

Analiza F1

Dokumentacja Projektu Hurtowni Danych

Hubert Sobociński i Paweł Pozorski

Politechnika Warszawska, Wydział Matematyki i Nauk Informacyjnych

8 czerwca 2025

Streszczenie

Celem projektu jest stworzenie kompleksowej hurtowni danych dedykowanej zespołom Formuły 1, umożliwiającej integrację, analizę oraz raportowanie danych wyścigowych z różnych sezonów. System będzie wspierał strategiczne decyzje konstruktorów, kierowców i analityków, zapewniając spójne i wysokiej jakości informacje o wynikach, postępach oraz statystykach. Planowana architektura rozwiązania opiera się na centralnym repozytorium danych, które pozwoli na łatwy dostęp, porównywanie sezonów oraz generowanie prognoz. Projekt przyniesie korzyści zarówno decydentom, jak i kibicom, zwiększając zaangażowanie i jakość analiz w świecie F1.

Spis treści

1 Cel projektu i planowane korzyści	3
1.1 Planowane korzyści z perspektywy odbiorcy rozwiązania	3
2 Diagram i Opis Architektury Rozwiązania	3
2.1 Opis architektury	4
2.2 Bezpieczeństwo połączenia z bazą danych	4
3 Wykorzystywane zbiory danych	4
4 Proces ETL	5
4.1 Extract	5
4.2 Transform	5
4.3 Load	6
5 Model fizyczny hurtowni danych	6
6 Opis kluczowych miar i atrybutów	9
6.1 Tabele faktów	9
6.1.1 Fact_race_data	9
6.1.2 Facts_entrant	10
6.2 Tabele wymiarów	10
6.2.1 Dim_constructor	10
6.2.2 Dim_driver	10
6.2.3 Dim_engine_manufacturer	10
6.2.4 Dim_race	11
7 Opis warstwy raportowej	11
8 Realizacja przykładowych raportów	12
9 Wizualizacja raportu	13
10 Podsumowanie rezultatów projektu	16
11 Testy	17
11.1 Testowanie ręczne z Prefect Deployment	17
11.2 Proces Deploymentu	17
11.3 Zadania w deploymentach	17
11.3.1 test-flow	17
11.3.2 racing-circuits	18
11.3.3 fl_attendance	18
11.3.4 flldb	18
11.4 DWH	18
11.5 Testowanie jakościowe kodu	19
11.5.1 Testowanie poprawności typów danych	19
11.5.2 Testowanie formatowania kodu	20
11.6 Testy z wykorzystaniem constraints	20
11.6.1 1. Przykłady constraintów na bazie danych	20
11.7 Testowanie Scrappingu	20
11.8 Zdjęcia przedstawiające działanie	21
11.9 Test ponownego wczytywania danych do bazy danych.	22
11.10 Test ponownego wczytywania danych do bazy danych scrapowanych ze strony z oglądalnością wyścigów oraz ze strony z informacjami torów.	24
11.11 Test ponownego procesu ETL z bazy do hurtowni danych.	27
11.12 Test zmiany danych, które nie mogą być zmieniane w hurtowni.	28
11.13 Test zmiany danych, które mogą być zmieniane w hurtowni.	30
11.14 Testy warstwy raportowej	31

1 Cel projektu i planowane korzyści

Celem projektu jest stworzenie hurtowni danych dla zespołów F1, która będzie gromadziła i analizowała informacje dotyczące konstruktorów, ich wyników oraz osiągnięć w poszczególnych sezonach wyścigowych. Hurtownia danych ma na celu integrację danych z różnych źródeł w jednym centralnym repozytorium, co umożliwi zaawansowaną analizę i generowanie raportów na temat wyników wyścigów, pozycji w mistrzostwach, oraz statystyk związanych z konkretnymi konstruktorami.

Główne cele projektu to:

- Zgromadzenie różnych szczegółowych danych na temat wyścigów F1.
- Umożliwienie łatwego dostępu do danych z różnych lat, co pozwoli na porównanie wyników konstruktorów oraz kierowców w danym sezonie oraz na przestrzeni lat.
- Tworzenie raportów dotyczących postępów konstruktorów czy kierowców, które będą użyteczne zarówno dla menedżerów wyścigów, jak i analityków sportowych.
- Zapewnienie wysokiej jakości danych, które będą zgodne z wymaganiami raportowania w branży wyścigowej.

1.1 Planowane korzyści z perspektywy odbiorcy rozwiązania

Hurtownia danych umożliwi właścicielom drużyn, konstruktorom, kierowcą czy kibicom dostęp do istotnych informacji, co pozwoli im na podejmowanie lepszych decyzji strategicznych, a dla kibiców na przykład wybór ciekawszych wyścigów do oglądania. Główne korzyści dla użytkowników końcowych to:

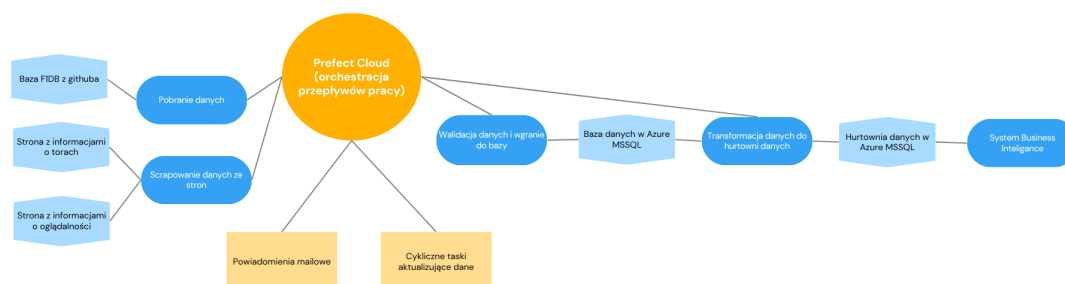
- **Lepsza analiza wyników wyścigów:** Hurtownia danych pozwoli na szybkie i dokładne analizowanie wyników poszczególnych wyścigów, co umożliwi ocenę efektywności konstruktorów w danym sezonie oraz porównanie ich wyników na przestrzeni lat.
- **Łatwiejsze raportowanie:** Zintegrowane dane w jednej hurtowni umożliwią szybkie generowanie raportów na potrzeby władz sportowych, sponsorów czy mediów, co usprawni procesy raportowe w wyścigach.
- **Lepsze prognozy:** Dzięki obszernym danym dotyczącym wyników i pozycji w mistrzostwach, możliwe będzie opracowanie bardziej trafnych prognoz dotyczących przyszłych wyników, które będą użyteczne zarówno dla analityków, jak i dla sponsorów czy mediów.
- **Lepsza organizacja i informacja dla kibiców:** Centralizacja danych pozwoli także na przygotowanie uporządkowanego terminarza wyścigów oraz łatwy dostęp do szczegółowych informacji o torach, co zwiększy zaangażowanie i satysfakcję kibiców.

Dzięki tym korzyściom, hurtownia danych przyczyni się do wzrostu efektywności zarządzania zespołami wyścigowymi, a także umożliwi lepsze podejmowanie decyzji opartych na danych. W dłuższej perspektywie projekt przyczyni się do poprawy konkurencyjności w wyścigach, dzięki dostarczeniu zespołom i analitykom narzędzi do lepszego zarządzania i analizy danych.

2 Diagram i Opis Architektury Rozwiązania

W ramach projektu zaplanowano następującą architekturę rozwiązania:

- Prefect jako narzędzie do organizacji procesów ETL,
- Zbiory danych w formacie CSV,
- Hurtownia danych oparta na bazie SQL, w której przechowywane będą wyniki wyścigów, dane o torach, uczestnikach i inne,
- Raportowanie z wykorzystaniem narzędzi Business Intelligence.



Rysunek 1: Diagram architektury rozwiązania

2.1 Opis architektury

- **Prefect Cloud:** Początkowo aplikacja była hostowana na chmurze Prefect, co umożliwiało centralne zarządzanie procesami ETL. Niestety, w darmowej wersji Prefect Cloud nie było wsparcia dla połączeń z bazą danych MSSQL, dlatego zdecydowaliśmy się na uruchomienie systemu lokalnie.
- **Azure MSSQL:** Baza danych MSSQL jest hostowana na platformie Azure ([hurtonie.database.windows.net](https://www.azure.com/hurtonie.database.windows.net)), co zapewnia wysoką dostępność i bezpieczeństwo danych. (Ze względu na ograniczone możliwości korzystania z platformy Azure to znaczy wykorzystanie darmowych zasobów, postawiliśmy później hurtownie lokalnie)
- **Lokalne hostowanie:** Testowo aplikacja działa z wykorzystaniem lokalnych zasobów i pełnej kontroli nad połączeniami do baz danych. System wciąż automatycznie pobiera najnowszy kod z repozytorium GitHub, co ułatwia rozwój i aktualizację aplikacji bez potrzeby jej resetowania.

2.2 Bezpieczeństwo połączenia z bazą danych

Dostęp do bazy MSSQL jest bezpieczny dzięki wdrożeniu kilku mechanizmów ochrony:

- **Firewall z whitelistą IP:** Aby zapewnić, że tylko autoryzowane adresy IP mogą uzyskać dostęp do bazy danych, skonfigurowano firewall z whitelistą dozwolonych adresów IP.
- **Logowanie po SQL:** Użytkownicy mogą logować się do bazy danych za pomocą standardowego logowania SQL, co zapewnia bezpieczeństwo dostępu oraz audyt działań użytkowników.
- **MS Authenticator:** Dodatkowo, osoby, które mają mieć dostęp do bazy danych przez narzędzia i muszą wykonywać operacje bezpośrednio na bazie danych, są zmuszone do korzystania z aplikacji MS Authenticator, co umożliwia uwierzytelnienie dwuetapowe i zwiększa bezpieczeństwo dostępu.

3 Wykorzystywane zbiory danych

Projekt wykorzystuje trzy główne źródła danych:

Dane z repozytorium `f1db` obejmują m.in. tabele: `drivers.csv`, `constructors.csv`, `races.csv`, `results.csv`, `circuits.csv` i inne, które razem tworzą pełną relacyjną bazę danych Formuły 1.

Zscrapowane dane z dwóch stron internetowych uzupełniają brakujące informacje niedostępne w `f1db`, takie jak liczba widzów na torze oraz szczegółowe parametry torów.

Zbiór danych	Źródło	Format danych	Częstotliwość odświeżania
Dane historyczne F1 (kierowcy, zespoły, wyścigi, wyniki, itd.)	f1db GitHub – zestaw plików CSV z pełną bazą danych F1	CSV	Co tydzień po wyścigu
Frekwencje na Grand Prix	Zescrapowane dane ze strony: f1destinations.com	Ręcznie stworzona CSV	Co tydzień po wyścigu
Dane o torach wyścigowych (lokalizacja, długość, typ, itp.)	Zescrapowane ze strony: racingcircuits.info	Ręcznie stworzona CSV	Co miesiąc

Tabela 1: Wykorzystywane zbiory danych

4 Proces ETL

ETL to proces służący do ekstrakcji danych, ich transformacji oraz załadowania do systemu docelowego. Nasz proces ETL wykorzystuje Prefect do zarządzania i harmonogramowania zadań, które realizują poszczególne etapy tego procesu.

4.1 Extract

Dane są ekstraktowane z różnych źródeł, w tym:

- Z repozytorium GitHub, gdzie znajduje się kod aplikacji.
- Z baz danych MSSQL, z których pobierane są dane wymagające przetworzenia.
- Z zewnętrznych źródeł CSV.

4.2 Transform

Proces transformacji obejmuje następujące kroki:

- **Czyszczenie danych:** Usuwanie nieprawidłowych rekordów, uzupełnianie brakujących danych.
- **Zmiana struktury danych:** Normalizacja danych.
- **Walidacja danych:** Sprawdzanie poprawności danych.
- **Agregacja danych:** Grupowanie danych w celu uzyskania bardziej skondensowanej i użytecznej formy.

W ramach procesu ETL (Extract, Transform, Load) dokonaliśmy szeregu przekształceń danych, które umożliwiają uzyskanie wymaganych wyników w strukturze analitycznej. Poniżej przedstawiono szczegółowy opis poszczególnych kroków transformacji danych.

1. Season_tyre_manufacturer → usunięte

Opis: Sezonowe wyniki producenta opon (Season_tyre_manufacturer) da się wyliczyć za pomocą Fact_Race_Data

2. Season_constructor + constructor → usunięte

Opis: Sezonowe wyniki konstruktora (Season_constructor) da się wyliczyć za pomocą Fact_Race_Data

3. Season_engine_manufacturer → usunięte

Opis: Sezonowe wyniki producenta silników (Season_engine_manufacturer) da się wyliczyć za pomocą Fact_Race_Data

4. Season_driver + Season_driver_standing → usunięte

Opis: Sezonowe wyniki kierowców (Season_driver + Season_driver_standing) da się wyliczyć za pomocą Fact_Race_Data

5. **Season_entrant + entrant → entrant**

Opis: Dane zespołów z sezonów oraz ogólne informacje o uczestnikach (entrant) zostały scalone w jedną tabelę **entrant**.

Agregacja: Zapewnia pełne dane o zespołach i ich wynikach sezonowych.

6. **Race + grand_prix → Dim_race**

Opis: Połączenie danych o wyścigach (Race) oraz Grand Prix (grand_prix) w tabeli **Dim_race**. Dodano również nazwy krajów i nazwiska związane z Grand Prix.

Usunięcie: Kolumna **total_races_held** została usunięta, ponieważ liczbę wyścigów można łatwo obliczyć przez agregację.

7. **Country + continent → Dim_country**

Opis: Dane o kraju (Country) oraz kontynencie (continent) zostały połączone w tabeli **Dim_country**, gdzie kontynent jest zawarty w danych o kraju.

8. **Season_entrant_* – Usunięte**

Opis: Kolumny **season_entrant_*** zostały usunięte, ponieważ dane te są już zawarte w tabeli **entrant**.

9. **Season_constructor_standing → usunięto**

Opis: Ranking konstruktorów da się uzyskać z tabeli *Fact_Race_Data*

10. **Chassis + engine + entrant → entrant**

Opis: Informacje o podwoziu (**chassis**) i silniku (**engine**) połączono z tabelą **entrant**, uzupełniając ją o dane specyfikacyjne zespołów.

Opis: Informacje o podwoziu (**chassis**) i silniku (**engine**) połączono z tabelą **entrant**, uzupełniając ją o dane specyfikacyjne zespołów.

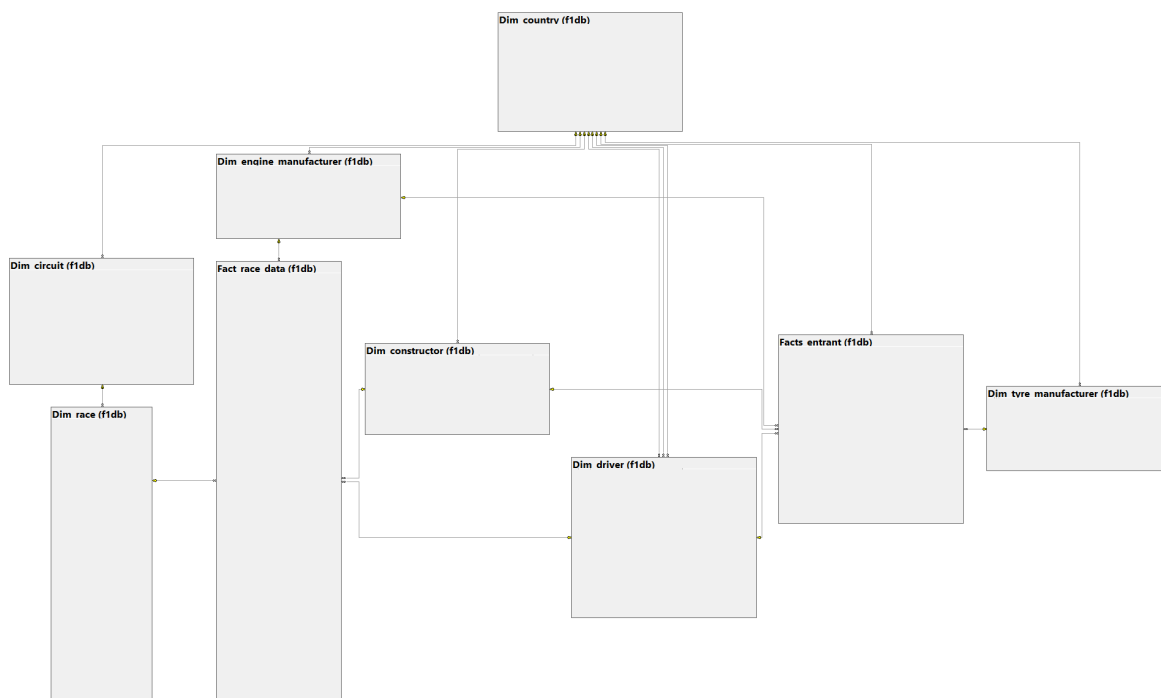
4.3 Load

Po przetworzeniu danych, są one ładowane do **Bazy MSSQL na Azure**

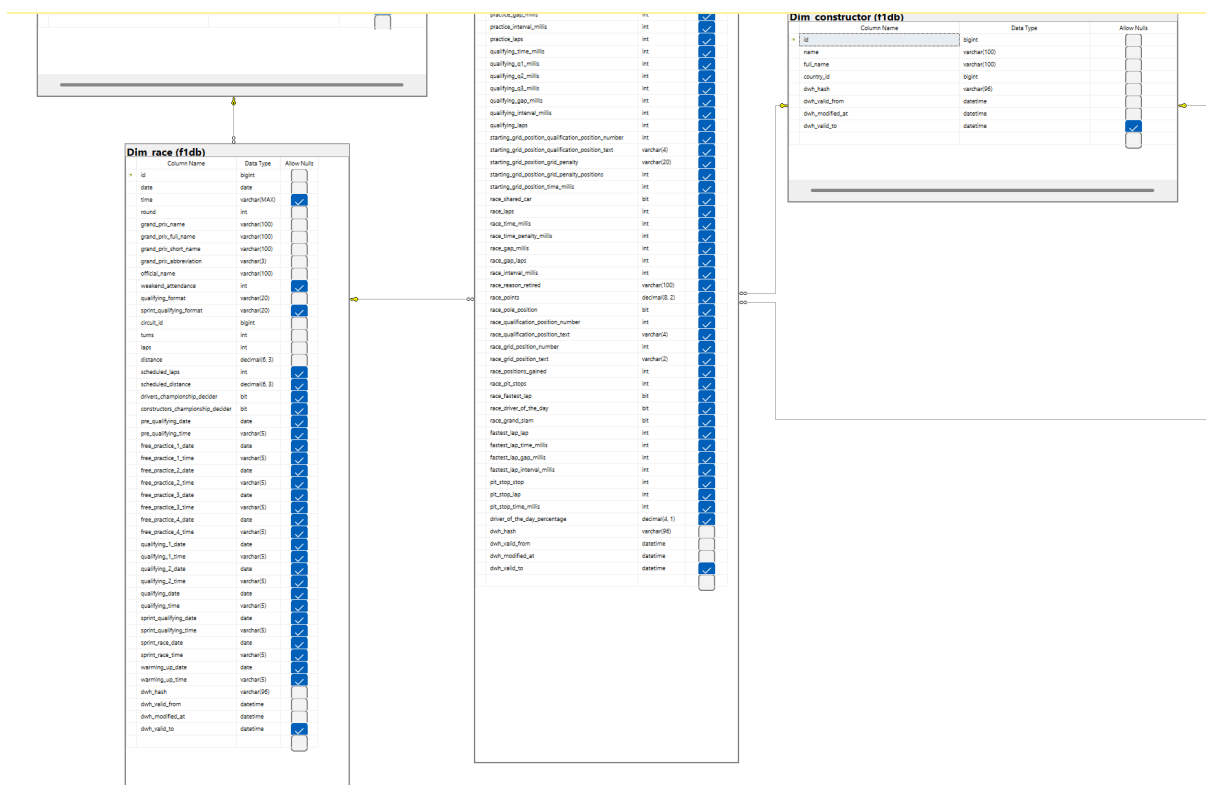
5 Model fizyczny hurtowni danych

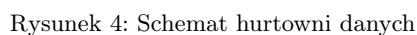
Ze względu na rozbudowaną strukturę hurtowni danych oraz ograniczenia środowiska SQL Server Management Studio, nie było możliwe przejrzyste i estetyczne wyeksportowanie diagramu struktury bazy danych w formie graficznej.

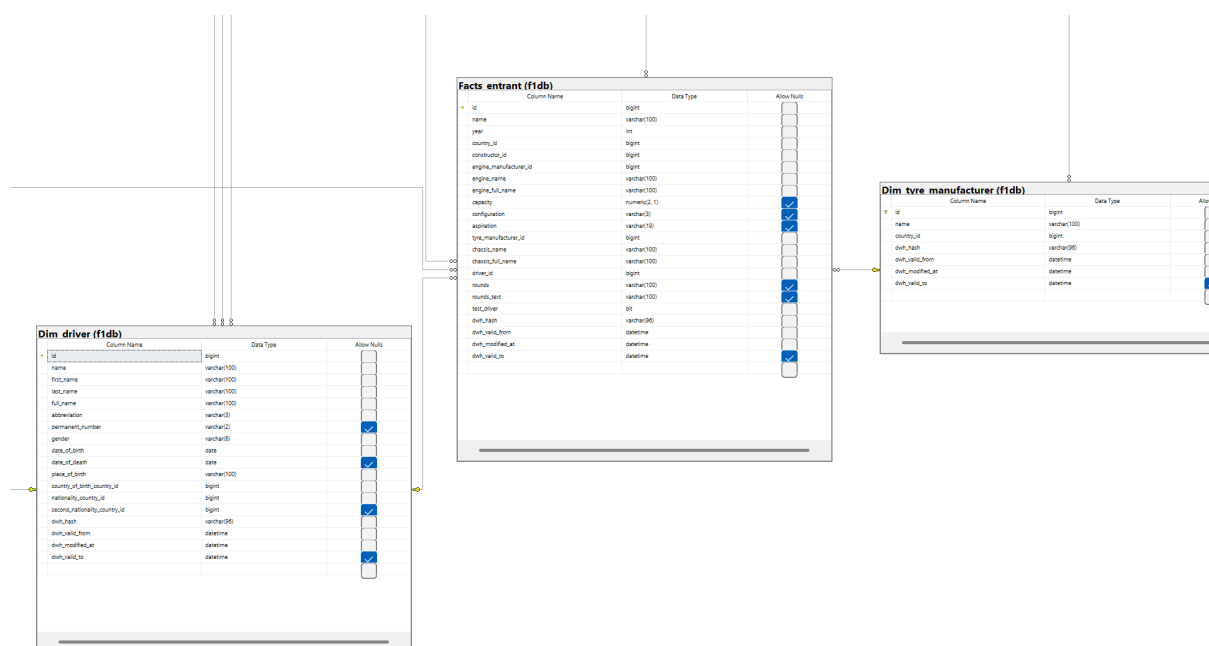
W związku z tym, pełny model logiczny hurtowni został odtworzony w postaci skryptu SQL. Skrypt ten znajduje się w pliku **skrypt_hurtowni_danych.sql** i zawiera definicje wszystkich tabel, kluczy głównych oraz relacji między encjami.



Rysunek 2: Schemat hurtowni danych







Rysunek 6: Schemat hurtowni danych

Powyższy diagram stanowi fragment przykładowej wizualizacji struktury. Pełna implementacja została załączona jako skrypt SQL.

6 Opis kluczowych miar i atrybutów

6.1 Tabele faktów

Tabele faktów zawierają miary ilościowe, które są przedmiotem analiz. Poniżej przedstawiono główne tabele faktów wraz z opisem kluczowych miar.

6.1.1 Fact race data

Kolumna	Opis
race_id	Identyfikator wyścigu (klucz obcy do Dim_race).
type	Typ sesji: race, qualifying, practice.
driver_id	Identyfikator kierowcy (Dim_driver).
constructor_id	Identyfikator zespołu (Dim_constructor).
race_laps	Liczba ukończonych okrążeń.
race_time_millis	Całkowity czas wyścigu w milisekundach.
race_gap_millis	Różnica czasowa do lidera w milisekundach.
race_positions_gained	Liczba pozycji zyskanych względem startu.
race_pit_stops	Liczba pit stopów wykonanych w wyścigu.
race_points	Punkty zdobyte za wyścig.
race_fastest_lap	Czy kierowca uzyskał najszybsze okrążenie (TAK/NIE).
race_driver_of_the_day	Czy kierowca został Driver of the Day.
qualifying_time_millis	Najlepszy czas kwalifikacyjny w milisekundach.
qualifying_laps	Liczba okrążeń w kwalifikacjach.
starting_grid_position_grid_penalty_positions	Liczba pozycji kary na starcie.
fastest_lap_time_millis	Czas najszybszego okrążenia w milisekundach.
pit_stop_time_millis	Czas trwania pit-stopu w milisekundach.
driver_of_the_day_percentage	Procent głosów oddanych na Driver of the Day.

Tabela 2: Najważniejsze miary i atrybuty w tabeli Fact race data

6.1.2 Facts_entrant

Kolumna	Opis
id	Unikalny identyfikator zespołu.
name	Nazwa wyświetlana uczestnika.
year	Rok sezonu.
country_id	Kraj uczestnika.
constructor_id	Konstruktor.
engine_manufacturer_id	Producent silnika.
engine_name	Skrócona nazwa silnika.
engine_full_name	Pełna nazwa silnika.
capacity	Pojemność silnika w litrach.
configuration	Konfiguracja silnika (np. V6, V8).
aspiration	Typ doładowania (np. Turbo, N/A).
tyre_manufacturer_id	Producent opon.
chassis_name	Skrócona nazwa nadwozia.
chassis_full_name	Pełna nazwa nadwozia.
driver_id	Identyfikator przypisanego kierowcy.
rounds	Numery rund, w których uczestniczył kierowca.
rounds_text	Opis tekstowy rund.
test_driver	Czy kierowca był testowym/rezerwowym.

Tabela 3: Najważniejsze miary i atrybuty w tabeli Facts_entrant

6.2 Tabele wymiarów

Tabele wymiarów zawierają atrybuty opisujące kontekst, którego dotyczą miary w tabelach faktów.

6.2.1 Dim_constructor

Kolumna	Opis
id	Unikalny identyfikator konstruktora.
name	Skrócona nazwa konstruktora.
full_name	Pełna oficjalna nazwa konstruktora.
country_id	Kraj pochodzenia konstruktora.

Tabela 4: Najważniejsze miary i atrybuty w tabeli Dim_constructor

6.2.2 Dim_driver

Kolumna	Opis
id	Unikalny identyfikator kierowcy.
full_name	Pełne imię i nazwisko kierowcy.
abbreviation	Skrót nazwiska kierowcy (3 litery).
gender	Płeć kierowcy.
date_of_birth	Data urodzenia kierowcy.
country_of_birth_country_id	Kraj urodzenia kierowcy.
nationality_country_id	Główna narodowość kierowcy.

Tabela 5: Najważniejsze miary i atrybuty w tabeli Dim_driver

6.2.3 Dim_engine_manufacturer

Kolumna	Opis
id	Unikalny identyfikator producenta silników.

Kolumna	Opis
name	Nazwa producenta silników.
country_id	Kraj producenta silników.

Tabela 6: Najważniejsze miary i atrybuty w tabeli Dim_engine_manufacturer

6.2.4 Dim_race

Kolumna	Opis
id	Unikalny identyfikator wyścigu.
year	Rok rozegrania wyścigu.
round	Numer wyścigu w sezonie.
date	Data wyścigu.
time	Godzina rozpoczęcia wyścigu.
grand_prix_name	Nazwa Grand Prix.
grand_prix_full_name	Pełna nazwa Grand Prix.
grand_prix_short_name	Skrócona nazwa Grand Prix.
grand_prix_abbreviation	Trzyliterowy skrót Grand Prix.
country_id	Identyfikator kraju, w którym odbywa się wyścig.
official_name	Oficjalna nazwa wydarzenia.
weekend_attendance	Frekwencja podczas weekendu wyścigowego.
qualifying_format	Format kwalifikacji.
sprint_qualifying_format	Format kwalifikacji sprinterskich.

Tabela 7: Atrybuty wymiaru wyścigu w tabeli Dim_race

7 Opis warstwy raportowej

Warstwa raportowa w hurtowni danych dla środowiska Formuły 1 odgrywa kluczową rolę w prezentacji przetworzonych informacji użytkownikom końcowym – analitykom, strategom zespołów, mediom oraz organizatorom. Zapewnia ona ustrukturyzowany i zrozumiały dostęp do danych poprzez zdefiniowane modele biznesowe, przekształcenia oraz hierarchie.

Dostęp do danych

Użytkownicy uzyskują dostęp do warstwy raportowej za pośrednictwem:

- narzędzia analitycznego Power BI,

które umożliwia interaktywne przeglądanie raportów, wizualizację danych oraz wykonywanie zapytań ad hoc. Dostęp do informacji jest kontrolowany w oparciu o system ról i uprawnień.

Model biznesowy danych

Model danych oparty jest na schemacie galaktyki (ang. *galaxy schema*). Główne fakty to:

- **Fact_race_data** – szczegółowe dane wyścigowe,
- **Fact_entrant** – dane związane z udziałem bolidów i kierowców w wyścigach.

Modele te są otoczone przez odpowiednie wymiary, takie jak czas, tor, kierowca, zespół czy lokalizacja, umożliwiając tworzenie zaawansowanych analiz przekrojowych.

Transformacje w obrębie warstwy raportowej

Transformacje stosowane w warstwie raportowej umożliwiają prezentację danych w formie zoptymalizowanej pod kątem raportowania i analizy. Wśród kluczowych transformacji znajdują się:

- obliczanie skumulowanej liczby punktów w obrębie sezonu,
- formatowanie czasu trwania pit stopów,
- oznaczenie ostatniego rozegranego wyścigu,
- identyfikacja najbliższego zaplanowanego wyścigu,
- określenie, czy kierowca znalazł się na podium w danym wyścigu,
- agregacja i formatowanie łącznego czasu ukończenia wyścigu,
- obliczenia zagregowanej oraz średniej oglądalności wyścigów,
- porównania oglądalności sezonów (wzrosty/spadki),
- wyodrębnienie roku wyścigu z daty wydarzenia,
- sumowanie liczby miejsc na podium i zwycięstw danego kierowcy,
- kalkulacja łącznych punktów kierowców oraz zespołów w sezonie,
- oznaczenie, czy wyścig się odbył oraz czy należy do najbliższych Grand Prix.

8 Realizacja przykładowych raportów

1. Rankingi

Cel: Prezentacja bieżącej klasyfikacji kierowców i zespołów.

Zakres:

- klasyfikacja kierowców w danym sezonie,
- klasyfikacja zespołów w danym sezonie,
- wykres przedstawiający zmiany punktacji kierowców w trakcie sezonu,
- ranking popularności kierowców na podstawie danych demograficznych i interakcji fanów.

2. Kierowcy

Cel: Analiza osiągnięć i kariery poszczególnych kierowców.

Zakres:

- wyszukiwanie kierowców według imienia i nazwiska,
- liczba zdobytych punktów,
- narodowość i data urodzenia,
- liczba miejsc na podium,
- liczba zwycięstw w wyścigach.

3. Tory i oglądalność wyścigów

Cel: Analiza atrakcyjności poszczególnych torów na podstawie danych oglądalności.

Zakres:

- łączna i średnia oglądalność w sezonie,
- mapa torów wyścigowych zawierająca:
 - długość toru,
 - datę nadchodzącego Grand Prix,
 - link do oficjalnej strony toru.

4. Terminarz wyścigów

Cel: Ułatwienie śledzenia przebiegu sezonu dla kibiców oraz użytkowników komercyjnych.

Zakres:

- liczba zakończonych i planowanych wyścigów w sezonie,
- szczegóły dotyczące najbliższego wyścigu (data, lokalizacja),
- mapa torów wraz z podziałem geograficznym (kontynenty),
- wyniki ostatniego rozegranego wyścigu.

5. Przegląd zespołów

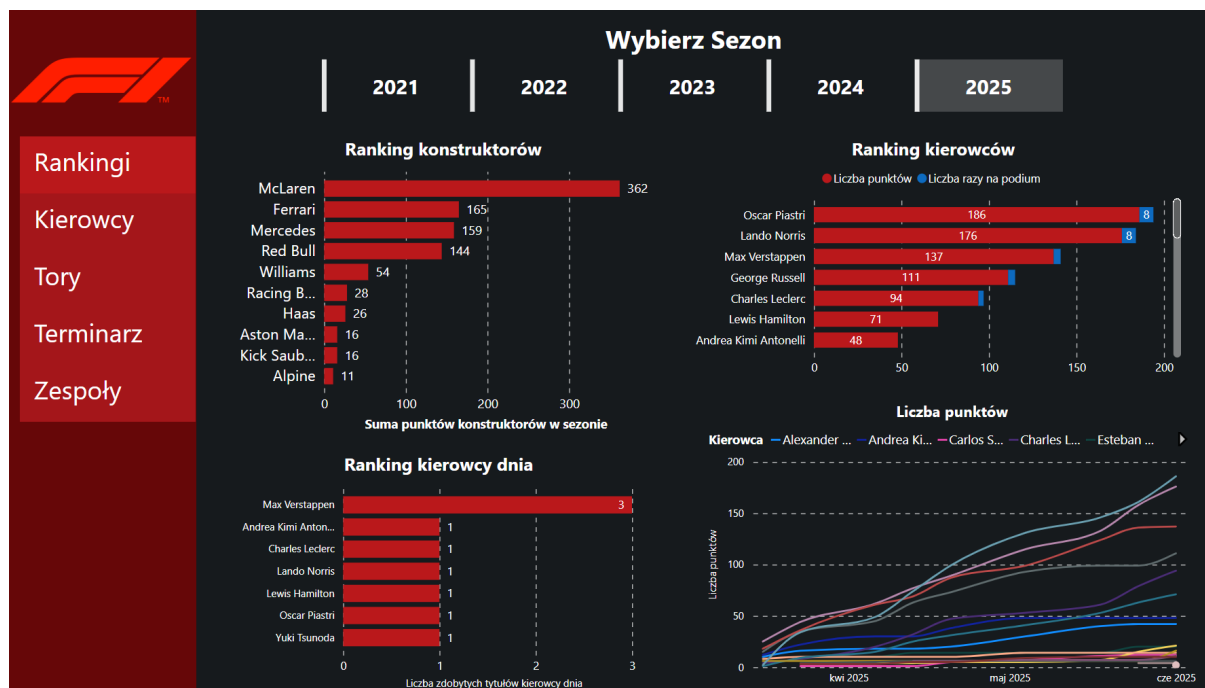
Cel: Przedstawienie danych technicznych bolidów oraz efektywności operacyjnej zespołów.

Zakres:

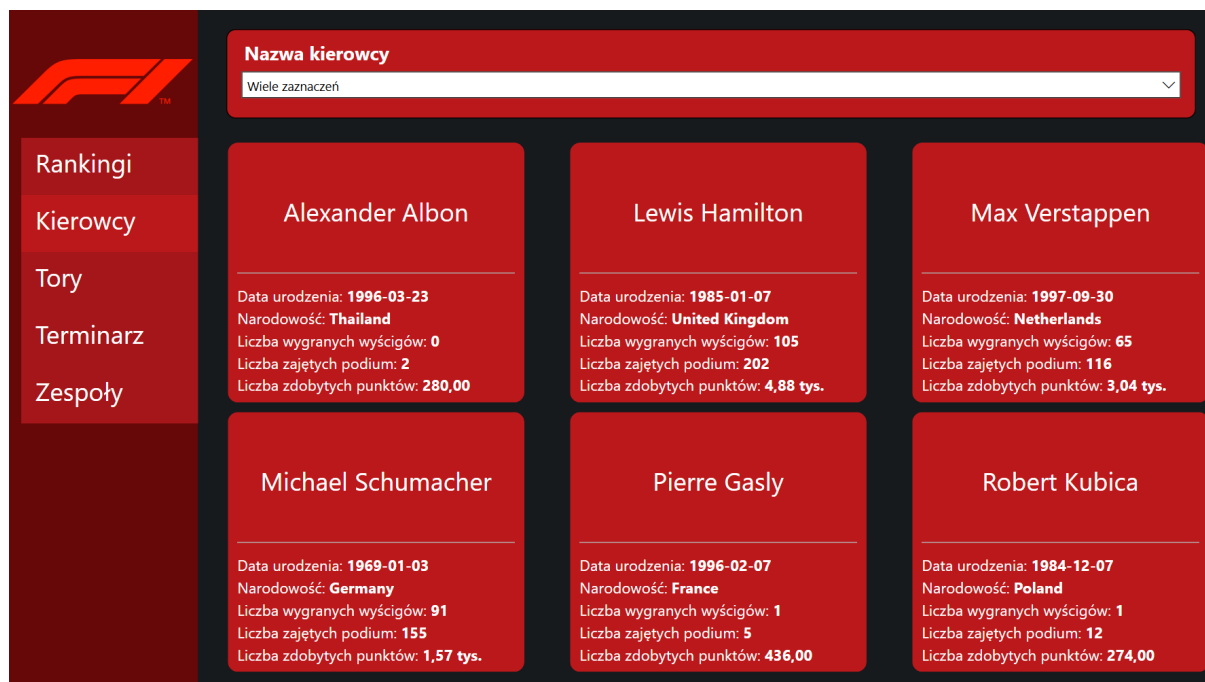
- szczegóły techniczne dotyczące podwozia i silnika bolidu,
- wizualizacja modelu bolidu,
- wykresy czasu pit-stopów z podziałem na zespoły i wyścigi.

9 Wizualizacja raportu

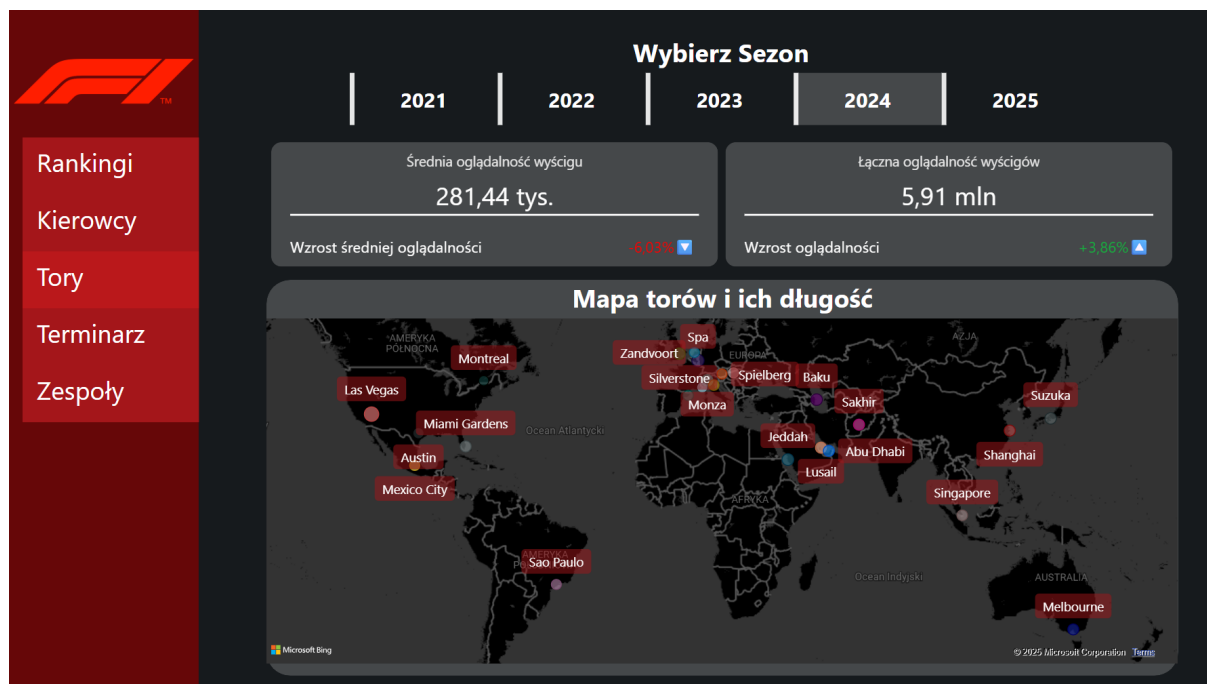
Poniżej znajdują się zdjęcia ilustrujące wygląd raportu w *Power BI*.



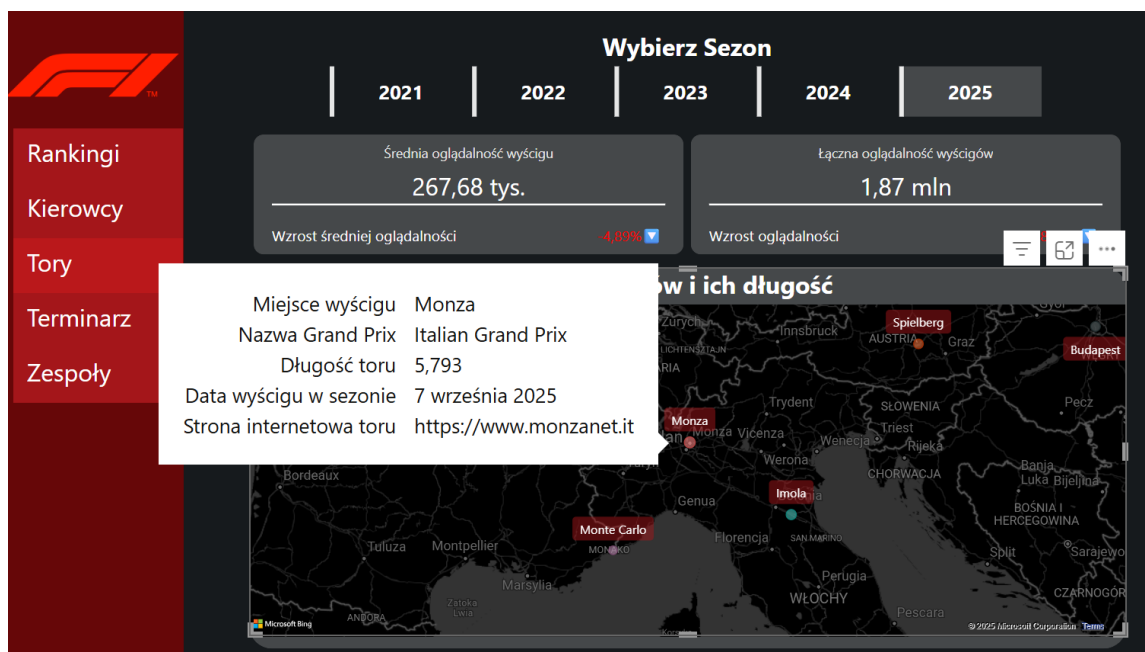
Rysunek 7: Strona przedstawiająca aktualne rankingi kierowców i zespołów, wraz z wizualizacją zmian punktacji w trakcie sezonu oraz analizą popularności wśród fanów.



Rysunek 8: Widok umożliwiający przegląd i porównanie kierowców, ich statystyk, narodowości, liczby zwycięstw i miejsc na podium.



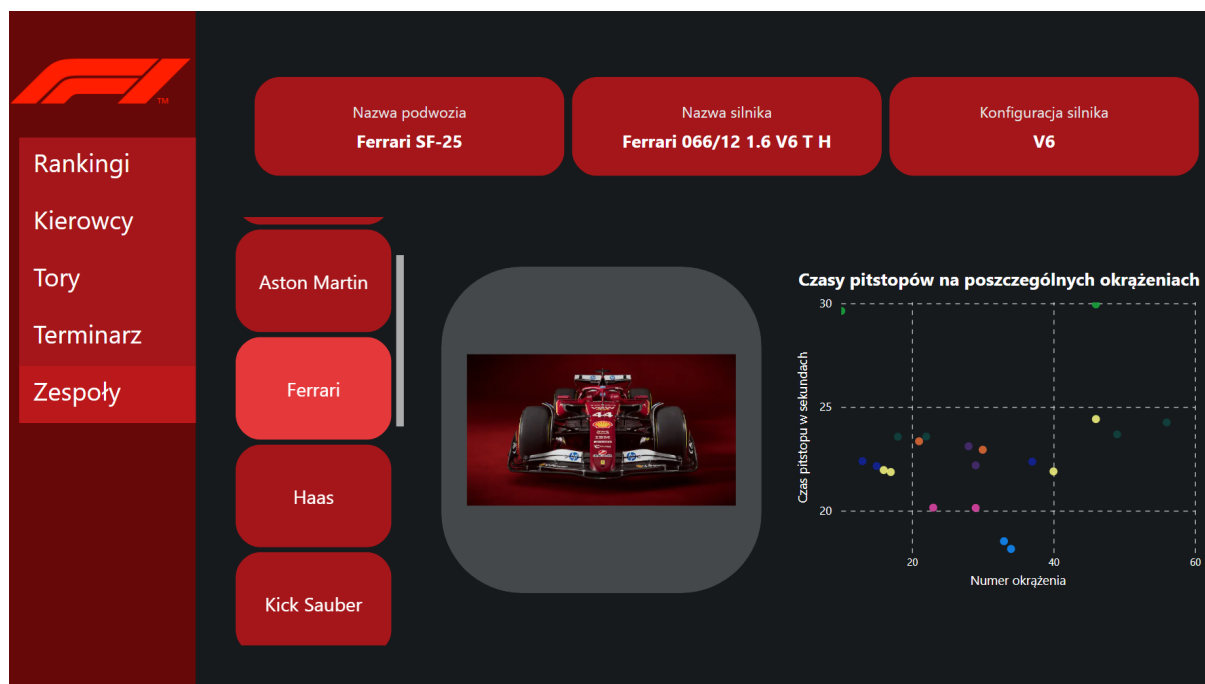
Rysunek 9: Strona z mapą torów wyścigowych i szczegółami dotyczącymi długości torów, nadchodzących wydarzeń oraz statystyk oglądalności.



Rysunek 10: Strona z mapą torów wyścigowych i szczegółami dotyczącymi długości torów, nadchodzących wydarzeń oraz statystyk oglądalności.



Rysunek 11: Zestawienie zakończonych i planowanych wyścigów w sezonie, uzupełnione mapą lokalizacji i wynikami ostatniego Grand Prix.



Rysunek 12: Prezentacja danych technicznych bolidów, wizualizacji modeli oraz analiz czasów pit-stopów dla każdego zespołu.

10 Podsumowanie rezultatów projektu

Zrealizowany projekt budowy hurtowni danych dla środowiska Formuły 1 dostarczył kompleksowe i skalowalne rozwiązanie wspierające analizę wyników sportowych, planowanie strategiczne oraz komunikację z interesariuszami.

Na podstawie przeprowadzonych testów oraz przykładowych raportów można wskazać szereg korzyści biznesowych wynikających z wdrożenia systemu:

- **Wzrost efektywności analizy danych:** Integracja danych z różnych źródeł (wyniki wyścigów, statystyki kierowców, oglądalność, dane o torach) umożliwiła tworzenie przekrojowych raportów wspomagających decyzje analityków i strategów zespołów.
- **Usprawnienie raportowania:** Zcentralizowana warstwa raportowa pozwala na automatyzację procesu tworzenia raportów dla sponsorów, federacji oraz mediów, co znacząco skraca czas przygotowania analiz sezonowych i bieżących.
- **Lepsze planowanie strategiczne:** Dzięki wdrożeniu zaawansowanych transformacji danych i wskaźników (np. rankingów, wzrostów/spadków oglądalności, formy kierowców), możliwe jest podejmowanie trafniejszych decyzji dotyczących składu zespołów, logistyki czy marketingu.
- **Wsparcie zaangażowania kibiców:** Raporty prezentujące terminarz, lokalizacje torów, dane historyczne oraz nadchodzące wyścigi umożliwiają tworzenie atrakcyjnych i angażujących treści dla fanów motorsportu.
- **Elastyczność i skalowalność:** Zastosowana architektura hurtowni danych pozwala na łatwe rozszerzenie systemu o dodatkowe źródła danych oraz nowe raporty.

Pod względem biznesowym projekt spełnił swoje cele, zapewniając wartość dodaną zarówno dla użytkowników operacyjnych, jak i decyzyjnych. Przyjęte rozwiązania wspierają rozwój analityki w organizacji oraz budowanie przewagi konkurencyjnej w sektorze sportów motorowych.

11 Testy

11.1 Testowanie ręczne z Prefect Deployment

W ramach testów, osoba odpowiedzialna za nadzór powinna okresowo monitorować tablicę z wynikami, aby upewnić się, że wszystkie zadania wykonują się zgodnie z planem. Dodatkowo, powiadomienia mailowe zostały skonfigurowane, aby informować zespół o statusie zadań.

11.2 Proces Deploymentu

Deployment w systemie jest wykonywany na podstawie kilku zadań (tasks), które są uruchamiane zgodnie z harmonogramem. Każdy task jest odpowiedzialny za wykonanie konkretnego etapu procesu, np. klonowanie repozytorium, instalowanie zależności, uruchamianie skryptów ELT, itp.

Harmonogram uruchamiania jest zarządzany przez Prefect, który obsługuje wszystkie zadania (tasks) w ramach deploymentu, zgodnie z poniższymi ustawieniami.

11.3 Zadania w deploymentach

11.3.1 test-flow

- **Cel:** Testowanie połączeń oraz sprawdzenie poprawności uruchamiania zadań w systemie.
- **Harmonogram:** Ręczne uruchomienie zadania.
- **Opis:** Task testowy uruchamiany ręcznie w celu weryfikacji, czy wszystkie połączenia (np. do bazy danych, repozytorium GitHub) działają prawidłowo.
- **Kroki:**
 - Klonowanie repozytorium z GitHub.
 - Instalowanie wymaganych zależności.
 - Uruchomienie testów (np. testów jednostkowych lub połączeń).

11.3.2 racing-circuits

- **Cel:** Uruchomienie procesu ETL dla danych o torach wyścigowych.
- **Harmonogram:** Uruchamiane co miesiąc o godzinie 3:00 w pierwszym dniu miesiąca.
- **Opis:** Pobieranie danych o torach wyścigowych, przetwarzanie ich i ładowanie do systemu.
- **Cron:** 0 3 1 * * (co miesiąc, 3:00 AM, 1 dnia)

11.3.3 fl_attendance

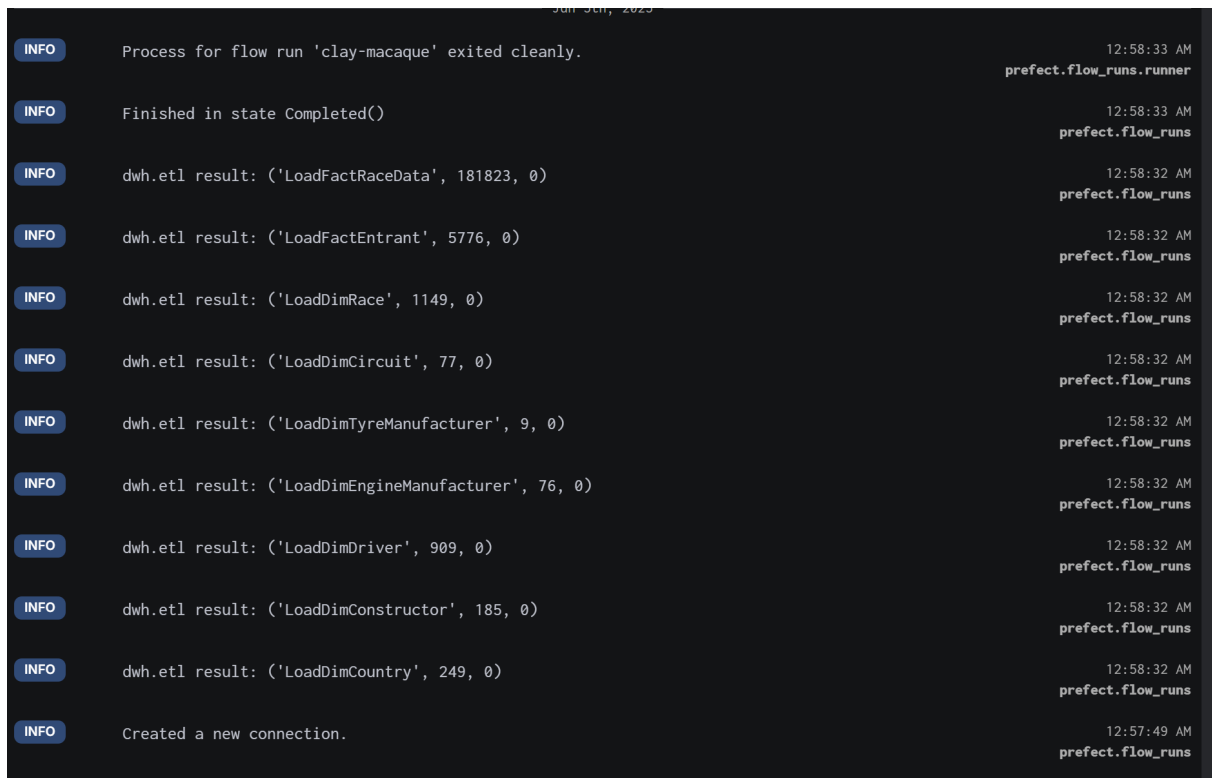
- **Cel:** Uruchomienie procesu ETL dla danych o obecności.
- **Harmonogram:** Uruchamiane co poniedziałek o godzinie 3:00.
- **Opis:** Pobieranie danych o obecności kibiców i ich wprowadzenie do bazy danych.
- **Cron:** 0 3 * * 1 (co tydzień, poniedziałek, 3:00 AM)

11.3.4 fldb

- **Cel:** Przesyłanie danych do bazy danych F1.
- **Harmonogram:** Uruchamiane co poniedziałek o godzinie 3:00.
- **Opis:** Proces ETL do przesyłania danych do bazy danych.
- **Cron:** 0 3 * * 1 (co tydzień, poniedziałek, 3:00 AM)

11.4 DWH

- **Cel:** Przeprowadzenie procesu ETL dla danych z naszej bazy F1DB do naszej hurtowni danych F1DWH.
- **Harmonogram:** Uruchamiane co poniedziałek o godzinie 5:00.
- **Opis:** Proces ETL danych z bazy do hurtowni danych.
- **Cron:** 0 5 * * 1 (co tydzień, poniedziałek, 5:00 AM)
- **Test:** Test w postaci logów czy dane zostały załadowane w poprawny sposób.



Rysunek 13: Logi dla tego procesu

Deployments					
4 Deployments					
		Search deployments...		All tags	A to Z
Deployment	Status	Activity	Tags	Schedules	
<input type="checkbox"/> f1_attendance elt_f1_attendance	Ready	web	At 03:00 AM, only on Monday	
<input type="checkbox"/> f1db elt_f1db	Ready	sql	At 03:00 AM, only on Monday	
<input type="checkbox"/> racing-circuits elt_circuits	Ready	web	+1	
<input type="checkbox"/> test-flow test_deployment	Ready	test	+1	

Rysunek 14: Zaplanowane procesy

11.5 Testowanie jakościowe kodu

Aby zapewnić, że system będzie rozwijalny, łatwy w utrzymaniu i niezawodny, przeprowadzono również testy jakościowe kodu. Testy te koncentrują się na trzech kluczowych aspektach: poprawności typów danych, formatowaniu kodu oraz dokumentacji. Poniżej przedstawiono szczegóły tych testów.

11.5.1 Testowanie poprawności typów danych

Testy sprawdzają, czy kod gwarantuje poprawność typów danych oraz unika niejawnych rzutowań (castów), które mogłyby prowadzić do błędów w działaniu systemu. Kluczowe kroki w tym procesie to:

- **Sprawdzanie deklaracji typów:** Weryfikacja, czy zmienne są odpowiednio zadeklarowane z właściwymi typami danych. Na przykład, zmienne przechowujące liczby całkowite powinny być deklarowane jako `int`, a zmienne przechowujące dane tekstowe jako `str`.
- **Brak niejawnych rzutowań:** Zapewnienie, że w kodzie nie występują niejawne rzutowania danych, takie jak konwersja zmiennych typu `str` do `int` czy `float` bez jawnej weryfikacji poprawności. Każda konwersja powinna być wyraźnie zdefiniowana w kodzie, aby uniknąć nieoczekiwanych błędów.

11.5.2 Testowanie formatowania kodu

Testy formatowania kodu mają na celu zapewnienie spójności i zgodności przy dalszym rozwoju hurtowni.

11.6 Testy z wykorzystaniem constraints

W ramach testowania bazy danych wykorzystaliśmy system constraintów w celu weryfikacji poprawności wprowadzanych danych.

11.6.1 1. Przykłady constraintów na bazie danych

Przykład constraintu na bazie danych:

- **Constraint UNIQUE:** Zapewnienie, że wartości w określonej kolumnie są unikalne. Na przykład, w tabeli krajów, nazwa kraju musi być unikalna

```
UniqueConstraint("name", name="uq_country_name")
```

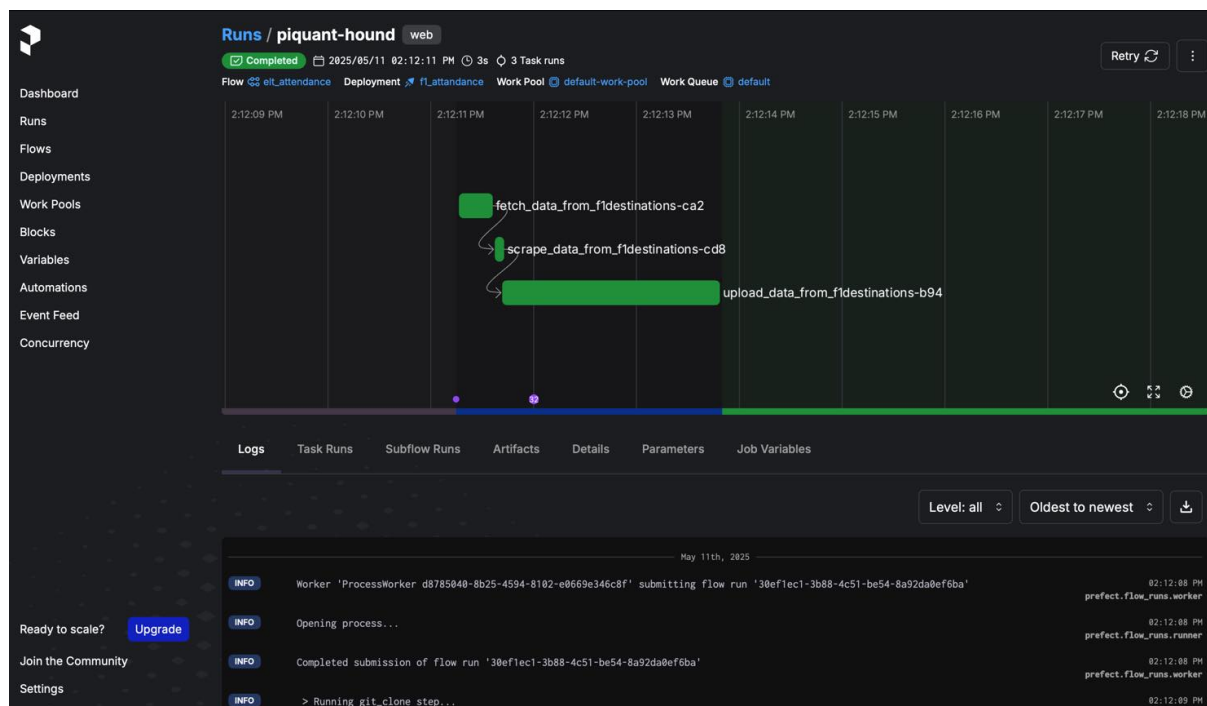
- **Constraint CHECK:** Weryfikacja, że wartości w kolumnie spełniają określony warunek. Na przykład, pozycja w wyścigu musi być większa, bądź równa 1 i nie może być NULL.

```
CheckConstraint(  
    "position_number IS NULL OR position_number >= 1",  
    name="check_sem_position_number_min",  
)
```

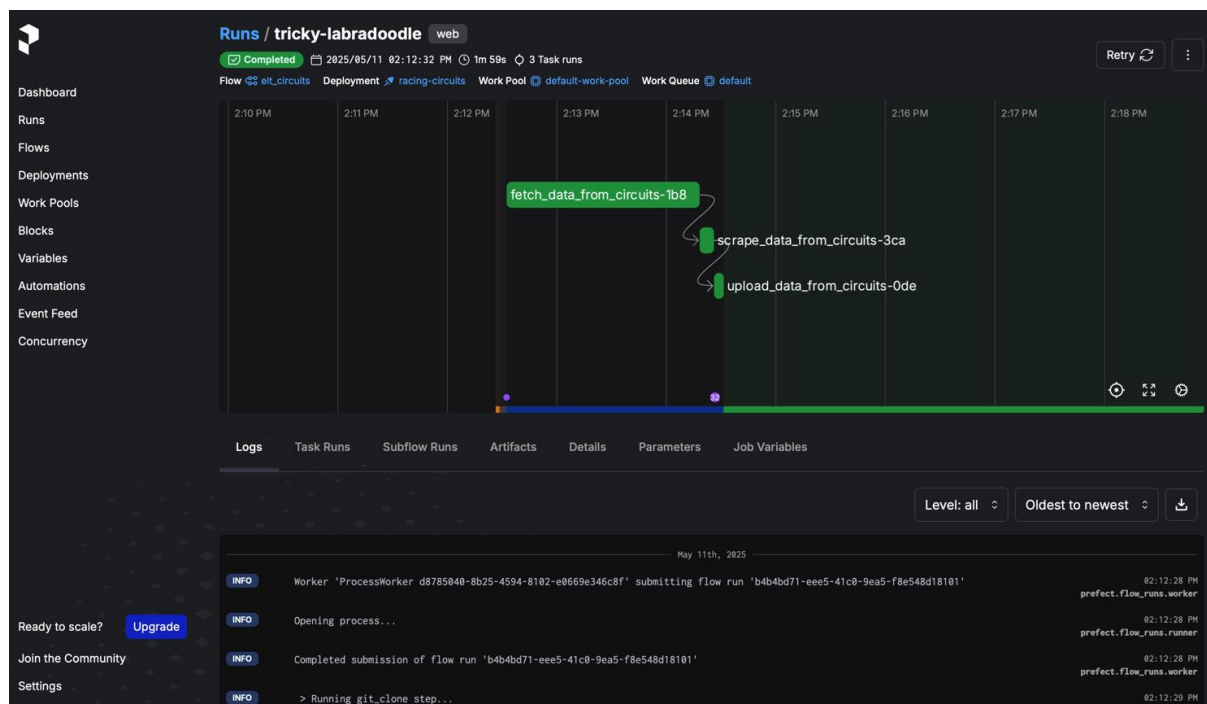
11.7 Testowanie Scrappingu

Podczas procesu scrappingu danych, kluczowe jest, aby wszystkie dane były zbierane w oczekiwanym formacie i z odpowiednimi typami danych. Każdy scrapowany element jest walidowany pod kątem oczekiwanego typu danych oraz struktury.

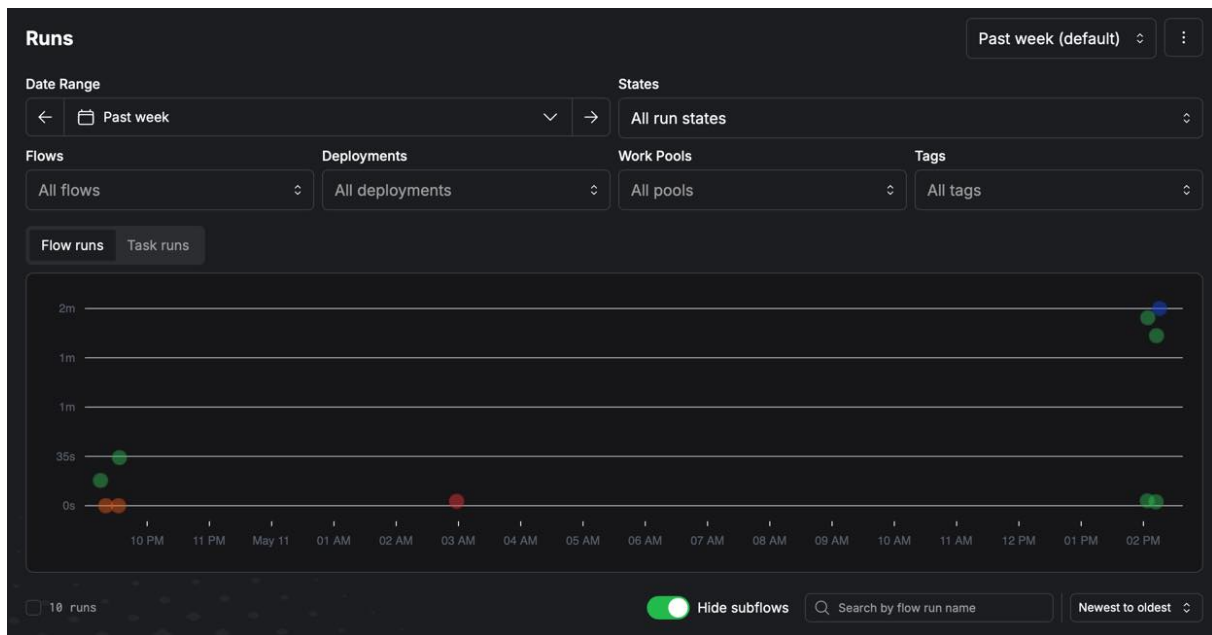
11.8 Zdjęcia przedstawiające działanie



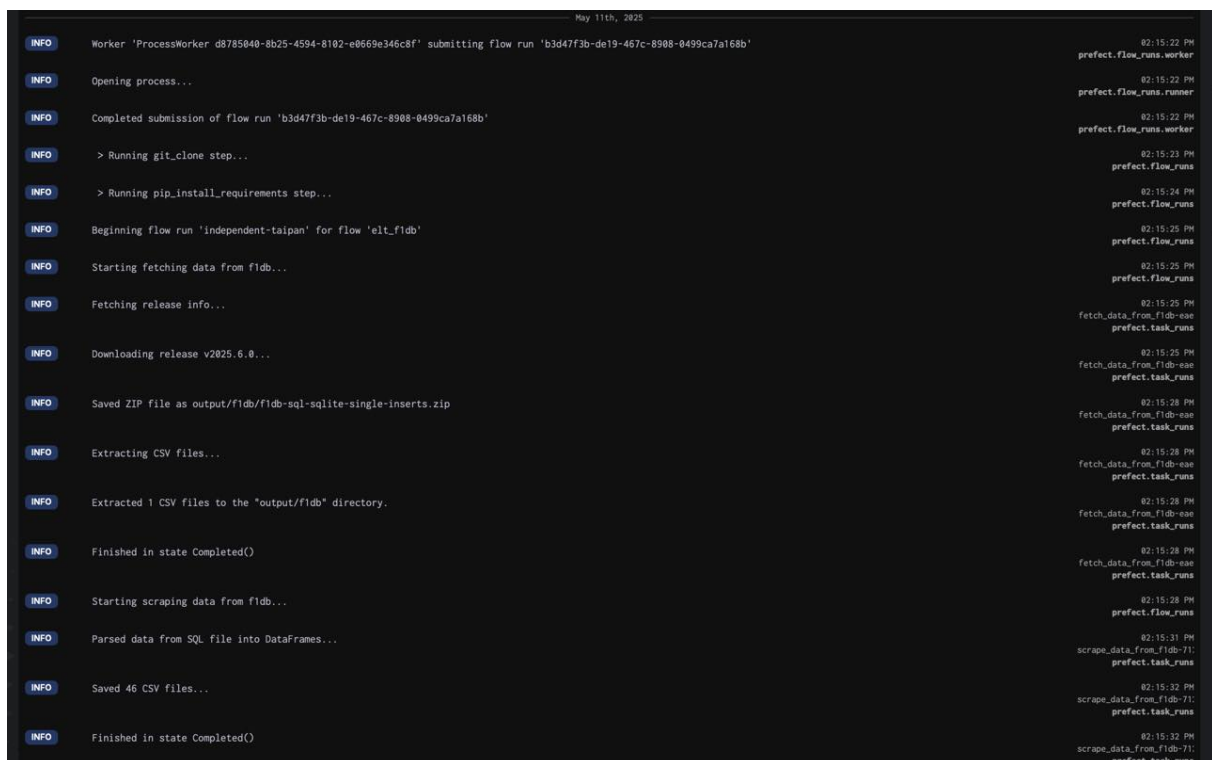
Rysunek 15: Proces zbierania danych dotyczących liczby osób na poszczególnych wyścigach



Rysunek 16: Proces zbierania danych dotyczących torów wyścigowych



Rysunek 17: Przeprowadzone procesy



Rysunek 18: Przykład logów podczas ładowania danych

11.9 Test ponownego wczytywania danych do bazy danych.

W przypadku ponownego włączenia procesu ładującego dane do bazy w celu uaktualnienia danych po ostatnich wyścigach widzimy, że zostały dodane tylko ostatnie rekordy związane z ostatnim wyścigiem.

- *Cel:* Testowanie poprawności aktualizacji danych w bazie F1

- *Sposób*: Testy są wykonane za pomocą sprawdzenia logów z Prefecta oraz wyświetleniu liczby wierszy przed wczytaniem danych oraz po wczytaniu i sprawdzenia.
- *Oczekiwany wynik*: Liczba wierszy wczytanych w logu powinna być zgodna z liczbą wierszy po aktualizacji, liczba wierszy modyfikowanych plus wierszy przed aktualizacją jest równa liczbie wierszy po aktualizacji.
- *Potwierdzenie*:

The screenshot shows a SQL query in a text editor: `SELECT COUNT(*) AS LiczbaRekordów FROM [F1DB].[f1db].[race_data];`. Below the query, there is a progress bar at 74% and two tabs: 'Results' and 'Messages'. The 'Results' tab is active, showing a table with one row and one column.

	LiczbaRekordów
1	181883

Rysunek 19: Liczba wierszy w tabeli race_data po pierwszym załadowaniu danych

The screenshot shows a SQL query in a text editor: `SELECT COUNT(*) AS LiczbaRekordów FROM [F1DB].[f1db].[race_driver_standing];`. Below the query, there is a progress bar at 74% and two tabs: 'Results' and 'Messages'. The 'Results' tab is active, showing a table with one row and one column.

	LiczbaRekordów
1	20872

Rysunek 20: Liczba wierszy w tabeli race_driver_standing po pierwszym załadowaniu danych

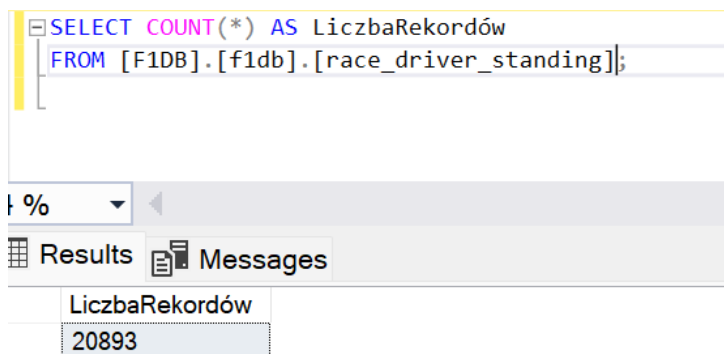
Ładujemy dane po ostatnim wyścigu. Poniżej widzimy procedurę ładowania oraz wyniki po aktualizacji dla tych 2 tabel:

The screenshot shows a log file with several entries. The first entry indicates that the data upload process is completed, with 28893 rows processed and 21 modified. Subsequent entries show the start of the data upload process, adding metadata columns, and uploading data to the race_driver_standing table.

INFO	Message	Timestamp
INFO	Data upload process completed. 28893 rows processed, 21 modified.	04:38:25 PM
INFO	Starting data upload process...	upload_data_from_f1db-a0-prefect.task_runs
INFO	Adding metadata columns...	04:36:58 PM
INFO	Uploading data to race_driver_standing (0)...	upload_data_from_f1db-a0-prefect.task_runs

Rysunek 21: Pokaz logów z ładowania danych dla tabeli race_driver_standing

Widzimy, że zostało załadowane 20893 z czego 21 zmodyfikowanych oznaczających zmianę rankingu po wyścigu co by się zgadzało.

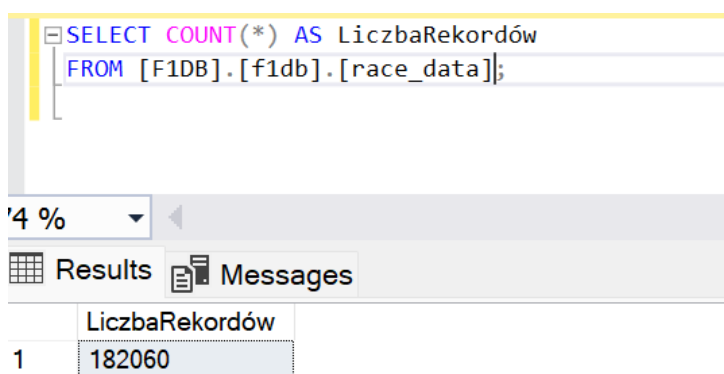


```
SELECT COUNT(*) AS LiczbaRekordów
FROM [F1DB].[f1db].[race_driver_standing];
```

	LiczbaRekordów
	20893

Rysunek 22: Liczba wierszy w tabeli race_driver_standing po aktualizacji z ostatniego wyścigu

Widzimy że liczba wierszy się zgadza. Poniżej sprawdzamy czy dane dla race_data też załadowały się poprawnie.



```
SELECT COUNT(*) AS LiczbaRekordów
FROM [F1DB].[f1db].[race_data];
```

	LiczbaRekordów
1	182060

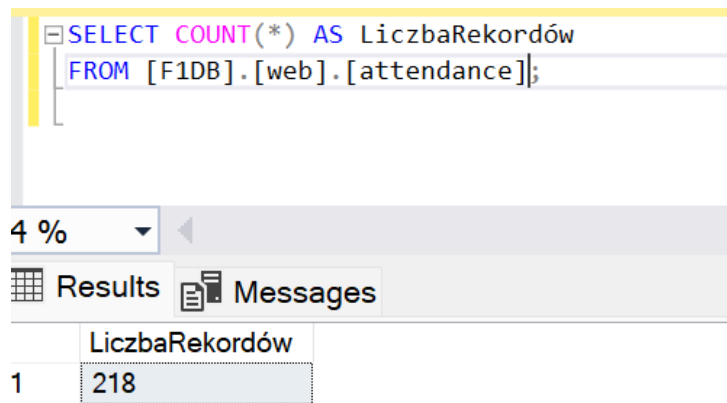
Rysunek 23: Liczba wierszy w tabeli race_data po aktualizacji z ostatniego wyścigu

Widzimy, że dane zostały faktycznie dodane.

11.10 Test ponownego wczytywania danych do bazy danych scrapowanych ze strony z oglądalnością wyścigów oraz ze strony z informacjami torów.

Testy przebiegają identycznie co w przypadku powyżej.

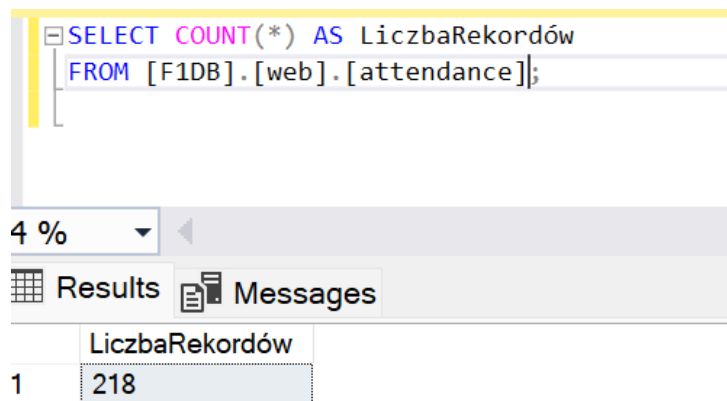
- *Cel:* Testowanie poprawności aktualizacji danych w bazie F1 ze scrapowanych stron.
- *Sposób:* Testy są wykonane za pomocą sprawdzenia logów z Prefecta oraz wyświetleniu liczby wierszy przed wczytaniem danych oraz po wczytaniu i sprawdzenia.
- *Oczekiwany wynik:* Liczba wierszy wczytanych w logu powinna być zgodna z liczbą wierszy po aktualizacji oraz z liczbą wiersz przed, ponieważ strona z wynikiem oglądalności nie została zaktualizowana.
- *Potwierdzenie:*



```
SELECT COUNT(*) AS LiczbaRekordów
FROM [F1DB].[web].[attendance];
```

	LiczbaRekordów
1	218

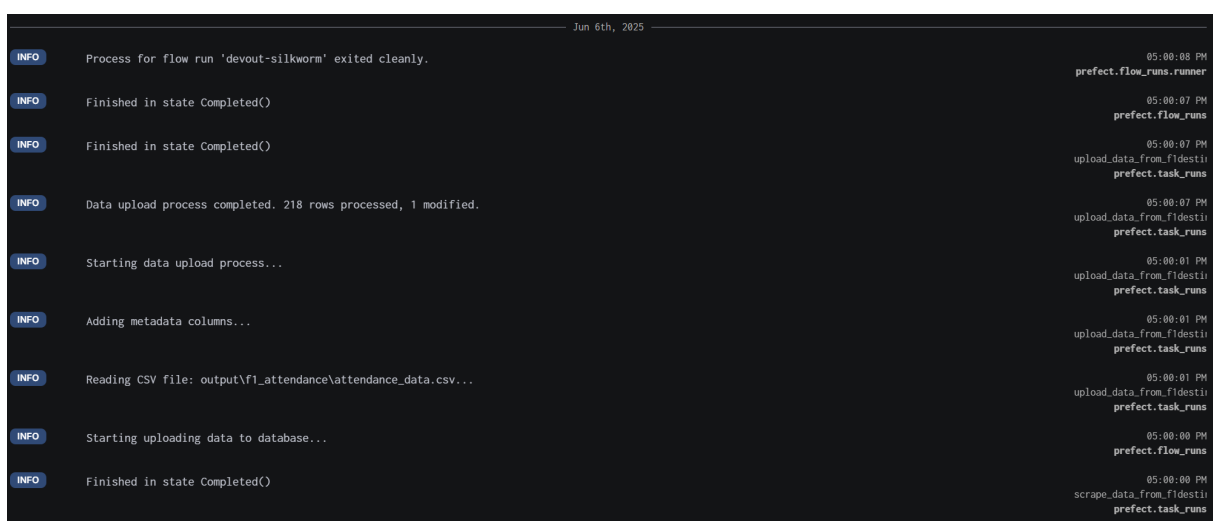
Rysunek 24: Liczba wierszy w tabeli web_attendence przed aktualizacją



```
SELECT COUNT(*) AS LiczbaRekordów
FROM [F1DB].[web].[attendance];
```

	LiczbaRekordów
1	218

Rysunek 25: Liczba wierszy w tabeli web_attendence po aktualizacji



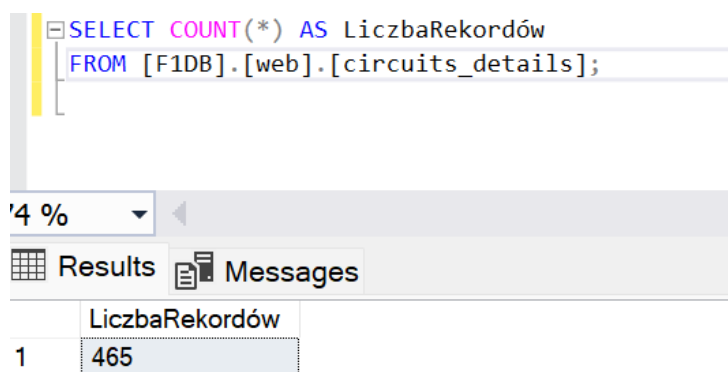
```
Jun 6th, 2025

INFO Process for flow run 'devout-silkworm' exited cleanly.
INFO Finished in state Completed()
INFO Finished in state Completed()
INFO Data upload process completed. 218 rows processed, 1 modified.
INFO Starting data upload process...
INFO Adding metadata columns...
INFO Reading CSV file: output\fl_attendance\attendance_data.csv...
INFO Starting uploading data to database...
INFO Finished in state Completed()
```

05:00:08 PM prefect.flow_runs.runner
05:00:07 PM prefect.flow_runs
05:00:07 PM upload_data_from_f1destii: prefect.task_runs
05:00:07 PM upload_data_from_f1destii: prefect.task_runs
05:00:01 PM upload_data_from_f1destii: prefect.task_runs
05:00:01 PM upload_data_from_f1destii: prefect.task_runs
05:00:01 PM upload_data_from_f1destii: prefect.task_runs
05:00:00 PM prefect.flow_runs
05:00:00 PM scrape_data_from_f1destii: prefect.task_runs

Rysunek 26: Logi z Prefecta podczas aktualizacji

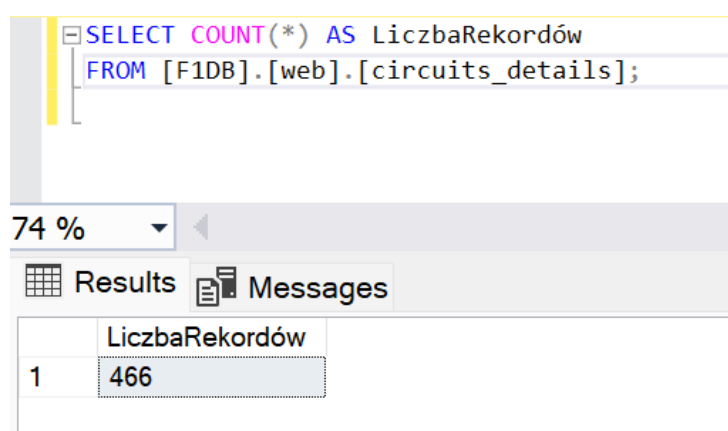
W przypadku *circuit_details* robimy to samo:



The screenshot shows a SQL query in a text editor: `SELECT COUNT(*) AS LiczbaRekordów FROM [F1DB].[web].[circuits_details];`. Below the query, a progress bar is at 4%. The 'Results' tab is active, displaying a table with one row and one column.

	LiczbaRekordów
1	465

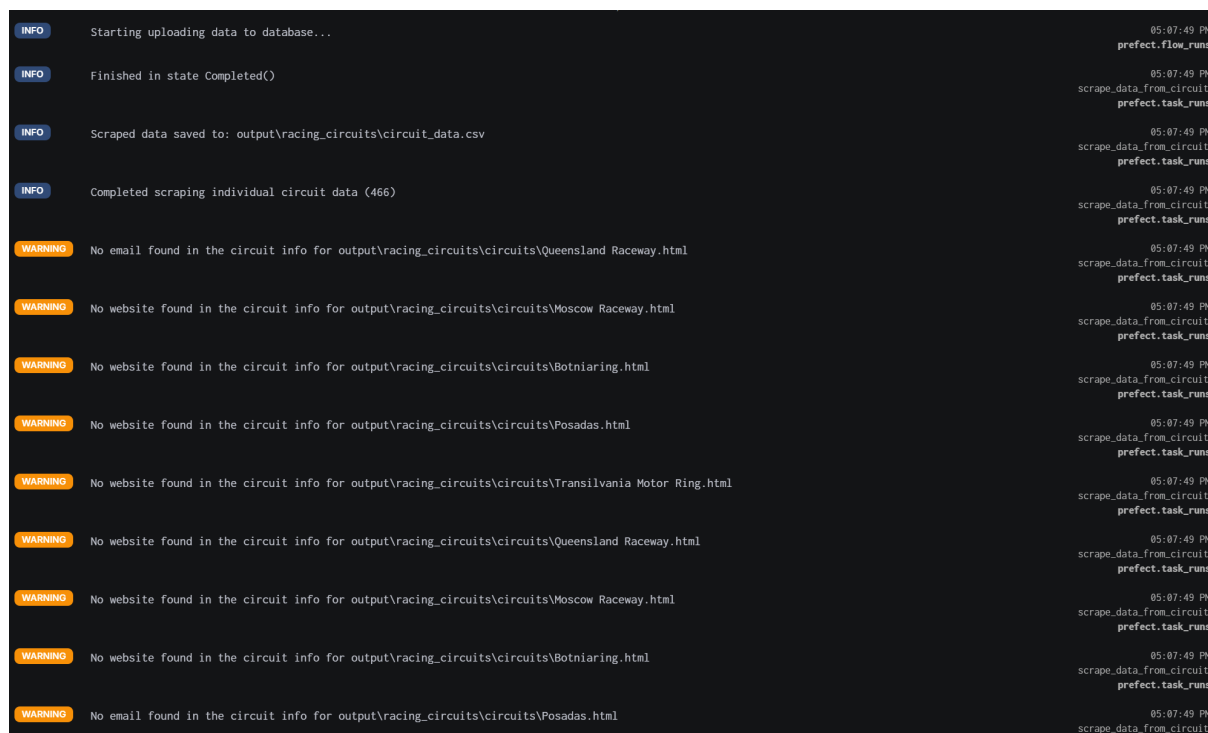
Rysunek 27: Liczba wierszy w tabeli web_circuit_details przed aktualizacją



The screenshot shows the same SQL query as in Figure 27. The progress bar is now at 74%. The 'Results' tab is active, displaying a table with one row and one column.

	LiczbaRekordów
1	466

Rysunek 28: Liczba wierszy w tabeli web_circuit_details po aktualizacji

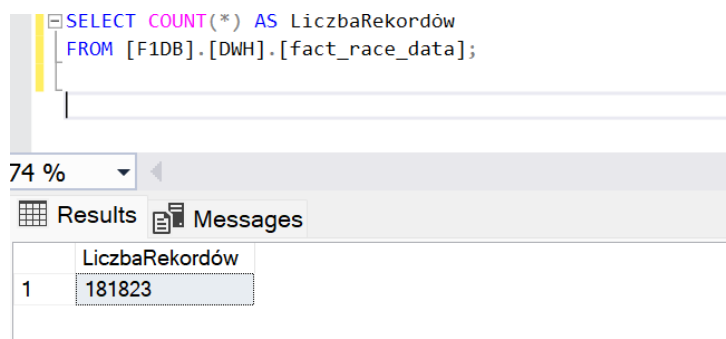


Rysunek 29: Logi z Prefecta podczas aktualizacji

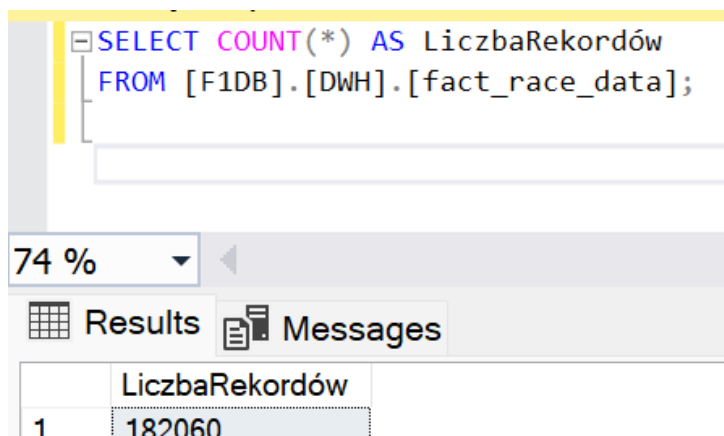
W logach widzimy ostrzeżenia, że w przypadku niektórych torów informacje na stronie są niekompletne. Widzimy, że liczba wierszy z logów zgadza się z tym co jest po w SQL, został dodany jeden nowy tor.

11.11 Test ponownego procesu ETL z bazy do hurtowni danych.

- *Cel:* Testowanie poprawności aktualizacji danych w hurtowni F1 po procesie ETL.
- *Sposób:* Testy są wykonane za pomocą sprawdzenia logów z Prefecta oraz wyświetleniu liczby wierszy przed wczytaniem danych oraz po wczytaniu i sprawdzenia.
- *Oczekiwany wynik:* Liczba nowych wierszy w logu plus liczba wierszy w hurtowni przed aktualizacją powinna być równa liczbie wierszy po aktualizacji.
- *Potwierdzenie:*



Rysunek 30: Liczba wierszy w tabeli DW_fact_race_data przed aktualizacją



The screenshot shows a SQL query execution window. The query is: `SELECT COUNT(*) AS LiczbaRekordów FROM [F1DB].[DWH].[fact_race_data];`. Below the query, there is a progress bar at 74%. At the bottom, there are two tabs: 'Results' and 'Messages'. The 'Results' tab is active, showing a table with one row and one column.

	LiczbaRekordów
1	182060

Rysunek 31: Liczba wierszy w tabeli DW_fact_race_data po aktualizacją



The screenshot shows a log from Prefect. It contains several 'INFO' messages indicating the completion of ETL tasks. Each message shows the task name, the number of records loaded, and the number of errors. The tasks are: 'LoadFactRaceData', 'LoadFactEntrant', 'LoadDimRace', 'LoadDimCircuit', 'LoadDimTyreManufacturer', 'LoadDimEngineManufacturer', 'LoadDimDriver', 'LoadDimConstructor', and 'LoadDimCountry'.

Task Name	Records Loaded	Errors
LoadFactRaceData	237	0
LoadFactEntrant	1	0
LoadDimRace	0	0
LoadDimCircuit	0	0
LoadDimTyreManufacturer	0	0
LoadDimEngineManufacturer	0	0
LoadDimDriver	0	0
LoadDimConstructor	0	0
LoadDimCountry	0	0

Rysunek 32: Logi z Prefecta podczas aktualizacji

Jaki widać po zsumowaniu wszystko się zgadza. Widzimy też że zostały dodane wiersze tylko w tabelach faktowych gdzie znajdują się nowe dane z ostatniego wyścigu.

11.12 Test zmiany danych, które nie mogą być zmieniane w hurtowni.

- *Cel:* Sprawdzenie czy na pewno nie da się zmienić danych w hurtowni, których nie powinno dać się zmienić
- *Sposób:* Zmieniamy ręcznie dane, których nie powinno dać się zmienić, przeprowadzamy proces ETL i sprawdzamy czy się zmodyfikowały.
- *Oczekiwany wynik:* Testowany wiersz nie uległ zmianie, wyrzuca błąd, że nie da się go zmienić
- *Potwierdzenie:*

```

CREATE PROCEDURE dbo.Update_First_Race_Row
AS
BEGIN
    SET NOCOUNT ON;

    UPDATE TOP (1) [F1DB].[f1db].[race_data]
    SET race_points = 99
    WHERE race_id = (
        SELECT TOP 1 race_id
        FROM [F1DB].[f1db].[race_data]
        ORDER BY race_id ASC
    );
END;

EXEC dbo.Update_First_Race_Row;

```

Rysunek 33: Testowa kwerenda zmieniając wiersz

	ace_gap	race_gap_millis	race_gap_laps	race_interval	race_interval_millis	race_reason_retired	race_points	race_pole_position	race_qualification_position_number	race_qualification_position_text	race_grid_position
1	NULL	NULL	NULL	NULL	NULL	NULL	99.00	NULL	NULL	NULL	NULL
2	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
3	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
4	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
5	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
6	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
7	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Rysunek 34: Rezultat kwerendy

```

ERROR Engine execution exited with unexpected exception
Traceback (most recent call last):
  File "C:\Users\soboc\AppData\Local\Programs\Python\Python313\Lib\site-packages\sqlalchemy\engine\base.py", line 1964, in _exec_single_context
    self.dialect.do_execute(
    ~~~~~^
        cursor, str_statement, effective_parameters, context
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    )
    ^
  File "C:\Users\soboc\AppData\Local\Programs\Python\Python313\Lib\site-packages\sqlalchemy\engine\default.py", line 945, in do_execute
    cursor.execute(statement, parameters)
    ~~~~~^
pyodbc.Error: ('01003', '[01003] [Microsoft][ODBC Driver 18 for SQL Server][SQL Server]Warning: Null value is eliminated by an aggregate or other SET operation. (8153) (SQLExecDirectW); [01003] [Microsoft][ODBC Driver 18 for SQL Server][SQL Server]dwh_hash has changed for one or more records (50000)')

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "C:\Users\soboc\AppData\Local\Programs\Python\Python313\Lib\site-packages\prefect\flow_engine.py", line 1527, in run_flow
    ret_val = run_flow_sync(**kwargs)
  File "C:\Users\soboc\AppData\Local\Programs\Python\Python313\Lib\site-packages\prefect\flow_engine.py", line 1372, in run_flow_sync
    return engine.state if return_type == "state" else engine.result()
    ~~~~~^
  File "C:\Users\soboc\AppData\Local\Programs\Python\Python313\Lib\site-packages\prefect\flow_engine.py", line 350, in result
    raise self._raised
  File "C:\Users\soboc\AppData\Local\Programs\Python\Python313\Lib\site-packages\prefect\flow_engine.py", line 763, in run_context
    yield self
  File "C:\Users\soboc\AppData\Local\Programs\Python\Python313\Lib\site-packages\prefect\flow_engine.py", line 1370, in run_flow_sync
    engine.call_flow_fn()
    ~~~~~^
  File "C:\Users\soboc\AppData\Local\Programs\Python\Python313\Lib\site-packages\prefect\flow_engine.py", line 783, in call_flow_fn
    result = call_with_parameters(self.flow.fn, self.parameters)
  File "C:\Users\soboc\AppData\Local\Programs\Python\Python313\Lib\site-packages\prefect\utilities\callables.py", line 210, in call_with_parameters
    return fn(*args, **kwargs)
  File "C:\Users\soboc\AppData\Local\Temp\tmpguq1hszeprefect\F1-main\src\f1\flows\dwh\elt.py", line 45, in elt
    result = connection.execute(text("EXEC [dwh].[etl]"))
  File "C:\Users\soboc\AppData\Local\Programs\Python\Python313\Lib\site-packages\sqlalchemy\engine\base.py", line 1416, in execute
    return meth(
           self,

```

Rysunek 35: Logi z prefecta. Wyskakuj błąd, że nie da się zmienić tego wiersza.

11.13 Test zmiany danych, które mogą być zmieniane w hurtowni.

- *Cel:* Sprawdzenie czy da się zmienić dane w hurtowni, które mogą być zmienione.
- *Sposób:* Zmieniamy ręcznie dane, których mogą być zmienione, przeprowadzamy proces ETL i sprawdzamy czy się zmodyfikowały.
- *Oczekiwany wynik:* Testowany wiersz uległ zmianie.
- *Potwierdzenie:*

```
UPDATE [F1DB].[f1db].[driver]
SET permanent_number = '1'
WHERE name = 'Michael Schumacher';
```

Rysunek 36: Testowa kwerenda zmieniając wiersz

616	michael-bleekemolen	Michael Bleekemolen	Michael	Bleekemolen	Michael Bleekemolen	BLE	NULL	MALE	1949-10-02	NULL	Amsterdam	netherlands
617	michael-may	Michael May	Michael	May	Michael May	MAY	NULL	MALE	1934-08-18	NULL	Stuttgart	germany
618	michael-schumacher	Michael Schumacher	Michael	Schumacher	Michael Schumacher	MSC	1	MALE	1969-01-03	NULL	Hürth	germany
619	michele-alboreto	Michele Alboreto	Michele	Alboreto	Michele Alboreto	ALB	NULL	MALE	1956-12-23	2001-04-25	Milan	italy

Rysunek 37: Rezultat kwerendy

INFO	Process for flow run 'gabby-fulmar' exited cleanly.	06:01:05 PM	prefect.flow_runs.runner
INFO	Finished in state Completed()	06:01:04 PM	prefect.flow_runs
INFO	dwh.etl result: ('LoadFactRaceData', 0, 0)	06:01:04 PM	prefect.flow_runs
INFO	dwh.etl result: ('LoadFactEntrant', 0, 0)	06:01:04 PM	prefect.flow_runs
INFO	dwh.etl result: ('LoadDimRace', 0, 0)	06:01:04 PM	prefect.flow_runs
INFO	dwh.etl result: ('LoadDimCircuit', 0, 0)	06:01:04 PM	prefect.flow_runs
INFO	dwh.etl result: ('LoadDimTyreManufacturer', 0, 0)	06:01:04 PM	prefect.flow_runs
INFO	dwh.etl result: ('LoadDimEngineManufacturer', 0, 0)	06:01:04 PM	prefect.flow_runs
INFO	dwh.etl result: ('LoadDimDriver', 0, 1)	06:01:04 PM	prefect.flow_runs
INFO	dwh.etl result: ('LoadDimConstructor', 0, 0)	06:01:04 PM	prefect.flow_runs
INFO	dwh.etl result: ('LoadDimCountry', 0, 0)	06:01:04 PM	prefect.flow_runs

Rysunek 38: Logi z prefecta pokazujące 1 modyfikację w dim_driver

```
SELECT *
FROM [F108].[DW].[dim_driver]
WHERE driver_name = 'Michael Schumacher';
```

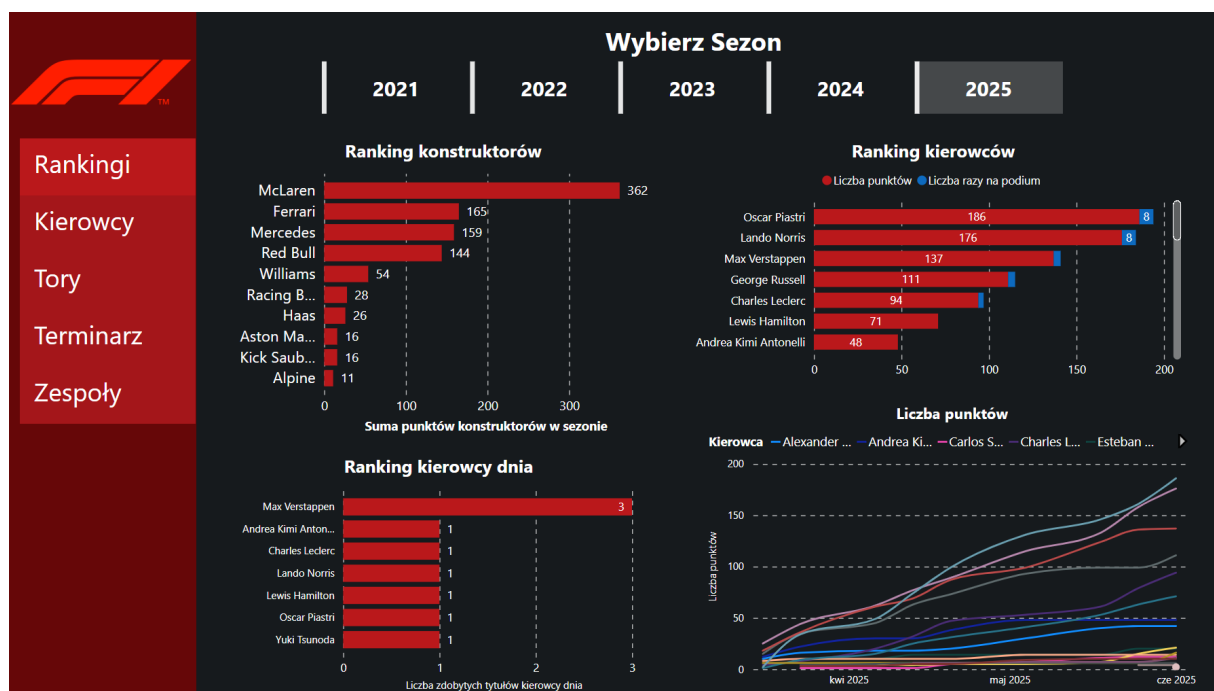
driver_name	driver_first_name	driver_last_name	driver_full_name	driver_abbreviation	driver_permanent_number	driver_gender	driver_date_of_birth	driver_date_of_death	driver_place_of_birth	driver_co
Michael Schumacher	Michael	Schumacher	Michael Schumacher	MSC	1	MALE	1969-01-03	NULL	Hürth	84

Rysunek 39: Wynik po modyfikacji

11.14 Testy warstwy raportowej

1 strona raportu

- *Cel*: Test czy dane na wykresach rysują się prawidłowo
- *Sposób*: Patrzymy co jest na wykresie i wywołamy odpowiednią kwerendę w SQL, żeby potwierdzić zgodność danych
- *Oczekiwany wynik*: Zarówno w raporcie jak i SQL mamy otrzymać te same wartości.
- *Potwierdzenie*:



Rysunek 40: Testowanie wykresu

```

SELECT
    d.driver_full_name,
    SUM(frd.race_data_race_points) AS total_points
FROM
    [F1DB].[DWH].[fact_race_data] frd
JOIN
    [F1DB].[DWH].[dim_race] dr ON frd.race_id = dr.dwh_id
JOIN
    [F1DB].[DWH].[dim_driver] d ON frd.driver_id = d.dwh_id
WHERE
    YEAR(dr.race_date) = 2025
GROUP BY
    d.driver_full_name
ORDER BY
    total_points DESC;

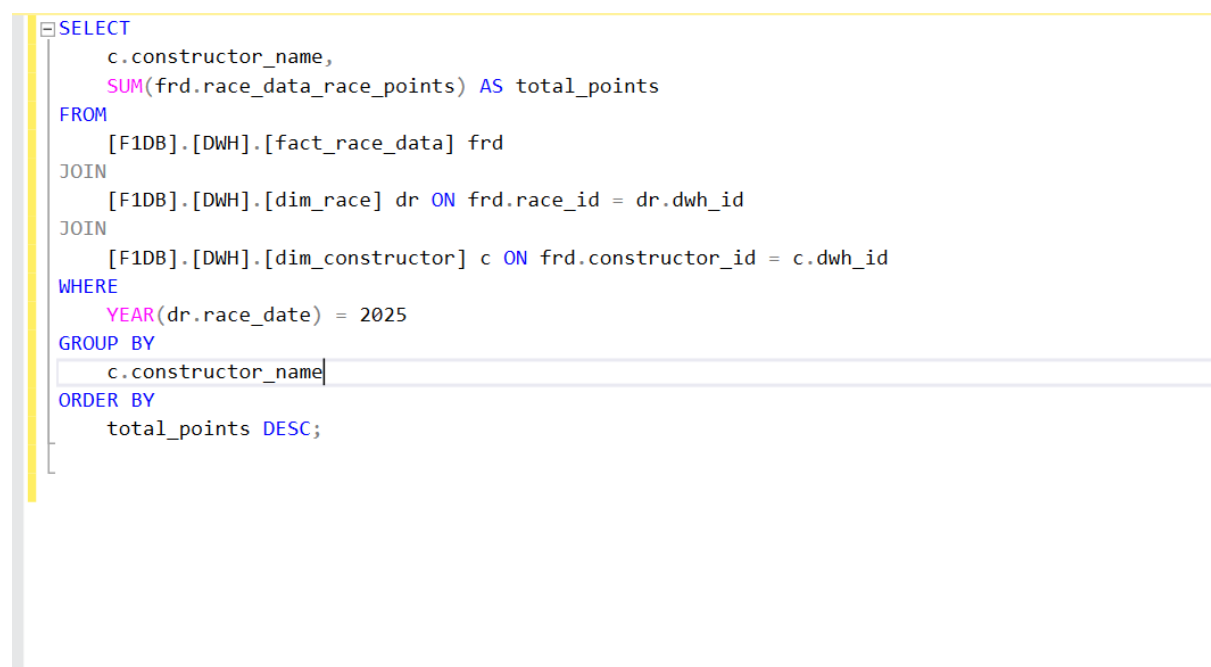
```

4 %

Results Messages

	driver_full_name	total_points
1	Oscar Piastri	186.00
2	Lando Norris	176.00
3	Max Emilian Verstappen	137.00
4	George Russell	111.00
5	Charles Marc Hervé Perceval Leclerc	94.00
6	Lewis Carl Davidson Hamilton	71.00
7	Andrea Kimi Antonelli	48.00
8	Alexander Albon	42.00
9	Isack Alexandre Hadjar	21.00
10	Esteban Ocon	20.00
11	Nicolas Hülkenberg	16.00
12	Lance Stroll	14.00
13	Carlos Sainz Vázquez de Castro	12.00
14	Pierre Gasly	11.00
15	Yuki Tsunoda	10.00
16	Oliver James Bearman	6.00
17	Liam Lawson	4.00
18	Fernando Alonso Díaz	2.00
19	Frederik Vesti Stamm	NULL
20	Jack Doohan	NULL
21	Franco Alejandro Colapinto	NULL
22	Felipe Drugovich Roncato	NULL
23	Victor Martins	NULL

Rysunek 41: Odpowiednia kwerenda



```
SELECT
    c.constructor_name,
    SUM(frd.race_data_race_points) AS total_points
FROM
    [F1DB].[DWH].[fact_race_data] frd
JOIN
    [F1DB].[DWH].[dim_race] dr ON frd.race_id = dr.dwh_id
JOIN
    [F1DB].[DWH].[dim_constructor] c ON frd.constructor_id = c.dwh_id
WHERE
    YEAR(dr.race_date) = 2025
GROUP BY
    c.constructor_name
ORDER BY
    total_points DESC;
```

4 %

Results Messages

	constructor_name	total_points
1	McLaren	362.00
2	Ferrari	165.00
3	Mercedes	159.00
4	Red Bull	144.00
5	Williams	54.00
6	Racing Bulls	28.00
7	Haas	26.00
8	Kick Sauber	16.00
9	Aston Martin	16.00
10	Alpine	11.00

Rysunek 42: Odpowiednia kwerenda

```

SELECT
    d.driver_full_name,
    COUNT(*) AS driver_of_the_day_count
FROM
    [F1DB].[DWH].[fact_race_data] frd
JOIN
    [F1DB].[DWH].[dim_race] dr ON frd.race_id = dr.dwh_id
JOIN
    [F1DB].[DWH].[dim_driver] d ON frd.driver_id = d.dwh_id
WHERE
    YEAR(dr.race_date) = 2025
    AND frd.race_data_race_driver_of_the_day = 1
GROUP BY
    d.driver_full_name
ORDER BY
    driver_of_the_day_count DESC;

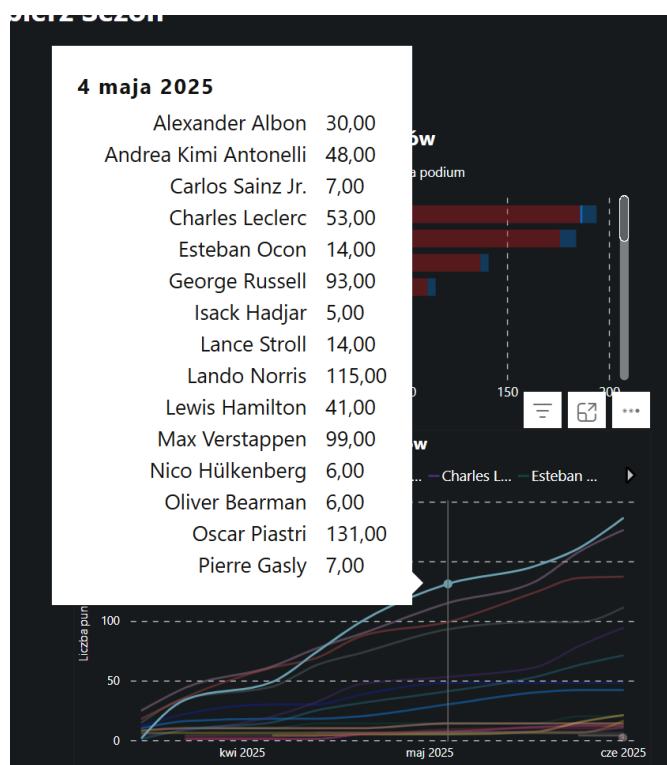
```

74 %

Results Messages

	driver_full_name	driver_of_the_day_count
1	Max Emilian Verstappen	3
2	Oscar Piastri	1
3	Yuki Tsunoda	1
4	Andrea Kimi Antonelli	1
5	Charles Marc Hervé Perceval Leclerc	1
6	Lando Norris	1
7	Lewis Carl Davidson Hamilton	1

Rysunek 43: Odpowiednia kwerenda



Rysunek 44: Test wykresu

```

SELECT
    d.driver_full_name,
    SUM(frd.race_data_race_points) AS total_points
FROM
    [F1DB].[DWH].[fact_race_data] frd
JOIN
    [F1DB].[DWH].[dim_race] dr ON frd.race_id = dr.dwh_id
JOIN
    [F1DB].[DWH].[dim_driver] d ON frd.driver_id = d.dwh_id
WHERE
    dr.race_date BETWEEN '2025-01-01' AND '2025-05-04'
GROUP BY
    d.driver_full_name
ORDER BY
    total_points DESC;

```

74 %

Results Messages

	driver_full_name	total_points
1	Oscar Piastri	131.00
2	Lando Norris	115.00
3	Max Emilian Verstappen	99.00
4	George Russell	93.00
5	Charles Marc Hervé Perceval Leclerc	53.00
6	Andrea Kimi Antonelli	48.00
7	Lewis Carl Davidson Hamilton	41.00
8	Alexander Albon	30.00
9	Esteban Ocon	14.00
10	Lance Stroll	14.00
11	Yuki Tsunoda	9.00
12	Carlos Sainz Vázquez de Castro	7.00
13	Pierre Gasly	7.00
14	Nicolas Hülkenberg	6.00
15	Oliver James Bearman	6.00
16	Isack Alexandre Hadjar	5.00
17	Ayumu Iwasa	NULL

Rysunek 45: Odpowiednia kwerenda

Widzimy, że się wszystko zgadza.

2 strona raportu

Cel i założenia są te same co wyżej.



Rysunek 46: Test wykresu

```

SELECT
  d.driver_full_name,
  c.country_name,
  d.driver_date_of_birth,
  COUNT(CASE
    WHEN frd.race_data_position_number = 1 AND frd.race_data_type = 'RACE_RESULT' THEN 1
  END) AS race_wins,
  COUNT(CASE
    WHEN frd.race_data_position_number IN (1, 2, 3) AND frd.race_data_type = 'RACE_RESULT' THEN 1
  END) AS podiums,
  SUM(CASE
    WHEN frd.race_data_type = 'RACE_RESULT' THEN frd.race_data_race_points
    ELSE 0
  END) AS total_race_points
FROM
  [F1DB].[DWH].[fact_race_data] frd
JOIN
  [F1DB].[DWH].[dim_driver] d ON frd.driver_id = d.dwh_id
JOIN
  [F1DB].[DWH].[dim_country] c ON d.driver_nationality_country_id = c.dwh_id
WHERE
  d.driver_name = 'Robert Kubica'
GROUP BY
  d.driver_full_name, c.country_name, d.driver_date_of_birth;

```

74 %

Results Messages

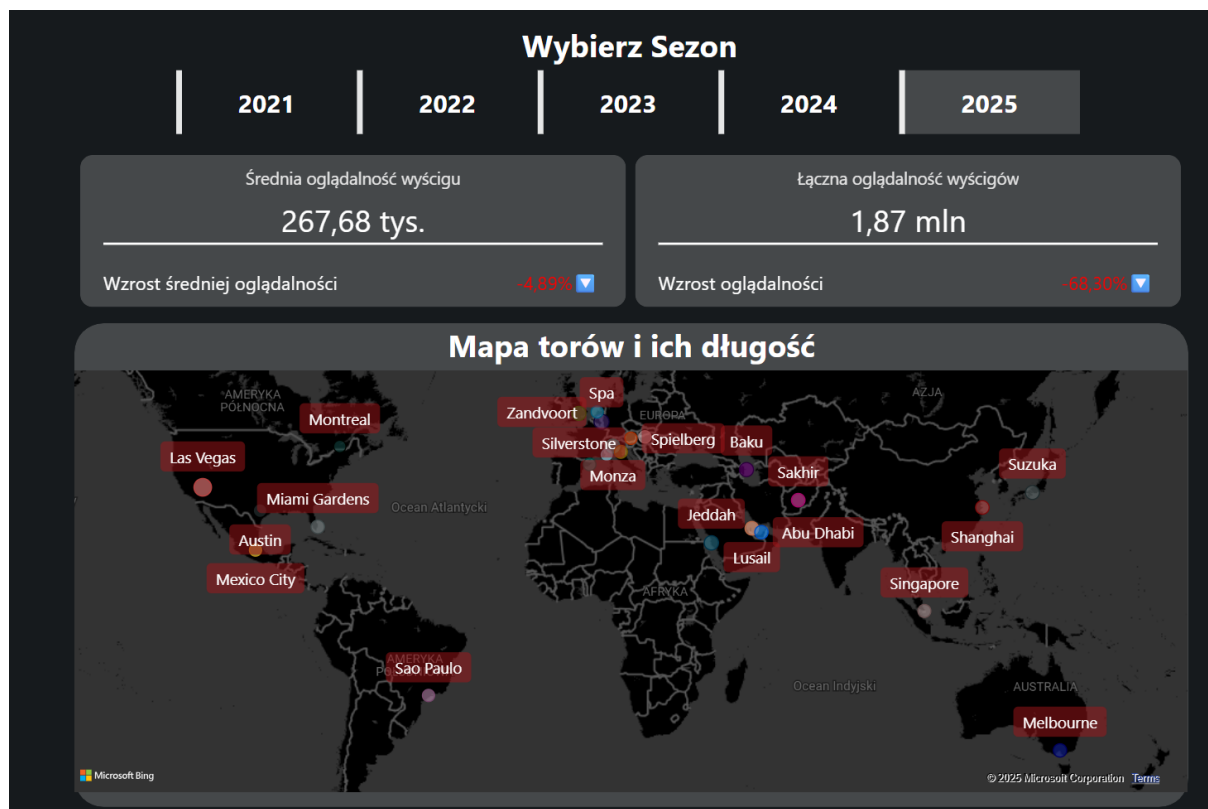
	driver_full_name	country_name	driver_date_of_birth	race_wins	podiums	total_race_points
1	Robert Józef Kubica	Poland	1984-12-07	1	12	274.00

Rysunek 47: Odpowiednia kwerenda

Wszystko zgadza się. Jak wiadomo w czerwcu 2008, podczas Grand Prix Kanady, Robert Kubica odniósł swoje pierwsze i jedyne zwycięstwo w Formule 1, stając się pierwszym Polakiem w historii, który tego dokonał.

3 strona raportu

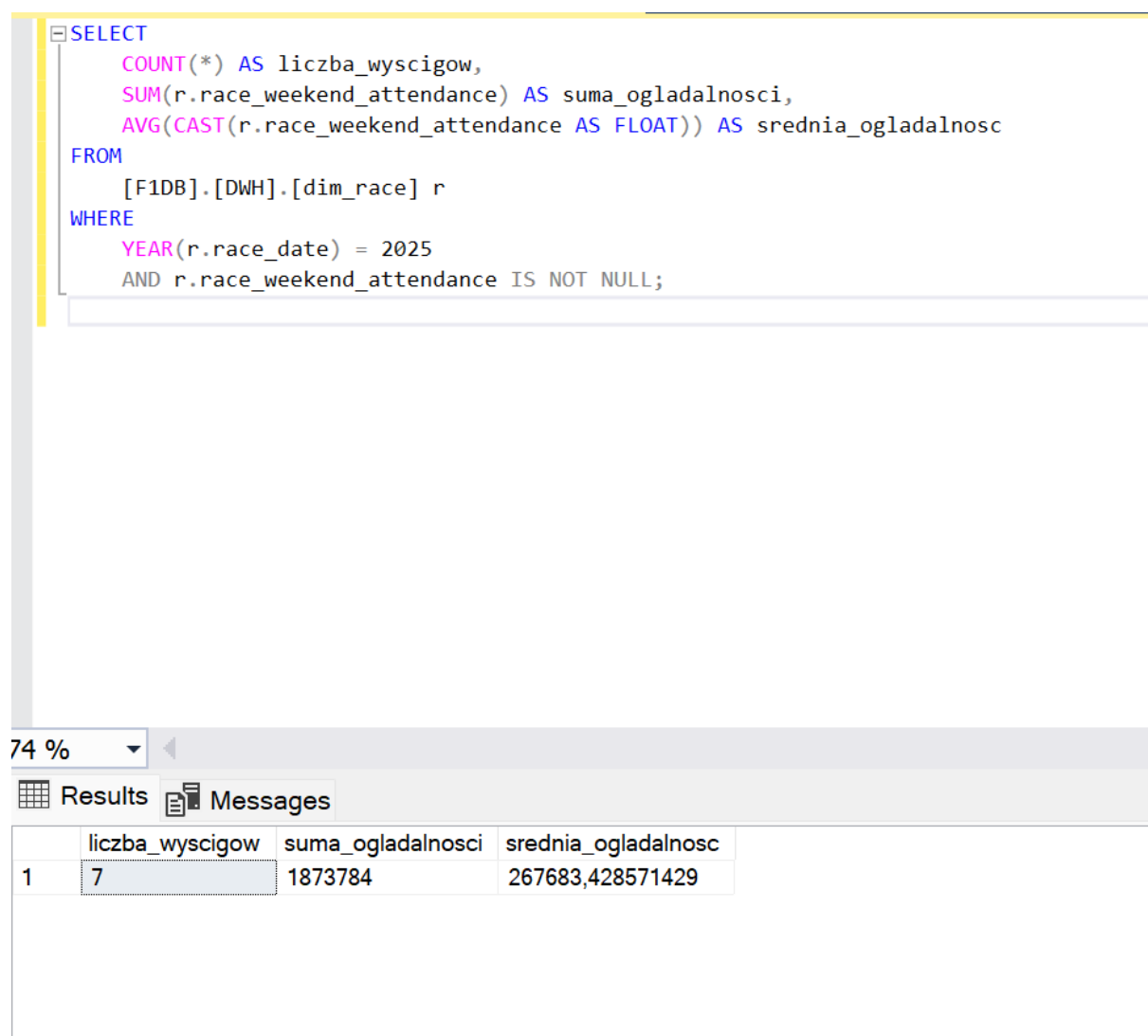
Cel i założenia są te same co wyżej.



Rysunek 48: Test wykresu

<pre> SELECT DISTINCT c.circuit_name, c.circuit_place_name, c.circuit_length, c.circuit_rating, c.circuit_website FROM [F1DB].[DWH].[dim_race] r JOIN [F1DB].[DWH].[dim_circuit] c ON r.circuit_id = c.dwh_id WHERE YEAR(r.race_date) = 2025 ORDER BY c.circuit_name; </pre>					
4 %					
Results Messages					
	circuit_name	circuit_place_name	circuit_length	circuit_rating	circuit_website
1	Americas	Austin	5.513	3,14	http://www.circuitoftheamericas.com
2	Bahrain	Sakhir	5.412	2,97	https://www.bahraingp.com
3	Baku	Baku	6.003	3,24	https://www.bakucitycircuit.com
4	Catalunya	Montmeló	4.657	2,97	https://www.circuitcat.com
5	Enzo e Dino Ferrari	Imola	4.909	2,93	https://www.autodromoimola.it
6	Gilles Villeneuve	Montreal	4.361	3,17	http://www.gpcanada.ca
7	Hermanos Rodríguez	Mexico City	4.304	3,03	https://ahr.notiauto.com
8	Hungaroring	Budapest	4.381	2,99	https://www.hungaroring.hu
9	Jeddah	Jeddah	6.174	3,16	https://www.saudiarabiangp.com
10	José Carlos Pace	Sao Paulo	4.309	3,16	http://www.autodromodeinterlagos.com.br
11	Las Vegas	Las Vegas	6.201	2,32	https://www.f1lasvegasgp.com
12	Lusail	Lusail	5.419	3,15	https://www.lcsc.qa
13	Marina Bay	Singapore	4.940	2,93	https://singaporegp.sg/en
14	Melbourne	Melbourne	5.278	3,04	https://www.grandprix.com.au
15	Miami	Miami Gardens	5.412	2,72	https://f1miamigp.com
16	Monaco	Monte Carlo	3.337	3,15	https://www.acm.mc
17	Monza	Monza	5.793	2,93	https://www.monzanet.it
18	Red Bull Ring	Spielberg	4.318	2,98	https://www.projekt-spielberg.at
19	Shanghai	Shanghai	5.451	3,18	NULL
20	Silverstone	Silverstone	5.891	3,04	https://www.silverstone.co.uk
21	Spa-Francorchamps	Spa	7.004	3,31	https://www.spa-francorchamps.be
22	Suzuka	Suzuka	5.807	2,98	http://www.suzukacircuit.jp
23	Yas Marina	Abu Dhabi	5.281	2,85	https://www.yasmarinacircuit.com
24	Zandvoort	Zandvoort	4.259	3,07	https://www.circuitzandvoort.nl

Rysunek 49: Odpowiednia kwerenda



The screenshot shows a SQL query editor with a yellow header bar. The query is as follows:

```
SELECT
    COUNT(*) AS liczba_wyscigow,
    SUM(r.race_weekend_attendance) AS suma_ogladalnosci,
    AVG(CAST(r.race_weekend_attendance AS FLOAT)) AS srednia_ogladalnosc
FROM
    [F1DB].[DWH].[dim_race] r
WHERE
    YEAR(r.race_date) = 2025
    AND r.race_weekend_attendance IS NOT NULL;
```

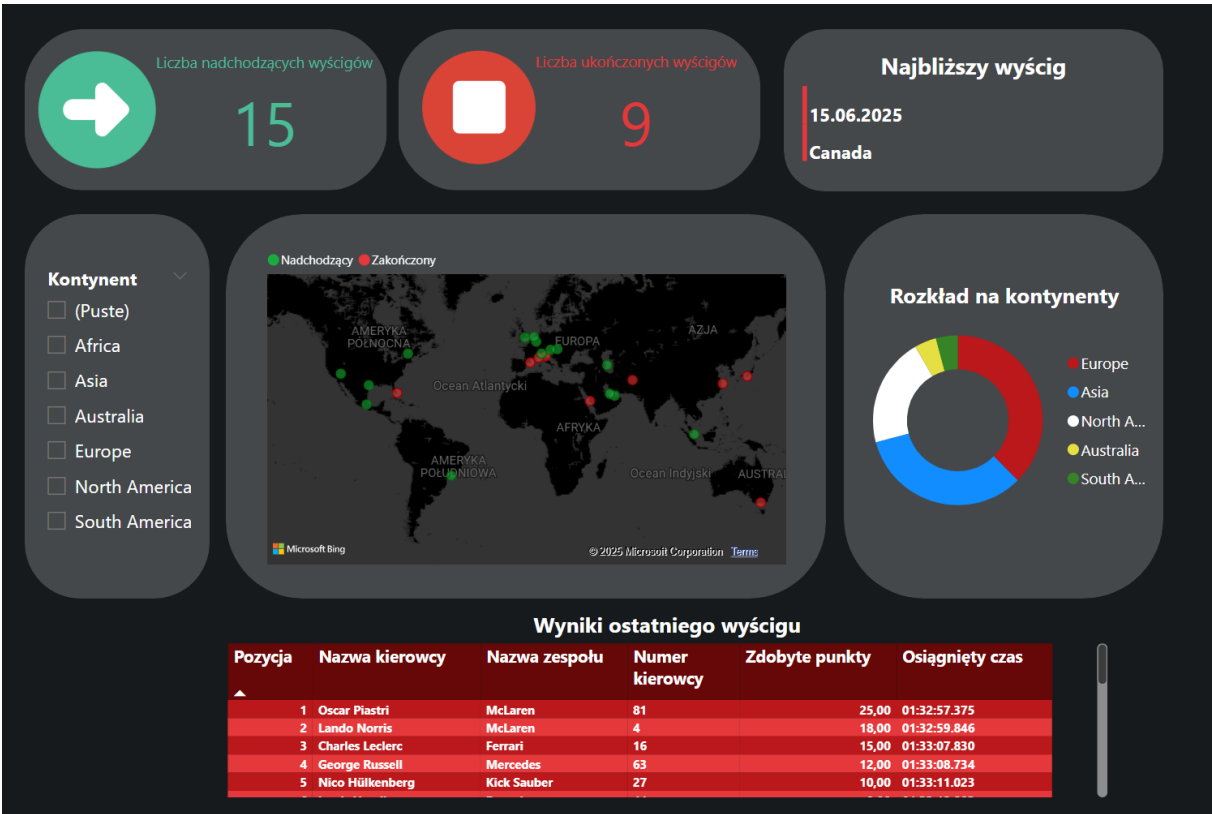
Below the query editor, there is a toolbar with a zoom dropdown set to 74 %, a 'Results' button (active), and a 'Messages' button. The results are displayed in a table with the following data:

	liczba_wyscigow	suma_ogladalnosci	srednia_ogladalnosc
1	7	1873784	267683,428571429

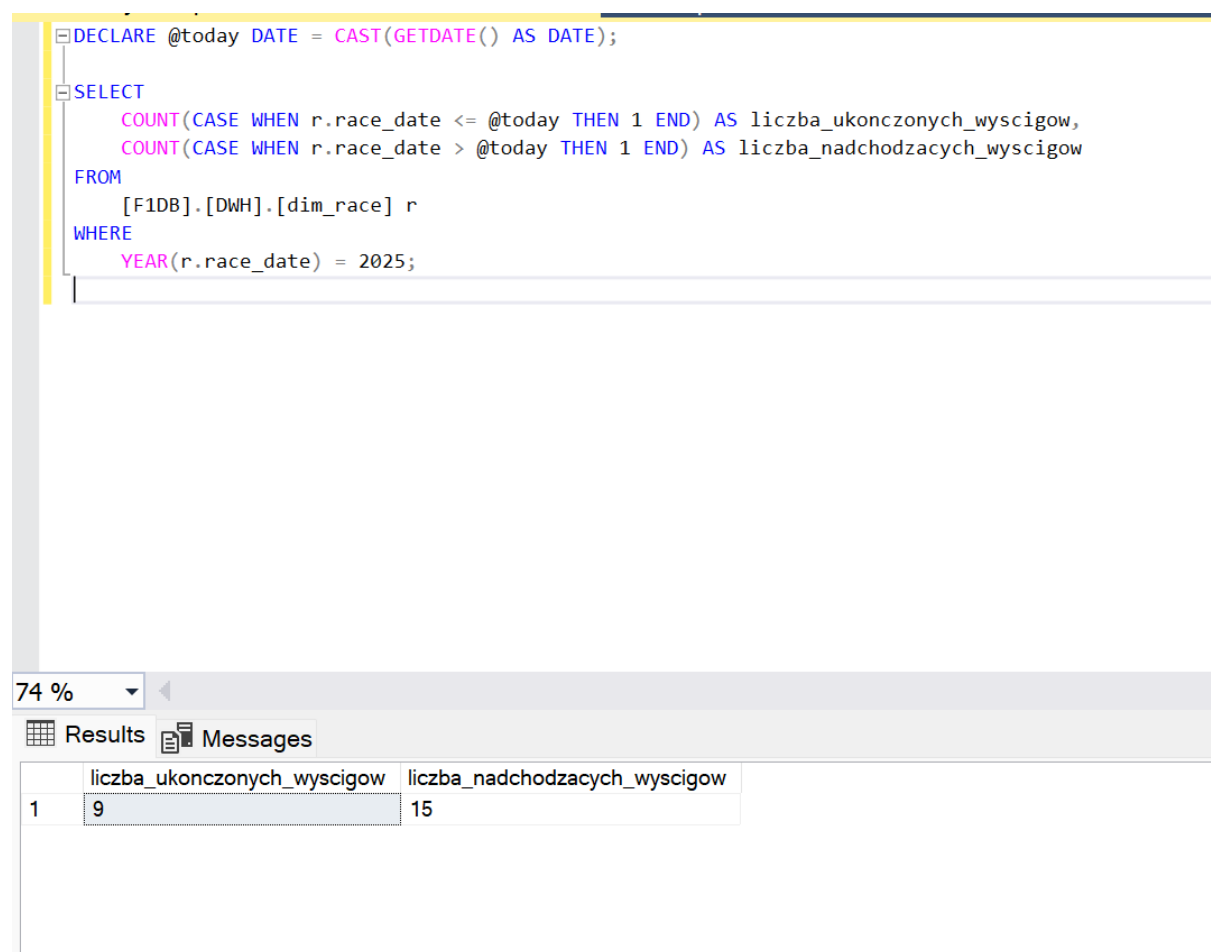
Rysunek 50: Odpowiednia kwerenda

4 strona raportu

Cel i założenia są te same co wyżej.



Rysunek 51: Test wykresu



The screenshot displays a SQL query in the SQL Server Enterprise Manager interface. The query is as follows:

```
DECLARE @today DATE = CAST(GETDATE() AS DATE);

SELECT
    COUNT(CASE WHEN r.race_date <= @today THEN 1 END) AS liczba_ukonczonych_wyscigow,
    COUNT(CASE WHEN r.race_date > @today THEN 1 END) AS liczba_nadchodzacych_wyscigow
FROM
    [F1DB].[DWH].[dim_race] r
WHERE
    YEAR(r.race_date) = 2025;
```

Below the query editor, the 'Results' tab is active, showing a single row of data:

	liczba_ukonczonych_wyscigow	liczba_nadchodzacych_wyscigow
1	9	15

Rysunek 52: Odpowiednia kwerenda

```
DECLARE @today DATE = CAST(GETDATE() AS DATE);

SELECT TOP 1
    r.race_date,
    r.race_grand_prix_name,
    c.circuit_name,
    c.circuit_place_name
FROM
    [F1DB].[DWH].[dim_race] r
JOIN
    [F1DB].[DWH].[dim_circuit] c ON r.circuit_id = c.dwh_id
WHERE
    r.race_date >= @today
ORDER BY
    r.race_date ASC;
```

4 %

Results Messages

	race_date	race_grand_prix_name	circuit_name	circuit_place_name
1	2025-06-15	Canada	Gilles Villeneuve	Montreal

Rysunek 53: Odpowiednia kwerenda

```

DECLARE @today DATE = CAST(GETDATE() AS DATE);
|
DECLARE @last_race_id INT = (
    SELECT TOP 1 dwh_id
    FROM [F1DB].[DWH].[dim_race]
    WHERE race_date <= @today
    ORDER BY race_date DESC
);
|
SELECT
    d.driver_full_name,
    frd.race_data_race_time_millis,
    frd.race_data_race_points,
    FORMAT(DATEADD(MILLISECOND, frd.race_data_race_time_millis, 0), 'HH:mm:ss') AS formatted_race_time
FROM
    [F1DB].[DWH].[fact_race_data] frd
JOIN
    [F1DB].[DWH].[dim_driver] d ON frd.driver_id = d.dwh_id
WHERE
    frd.race_id = @last_race_id

```

74 %

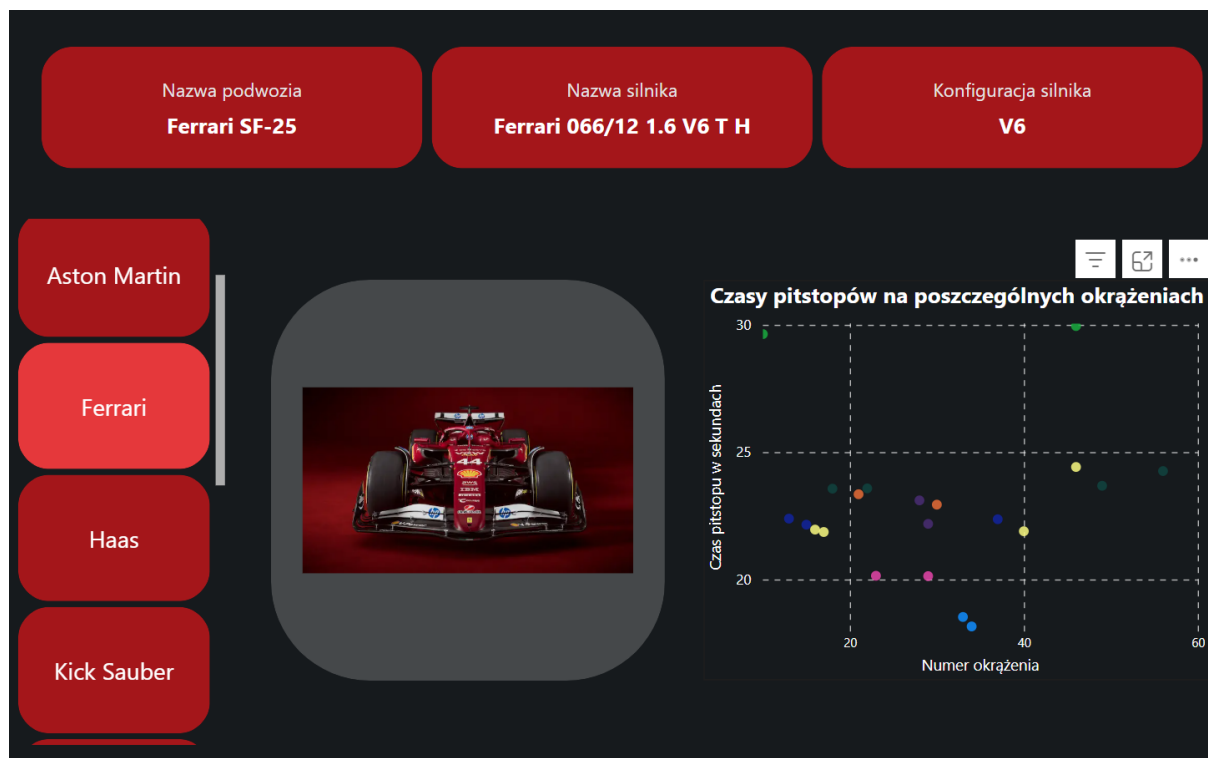
Results Messages

	driver_full_name	race_data_race_time_millis	race_data_race_points	formatted_race_time
1	Lance Stroll	NULL	NULL	NULL
2	Andrea Kimi Antonelli	NULL	NULL	NULL
3	Alexander Albon	NULL	NULL	NULL
4	Oscar Piastri	5577375	25.00	01:32:57
5	Lando Norris	5579846	18.00	01:32:59
6	Charles Marc Hervé Perceval Leclerc	5587830	15.00	01:33:07
7	George Russell	5588734	12.00	01:33:08
8	Nicolas Hülkenberg	5591023	10.00	01:33:11
9	Lewis Carl Davidson Hamilton	5592883	8.00	01:33:12
10	Isack Alexandre Hadjar	5593397	6.00	01:33:13
11	Pierre Gasly	5595257	4.00	01:33:15
12	Fernando Alonso Díaz	5598939	2.00	01:33:18
13	Max Emilian Verstappen	5599201	1.00	01:33:19
14	Liam Lawson	5602907	NULL	01:33:22
15	Gabriel Bortoleto Oliveira	5603371	NULL	01:33:23
16	Yuki Tsunoda	5606197	NULL	01:33:26
17	Carlos Sainz Vázquez de Castro	5606684	NULL	01:33:26
18	Franco Alejandro Colapinto	5608756	NULL	01:33:28
19	Esteban Ocon	5609572	NULL	01:33:29
20	Oliver James Bearman	5614440	NULL	01:33:34

Rysunek 54: Odpowiednia kwerenda

5 strona raportu

Cel i założenia są te same co wyżej.



Rysunek 55: Test wykresu

```
DECLARE @ferrari_id INT = (  
    SELECT dwh_id  
    FROM [F1DB].[DWH].[dim_constructor]  
    WHERE constructor_name = 'Ferrari'  
);  
  
DECLARE @last_year INT = (  
    SELECT MAX(entrant_year)  
    FROM [F1DB].[DWH].[fact_entrant]  
    WHERE constructor_id = @ferrari_id  
);  
  
SELECT DISTINCT  
    entrant_engine_full_name AS engine_name,  
    entrant_chassis_full_name AS chassis_name,  
    entrant_engine_configuration AS engine_configuration,  
    entrant_year  
FROM  
    [F1DB].[DWH].[fact_entrant]  
WHERE  
    constructor_id = @ferrari_id  
    AND entrant_year = @last_year;
```

74 %

Results Messages

	engine_name	chassis_name	engine_configuration	entrant_year
1	Ferrari 066/12 1.6 V6 T H	Ferrari SF-25	V6	2025

Rysunek 56: Test wykresu