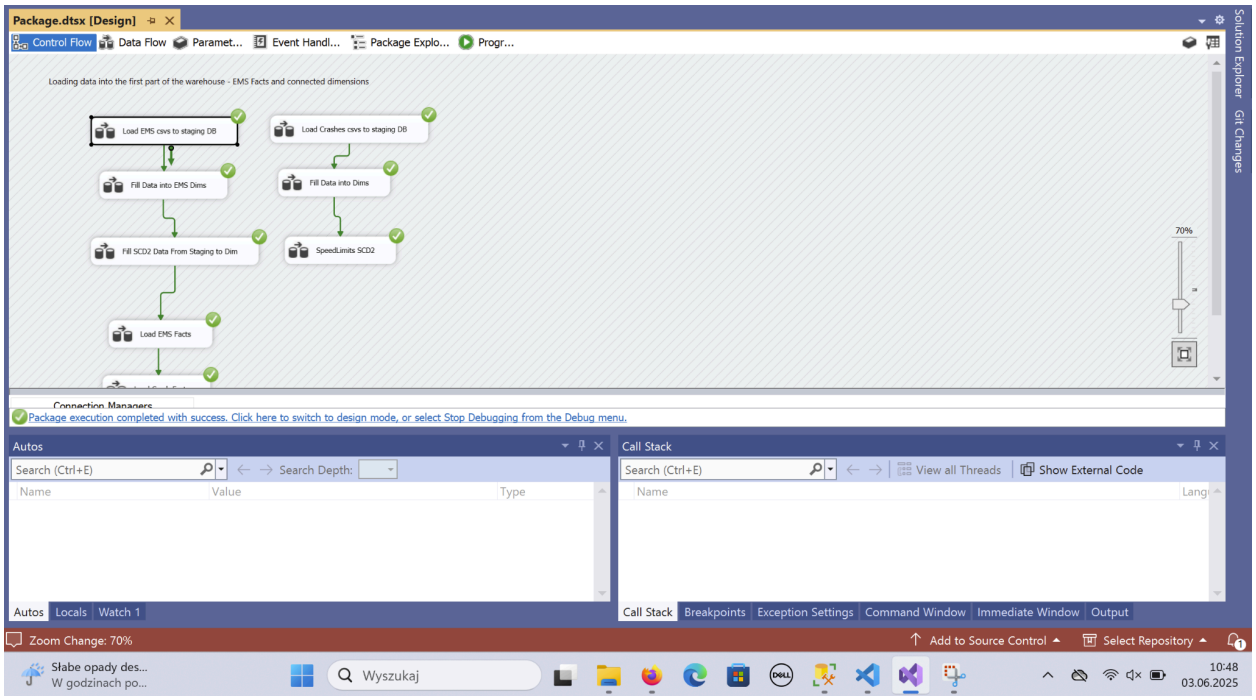
Testy kolejnych iteracji

Poniżej przedstawiono zrzuty ekranu z widoczną godziną ich wykonania przedstawiające bazę danych już zainicjowaną, następnie wywołanie kolejnych iteracji:

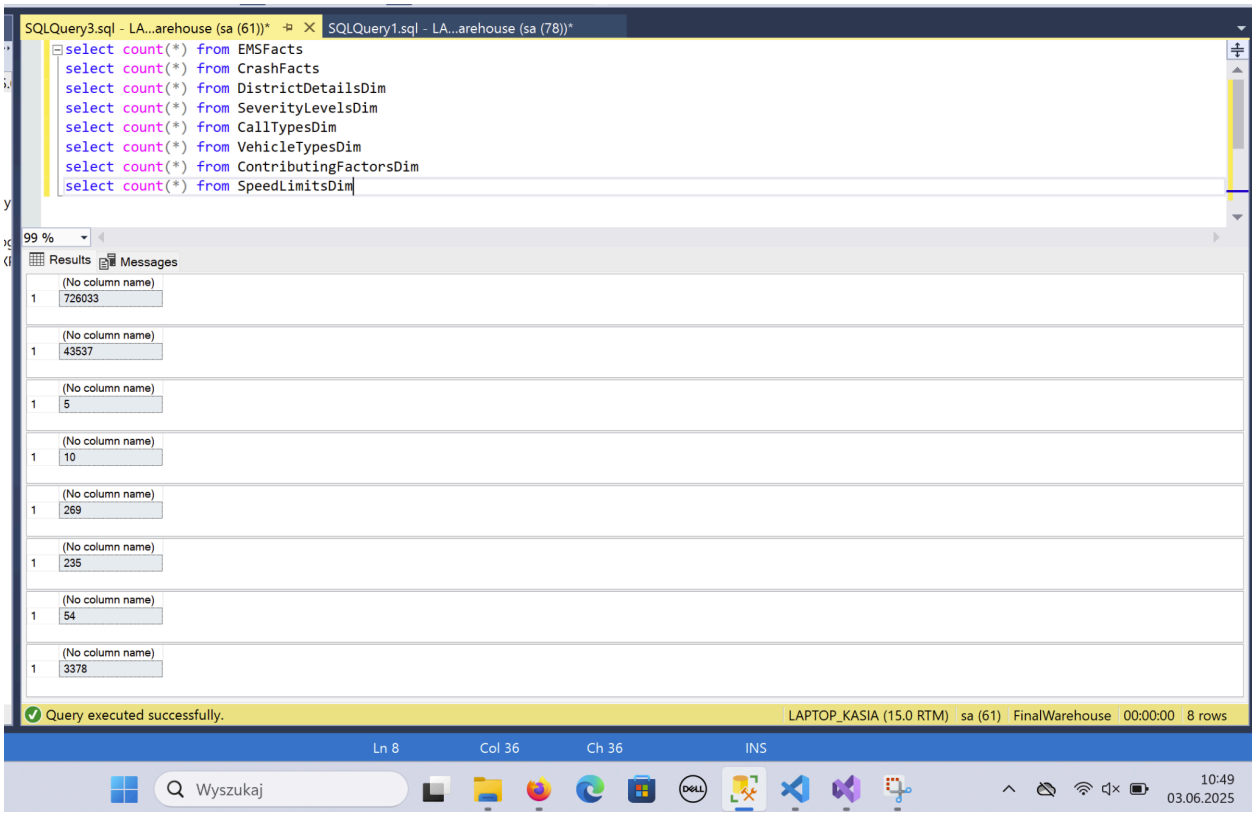
a) Bez zmian rekordów
b) Ze zmianą rekordu w danych źródłowych



Zainicjalizowana wcześniej baza danych (10:46)

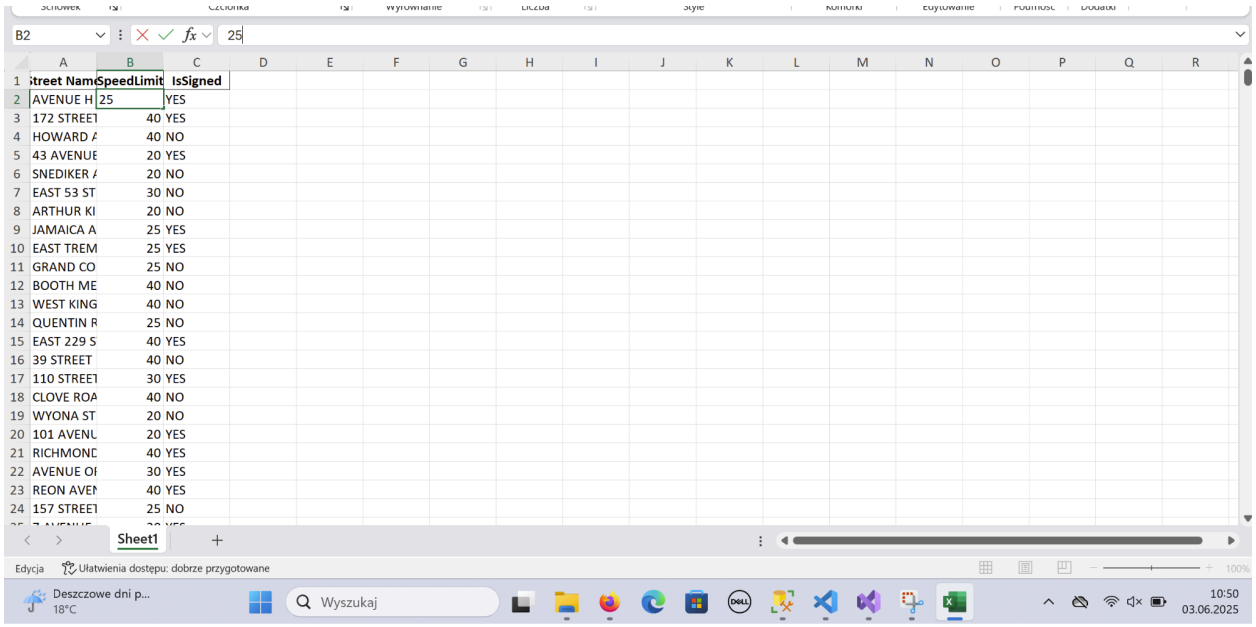


Ponownie wywołany proces ETL bez zmian w danych (10:48)

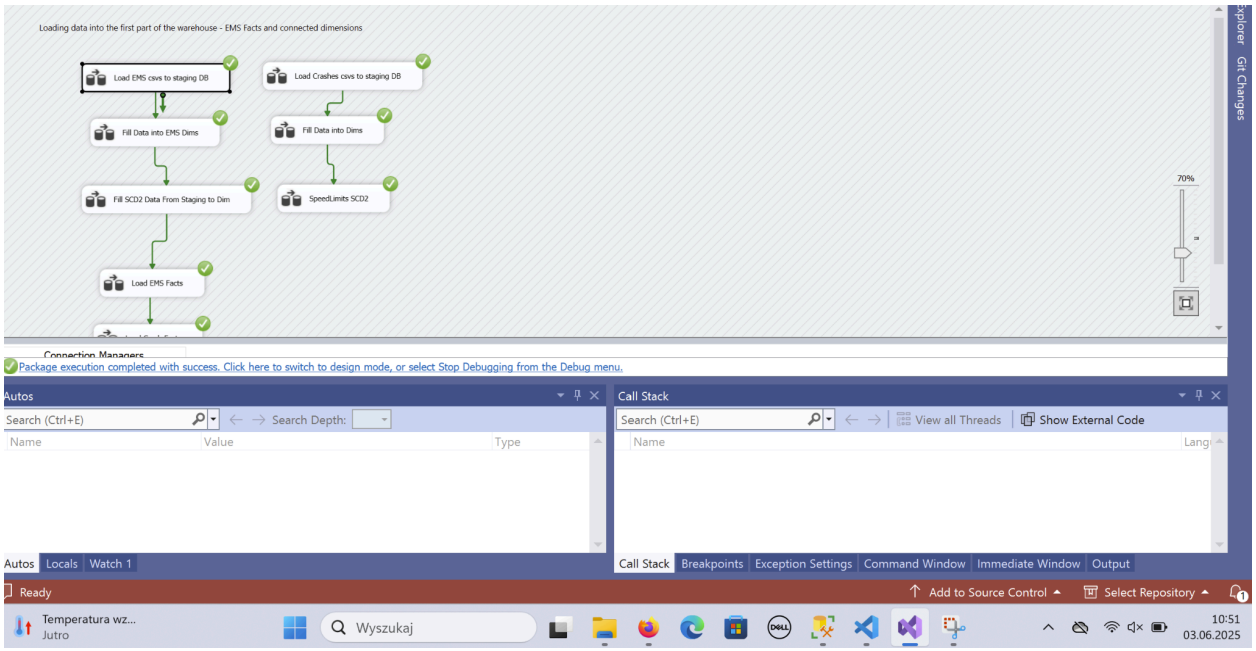


Wybór danych po kolejnej iteracji (10:49)

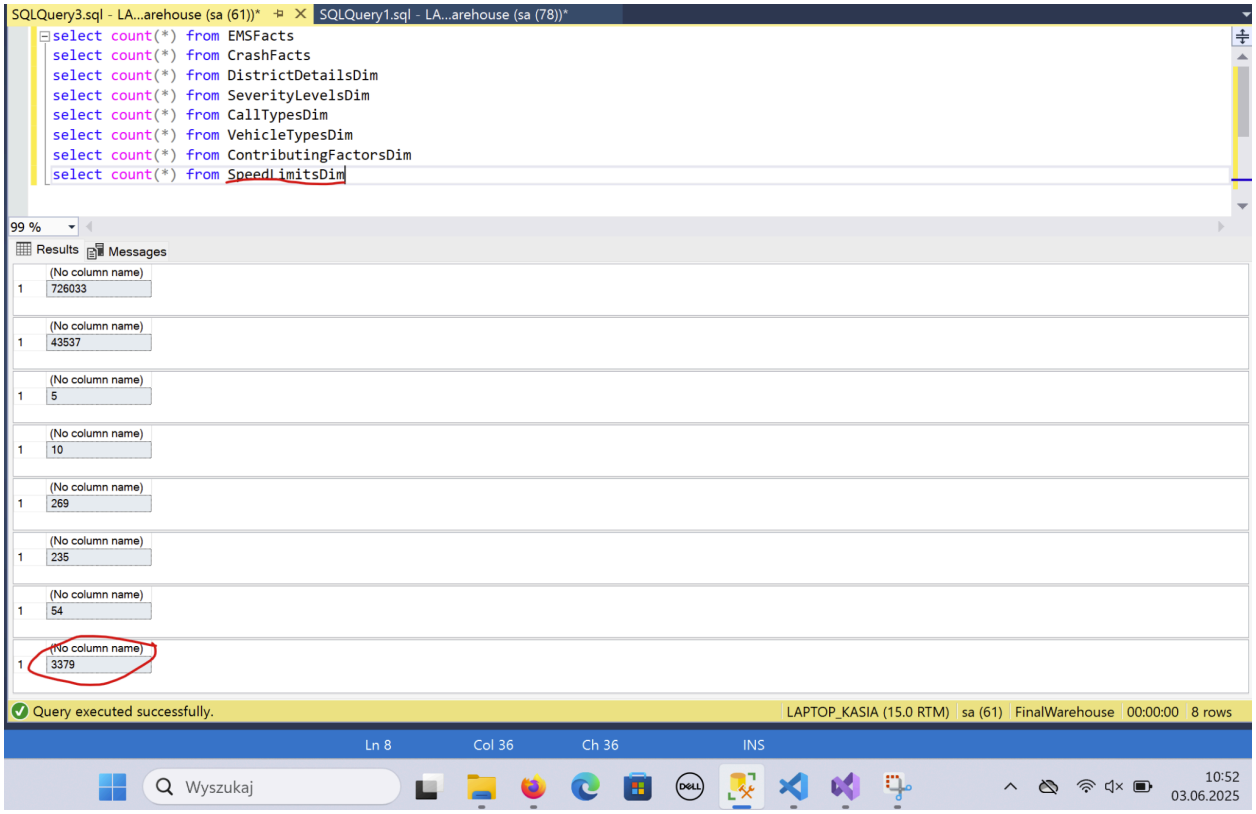
Jak widać ilość wierszy w hurtowni nie uległa zmianie, gdyż dane wejściowe nie zostały zmienione, co jest zgodne z oczekiwaniami.
W kolejnym scenariuszu zmieniono wartość kolumny SpeedLimit dla pewnej ulicy w pliku źródłowym. Spodziewany wynik po wybraniu danych po kolejnej iteracji to jeden więcej rekord w tabeli SpeedLimits Dim oraz niezmieniona ilość wierszy w pozostałych tabelach.



Zmiana danych (10:50)



Wywołanie kolejnej iteracji po zmianie danych źródłowych (10:51)



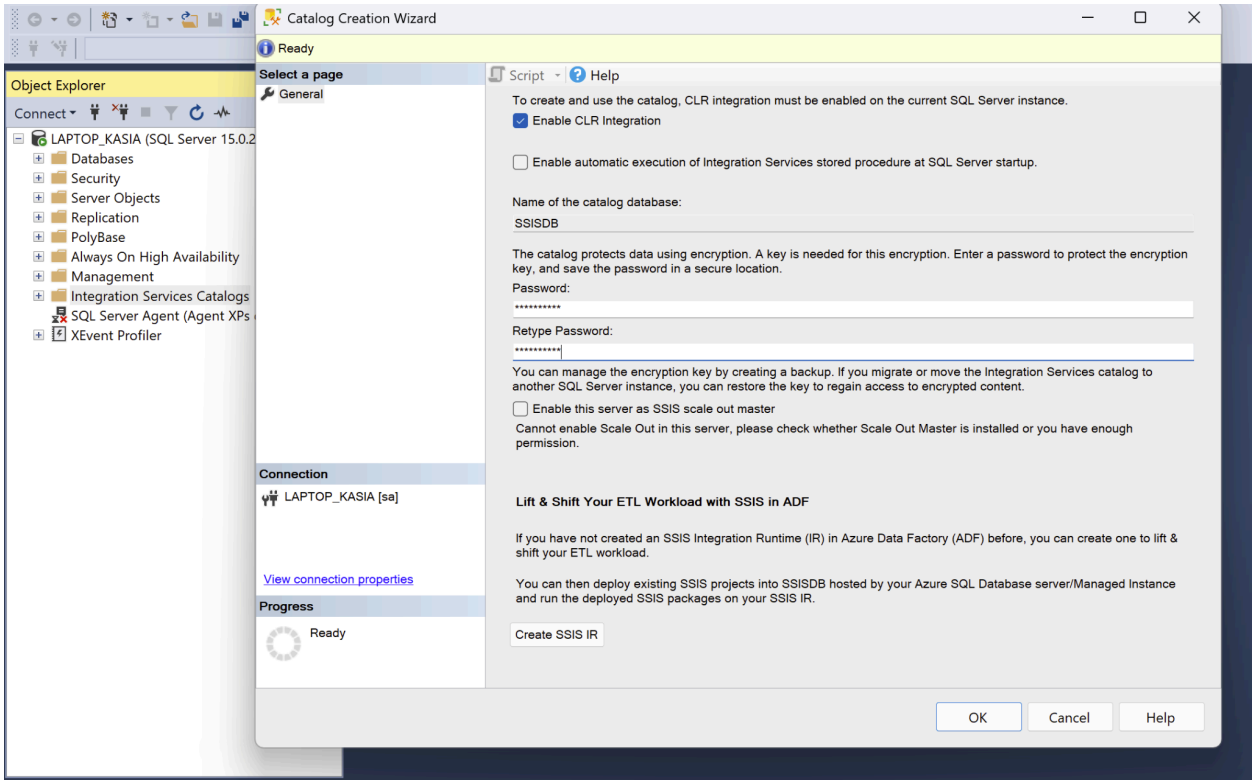
Wybór danych z hurtowni (10:52)

Można zauważyć, że zgodnie z oczekiwaniami został dodany nowy rekord do tabeli SpeedLimits Dim w ramach mechanizmu SCD2, a inne tabele pozostały z tą samą ilością wierszy.

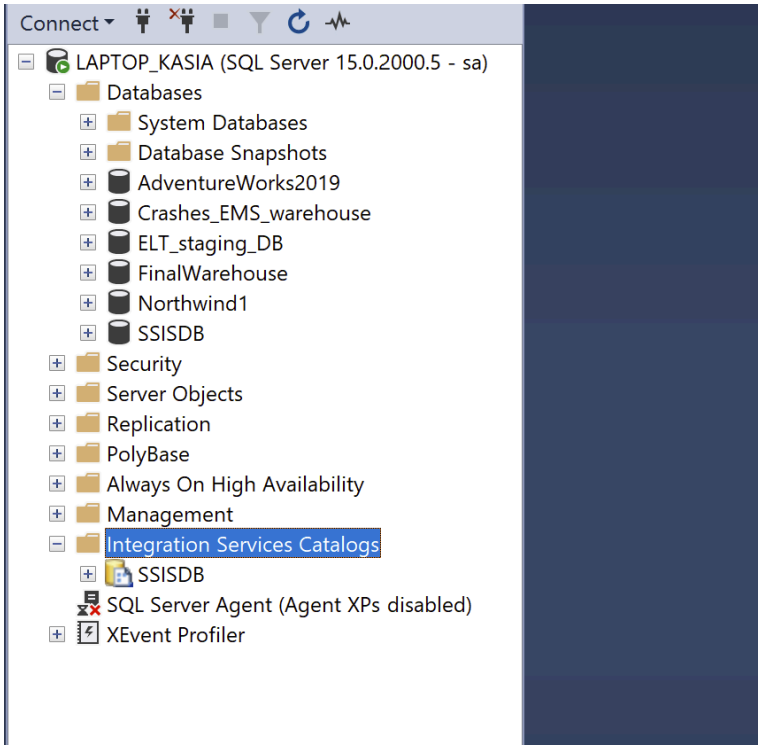
Deployment

W tej części testów zamieszczone zostały zrzuty ekranu potwierdzające wykonanie procesu deployment pakietu z Visual Studio do SQL Server.

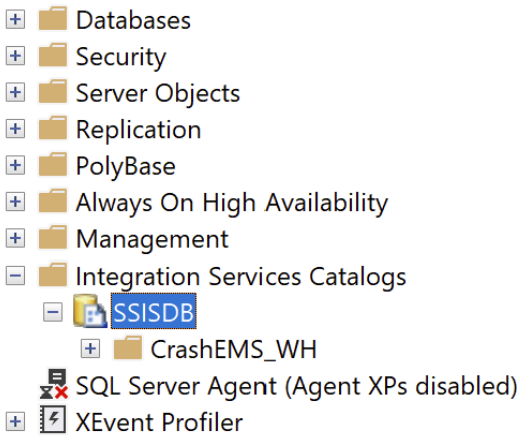
Krok 1 - Utworzenie katalogu Integration Services



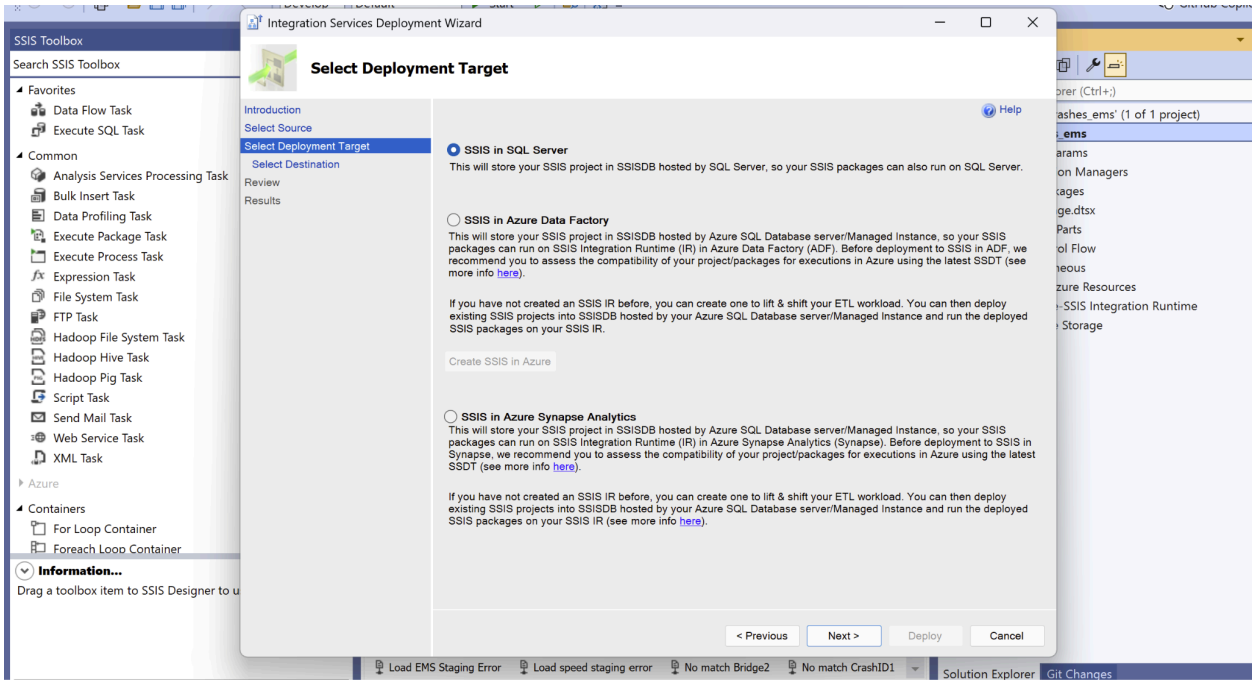
Krok 2 - Weryfikacja utworzenia nowej bazy

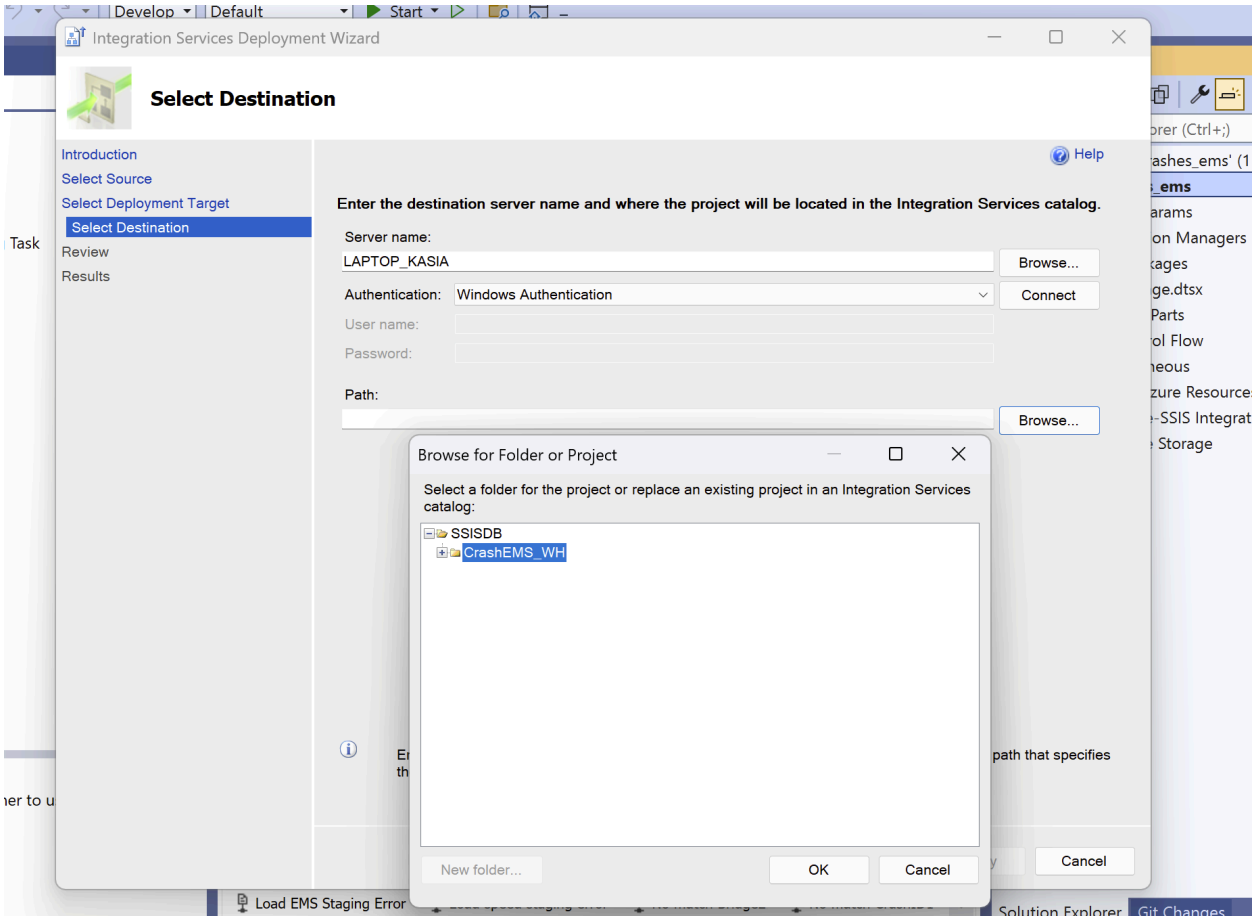


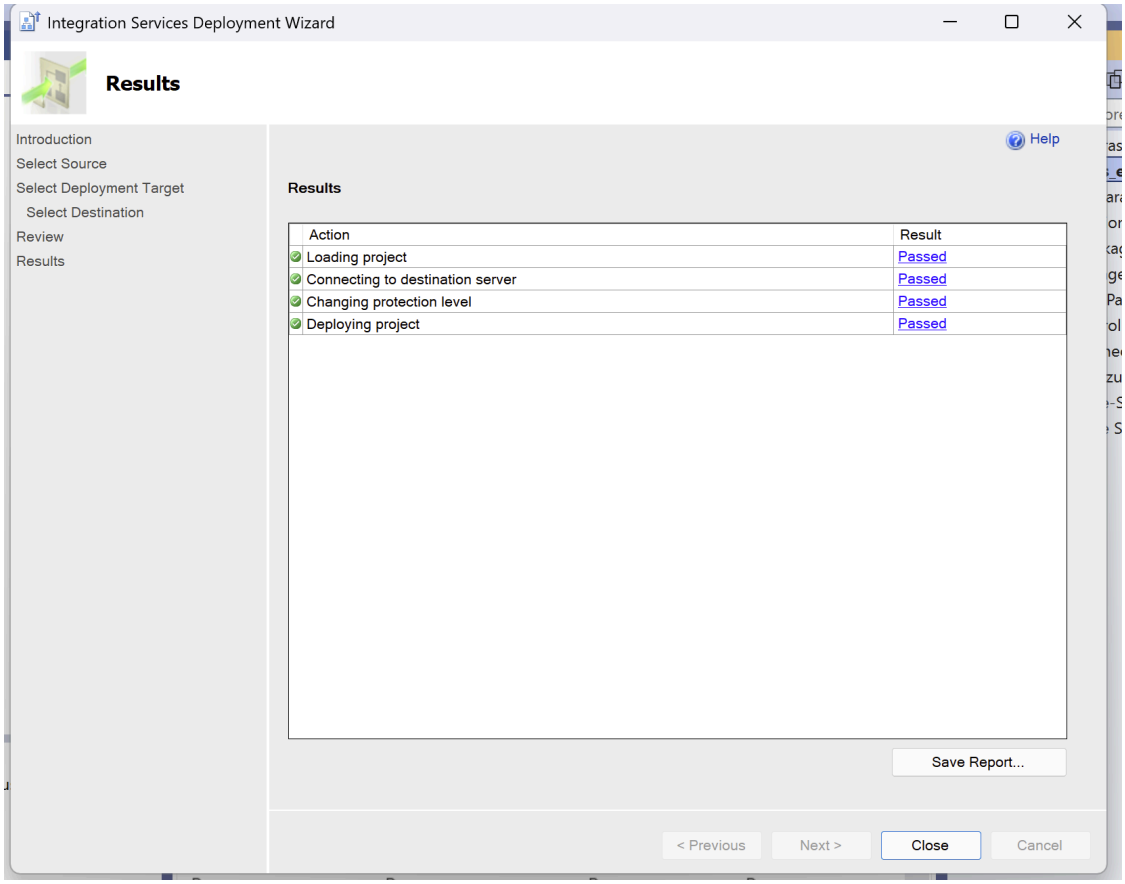
Krok 3 - Utworzenie folderu na nasz projekt



Krok 4 - Deploment pakitu w Visual Studio




Integration Services Deployment Wizard
Select Destination
Introduction
Select Source
Select Deployment Target
Select Destination
Review
Results
Help
Enter the destination server name and where the project will be located in the Integration Services catalog.
Server name:
LAPTOP_KASIA
Browse...
Authentication:
Windows Authentication
Connect
User name:
Password:
Path:
Browse...
Browse for Folder or Project
Select a folder for the project or replace an existing project in an Integration Services catalog:
SSISDB
CrashEMS_WH
New folder...
OK
Cancel
Cancel
Load EMS Staging Error

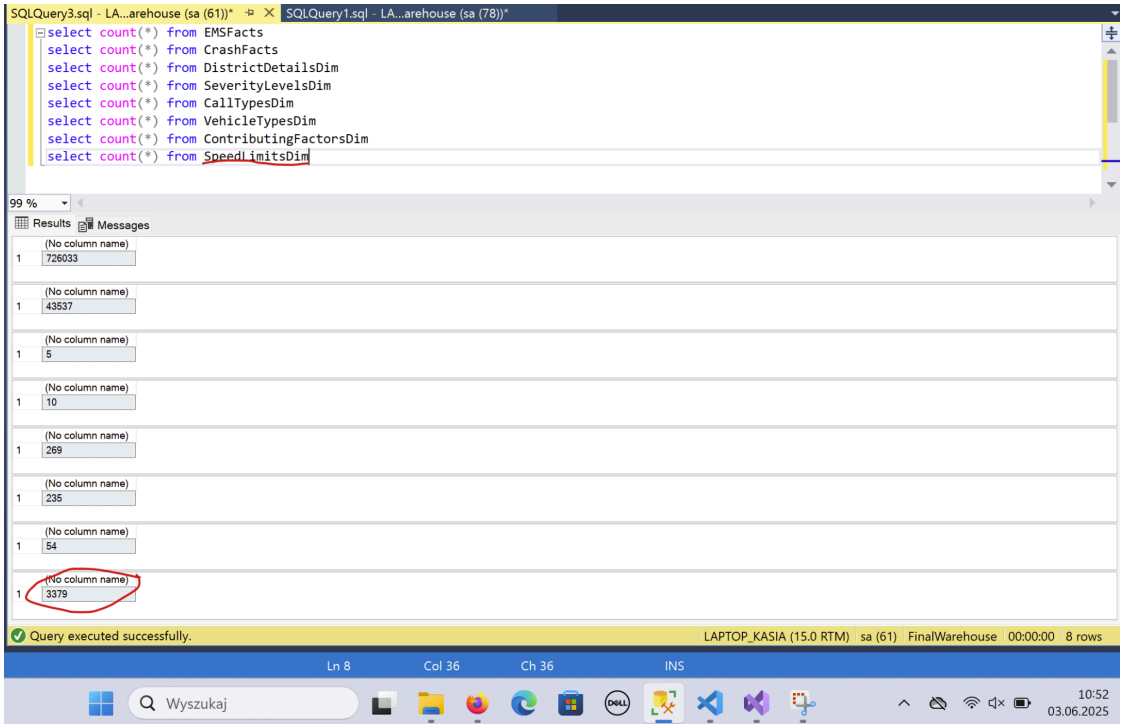


Teraz nasz pakiet jest już gotowy do ewentualnego wdrożenia i automatyzacji zadań w przyszłości.

Test End to End

Z powodu problemów technicznych oraz chronologicznego podziału prac w grupie, test całościowego działania systemu został zaprezentowany w następujący sposób.

1) Przedstawienie daty załadowania finalnej bazy danych w SQL na serwerze LAPTOP_KASIA oraz wywołanie zapytań Select w celu zweryfikowania ilości rekordów

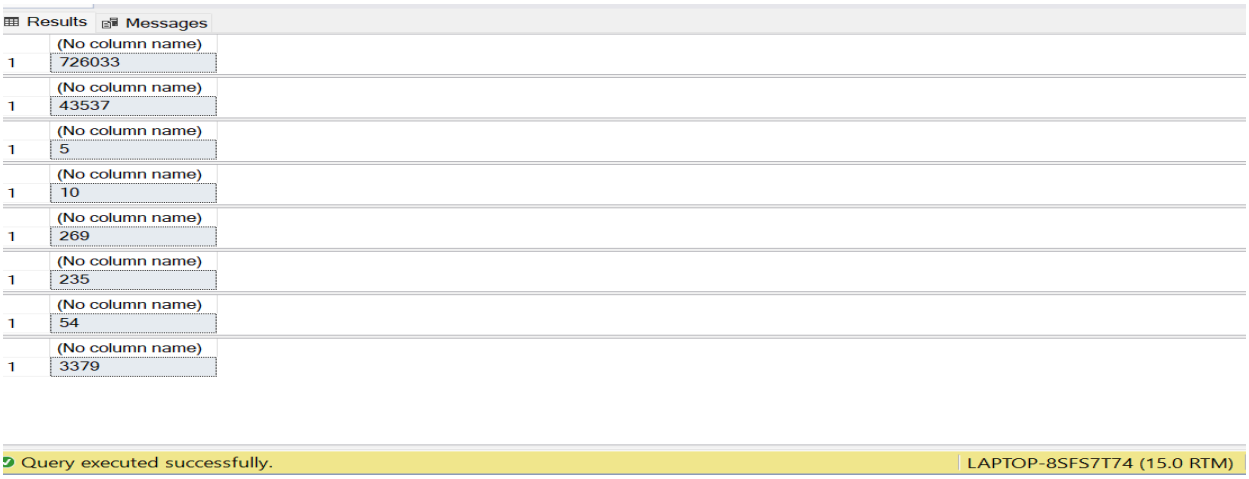


(Zdjęcie z testu iteracji) Data: 03.06.2025 godzina 10:52

2) Przedstawienie daty i godziny zapisu finalnej hurtowni do pliku .bak

3) Przedstawienie daty commita na Githubie zawierającego wygenerowany plik .bak

4) Pobranie danych i wykonanie tych samych zapytań select na drugim urządzeniu w celu potwierdzenia spójności



Test spójności danych hurtowni i raportu

Dla każdej wizualizacji przeprowadzono testy porównawcze raportu i zawartości hurtowni danych. Wszystkie skrypty do testów są oddzielnie załączone.

Strona raportowa 1 - Overview

Testy dla KPI Cards

```
SELECT COUNT(DISTINCT Crash_ID) AS TotalCrashCount
FROM CrashFacts;

SELECT SUM(PeopleInjured) AS TotalPeopleInjured
FROM CrashFacts;

SELECT COUNT(DISTINCT EMS_ID) AS TotalEMSDispatchInterventions
FROM EMSFacts;

SELECT
    COUNT(CASE WHEN OnSceneArrivalDuration < 8 THEN 1 END) * 100.0 /
    COUNT(*) AS FastResponsePercentage
FROM EMSFactsT
```

Wyniki KPI Cards w raporcie, dla testu zmieniony format, aby potwierdzić zgodność z wynikiem zapytań SQL

Results Messages

	TotalCrashCount
1	43537

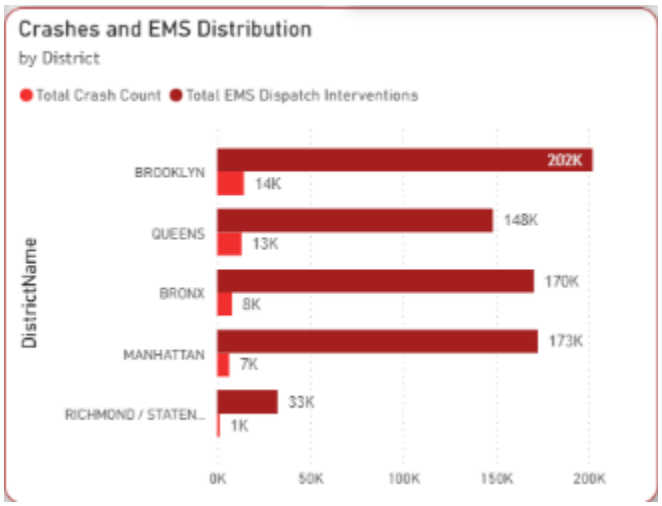
	TotalPeopleInjured
1	19656

	TotalEMSDispatchInterventions
1	726033

	FastResponsePercentage
1	70.844575935253

Testy dla wykresu skumulowanego:

```
SELECT
    d.DistrictName,
    COUNT(DISTINCT c.Crash_ID) AS TotalCrashCount,
    COUNT(DISTINCT e.EMS_ID) AS TotalEMSDispatchCount
FROM
    DistrictDetailsDim d
LEFT JOIN
    CrashFacts c ON d.District_ID = c.BoroughKey
LEFT JOIN
    EMSFacts e ON d.District_ID = e.BoroughKey
GROUP BY
    d.DistrictName
ORDER BY
    d.DistrictName;
```



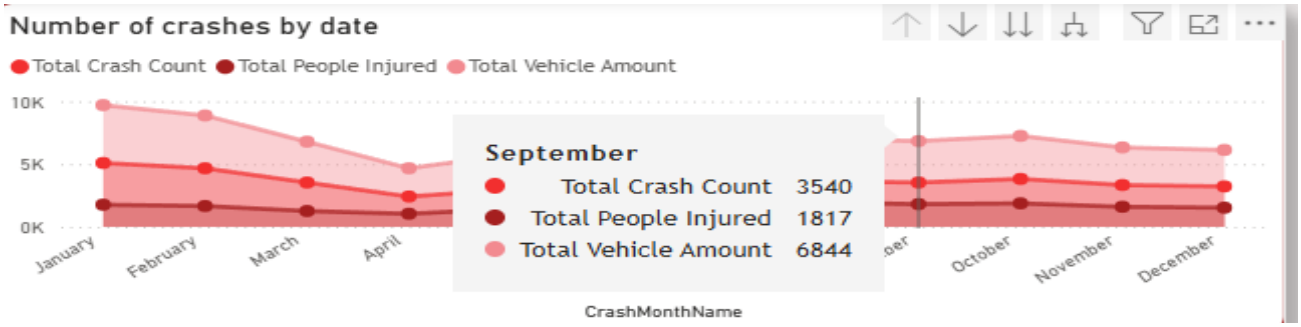
Results | Messages

	DistrictName	TotalCrashCount	TotalEMSDispatchCount
1	BRONX	7947	170415
2	BROOKLYN	14412	202018
3	MANHATTAN	6569	172552
4	QUEENS	13156	148411
5	RICHMOND / STATEN ISLAND	1453	32637

Strona raportowa 2 - Crashes Analysis

Wykres liniowy - liczba wypadków, poszkodowanych i ofiar śmiertelnych w ujęciu miesięcznym

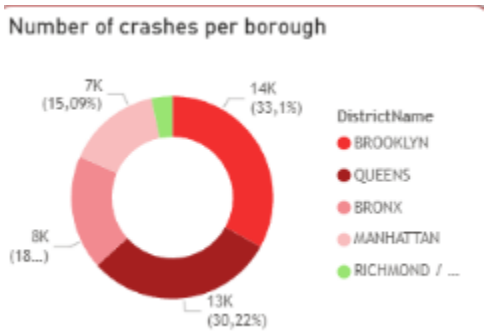
```
SELECT
    d.IncidentMonth,
    d.IncidentMonthName,
    COUNT(c.Crash_ID) AS CrashCount,
    SUM(c.VehiclesAmount) AS TotalVehicleAmount,
       SUM(c.PeopleInjured) AS TotalInjured
FROM
    CrashFacts c
JOIN
    vwDimDate_Incident d ON c.CrashDateKey = d.IncidentDateKey
GROUP BY
    d.IncidentMonth, d.IncidentMonthName
ORDER BY
    d.IncidentMonth;
```



	IncidentMonth	IncidentMonthName	CrashCount	TotalVehicleAmount	TotalInjured
1	1	January	5097	9706	1771
2	2	February	4673	8877	1667
3	3	March	3538	6794	1268
4	4	April	2423	4673	1049
5	5	May	2980	5774	1445
6	6	June	3543	6829	1819
7	7	July	3684	7028	1841
8	8	August	3664	7024	1961
9	9	September	3540	6844	1817
10	10	October	3825	7254	1880
11	11	November	3340	6335	1605
12	12	December	3230	6131	1533

Donut chart - procentowy udział dzielnic w liczbie wypadków

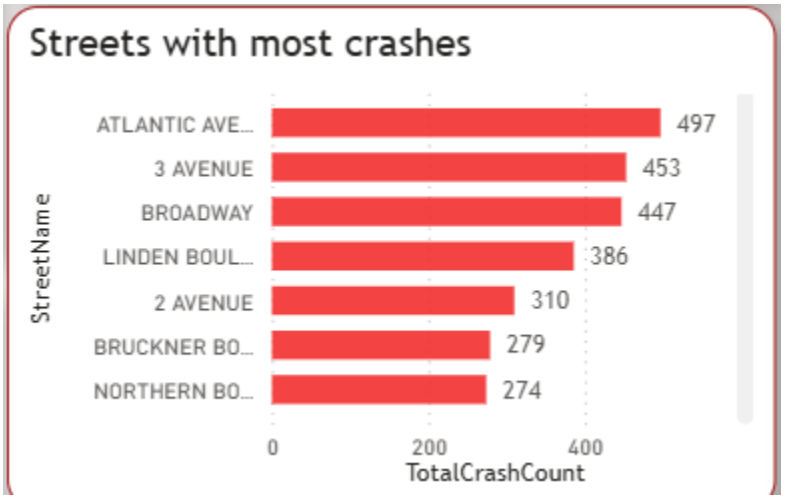
```
SELECT
    dd.DistrictName,
    COUNT(c.Crash_ID) AS CrashCount,
    ROUND(
        100.0 * COUNT(c.Crash_ID) / SUM(COUNT(c.Crash_ID)) OVER(),
        1
    ) AS DistrictShareInYearPct
FROM
    CrashFacts c
JOIN
    vwDimDate_Crash d ON c.CrashDateKey = d.CrashDateKey
JOIN
    DistrictDetailsDim dd ON c.BoroughKey = dd.District_ID
WHERE
    dd.IsValid = 1
GROUP BY
    dd.DistrictName
ORDER BY
    DistrictShareInYearPct DESC;
```



	DistrictName	CrashCount	DistrictShareInYearPct
1	BROOKLYN	14412	33.100000000000
2	QUEENS	13156	30.220000000000
3	BRONX	7947	18.250000000000
4	MANHATTAN	6569	15.090000000000
5	RICHMOND / STATEN ISLAND	1453	3.340000000000

Wykres słupkowy – ulice o największej liczbie wypadków

```
SELECT
    s.StreetName,
    COUNT(c.Crash_ID) AS TotalCrashCount
FROM
    CrashFacts c
    JOIN SpeedLimitsDim s ON c.StreetKey = s.Street_ID
WHERE
    s.IsValid = 1
GROUP BY
    s.StreetName
ORDER BY
    TotalCrashCount DESC;
```

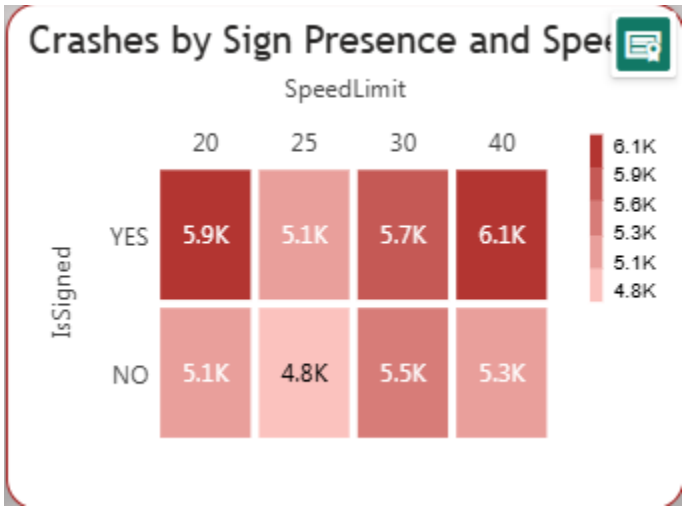


Results | Messages

	StreetName	TotalCrashCount
1	ATLANTIC AVENUE	497
2	3 AVENUE	453
3	BROADWAY	447
4	LINDEN BOULEVARD	386
5	2 AVENUE	310
6	BRUCKNER BOULEVARD	279
7	NORTHERN BOULEVARD	274
8	QUEENS BOULEVARD	244
9	FLATBUSH AVENUE	234
10	JEROME AVENUE	220
11	JAMAICA AVENUE	216

Mapa korelacji – zależność liczby wypadków od obecności oznakowania drogowego i prędkości, na raporcie liczby w zaokrąglone

```
SELECT
    s.SpeedLimit,
    s.IsSigned,
    COUNT(c.Crash_ID) AS CrashCount
FROM
    CrashFacts c
    JOIN SpeedLimitsDim s ON c.StreetKey = s.Street_ID
WHERE
    s.IsValid = 1
GROUP BY
    s.SpeedLimit, s.IsSigned
ORDER BY
    s.SpeedLimit, s.IsSigned;
```



Results | Messages

	SpeedLimit	IsSigned	CrashCount
1	20	NO	5095
2	20	YES	5893
3	25	NO	4800
4	25	YES	5116
5	30	NO	5538
6	30	YES	5680
7	40	NO	5281
8	40	YES	6134

Tabela podsumowująca – statystyki wypadków w podziale na dzielnice

```
WITH DistrictStats AS (
    SELECT
        d.DistrictName,
        COUNT(c.Crash_ID) AS TotalCrashCount,
        SUM(c.PeopleInjured) AS TotalPeopleInjured,
        SUM(c.PeopleKilled) AS TotalPeopleKilled
    FROM
        CrashFacts c
        JOIN DistrictDetailsDim d ON c.BoroughKey = d.District_ID
    WHERE
        d.IsValid = 1
    GROUP BY
        d.DistrictName
),
TotalStats AS (
    SELECT
        'Total' AS DistrictName,
        SUM(TotalCrashCount) AS TotalCrashCount,
        SUM(TotalPeopleInjured) AS TotalPeopleInjured,
        SUM(TotalPeopleKilled) AS TotalPeopleKilled
    FROM
        DistrictStats
)

SELECT * FROM DistrictStats
UNION ALL
SELECT * FROM TotalStats;
```

DistrictName	TotalCrashCount	Total People Injured	TotalPeopleKilled
QUEENS	13156	5683	42
BROOKLYN	14412	7008	41
BRONX	7947	3653	20
MANHATTAN	6569	2575	19
RICHMOND / STATEN ISLAND	1453	737	8
Total	**43537**	**19656**	**130**

Results | Messages

	DistrictName	TotalCrashCount	TotalPeopleInjured	TotalPeopleKilled
1	BRONX	7947	3653	20
2	BROOKLYN	14412	7008	41
3	MANHATTAN	6569	2575	19
4	QUEENS	13156	5683	42
5	RICHMOND / STATEN ISLAND	1453	737	8
6	Total	43537	19656	130

Strona raportowa 3 - Crashes Contributing Factors

Macierz zależności – dzień tygodnia vs godzina zdarzenia

```
SELECT
    DATEPART(HOUR, t.CrashTimeValue) AS CrashHour,
    SUM(CASE WHEN d.CrashWeekdayName = 'Sunday' THEN 1 ELSE 0 END) AS Sunday,
    SUM(CASE WHEN d.CrashWeekdayName = 'Monday' THEN 1 ELSE 0 END) AS Monday,
    SUM(CASE WHEN d.CrashWeekdayName = 'Tuesday' THEN 1 ELSE 0 END) AS Tuesday,
    SUM(CASE WHEN d.CrashWeekdayName = 'Wednesday' THEN 1 ELSE 0 END) AS Wednesday,
    SUM(CASE WHEN d.CrashWeekdayName = 'Thursday' THEN 1 ELSE 0 END) AS Thursday,
    SUM(CASE WHEN d.CrashWeekdayName = 'Friday' THEN 1 ELSE 0 END) AS Friday,
    SUM(CASE WHEN d.CrashWeekdayName = 'Saturday' THEN 1 ELSE 0 END) AS Saturday,
    COUNT(*) AS Total
FROM
    CrashFacts c
JOIN
    vwDimDate_Crash d ON c.CrashDateKey = d.CrashDateKey
JOIN
    vwTimeDim_Crash t ON c.CrashTimeKey = t.CrashTimeKey
GROUP BY
    DATEPART(HOUR, t.CrashTimeValue)
ORDER BY
    DATEPART(HOUR, t.CrashTimeValue);
```

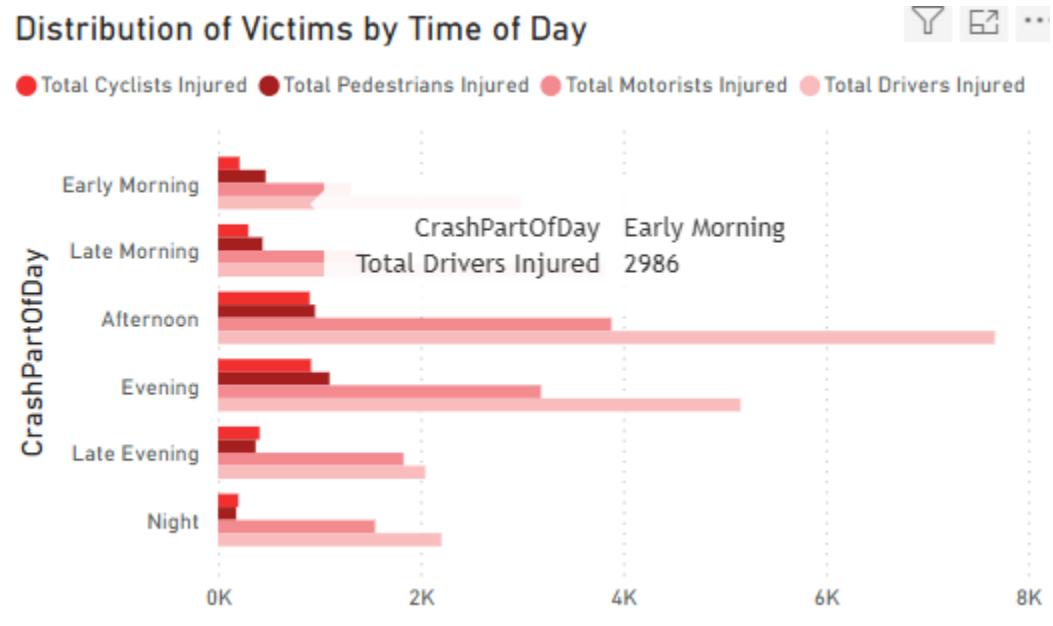
CrashHour	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Total
0	272	219	219	210	243	251	262	**1676**
1	145	112	93	99	109	110	162	**830**
2	135	65	53	81	77	86	129	**626**
3	104	61	52	56	67	66	107	**513**
4	96	63	75	56	63	49	87	**489**
5	96	76	78	80	81	78	93	**582**
6	103	160	161	141	154	161	106	**986**
7	133	185	211	208	250	207	130	**1324**
8	206	312	346	301	354	338	228	**2085**
9	204	282	305	282	296	337	206	**1912**
10	243	311	274	286	294	298	238	**1944**
11	261	313	322	307	341	321	268	**2133**
12	266	359	295	324	347	379	321	**2291**
13	326	379	354	351	374	361	345	**2490**
14	334	388	409	460	418	454	390	**2853**
15	318	434	399	351	381	434	343	**2660**
16	390	421	475	462	472	499	401	**3120**
17	364	441	490	485	470	462	402	**3114**
18	352	373	436	464	407	436	380	**2848**
19	278	326	324	357	378	368	319	**2350**
20	283	267	285	277	335	319	287	**2053**
21	242	211	210	279	251	262	259	**1714**
22	239	170	232	246	222	262	249	**1620**
23	181	147	186	173	207	210	220	**1324**
Total	**5571**	**6075**	**6284**	**6336**	**6591**	**6748**	**5932**	**43537**

Results Messages

	CrashHour	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Total
1	0	272	219	219	210	243	251	262	1676
2	1	145	112	93	99	109	110	162	830
3	2	135	65	53	81	77	86	129	626
4	3	104	61	52	56	67	66	107	513
5	4	96	63	75	56	63	49	87	489
6	5	96	76	78	80	81	78	93	582
7	6	103	160	161	141	154	161	106	986
8	7	133	185	211	208	250	207	130	1324
9	8	206	312	346	301	354	338	228	2085
10	9	204	282	305	282	296	337	206	1912
11	10	243	311	274	286	294	298	238	1944
12	11	261	313	322	307	341	321	268	2133
13	12	266	359	295	324	347	379	321	2291
14	13	326	379	354	351	374	361	345	2490
15	14	334	388	409	460	418	454	390	2853
16	15	318	434	399	351	381	434	343	2660
17	16	390	421	475	462	472	499	401	3120
18	17	364	441	490	485	470	462	402	3114
19	18	352	373	436	464	407	436	380	2848
20	19	278	326	324	357	378	368	319	2350
21	20	283	267	285	277	335	319	287	2053
22	21	242	211	210	279	251	262	259	1714
23	22	239	170	232	246	222	262	249	1620

Wykres skumulowany z podziałem – liczba poszkodowanych wg typu uczestnika ruchu i pory dnia

```
SELECT
    t.CrashPartOfDay AS TimeCategory,
    SUM(c.PedestriansInjured) AS TotalPedestriansInjured,
    SUM(c.CyclistsInjured) AS TotalCyclistsInjured,
    SUM(c.MotoristInjured) AS TotalMotoristsInjured,
    SUM(c.PeopleInjured) AS TotalPeopleInjured
FROM
    CrashFacts c
JOIN
    vwTimeDim_Crash t ON c.CrashTimeKey = t.CrashTimeKey
GROUP BY
    t.CrashPartOfDay
ORDER BY
    CASE t.CrashPartOfDay
        WHEN 'Early Morning' THEN 1
        WHEN 'Late Morning' THEN 2
        WHEN 'Afternoon' THEN 3
        WHEN 'Evening' THEN 4
        WHEN 'Late Evening' THEN 5
        WHEN 'Night' THEN 6
        ELSE 0
    END;
```

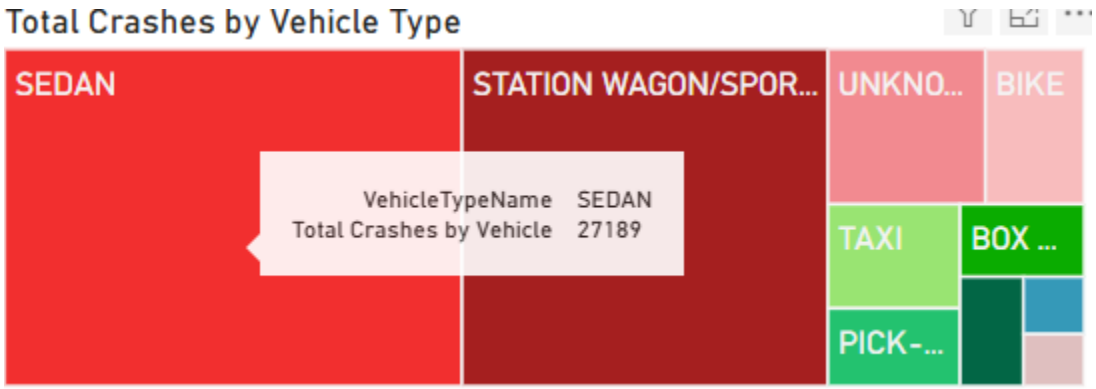


Results | Messages

	TimeCategory	TotalPedestriansInjured	TotalCyclistsInjured	TotalMotoristsInjured	TotalPeopleInjured
1	Early Morning	469	212	1310	1991
2	Late Morning	439	299	1436	2174
3	Afternoon	957	904	3883	5744
4	Evening	1100	918	3188	5206
5	Late Evening	371	410	1831	2612
6	Night	179	200	1550	1929

Tree map – 10 najczęściej uczestniczących w wypadkach typów pojazdów

```
SELECT TOP 10
    vt.VehicleTypeName,
    COUNT(*) AS CrashCount
FROM
    CrashFacts c
JOIN
    CrashVehicleBridge cvb ON c.Crash_ID = cvb.Crash_ID
JOIN
    VehicleTypesDim vt ON cvb.Vehicle_ID = vt.Vehicle_ID
GROUP BY
    vt.VehicleTypeName
ORDER BY
    COUNT(*) DESC;
```

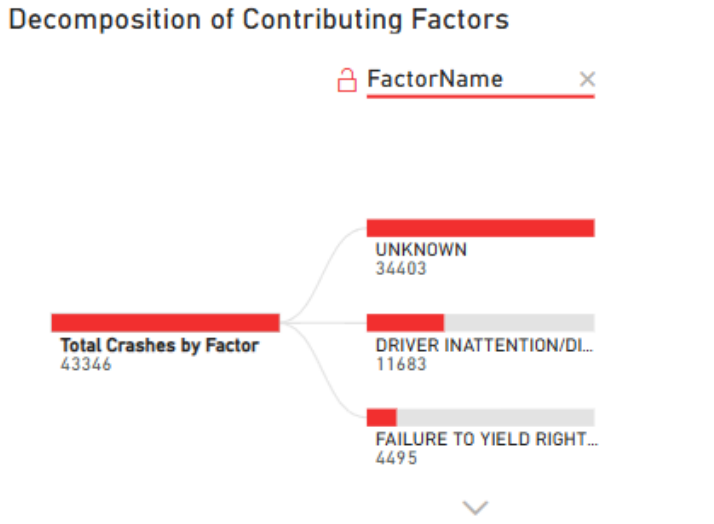


Results | Messages

	VehicleTypeName	CrashCount
1	SEDAN	27189
2	STATION WAGON/SPORT UTILITY VEHICLE	21748
3	UNKNOWN	4315
4	BIKE	2708
5	TAXI	2426
6	PICK-UP TRUCK	1809
7	BOX TRUCK	1585
8	BUS	1217
9	MOTORCYCLE	614
10	E-SCOOTER	556

Drzewo dekompozycji – analiza przyczyn wypadków (Contributing Factors)

```
WITH FactorAnalysis AS (
    SELECT
        cf.FactorName,
        COUNT(DISTINCT c.Crash_ID) AS CrashCount
    FROM
        CrashFacts c
    JOIN
        CrashFactorBridge cfb ON c.Crash_ID = cfb.Crash_ID
    JOIN
        ContributingFactorsDim cf ON cfb.Factor_ID = cf.Factor_ID
    GROUP BY
        cf.FactorName
)

SELECT * FROM FactorAnalysis
ORDER BY
    CrashCount DESC;
```



Results | Messages

	FactorName	CrashCount
1	UNKNOWN	34403
2	DRIVER INATTENTION/DISTRACTION	11683
3	FAILURE TO YIELD RIGHT-OF-WAY	4495
4	FOLLOWING TOO CLOSELY	2484
5	PASSING OR LANE USAGE IMPROPER	1855
6	TRAFFIC CONTROL DISREGARDED	1722
7	PASSING TOO CLOSELY	1536
8	OTHER VEHICULAR	1419
9	BACKING UNSAFELY	1282
10	UNSAFE SPEED	1199
11	TURNING IMPROPERLY	1160

Strona raportowa 4 - EMS Dispatch Analysis

KPI Cards:

```
SELECT
    COUNT(*) AS Total_EMS_Dispatch_Interventions,
    FORMAT(COUNT(*), 'N0') + 'K' AS Formatted_Value
FROM EMSFacts;

SELECT
    AVG(InterventionDuration) AS Avg_Duration_Decimal,
    CONCAT(
        CAST(AVG(InterventionDuration) AS INT), ' min ',
        CAST(ROUND((AVG(InterventionDuration) - CAST(AVG(InterventionDuration) AS
INT)) * 60, 0) AS INT), ' s'
    ) AS Avg_Intervention_Duration
FROM EMSFacts;

SELECT
    AVG(OnSceneArrivalDuration) AS Avg_Response_Decimal,
    CONCAT(
        CAST(AVG(OnSceneArrivalDuration) AS INT), ' min ',
        CAST(ROUND((AVG(OnSceneArrivalDuration) - CAST(AVG(OnSceneArrivalDuration) AS
INT)) * 60, 0) AS INT), ' s'
    ) AS Avg_Response_Time
FROM EMSFacts;
```

Results Messages

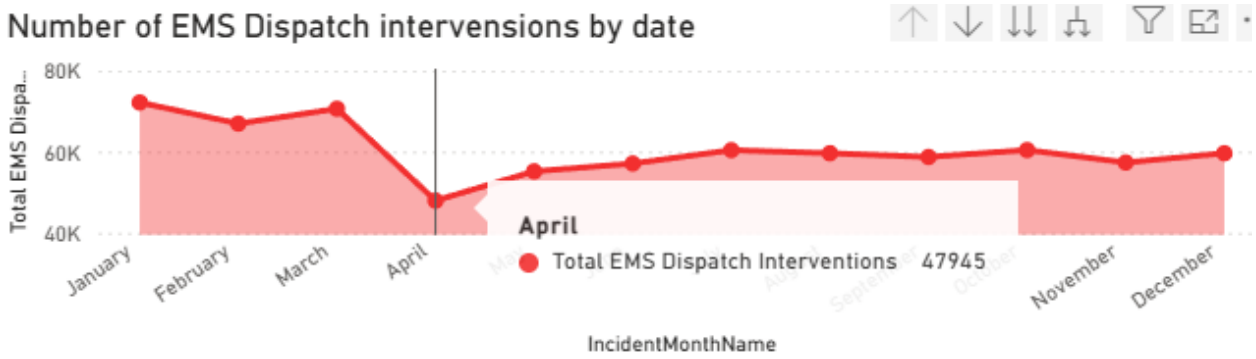
	Total_EMS_Dispatch_Interventions	Formatted_Value
1	726033	726,033K

	Avg_Duration_Decimal	Avg_Intervention_Duration
1	77.485937	77 min 29 s

	Avg_Response_Decimal	Avg_Response_Time
1	7.093565	7 min 6 s

Wykres liniowy - liczba interwencji w zależności od miesiąca

```
SELECT
    t.IncidentMonthName AS IncidentMonthName,
    COUNT(*) AS InterventionCount
FROM EMSFacts e
JOIN vwDimDate_Incident t ON e.IncidentDateKey = t.IncidentDateKey
GROUP BY t.IncidentMonthName, t.IncidentMonth
ORDER BY t.IncidentMonth;
```

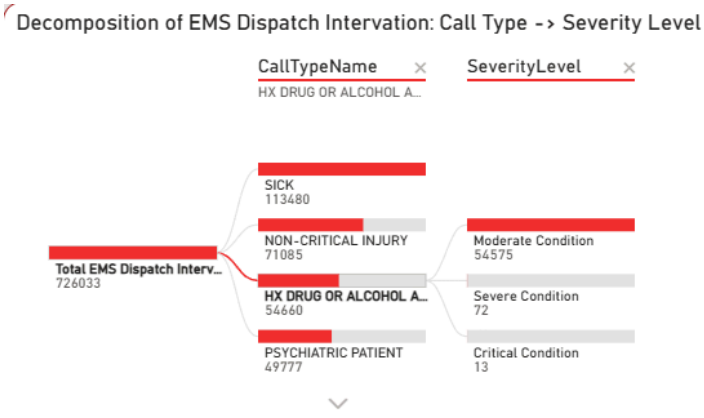


Results Messages

	IncidentMonthName	InterventionCount
1	January	72150
2	February	66943
3	March	70610
4	April	47945
5	May	55160
6	June	57094
7	July	60394
8	August	59631
9	September	58710
10	October	60422
11	November	57343

Decomposition Tree – interwencje służb ratunkowych według typu zdarzenia i poziomu ciężkości

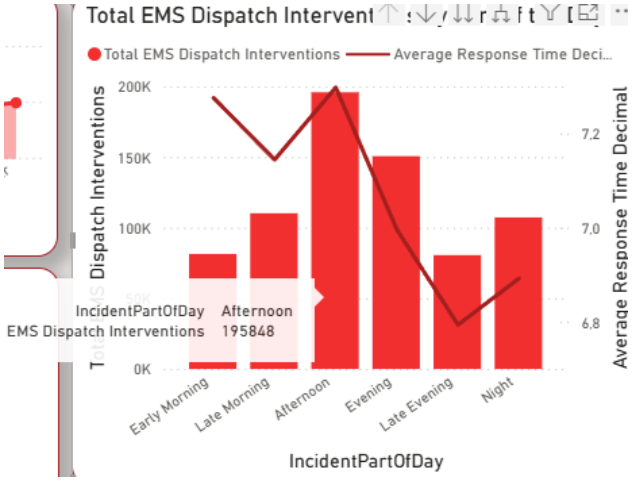
```
WITH CallTypeSeverity AS (
    SELECT
        ct.CallTypeName,
        sl.SeverityLevel,
        COUNT(*) AS InterventionCount
    FROM EMSFacts e
    JOIN CallTypesDim ct ON e.FinalCallTypeKey = ct.CallType_ID
    JOIN SeverityLevelsDim sl ON e.FinalSeverityLevelKey = sl.SeverityLevel_ID
    GROUP BY ct.CallTypeName, sl.SeverityLevel
)

SELECT
    CallTypeName,
    SeverityLevel,
    InterventionCount,
    SUM(InterventionCount) OVER (PARTITION BY CallTypeName) AS TotalForCallType
FROM CallTypeSeverity
ORDER BY
    SUM(InterventionCount) OVER (PARTITION BY CallTypeName) DESC,
    CallTypeName,
    InterventionCount DESC;
```

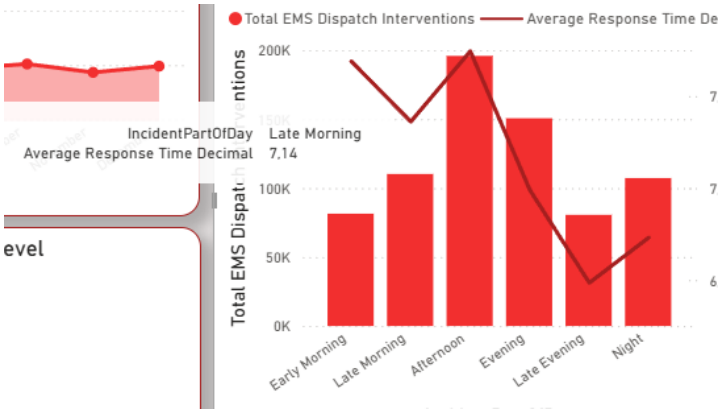


	CallTypeName	SeverityLevel	InterventionCount	TotalForCallType
7	NON-CRITICAL INJURY	Minor Condition	68090	71085
8	NON-CRITICAL INJURY	Moderate Condition	2858	71085
9	NON-CRITICAL INJURY	Severe Condition	105	71085
10	NON-CRITICAL INJURY	Critical Condition	32	71085
11	HX DRUG OR ALCOHOL ABUSE	Moderate Condition	54575	54660
12	HX DRUG OR ALCOHOL ABUSE	Severe Condition	72	54660
13	HX DRUG OR ALCOHOL ABUSE	Critical Condition	13	54660
14	PSYCHIATRIC PATIENT	Informational Call	49544	49777
15	PSYCHIATRIC PATIENT	Severe Condition	189	49777
16	PSYCHIATRIC PATIENT	Critical Condition	27	49777
17	PSYCHIATRIC PATIENT	Moderate Condition	16	49777

Line and clustered column chart ilości interwencji i średniego czas reakcji

```
SELECT
    t.IncidentPartOfDay,
    COUNT(*) AS InterventionCount,
    ROUND(AVG(e.OnSceneArrivalDuration), 2) AS AvgResponseTimeMinutes
FROM EMSFacts e
JOIN vwTimeDim_Incident t ON e.IncidentTimeKey = t.IncidentTimeKey
GROUP BY t.IncidentPartOfDay
```



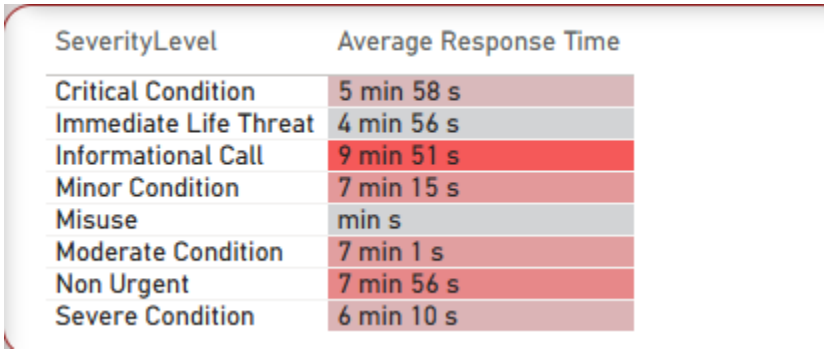


Results | Messages

	IncidentPartOfDay	InterventionCount	AvgResponseTimeMinutes
1	Afternoon	195848	7.300000
2	Early Morning	81439	7.280000
3	Evening	150628	7.000000
4	Late Evening	80604	6.790000
5	Late Morning	110230	7.140000
6	Night	107284	6.890000

Tabela – Poziom ciężkości zdarzenia a średni czas reakcji

```
SELECT
    sl.SeverityLevel,
    AVG(e.OnSceneArrivalDuration) AS AvgResponseTimeMinutes,
    CONCAT(
        FLOOR(AVG(e.OnSceneArrivalDuration)), ' min ',
        ROUND((AVG(e.OnSceneArrivalDuration) * 60) % 60, 0), ' s'
    ) AS AvgResponseTimeFormatted
FROM EMSFacts e
JOIN SeverityLevelsDim sl ON e.FinalSeverityLevelKey = sl.SeverityLevel_ID
GROUP BY sl.SeverityLevel
ORDER BY AVG(e.OnSceneArrivalDuration);
```



SeverityLevel	Average Response Time
Critical Condition	5 min 58 s
Immediate Life Threat	4 min 56 s
Informational Call	9 min 51 s
Minor Condition	7 min 15 s
Misuse	min s
Moderate Condition	7 min 1 s
Non Urgent	7 min 56 s
Severe Condition	6 min 10 s

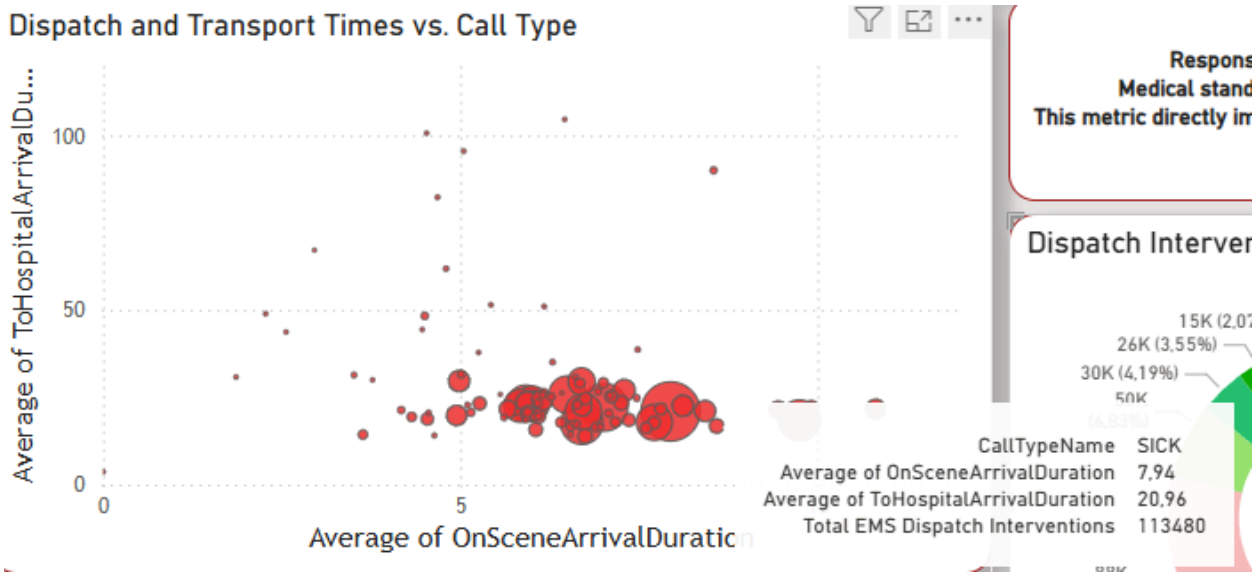
Results Messages

	SeverityLevel	AvgResponseTimeMinutes	AvgResponseTimeFormattec
1	Immediate Life Threat	4.929088	4 min 56.000000 s
2	Critical Condition	5.969885	5 min 58.000000 s
3	Severe Condition	6.162058	6 min 10.000000 s
4	Moderate Condition	7.017605	7 min 1.000000 s
5	Minor Condition	7.252850	7 min 15.000000 s
6	Non Urgent	7.937763	7 min 56.000000 s
7	Informational Call	9.846691	9 min 51.000000 s

Strona raportowa 5 - EMS Dispatch Effectiveness

Scatter Plot – rozkład średnich czasów reakcji według typu zgłoszenia

```
SELECT
    ct.CallTypeName,
    AVG(f.OnSceneArrivalDuration) AS AvgOnSceneArrivalDuration,
    AVG(f.ToHospitalArrivalDuration) AS AvgToHospitalArrivalDuration,
    COUNT(*) AS InterventionCount
FROM EMSFacts f
JOIN CallTypesDim ct ON f.FinalCallTypeKey = ct.CallType_ID
GROUP BY ct.CallTypeName, ct.CallType_ID
ORDER BY AvgOnSceneArrivalDuration DESC;
```

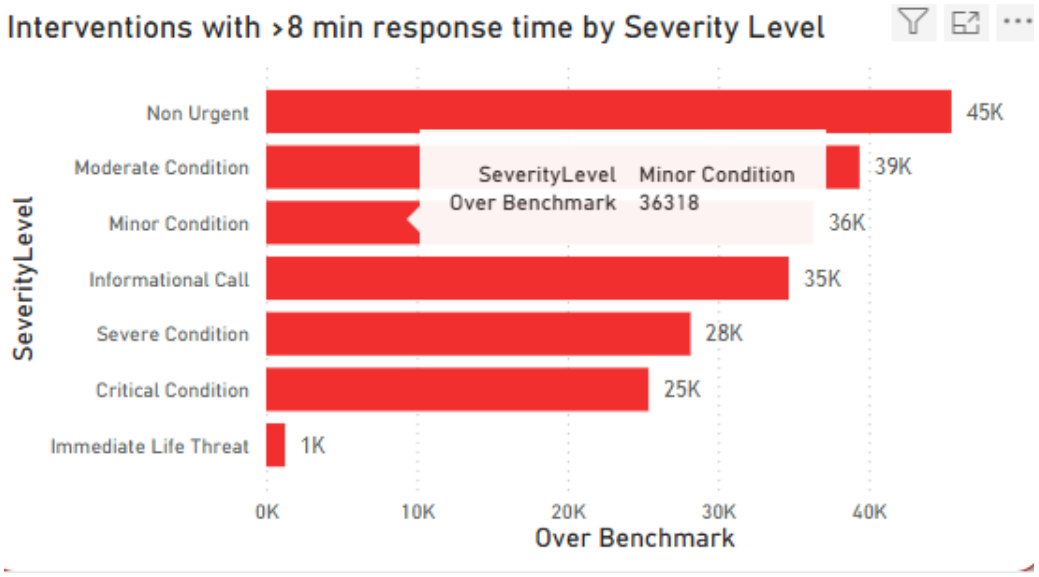


Results | Messages

	CallTypeName	AvgOnSceneArrivalDuration	AvgToHospitalArrivalDuration	InterventionCount
7	HOSTAGE SITUATION / BARRICADED	8.545809	90.009206	315
8	RESPIRATORY DISTRESS	8.426504	21.073914	10774
9	SICK - COUGH & FEVER	8.115095	22.593646	10834
10	SICK	7.943809	20.956446	113480
11	RESP DISTRESS - FEVER&COUGH	7.808666	21.673238	2174
12	ABDOMINAL PAIN	7.717854	17.824770	38153
13	GYN BLEEDING/PT NOT PREGNANT	7.711458	17.723943	2716
14	MISCARRIAGE	7.599651	16.206946	1952
15	CHILD ABUSE	7.484054	38.669054	74
16	STROKE - FEVER & COUGH	7.468137	24.800344	290
17	ABDOMINAL PAIN-FEVER & COUGH	7.361244	18.507133	2861

Clustered Bar Chart – przekroczenie standardu 8 minut według poziomu ciężkości

```
SELECT
    sl.SeverityLevel,
    COUNT(*) AS InterventionsCount
FROM EMSFacts f
JOIN SeverityLevelsDim sl ON f.FinalSeverityLevelKey = sl.SeverityLevel_ID
WHERE
    f.OnSceneArrivalDuration > 8
GROUP BY sl.SeverityLevel, sl.SeverityLevel_ID
```

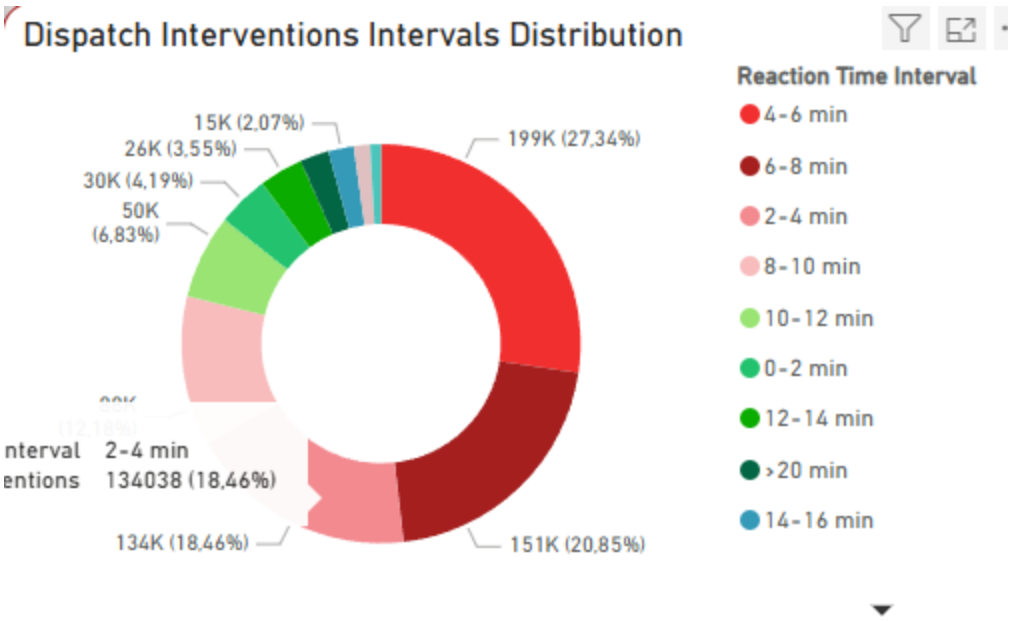


Results | Messages

	SeverityLevel	InterventionsCount
1	Moderate Condition	39392
2	Immediate Life Threat	1252
3	Critical Condition	25385
4	Informational Call	28
5	Severe Condition	28170
6	Informational Call	34666
7	Minor Condition	36318
8	Non Urgent	45494

Donut Chart – rozkład czasów dotarcia na miejsce zdarzenia

```
WITH TimeIntervals AS (
    SELECT
        CASE
            WHEN OnSceneArrivalDuration < 2 THEN '0-2 min'
            WHEN OnSceneArrivalDuration < 4 THEN '2-4 min'
            WHEN OnSceneArrivalDuration < 6 THEN '4-6 min'
            WHEN OnSceneArrivalDuration < 8 THEN '6-8 min'
            WHEN OnSceneArrivalDuration < 10 THEN '8-10 min'
            WHEN OnSceneArrivalDuration < 12 THEN '10-12 min'
            WHEN OnSceneArrivalDuration < 14 THEN '12-14 min'
            WHEN OnSceneArrivalDuration < 16 THEN '14-16 min'
            WHEN OnSceneArrivalDuration < 18 THEN '16-18 min'
            WHEN OnSceneArrivalDuration < 20 THEN '18-20 min'
            ELSE '>20 min'
        END AS TimeInterval,
        OnSceneArrivalDuration
    FROM EMSFacts
),
IntervalStats AS (
    SELECT
        TimeInterval,
        COUNT(*) AS InterventionCount,
        ROUND(COUNT(*) * 100.0 / (SELECT COUNT(*) FROM TimeIntervals), 2) AS Percentage
    FROM TimeIntervals
    GROUP BY TimeInterval
)
SELECT
    TimeInterval,
    InterventionCount,
    Percentage
FROM IntervalStats;
```

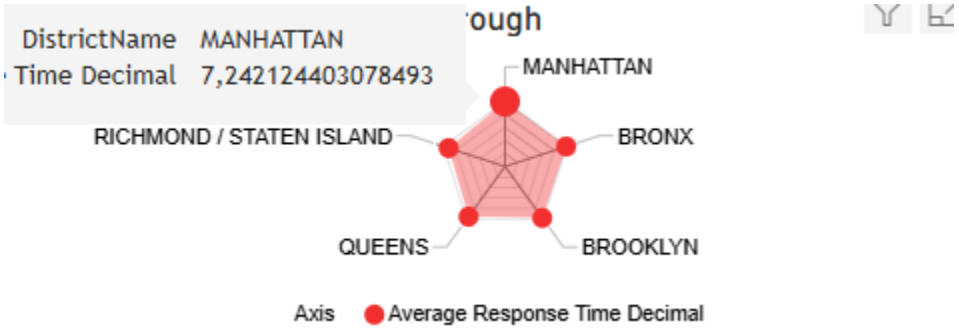


Results Messages

	TimeInterval	InterventionCount	Percentage
1	0-2 min	30418	4.190000000000
2	>20 min	16937	2.330000000000
3	12-14 min	25786	3.550000000000
4	10-12 min	49622	6.830000000000
5	2-4 min	134038	18.460000000000
6	16-18 min	9632	1.330000000000
7	6-8 min	151389	20.850000000000
8	8-10 min	88428	12.180000000000
9	18-20 min	6264	0.860000000000
10	14-16 min	15009	2.070000000000
11	4-6 min	198510	27.340000000000

Radar Chart – średni czas reakcji wg dzielnicy

```
SELECT
    d.DistrictName,
    AVG(f.OnSceneArrivalDuration) AS AvgResponseTime
FROM EMSFacts f
JOIN DistrictDetailsDim d ON f.BoroughKey = d.District_ID
GROUP BY d.DistrictName
ORDER BY AvgResponseTime DESC;
```



Results Messages

	DistrictName	AvgResponseTime
1	MANHATTAN	7.242124
2	BRONX	7.196689
3	BROOKLYN	7.086363
4	QUEENS	6.917168
5	RICHMOND / STATEN ISLAND	6.616372

Strona raportowa 6 - Crashes and EMS Dispatch by Borough

Tabela podsumowująca charakterystyki dzielnic

```
SELECT
    d.DistrictName AS Borough,
    d.AvgPopulation,
    ROUND(1.0 * d.TotalSNAPRecipients / d.AvgPopulation * 100, 2) AS [% SNAP
Recipients],
    ROUND(1.0 * d.TotalCashAssistanceRecipients / d.AvgPopulation * 100, 2) AS [% Cash
Assistance],
    ROUND(1.0 * d.TotalMEDICAIDRecipients / d.AvgPopulation * 100, 2) AS [% Medicaid],
    (SELECT COUNT(DISTINCT c.Crash_ID)
     FROM CrashFacts c
     WHERE c.BoroughKey = d.District_ID) AS TotalCrashes,
    (SELECT COUNT(DISTINCT e.EMS_ID)
     FROM EMSFacts e
     WHERE e.BoroughKey = d.District_ID) AS TotalEMSInterventions,
    CAST(ROUND(
        (SELECT COUNT(DISTINCT c.Crash_ID)
         FROM CrashFacts c
         WHERE c.BoroughKey = d.District_ID) * 100000.0 / d.AvgPopulation, 0
    ) AS INT) AS CrashesPer100k,
    CAST(ROUND(
        (SELECT COUNT(DISTINCT e.EMS_ID)
         FROM EMSFacts e
         WHERE e.BoroughKey = d.District_ID) * 100000.0 / d.AvgPopulation, 0
    ) AS INT) AS EMSInterventionsPer100k
FROM
    DistrictDetailsDim d
WHERE
    d.IsValid = 1
ORDER BY
    d.AvgPopulation DESC;
```

DistrictName	Sum of Population	% Cash Assistance Recipients	% Medicaid Recipients	% SNAP Recipients	Number of crashes per 100k residents	Number of EMS per 100k residents
BRONX	1446788	0,78%	2,24%	2,63%	549	11779
BROOKLYN	2648452	0,23%	1,11%	1,20%	544	7628
MANHATTAN	1638281	0,26%	1,09%	1,16%	401	10533
QUEENS	2330295	0,16%	1,03%	0,92%	565	6369
RICHMOND / STATEN ISLAND	476143	1,11%	4,57%	4,44%	305	6854
Total	**8539959**	**0,36%**	**1,47%**	**1,54%**	**510**	**8502**

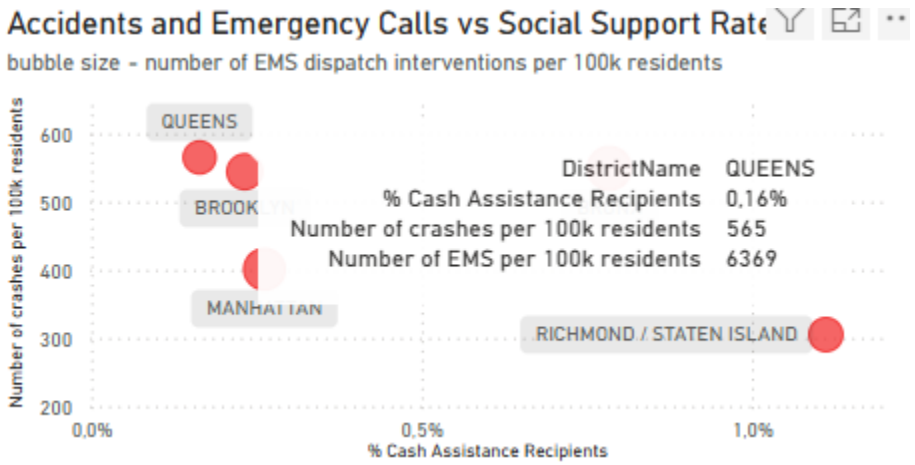
Results | Messages

	Borough	AvgPopulation	% SNAP Recipients	% Cash Assistance	% Medicaid	TotalCrashes	TotalEMSInterventions	CrashesPer100k	EMSInterventionsPer100k
1	BROOKLYN	2648452	1.200000000000	0.230000000000	1.110000000000	14412	202018	544	7628
2	QUEENS	2330295	0.920000000000	0.160000000000	1.030000000000	13156	148411	565	6369
3	MANHATTAN	1638281	1.160000000000	0.260000000000	1.090000000000	6569	172552	401	10533
4	BRONX	1446788	2.630000000000	0.780000000000	2.240000000000	7947	170415	549	11779
5	RICHMOND / STATEN ISLAND	476143	4.440000000000	1.110000000000	4.570000000000	1453	32637	305	6854

Scatter Plot – zależność między pomocą społeczną a liczbą zdarzeń ratunkowych i wypadków

```
WITH BoroughStats AS (
    SELECT
        d.District_ID,
        d.DistrictName AS Borough,
        d.TotalCashAssistanceRecipients,
        d.AvgPopulation,
        CAST(d.TotalCashAssistanceRecipients AS FLOAT) / d.AvgPopulation * 100 AS
CashAssistancePercentage
    FROM DistrictDetailsDim d
    WHERE d.IsValid = 1
),
EMSCounts AS (
    SELECT
        e.BoroughKey,
        COUNT(DISTINCT e.EMS_ID) AS EMSInterventions
    FROM EMSFacts e
    GROUP BY e.BoroughKey
),
CrashCounts AS (
    SELECT
        c.BoroughKey,
        COUNT(DISTINCT c.Crash_ID) AS Crashes
    FROM CrashFacts c
    GROUP BY c.BoroughKey
)
SELECT
    b.Borough,
    ROUND(b.CashAssistancePercentage, 2) AS CashAssistancePercentage,
    COALESCE(e.EMSInterventions, 0) AS EMSInterventions,
    COALESCE(c.Crashes, 0) AS Crashes,
    ROUND(
      CASE WHEN b.AvgPopulation > 0 THEN COALESCE(e.EMSInterventions, 0) * 100000.0 /
b.AvgPopulation ELSE 0 END
    , 0) AS EMSInterventionsPer100k,
    ROUND(
      CASE WHEN b.AvgPopulation > 0 THEN COALESCE(c.Crashes, 0) * 100000.0 /
b.AvgPopulation ELSE 0 END
    , 0) AS CrashesPer100k
FROM BoroughStats b
LEFT JOIN EMSCounts e ON b.District_ID = e.BoroughKey
```

```
LEFT JOIN CrashCounts c ON b.District_ID = c.BoroughKey
ORDER BY b.CashAssistancePercentage;
```

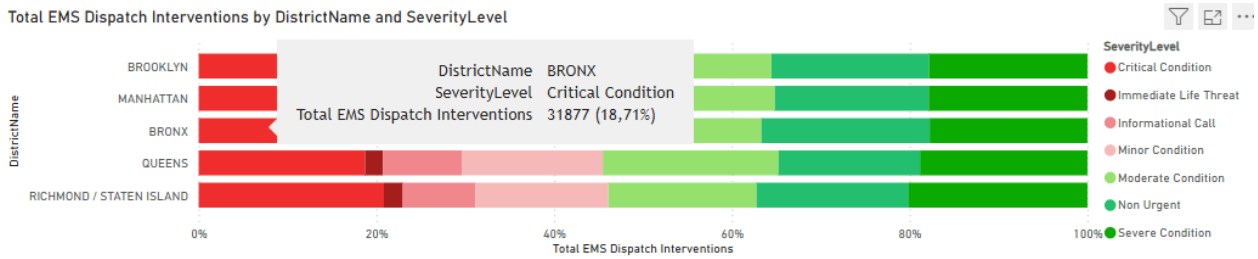


Results | Messages

	Borough	CashAssistancePercentage	EMSInterventions	Crashes	EMSInterventionsPer100k	CrashesPer100k
1	QUEENS	0,16	148411	13156	6369.000000000000	565.000000000000
2	BROOKLYN	0,23	202018	14412	7628.000000000000	544.000000000000
3	MANHATTAN	0,26	172552	6569	10533.000000000000	401.000000000000
4	BRONX	0,78	170415	7947	11779.000000000000	549.000000000000
5	RICHMOND / STATEN ISLAND	1,11	32637	1453	6854.000000000000	305.000000000000

100% Stacked Bar Chart – struktura poziomów ciężkości interwencji według dzielnicy

```
WITH SeverityByBorough AS (
    SELECT
        d.DistrictName AS Borough,
        s.SeverityLevel,
        COUNT(*) AS InterventionCount
    FROM EMSFacts e
    JOIN DistrictDetailsDim d ON e.BoroughKey = d.District_ID AND d.IsValid = 1
    JOIN SeverityLevelsDim s ON e.FinalSeverityLevelKey = s.SeverityLevel_ID
    GROUP BY d.DistrictName, s.SeverityLevel
),
TotalByBorough AS (
    SELECT
        Borough,
        SUM(InterventionCount) AS TotalInterventions
    FROM SeverityByBorough
    GROUP BY Borough
)
SELECT
    s.Borough,
    s.SeverityLevel,
    s.InterventionCount,
    t.TotalInterventions,
    CAST(s.InterventionCount AS FLOAT) / t.TotalInterventions * 100 AS Percentage
FROM SeverityByBorough s
JOIN TotalByBorough t ON s.Borough = t.Borough
ORDER BY s.Borough, s.SeverityLevel;
```

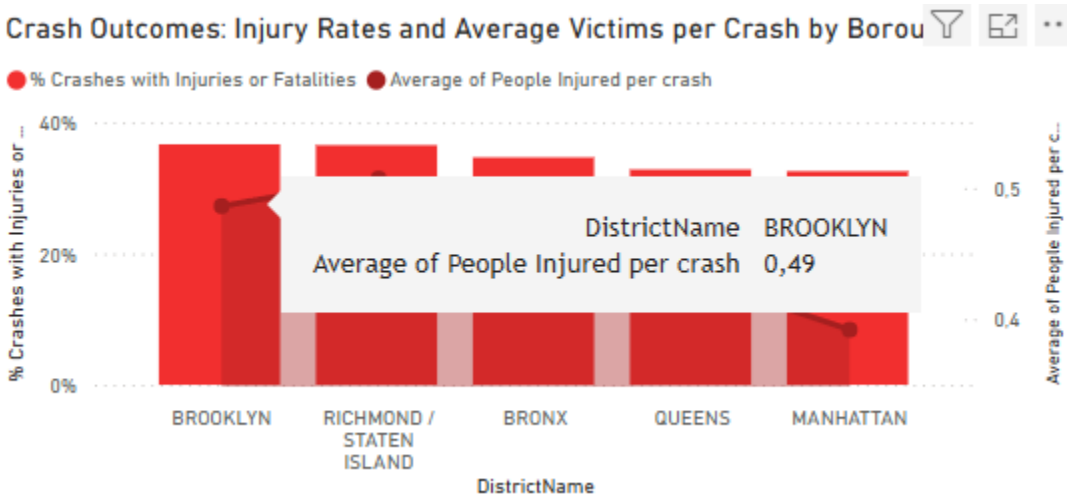


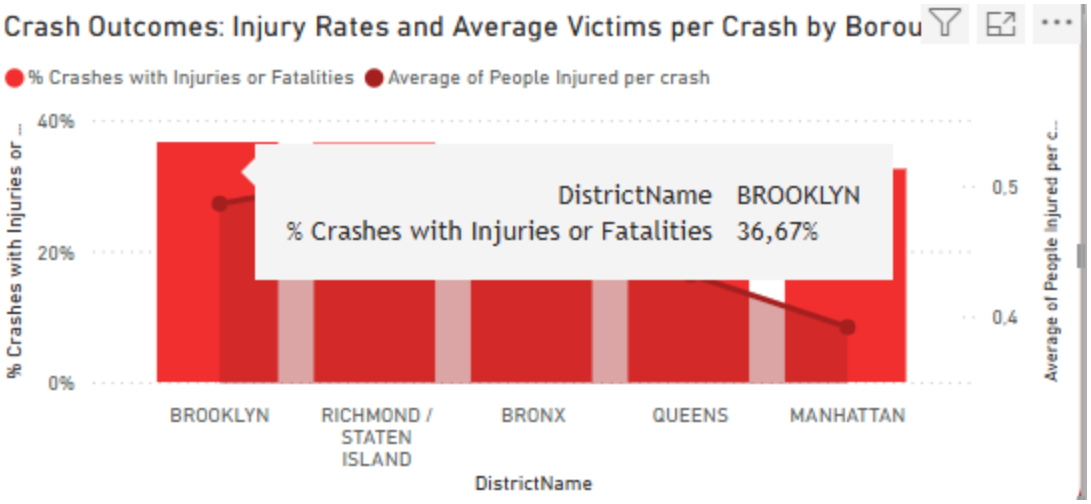
Results Messages

	Borough	SeverityLevel	InterventionCount	TotalInterventions	Percentage
1	BRONX	Critical Condition	31877	170415	18,7055130123522
2	BRONX	Immediate Life Threat	2726	170415	1,59962444620485
3	BRONX	Informational Call	14889	170415	8,73690696241528
4	BRONX	Minor Condition	29494	170415	17,307161928234
5	BRONX	Moderate Condition	28900	170415	16,9586010621131
6	BRONX	Non Urgent	32284	170415	18,9443417539536
7	BRONX	Severe Condition	30245	170415	17,747850834727
8	BROOKLYN	Critical Condition	38250	202018	18,9339563801245
9	BROOKLYN	Immediate Life Threat	3501	202018	1,7330138898514
10	BROOKLYN	Informational Call	17826	202018	8,82396618123138
11	BROOKLYN	Minor Condition	32679	202018	16,1762813214664

Line and clustered column plot - udział wypadków z ofiarami oraz średnia liczba poszkodowanych na wypadek per dzielnica

```
WITH CrashStats AS (
    SELECT
        dd.DistrictName,
        COUNT(CASE WHEN cf.PeopleInjured > 0 OR cf.PeopleKilled > 0 THEN 1 END) *
100.0 / COUNT(*) AS PercentageWithInjuries,
        SUM(cf.PeopleInjured * 1.0) / COUNT(*) AS AvgPeopleInjuredPerCrash
    FROM
        CrashFacts cf
    JOIN
        DistrictDetailsDim dd ON cf.BoroughKey = dd.District_ID
    WHERE
        dd.IsValid = 1
    GROUP BY
        dd.DistrictName
)
SELECT
    DistrictName,
    ROUND(PercentageWithInjuries, 2) AS PercentageWithInjuries,
    ROUND(AvgPeopleInjuredPerCrash, 2) AS AvgPeopleInjuredPerCrash
FROM
    CrashStats
ORDER BY
    DistrictName;
```





Results Messages

	DistrictName	PercentageWithInjuries	AvgPeopleInjuredPerCrash
1	BRONX	34.680000000000	0.460000
2	BROOKLYN	36.670000000000	0.490000
3	MANHATTAN	32.580000000000	0.390000
4	QUEENS	32.840000000000	0.430000
5	RICHMOND / STATEN ISLAND	36.550000000000	0.510000