



UNIwersytet Jagielloński
Wydział Matematyki i Informatyki

Projekt bazy danych dla portu lotniczego

Bartłomiej Szwaja
Yurii Titov
Informatyka, Rok II

Kraków 2021

Spis treści

Opis projektu	4
Schemat bazy danych	4
Diagram związków encji	5
Opis tabel	5
Główne założenia i ograniczenia	6
Strategia pielęgnacji bazy danych	7
Indeksy	7
Typowe zapytania.....	8
Skrypt tworzący bazę danych	11
Funkcje	27
Funkcja dbo.F_gr_obsługi_lotu	27
Funkcja dbo.F_polaczenia_przewoźnika	27
Funkcja dbo.F_Polaczenia_do_lotniska	28
Funkcja dbo.F_Samoloty_przewoźnika	28
Funkcja skalarna (pomocnicza) dbo.Czy_samolot_na_lotnisku	29
Funkcja skalarna dbo.Ilosc_rezerwacji_na_lot.....	30
Procedury	33
Procedura [Przesun_czas_lotu(o minut)]	33
Procedura Skasuj_bagaz.....	34
Procedura Zmien_samolot_obsługujący_lot.....	35
Procedura Wypowiedz_umowe_z_przewoźnikiem	38
Procedura [Przedluz_umowe_z_przewoźnikiem(o miesiecy)]	40
Procedura Usun_rezerwacje	41
Procedura Zmien_bramke.....	42
Wyzwalacze	45
Trigger TR_AU_rezerwacja	45
Trigger TR_dodaj_lot.....	46
Trigger TR_Bagaz	50
Trigger TR_gr_obsługi.....	50
Trigger TR_rezerwacje.....	53
Widoki	57

Widok dbo.vw_Szczegoly_lotow.....	57
Widok dbo.vw_przebieg_samolotow.....	58
Widok dbo.vw_BRAMKI	58
Widok dbo.vw_Odloty.....	59
Widok dbo.vw_Przyloty.....	59
Widok dbo.vw_Szczegoly_rezerwacji_pasazerow	60

Opis projektu

Celem projektu jest realizacja bazy danych dla portu lotniczego. W dzisiejszych czasach porty lotnicze są miejscami niezwykle ważnymi w podróżowaniu. Umożliwiają najszybsze dotarcie do wybranego celu podróży spośród dostępnych powszechnie środków transportu. Dlatego coraz więcej ludzi wyraża chęć skorzystania z takich udogodnień. Stąd efektem takiego działania jest rosnąca z roku na rok liczba wykonanych lotów na świecie. Jak przed chwilą wspomnieliśmy, zaletą podróży samolotem jest zyskany czas, jednakże z uwagi na duże zainteresowanie czas ten musi być starannie rozplanowany przez administrację portu lotniczego, by w ciągu doby lotnisko mogło obsłużyć jak najwięcej lotów. Dlatego zaprojektowana przez nas baza danych, jest przygotowana do zapewnienia optymalnego funkcjonowania wyżej wymienionego podmiotu.

Schemat bazy danych

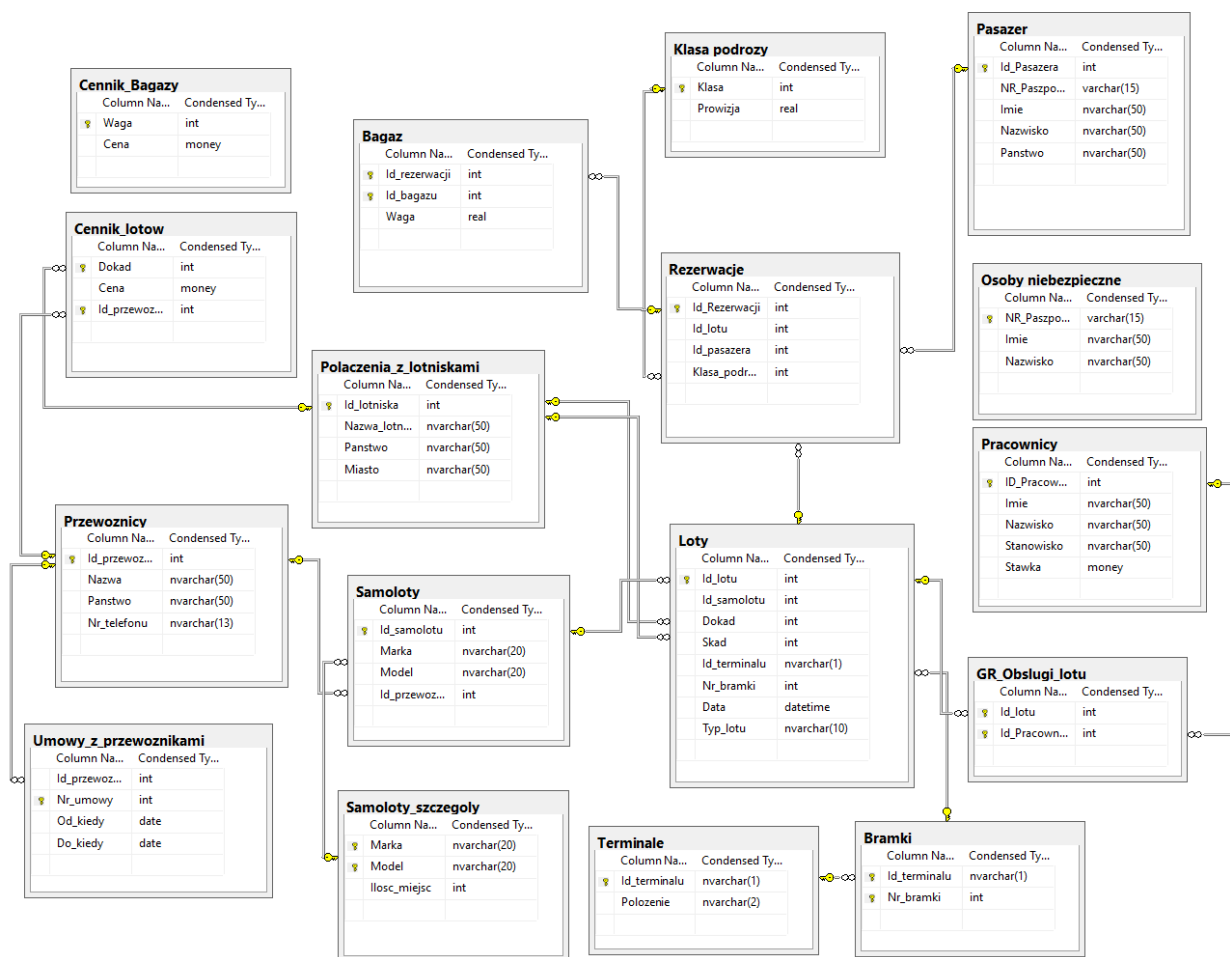
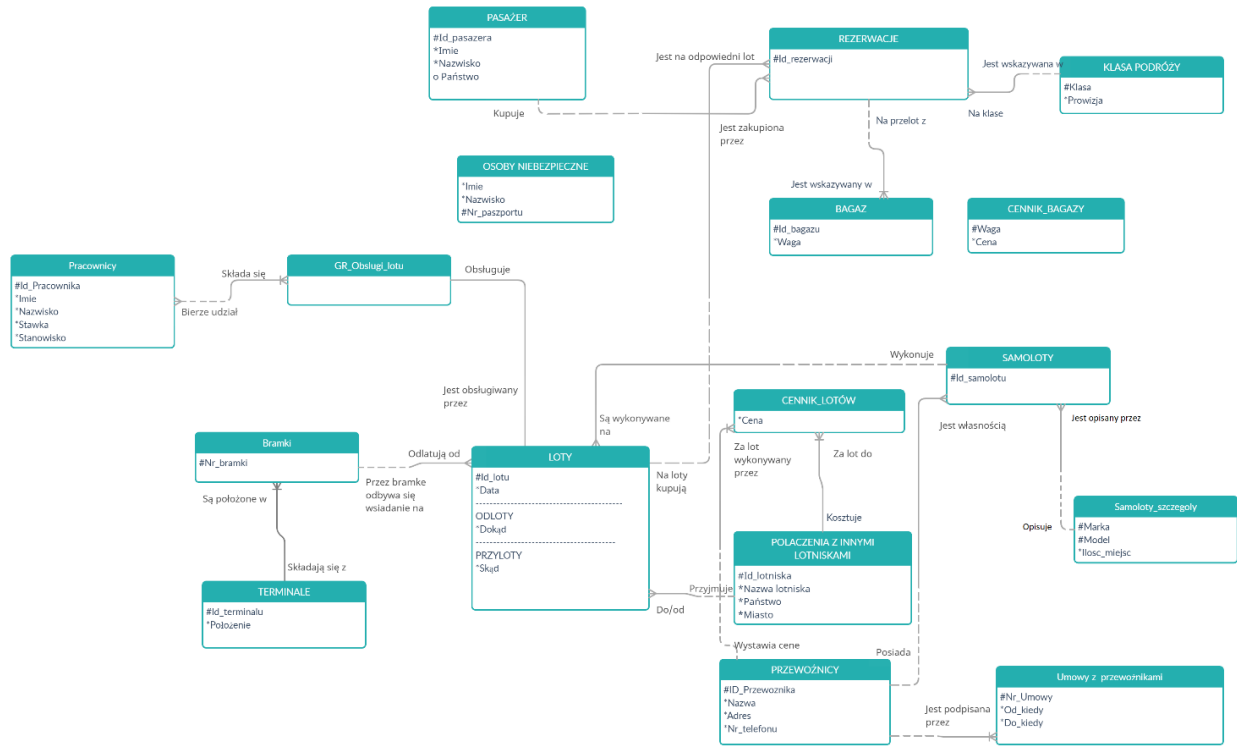


Diagram związków encji



Opis tabel

Baza danych zawiera 17 tabel. Tabelą najważniejszą z punktu funkcjonowania lotniska jest tabela **Loty**. Zawiera podstawowe informacje o zaplanowanych lotach. W celu wyznaczenia miejsca odlotu/przylotu samolotu istnieją tabele **Bramki** oraz **Terminale**. W bazie trzymane są również informacje o przewoźnikach (tabela **Przewoźnicy**) oraz umowach z nimi (tabela **Umowy_z_przewoźnikami**). W tej drugiej trzymane są informacje o dacie rozpoczęcia i zakończenia kontraktu z danym przewoźnikiem. By loty mogły funkcjonować prawidłowo istnieją również tabele przechowujące informacje o samolotach (**Samoloty** oraz **Samoloty_szczegoly**). Utworzyliśmy dwie tabele by wyeliminować redundancję, w postaci powtarzającej się informacji o ilości miejsc w tych samych modelach samolotów.

Baza zawiera również tabelę ***Polaczenia_z_innymi_lotniskami*** oraz tabelę ***Cennik_lotow***. Ta pierwsza przechowuje lotniska na które mogą latać samoloty z naszego portu. Zaś ta druga reguluje ceny biletów do wyżej wspomnianych lotnisk.

Każdy przewoźnik posiada własną cenę na każdy lot. Jednak nie każdy przewoźnik lata w każde miejsce. Stąd może być sytuacja w której pewne miejsce obsługuje tylko jeden przewoźnik. Zadbaliśmy również o informację, jacy pracownicy obsługują dany lot. Czyli jaką tożsamość posiadają piloci, stewardessy, stewardzi odpowiedzialni za każdą podróż. Te dane są przechowywane w tabelach **Pracownicy** oraz **GR_obsługi_lotu**.

Lotnisko musi obsługiwać również proces składania rezerwacji na dany lot. Stąd tabela **Rezerwacje**. Z tym wykazem powiązane są również tabele: **Klasa_podróży** – określająca prowizję za daną klasę, **Bagaz** – zawierająca informacje o bagażach danego pasażera, oraz tabela „informacyjna” **Cennik_bagazy** określająca cenę za bagaż w stosunku do jego wagi. Ponadto istnieje również tabela **Pasażer** przechowująca informacje na temat pasażerów. W celu zwiększenia bezpieczeństwa portu lotniczego utworzyliśmy również tabelę **Osoby_niebezpieczne**, która przechowuje dane osób poszukiwanych. Takie osoby nie mogą dokonać rezerwacji.

Główne założenia i ograniczenia:

- dana bramka może obsługiwać tylko jeden samolot w czasie 30 minut przed startem oraz 30 minut po starcie,
- łączna waga bagaży jednego pasażera na dany lot nie może przekroczyć 100kg,
- każdy samolot, który odlata z lotniska, musi na nie powrócić by móc wykonać kolejny lot,
- przylot samolotu nie musi być z miejsca w które poleciał,
- liczba rezerwacji na dany lot nie może przekroczyć liczby miejsc w samolocie,
- osoba niebezpieczna nie może dokonać rezerwacji,
- każdy pracownik w ciągu jednego dnia może obsługiwać tylko jeden lot

Więcej ograniczeń oraz założeń zostało szczegółowo opisane w opisach funkcji, procedur oraz wyzwalaczy.

Strategia pielęgnacji bazy danych

Zaleca się tworzyć kopie zapasowe bazy każdego dnia w ustalonym czasie na dwóch dyskach pamięci. Najlepiej w nocy w godzinach 3:00 – 4:00, ponieważ wtedy odbywa się najmniejsza ilość odlotów i przylotów, a ponadto wykonuje się minimalna ilość rezerwacji na loty oraz pracownicy lotniska nie wprowadzają żadnych nowych danych o grupach obsługi lotów. Nie są aktualizowane dane o dokumentacji tj. umowy z przewoźnikami, ponieważ nie są to dla nich godziny robocze. Ze względu na to, obciążenie systemu w tym czasie będzie najmniejsze. Kopii zapasowych bazy danych portu lotniczego nie należy usuwać przez okres czterech miesięcy. Ewentualnie termin ten może ulec zmianie ze względu na możliwości techniczne. Jednak nie zaleca się, żeby był on mniejszy niż jeden miesiąc, ze względu na obsługę klientów oraz linii lotniczych, by w przypadku wystąpienia pewnych niezgodności można było prześledzić chronologię wprowadzenia zmian. Dla ekonomii pamięci zaleca się usuwanie wszystkich kopii zapasowych starszych niż cztery miesiące, oprócz ostatniej kopii zrobionej w każdym miesiącu roku, która będzie służyć jako archiwum.

Indeksy

MS SQL Server automatycznie tworzy indeksy typu clustered dla kolumn, które pełnią funkcję klucza głównego w danej tabeli. Dodatkowo system tworzy również indeksy typu nonclustered dla każdej kolumny, która posiada ograniczenie UNIQUE.

Indeks dodatkowy stworzony przez nas:

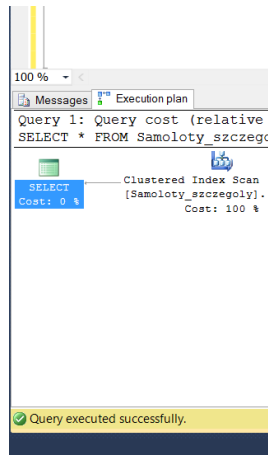
Indeks nonclustered IX_SamolotySzczegoly_IloscMiejsc

```
CREATE NONCLUSTERED INDEX [IX_SamolotySzczegoly_IloscMiejsc]  
ON Samoloty_szczegoly(Ilosc_miejsc)
```

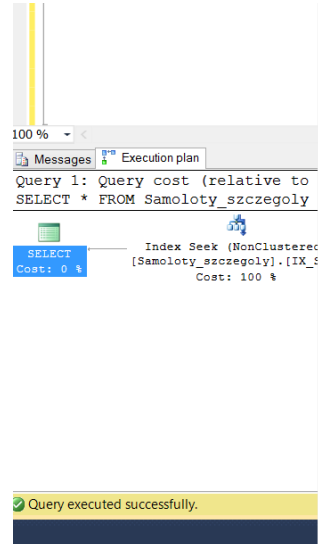
Sprawdzenie wydajności na zapytaniu:

```
SELECT * FROM Samoloty_szczegoly  
WHERE Ilosc_miejsc = 180
```

Przed utworzeniem nonclustered:

	
Clustered Index Scan (Clustered) Scanning a clustered index, entirely or only a range.	
Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Estimated Execution Mode	Row
Estimated Operator Cost	0,0033106 (100%)
Estimated I/O Cost	0,003125
Estimated CPU Cost	0,0001856
Estimated Subtree Cost	0,0033106
Estimated Number of Executions	1
Estimated Number of Rows	1
Estimated Row Size	57 B
Ordered	False
Node ID	0
Predicate	[Port_lotniczy].[dbo].[Samoloty_szczegoly].[Ilosc_miejsc] = CONVERT_IMPLICIT(int,[@1],0)
Object	[Port_lotniczy].[dbo].[Samoloty_szczegoly]. [PK_Samoloty_0D537D84D27D97D5]
Output List	[Port_lotniczy].[dbo].[Samoloty_szczegoly].Marka; [Port_lotniczy].[dbo].[Samoloty_szczegoly].Model; [Port_lotniczy].[dbo].[Samoloty_szczegoly].Ilosc_miejsc

Po utworzeniu nonclustered:

	
Index Seek (NonClustered) Scan a particular range of rows from a nonclustered index.	
Physical Operation	Index Seek
Logical Operation	Index Seek
Estimated Execution Mode	Row
Storage	RowStore
Estimated I/O Cost	0,003125
Estimated Operator Cost	0,0032831 (100%)
Estimated Subtree Cost	0,0032831
Estimated CPU Cost	0,0001581
Estimated Number of Executions	1
Estimated Number of Rows	1
Estimated Row Size	39 B
Ordered	True
Node ID	0
Object	[Port_lotniczy].[dbo].[Samoloty_szczegoly]. [IX_SamolotySzczegoly_IloscMiejsc]
Output List	[Port_lotniczy].[dbo].[Samoloty_szczegoly].Marka; [Port_lotniczy].[dbo].[Samoloty_szczegoly].Model; [Port_lotniczy].[dbo].[Samoloty_szczegoly].Ilosc_miejsc
Seek Predicates	Seek Keys[1]: Prefix: [Port_lotniczy].[dbo]. [Samoloty_szczegoly].Ilosc_miejsc = Scalar Operator (CONVERT_IMPLICIT(int,[@1],0))

Typowe zapytania

Zapytanie zwracające ilość lotów w każdy dzień:

```
SELECT Data, COUNT(*) AS [Ilość lotów] FROM
(SELECT CONVERT(date, Data) AS Data FROM Loty ) AS DataTable
GROUP BY Data
```

Zapytanie zwracające średnią pensję dla każdego stanowiska pracy:

```
SELECT Stanowisko, ROUND(AVG(Stawka), 2) AS [Średnia pensja]
FROM Pracownicy
GROUP BY Stanowisko
```

Zapytanie zwracające ilość pasażerów danego kraju:

```
SELECT Panstwo, COUNT(*) AS [Ilość pasażerów]
FROM Pasazer
GROUP BY Panstwo
```


Zapytanie zwracające najniższą cenę do danego lotniska, wraz z przewoźnikiem, który taką cenę oferuje:

```
SELECT Cennik_lotow.Dokad, Nazwa_lotniska, [Najniższa cena],
Przewoźnicy.Nazwa, Przewoźnicy.Id_przewoźnika
FROM Cennik_lotow

JOIN
(SELECT Dokad, MIN(Cena) AS [Najniższa cena]
FROM Cennik_lotow
GROUP BY Dokad) AS Dane
ON (Cennik_lotow.Dokad = Dane.Dokad
AND Cena = Dane.[Najniższa cena])

JOIN Polaczenia_z_lotniskami
ON Polaczenia_z_lotniskami.Id_lotniska = Cennik_lotow.Dokad

JOIN Przewoźnicy
ON Przewoźnicy.Id_przewoźnika = Cennik_lotow.Id_przewoźnika
```

Zapytanie zwracające ilość rezerwacji , dokonanych przez każdego pasażera, oraz wyświetla wszystkie dane tego pasażera

```
SELECT P.*, COUNT(Id_Rezerwacji) [Ilość dokonanych rezerwacji]
FROM Pasazer P LEFT JOIN Rezerwacje R
on P.Id_Pasazera = R.Id_pasazera
GROUP By P.Id_Pasazera, P.Imie, P.Nazwisko, P.NR_Paszportu, P.Panstwo
```

Zapytanie zwracające wolne w obecnej chwili Bramki , które mogą przyjąć lot (ewentualnie opóźniony lub nieprzewidywalny)

```
SELECT B.Id_terminalu, B.Nr_bramki from BRAMKI B
where not exists
(SELECT B1.Id_terminalu, B1.Nr_bramki from vw_BRAMKI B1
where B.Id_terminalu = B1.Id_terminalu and B.Nr_bramki = B1.Nr_bramki
and
(B1.Data between DATEADD(MINUTE, -30, CURRENT_TIMESTAMP) and
DATEADD(MINUTE, 30, CURRENT_TIMESTAMP)))
```

Zapytanie zwracające loty , na które już nie można dokonać rezerwację z powodu braku miejsc lub ich odlotu, lub tego , że jest ten lot przylotem .

```
Select L.Id_lotu,  dbo.Ilosc_rezerwacji_na_lot(L.Id_lotu),  
L.Typ_lotu, L.Data  
from Loty L  
JOIN Samoloty S on L.Id_samolotu = S.Id_samolotu  
JOIN Samoloty_szczegoly Sz  
on S.Marka = Sz.Marka and S.Model = Sz.Model  
where Sz.Ilosc_miejsc = dbo.Ilosc_rezerwacji_na_lot(L.Id_lotu)  
or L.Typ_lotu = 'Przylot' or L.Data < CURRENT_TIMESTAMP
```

Skrypt tworzący bazę danych

```
CREATE DATABASE Port_lotniczy
GO
USE Port_lotniczy

CREATE TABLE Przewoźnicy (
    Id_przewoźnika INT PRIMARY KEY,
    Nazwa NVARCHAR(50) NOT NULL,
    Państwo NVARCHAR(50) NOT NULL,
    Nr_telefonu NVARCHAR(13) NOT NULL
);

CREATE TABLE Samoloty_szczegoly (
    Marka NVARCHAR(20),
    Model NVARCHAR(20),
    Ilosc_miejsc INT NOT NULL,
    PRIMARY KEY(Marka, Model)
)

CREATE TABLE Samoloty (
    Id_samolotu INT PRIMARY KEY,
    Marka NVARCHAR(20) NOT NULL,
    Model NVARCHAR(20) NOT NULL,
    Id_przewoźnika INT NOT NULL,
    FOREIGN KEY (Marka, Model) REFERENCES Samoloty_szczegoly(Marka, Model) ON
    DELETE CASCADE ON UPDATE CASCADE
)

ALTER TABLE Samoloty
ADD CONSTRAINT FK_Samoloty FOREIGN KEY(Id_przewoźnika) REFERENCES
Przewoźnicy(Id_przewoźnika)
ON DELETE CASCADE
ON UPDATE CASCADE;

CREATE TABLE Umowy_z_przewoźnikami (
    Id_przewoźnika INT NOT NULL,
    Nr_umowy INT PRIMARY KEY,
    Od_kiedy DATE,
    Do_kiedy DATE,
    CHECK (Do_kiedy >= Od_kiedy),
    FOREIGN KEY (Id_przewoźnika) REFERENCES Przewoźnicy(Id_przewoźnika) ON DELETE
    CASCADE ON UPDATE CASCADE
);

CREATE TABLE Polaczenia_z_lotniskami (
    Id_lotniska INT PRIMARY KEY,
    Nazwa_lotniska NVARCHAR(50) NOT NULL,
    Państwo NVARCHAR(50) NOT NULL,
    Miasto NVARCHAR(50) NOT NULL
);

CREATE TABLE Cennik_lotow (
```

```

    Dokad INT NOT NULL,
    Cena MONEY NOT NULL,
    Id_przewoznika INT NOT NULL,
    PRIMARY KEY(Dokad, Id_przewoznika),
    FOREIGN KEY(Dokad) REFERENCES Polaczenia_z_lotniskami(Id_lotniska) ON DELETE
    CASCADE ON UPDATE CASCADE,
    FOREIGN KEY(Id_przewoznika) REFERENCES Przewoznicy(Id_przewoznika) ON DELETE
    CASCADE ON UPDATE CASCADE
);

CREATE TABLE Terminale (
    Id_terminalu NVARCHAR(1) PRIMARY KEY,
    Polozenie NVARCHAR(2) NOT NULL
);

CREATE TABLE Bramki (
    Id_terminalu NVARCHAR(1) NOT NULL,
    Nr_bramki INT NOT NULL,
    PRIMARY KEY (Id_terminalu, Nr_bramki),
    FOREIGN KEY(Id_terminalu) REFERENCES Terminale(Id_terminalu) ON DELETE CASCADE
ON UPDATE CASCADE
);

CREATE TABLE Loty (
    Id_lotu INT PRIMARY KEY,
    Id_samolotu INT NOT NULL,
    Dokad INT,
    Skad INT,
    Id_terminalu NVARCHAR(1) NOT NULL,
    Nr_bramki INT NOT NULL,
    Data DATETIME NOT NULL,
    Typ_lotu NVARCHAR(10) NOT NULL,
    FOREIGN KEY (Id_samolotu) REFERENCES Samoloty(Id_samolotu)
    ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (Id_terminalu, Nr_bramki)
    REFERENCES Bramki(Id_terminalu, Nr_bramki) ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (Dokad) REFERENCES Polaczenia_z_lotniskami(Id_lotniska)
    ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (Skad) REFERENCES Polaczenia_z_lotniskami(Id_lotniska),
    CONSTRAINT CHK_Typ_lotu CHECK ((Typ_lotu = 'Przylot' AND (Skad IS NOT NULL
    AND Dokad IS NULL)) OR (Typ_lotu = 'Odlot' AND (Dokad IS NOT NULL
    AND Skad IS NULL)))
)

-----

IF OBJECT_ID('Pracownicy', 'U') is not NULL DROP TABLE Pracownicy

CREATE TABLE Pracownicy(
    ID_Pracownika int not null,
    Imie nvarchar(50) not null,
    Nazwisko nvarchar(50) not null,
    Stanowisko nvarchar(50) not null,
    Stawka money not null

```

```

)

IF OBJECT_ID('GR_Obslugi_lotu', 'U') is not NULL DROP TABLE GR_Obslugi_lotu

CREATE TABLE GR_Obslugi_lotu(
    Id_lotu int not null,
    Id_Pracownika int not null
)

IF OBJECT_ID('Osoby niebezpieczne', 'U') is not NULL DROP TABLE [Osoby niebezpieczne]

CREATE TABLE [Osoby niebezpieczne](
    NR_Paszportu varchar(15) not null,
    Imie nvarchar(50) not null,
    Nazwisko nvarchar(50) not null,
)

IF OBJECT_ID('Pasazer', 'U') is not NULL DROP TABLE Pasazer

CREATE TABLE Pasazer(
    Id_Pasazera int not null,
    NR_Paszportu varchar(15) not null UNIQUE,
    Imie nvarchar(50) not null,
    Nazwisko nvarchar(50) not null,
    Panstwo nvarchar(50) null
)

IF OBJECT_ID('Klasa podrozy', 'U') is not NULL DROP TABLE [Klasa podrozy]

CREATE TABLE [Klasa podrozy](
    Klasa int not null,
    Prowizja real not null
)

IF OBJECT_ID('Rezerwacje', 'U') is not NULL DROP TABLE Rezerwacje

CREATE TABLE Rezerwacje(
    Id_Rezerwacji int not null,
    Id_lotu int not null,
    Id_pasazera int not null,
    Klasa_podrozy int not null
)

IF OBJECT_ID('Bagaz', 'U') is not NULL DROP TABLE Bagaz

CREATE TABLE Bagaz(
    Id_rezerwacji int not null,
    Id_bagazu int not null,
    Waga real not null
)

IF OBJECT_ID('Cennik_Bagazy', 'U') is not NULL DROP TABLE [Cennik_Bagazy]

```

```

CREATE TABLE [Cennik_Bagazy](
    Waga int not null, -- do X kg
    Cena money not null,
)

-- PRIMARY KEY :
Alter table [Cennik_Bagazy] add PRIMARY KEY (Waga)
Alter table Bagaz add Primary Key (Id_rezerwacji, Id_bagazu)
Alter table Rezerwacje add Primary Key (Id_Rezerwacji)
Alter table [Klasa podrozy] add Primary Key (Klasa)
Alter table Pasazer add Primary Key (Id_Pasazera)
Alter table [Osoby niebezpieczne] add Primary Key (NR_Paszportu)
Alter table GR_Obslugi_lotu add Primary Key (Id_lotu, ID_Pracownika)
Alter table Pracownicy add Primary Key (ID_Pracownika)

--FOREIGN KEY :

Alter table GR_Obslugi_lotu add Foreign Key (ID_Pracownika)
REFERENCES Pracownicy(ID_Pracownika)
ON DELETE CASCADE
ON UPDATE CASCADE

Alter table GR_Obslugi_lotu add Foreign Key (ID_lotu) REFERENCES Loty(ID_Lotu)
ON DELETE CASCADE
ON UPDATE CASCADE

Alter table Rezerwacje add FOREIGN KEY (Id_pasazera) REFERENCES Pasazer(Id_Pasazera)
ON DELETE CASCADE
ON UPDATE CASCADE

Alter table Rezerwacje add FOREIGN KEY (Klasa_podrozy)
REFERENCES [Klasa podrozy](Klasa)

Alter table Rezerwacje add FOREIGN KEY (Id_lotu) REFERENCES [Loty](Id_lotu)
ON DELETE CASCADE
ON UPDATE CASCADE

Alter table Bagaz add FOREIGN KEY (Id_rezerwacji) REFERENCES
Rezerwacje(Id_rezerwacji)
ON DELETE CASCADE
ON UPDATE CASCADE

-- UNIQUE
Alter TABLE REZERWACJE
ADD UNIQUE(ID_LOTU, ID_PASAZERA)

-----wstawianie danych
INSERT INTO Przewoźnicy VALUES
(1, 'Ryanair', 'Irlandia', '+48221522001'),
(2, 'Lufthansa', 'Niemcy', '+48225123917'),
(3, 'easyJet', 'Wielka Brytania', '+48616262022'),
(4, 'Emirates', 'Zjednoczone Emiraty Arabskie', '+48223060808'),

```

```
(5, 'Air France', 'Francja', '+48225123949'),
(6, 'British Airways', 'Wielka Brytania', '+48223060850'),
(7, 'Air Berlin', 'Niemcy', '+493034340'),
(8, 'KLM', 'Holandia', '+48225123947'),
(9, 'Delta Air Lines', 'Stany Zjednoczone', '+14047152600'),
(10, 'American Airlines', 'Stany Zjednoczone', '+48226253002'),
(11, 'Air China', 'Chiny', '00861095583'),
(12, 'Singapore Airlines', 'Singapur', '+6562238888'),
(13, 'Turkish Airlines', 'Turcja', '+48225297700'),
(14, 'LOT Polish Airlines', 'Polska', '+48225777755'),
(15, 'Ukraine International Airlines', 'Ukraina', '+444865486'),
(16, 'Qatar Airways', 'Katar', '+48717564055'),
(17, 'ANA All Nippon Airways', 'Japonia', '0367411120'),
(18, 'Qantas Airways', 'Australia', '1300650729'),
(19, 'Wizz Air', 'Węgry', '+48703703003'),
(20, 'Air Canada', 'Kanada', '+48226968520');
```

INSERT INTO Umowy_z_przewoźnikami VALUES

```
(1, 1001, '20150115', '20250101'),
(2, 1002, '20160315', '20210205'),
(3, 1003, '20160215', '20220211'),
(4, 1012, '20100425', '20300709'),
(5, 1005, '20100224', '20251111'),
(6, 1023, '20190102', '20220421'),
(7, 1007, '20200101', '20201231'),
(8, 1008, '20190717', '20220202'),
(9, 1009, '20100110', '20300110'),
(10, 1034, '20170108', '20250125'),
(11, 1011, '20190105', '20240404'),
(12, 1056, '20160119', '20270101'),
(13, 1067, '20000616', '20210202'),
(14, 1078, '20150115', '20240212'),
(15, 1089, '20180701', '20300101'),
(16, 1016, '20100110', '20290908'),
(17, 1017, '20100110', '20201231'),
(18, 1018, '20150115', '20220213'),
(19, 2001, '20191231', '20211231'),
(20, 1020, '20120112', '20230307');
```

INSERT INTO Samoloty_szczegoly VALUES

```
('Boeing', '737-300', 133),
('Boeing', '737-900', 215),
('Boeing', '747-8', 467),
('Boeing', '767-200', 216),
('Boeing', '777-300', 297),
('Boeing', '787-9', 275),
('Boeing', '787-10', 310),
('Airbus', 'A300', 266),
('Airbus', 'A310', 262),
('Airbus', 'A330', 330),
('Airbus', 'A340', 350),
('Airbus', 'A350', 325),
('Airbus', 'A380', 853),
('Airbus', 'A318', 132),
```

```
( 'Airbus', 'A320', 180),
( 'Airbus', 'A321', 220),
( 'Embraer', '170', 70),
( 'Embraer', '175', 80),
( 'Embraer', '190', 98),
( 'Embraer', '190LR', 106),
( 'Embraer', '195', 108),
( 'Embraer', '195LR', 122),
( 'Saab', '2000', 58),
( 'ATR', '42-300', 50),
( 'Cessna', 'Citation I', 8),
( 'Cessna', 'Citation X', 12)
```

INSERT INTO Samoloty VALUES

```
(1, 'Boeing', '737-300', 1),
(2, 'Boeing', '737-900', 2),
(3, 'Boeing', '747-8', 2),
(4, 'Boeing', '767-200', 1),
(5, 'Boeing', '777-300', 1),
(6, 'Boeing', '787-9', 1),
(7, 'Boeing', '787-10', 3),
(8, 'Airbus', 'A300', 1),
(9, 'Airbus', 'A310', 1),
(10, 'Airbus', 'A330', 3),
(11, 'Airbus', 'A340', 1),
(12, 'Airbus', 'A350', 3),
(13, 'Airbus', 'A380', 4),
(14, 'Airbus', 'A318', 1),
(15, 'Airbus', 'A320', 4),
(16, 'Airbus', 'A321', 1),
(17, 'Embraer', '170', 5),
(18, 'Embraer', '175', 1),
(19, 'Embraer', '190', 1),
(20, 'Embraer', '190LR', 1),
(21, 'Embraer', '195', 5),
(22, 'Embraer', '195LR', 7),
(23, 'Saab', '2000', 1),
(24, 'ATR', '42-300', 7),
(25, 'Cessna', 'Citation I', 8),
(26, 'Cessna', 'Citation X', 8),
-----
(27, 'Boeing', '737-300', 6),
(28, 'Boeing', '737-900', 6),
(29, 'Boeing', '747-8', 6),
(30, 'Boeing', '767-200', 4),
(31, 'Boeing', '777-300', 6),
(32, 'Boeing', '787-9', 6),
(33, 'Boeing', '787-10', 6),
(34, 'Airbus', 'A300', 6),
(35, 'Airbus', 'A310', 6),
(36, 'Airbus', 'A330', 9),
(37, 'Airbus', 'A340', 6),
(38, 'Airbus', 'A350', 6),
(39, 'Airbus', 'A380', 8),
(40, 'Airbus', 'A318', 6),
```



```

(41, 'Airbus', 'A320', 6),
(42, 'Airbus', 'A321', 6),
--
(43, 'Boeing', '737-300', 10),
(44, 'Boeing', '737-900', 10),
(45, 'Boeing', '747-8', 11),
(46, 'Boeing', '767-200', 10),
(47, 'Boeing', '777-300', 11),
(48, 'Boeing', '787-9', 10),
(49, 'Boeing', '787-10', 10),
(50, 'Airbus', 'A300', 10),
(51, 'Airbus', 'A310', 10),
(52, 'Airbus', 'A330', 10),
(53, 'Airbus', 'A340', 11),
(54, 'Airbus', 'A350', 10),
(55, 'Airbus', 'A380', 10),
(56, 'Airbus', 'A318', 10),
(57, 'Airbus', 'A320', 10),
(58, 'Airbus', 'A321', 10),
--
(59, 'Boeing', '737-300', 13),
(60, 'Boeing', '737-900', 12),
(61, 'Boeing', '747-8', 13),
(62, 'Boeing', '767-200', 13),
(63, 'Boeing', '777-300', 12),
(64, 'Boeing', '787-9', 12),
(65, 'Boeing', '787-10', 12),
(66, 'Airbus', 'A300', 12),
(67, 'Airbus', 'A310', 13),
(68, 'Airbus', 'A330', 12),
(69, 'Airbus', 'A340', 13),
(70, 'Airbus', 'A350', 12),
(71, 'Airbus', 'A380', 12),
(72, 'Airbus', 'A318', 13),
(73, 'Airbus', 'A320', 12),
(74, 'Airbus', 'A321', 12),
--
(75, 'Boeing', '737-300', 14),
(76, 'Boeing', '737-900', 14),
(77, 'Boeing', '747-8', 14),
(78, 'Boeing', '767-200', 14),
(79, 'Boeing', '777-300', 15),
(80, 'Boeing', '787-9', 14),
(81, 'Boeing', '787-10', 14),
(82, 'Airbus', 'A300', 14),
(83, 'Airbus', 'A310', 15),
(84, 'Airbus', 'A330', 14),
(85, 'Airbus', 'A340', 14),
(86, 'Airbus', 'A350', 15),
(87, 'Airbus', 'A380', 14),
(88, 'Airbus', 'A318', 15),
(89, 'Airbus', 'A320', 14),
(90, 'Airbus', 'A321', 14),
(91, 'Embraer', '170', 15),
(92, 'Embraer', '175', 14),

```

```

(93, 'Embraer', '190', 15),
(94, 'Embraer', '190LR', 14),
(95, 'Embraer', '195', 14),
(96, 'Embraer', '195LR', 14),
(97, 'Saab', '2000', 15),
(98, 'ATR', '42-300', 14),
(99, 'Cessna', 'Citation I', 15),
(100, 'Cessna', 'Citation X', 14),
--
(101, 'Airbus', 'A300', 16),
(102, 'Airbus', 'A310', 16),
(103, 'Airbus', 'A330', 16),
(104, 'Airbus', 'A340', 16),
(105, 'Airbus', 'A350', 16),
(106, 'Airbus', 'A380', 16),
(107, 'Airbus', 'A318', 16),
(108, 'Airbus', 'A320', 16),
(109, 'Airbus', 'A321', 16),
--
(110, 'Boeing', '737-300', 17),
(111, 'Boeing', '737-900', 17),
(112, 'Boeing', '747-8', 18),
(113, 'Boeing', '767-200', 19),
(114, 'Boeing', '777-300', 20),
(115, 'Boeing', '787-9', 17),
(116, 'Boeing', '787-10', 19),
(117, 'Airbus', 'A300', 18),
(118, 'Airbus', 'A310', 19),
(119, 'Airbus', 'A330', 20),
(120, 'Airbus', 'A340', 20),
(121, 'Airbus', 'A350', 19),
(122, 'Airbus', 'A380', 19),
(123, 'Airbus', 'A318', 19),
(124, 'Airbus', 'A320', 19),
(125, 'Airbus', 'A321', 18),
(126, 'Embraer', '170', 17),
(127, 'Embraer', '175', 18),
(128, 'Embraer', '190', 18),
(129, 'Embraer', '190LR', 19),
(130, 'Embraer', '195', 20),
(131, 'Embraer', '195LR', 20),
(132, 'Saab', '2000', 19),
(133, 'ATR', '42-300', 19),
(134, 'Cessna', 'Citation I', 19),
(135, 'Cessna', 'Citation X', 19)

```

INSERT INTO Polaczenia_z_lotniskami VALUES

```

(1, 'Hartsfield-Jackson Atlanta International Airport', 'Stany Zjednoczone',
'Atlanta'),
(2, 'Beijing Capital International Airport', 'Chiny', 'Pekin'),
(3, 'Dubai International Airport', 'Zjednoczone Emiraty Arabskie', 'Dubaj'),
(4, 'Haneda Airport', 'Japonia', 'Tokio'),
(5, 'Los Angeles International Airport', 'Stany Zjednoczone', 'Los Angeles'),
(6, 'Chicago-O'Hare', 'Stany Zjednoczone', 'Chicago'),

```

```
(7, 'London Heathrow', 'Wielka Brytania', 'Londyn'),
(8, 'Hong Kong International Airport', 'Chiny', 'Hongkong'),
(9, 'Shanghai Pudong International Airport', 'Chiny', 'Szanghaj'),
(10, 'Charles de Gaulle International Airport', 'Francja', 'Roissy-en-France'),
(11, 'Amsterdam Airport Schiphol', 'Holandia', 'Amsterdam'),
(12, 'Dallas/Fort Worth International Airport', 'Stany Zjednoczone', 'Dallas'),
(13, 'Guangzhou Baiyun International Airport', 'Chiny', 'Kanton'),
(14, 'Frankfurt am Main Airport', 'Niemcy', 'Frankfurt'),
(15, 'Ataturk Airport', 'Turcja', 'Stambul'),
(16, 'Indira Gandhi International Airport', 'Indie', 'Nowe Delhi'),
(17, 'Soekarno-Hatta International Airport', 'Indonezja', 'Dzakarta'),
(18, 'Singapore Changi Airport', 'Singapur', 'Singapur'),
(19, 'Incheon International Airport', 'Korea Południowa', 'Inczon'),
(20, 'Denver International Airport', 'Stany Zjednoczone', 'Denver'),
(21, 'Warsaw Chopin Airport', 'Polska', 'Warszawa'),
(22, 'Boryspil International Airport', 'Ukraina', 'Boryspol')
```

```
INSERT INTO Cennik_lotow VALUES
```

```
(1, 3000, 9),
(1, 3500, 10),
(1, 3300, 1),
(1, 3200, 14),
(2, 4400, 17),
(2, 3900, 11),
(2, 4800, 15),
(3, 5000, 4),
(4, 4000, 17),
(4, 4500, 2),
(4, 3990, 18),
(5, 2990, 19),
(5, 3450, 20),
(6, 2500, 6),
(6, 3500, 10),
(6, 4400, 14),
(6, 3150, 15),
(7, 2000, 6),
(7, 2500, 3),
(7, 2750, 5),
(8, 4400, 11),
(9, 4500, 11),
(10, 1800, 5),
(11, 2100, 8),
(12, 2900, 7),
(13, 4100, 16),
(14, 2150, 9),
(14, 2500, 7),
(15, 3900, 13),
(15, 4100, 14),
(16, 4500, 15),
(16, 4250, 20),
(16, 4300, 19),
(17, 2900, 18),
(17, 3300, 12),
(18, 4500, 12),
```

```
(18, 4110, 13),
(18, 3400, 3),
(19, 4700, 17),
(19, 4500, 14),
(20, 3900, 9),
(20, 3600, 1),
(21, 1800, 10),
(21, 2100, 14),
(21, 2700, 12),
(22, 2500, 15),
(22, 2000, 6),
(22, 2900, 4)
```

INSERT INTO Terminale VALUES

```
('A', 'N'),
('B', 'E'),
('C', 'S'),
('D', 'W')
```

INSERT INTO Bramki VALUES

```
('A', 1),
('A', 2),
('A', 3),
('B', 1),
('B', 2),
('B', 3),
('B', 4),
('B', 5),
('C', 1),
('C', 2),
('D', 1),
('D', 2),
('D', 3),
('D', 4),
('D', 5),
('D', 6)
```

INSERT INTO Loty VALUES

```
(1000, 4, 1, NULL, 'A', 1, '2021-01-06 12:00', 'Odlot'),
(1001, 36, 1, NULL, 'A', 2, '2021-01-07 19:00', 'Odlot'),
(1002, 53, 2, NULL, 'B', 5, '2021-01-07 21:15', 'Odlot'),
(1003, 126, 2, NULL, 'C', 1, '2021-01-08 08:50', 'Odlot'),
(1004, 4, NULL, 5, 'D', 6, '2021-01-09 10:00', 'Przylot'),
(1005, 13, 3, NULL, 'C', 2, '2021-01-10 14:15', 'Odlot'),
(1006, 111, 4, NULL, 'A', 3, '2021-01-11 16:40', 'Odlot'),
(1007, 128, 4, NULL, 'A', 1, '2021-01-11 22:10', 'Odlot'),
(1008, 134, 5, NULL, 'C', 1, '2021-01-12 04:30', 'Odlot'),
(1009, 97, 6, NULL, 'B', 1, '2021-01-12 23:45', 'Odlot'),
(1010, 53, NULL, 17, 'D', 5, '2021-01-13 14:20', 'Przylot'),
(1011, 36, NULL, 22, 'D', 6, '2021-01-13 14:30', 'Przylot'),
(1012, 33, 7, NULL, 'B', 4, '2021-01-14 16:00', 'Odlot'),
(1013, 53, 8, NULL, 'A', 1, '2021-01-15 13:30', 'Odlot'),
(1014, 45, 9, NULL, 'B', 1, '2021-01-16 21:00', 'Odlot'),
(1015, 21, 10, NULL, 'C', 2, '2021-01-17 11:40', 'Odlot'),
```

```
(1016, 26, 11, NULL, 'B', 4, '2021-01-18 15:05', 'Odlot'),
(1017, 126, NULL, 21, 'D', 3, '2021-01-19 10:50', 'Przylot'),
(1018, 13, NULL, 16, 'D', 2, '2021-01-20 14:25', 'Przylot'),
(1019, 24, 12, NULL, 'A', 1, '2021-01-21 19:20', 'Odlot'),
(1020, 106, 13, NULL, 'B', 3, '2021-01-22 20:00', 'Odlot'),
(1021, 36, 14, NULL, 'C', 2, '2021-01-23 10:10', 'Odlot'),
(1022, 111, NULL, 10, 'D', 4, '2021-01-24 22:00', 'Przylot'),
(1023, 81, 15, NULL, 'C', 2, '2021-01-25 10:15', 'Odlot'),
(1024, 59, 15, NULL, 'A', 1, '2021-01-26 19:30', 'Odlot'),
(1025, 133, 16, NULL, 'B', 3, '2021-01-27 15:50', 'Odlot'),
(1026, 112, 17, NULL, 'C', 2, '2021-01-27 20:00', 'Odlot'),
(1027, 128, NULL, 4, 'D', 4, '2021-01-27 23:45', 'Przylot'),
(1028, 7, 18, NULL, 'A', 2, '2021-01-28 10:00', 'Odlot'),
(1029, 134, NULL, 22, 'D', 6, '2021-01-28 14:00', 'Przylot'),
(1030, 97, NULL, 20, 'D', 5, '2021-01-28 16:00', 'Przylot'),
(1031, 111, 19, NULL, 'B', 3, '2021-01-29 17:45', 'Odlot'),
(1032, 1, 20, NULL, 'C', 2, '2021-01-29 20:15', 'Odlot'),
(1033, 55, 21, NULL, 'A', 3, '2021-01-30 14:00', 'Odlot'),
(1034, 79, 22, NULL, 'C', 1, '2021-01-31 10:10', 'Odlot')
```

INSERT INTO Pracownicy VALUES

```
(1, 'Jan', 'Valsman', 'pilot', 3750),
(2, 'Piotr', 'Rymarczyk', 'pilot', 4000),
(3, 'Jarosław', 'Widewski', 'pilot', 3500),
(4, 'Natalia', 'Kujawska', 'pilot', 4000),
(5, 'Wiktor', 'Adamiec', 'pilot', 3000),
(6, 'Wiktoria', 'Chomicka', 'stewardessa', 2500),
(7, 'Olga', 'Chomik', 'stewardessa', 2500),
(8, 'Kunigunda', 'von Bismark', 'stewardessa', 3000),
(9, 'Fatima', 'Istaimelova', 'stewardessa', 1800),
(10, 'Szymon', 'Aleksiev', 'steward', 2500),
(11, 'Mark', 'Wisniewski', 'steward', 3000),
(12, 'Jan', 'Bura', 'celnik', 5000),
(13, 'Peter', 'Peterson', 'pilot', 5000),
(14, 'Władysław', 'Guralski', 'strażnik', 3500),
(15, 'Włodzimierz', 'Boguski', 'strażnik', 3500),
(16, 'Witold', 'Boguski', 'strażnik', 3000),
(17, 'Semuel', 'Tesla', 'strażnik', 2800),
(18, 'Irena', 'Halicka', 'celnik', 5500),
(19, 'Krzysztof', 'Hejnek', 'celnik', 4500),
(20, 'Walter', 'Adamowski', 'pilot', 3750),
(21, 'Kim', 'Sen', 'pilot', 4000),
(22, 'Aleksandra', 'Drewniowska', 'pilot', 4500),
(23, 'Lilia', 'Komarowska', 'stewardessa', 2700),
(24, 'Anna', 'Gutowska', 'stewardessa', 2500),
(25, 'Samanta', 'Iwanicka', 'stewardessa', 3000),
(26, 'Wiktoria', 'Ekielska', 'stewardessa', 2200),
(27, 'Katarzyna', 'Kozubek', 'stewardessa', 3000),
(28, 'Olga', 'Jurczyk', 'stewardessa', 3100),
(29, 'Andrzej', 'Leja', 'pilot', 4000),
(30, 'Sirius', 'Black', 'pilot', 3500),
(31, 'Jerzy', 'Murawski', 'pilot', 4200)
```

```
INSERT INTO GR_Obslugi_lotu VALUES
```

```
(1000, 1), (1000, 2),  
(1000, 6), (1000, 8),  
(1000, 9), (1001, 4),  
(1001, 3), (1001, 7),  
(1001, 10), (1001, 11),  
(1002, 13), (1002, 20),  
(1002, 5), (1002, 23),  
(1002, 24), (1003, 20),  
(1003, 21), (1003, 26),  
(1003, 27), (1004, 29),  
(1004, 27),  
(1004, 28), (1005, 1),  
(1005, 2), (1005, 6),  
(1005, 7), (1006, 3),  
(1006, 4), (1006, 8),  
(1006, 10), (1006, 11),  
(1007, 13), (1007, 20),  
(1007, 27), (1007, 28),  
(1008, 31), (1008, 30),  
(1008, 24), (1008, 25),  
(1009, 22), (1009, 20),  
(1009, 9), (1009, 27),  
(1009, 28),  
(1010, 1), (1010, 2),  
(1010, 6), (1010, 8),  
(1010, 9), (1011, 4),  
(1011, 3), (1011, 7),  
(1011, 10), (1011, 11),  
(1012, 13), (1012, 20),  
(1012, 5), (1012, 23),  
(1012, 24), (1013, 20),  
(1013, 21), (1013, 26),  
(1013, 27),  
(1014, 29), (1014, 27),  
(1014, 28), (1015, 1),  
(1015, 2), (1015, 6),  
(1015, 7), (1016, 3),  
(1016, 4), (1016, 8),  
(1016, 10), (1016, 11),  
(1017, 13), (1017, 20),  
(1017, 27), (1017, 28),  
(1018, 31), (1018, 30),  
(1018, 24), (1018, 25),  
(1019, 22), (1019, 20),  
(1019, 9), (1019, 27),  
(1019, 28),  
(1020, 1), (1020, 2),  
(1020, 6), (1020, 8),  
(1020, 9), (1021, 4),  
(1021, 3), (1021, 7),  
(1021, 10), (1021, 11),  
(1022, 13), (1022, 20),  
(1022, 5), (1022, 23),
```

```
(1022, 24), (1023, 20),
(1023, 21), (1023, 26),
(1023, 27),
(1024, 29), (1024, 27),
(1024, 28), (1025, 1),
(1025, 2), (1025, 6),
(1025, 7), (1026, 3),
(1026, 4), (1026, 8),
(1026, 10), (1026, 11),
(1027, 13), (1027, 20),
(1027, 27), (1027, 28),
(1028, 31), (1028, 30),
(1028, 24), (1028, 25),
(1029, 22), (1029, 20),
(1029, 9), (1029, 27),
(1029, 28),
(1030, 1), (1030, 2),
(1030, 6), (1030, 8),
(1030, 9), (1031, 4),
(1031, 3), (1031, 7),
(1031, 10), (1031, 11),
(1032, 13), (1032, 20),
(1032, 5), (1032, 23),
(1032, 24), (1033, 20),
(1033, 21), (1033, 26),
(1033, 27),
(1034, 29), (1034, 27),
(1034, 28)
```

INSERT INTO [Osoby niebezpieczne] VALUES

```
('FN645678' , 'Aleksandr', 'Fibonacci'),
('KK334556' , 'Jan', 'Kordemski'),
('4455284L' , 'Ann', 'Brown'),
('OP569832' , 'Otto', 'Den'),
('UU4567TR' , 'Witold', 'Kwiek'),
('FE789678' , 'Józef', 'Kuniek'),
('PR324500' , 'Aleks', 'Mederski'),
('HP345289' , 'Ludowik', 'Olivier')
```

INSERT INTO PASAZER VALUES

```
(1, 'GH454747' , 'Jan', 'Bieler', 'Niemcy'),
(2, 'GH565755' , 'Mia', 'Krause', 'Niemcy'),
(3, 'UJ746235' , 'Emma', 'Remus', 'Wielka Brytania'),
(4, 'KF84U3P2' , 'Marie', 'Cooper', 'Stany Zjednoczone'),
(5, 'KC489920' , 'Lena', 'Red', 'Wielka Brytania'),
(6, 'KD894492' , 'Felix', 'Montgomery', 'Stany Zjednoczone'),
(7, 'DL482029' , 'Max', 'Simpson', 'Stany Zjednoczone'),
(8, 'EE372024' , 'Julian', 'Oldenburg', 'Luksemburg'),
(9, 'SS203039' , 'Witold', 'Kujawinski', 'Polska'),
(10, 'DL394753' , 'Włodzimierz', 'Michałek', 'Polska'),
(11, 'DL220395' , 'Szymon', 'Kwiek', 'Polska'),
(12, 'ZE303922' , 'Olga', 'Larek', 'Polska'),
(13, 'OJ844840' , 'Jarosław', 'Kwiatkowski', 'Polska'),
(14, 'VKR49394' , 'Ivan', 'Malychev', 'Rosja'),
(15, 'CL394849' , 'Tadeusz', 'Razmus', 'Ukraina'),
```

```

(16, 'DF444002' , 'Yaryna', 'Sych', 'Ukraina'),
(17, 'AA302MF3' , 'Harry', 'Coldman', 'Wielka Brytania'),
(18, '30493MMF' , 'Lucius', 'Malfoy', 'Wielka Brytania'),
(19, '23D0E333' , 'Taras', 'Petrenko', 'Ukraina'),
(20, 'RR394203' , 'Peter', 'Oldenburg', 'Luksemburg'),
(21, 'IJ444932' , 'Olivier', 'Twen', 'Francja'),
(22, 'FC389393' , 'Caton', 'de Cavi', 'Hiszpania'),
(23, 'W0483624' , 'Jose', 'de Pola', 'Hiszpania'),
(24, 'UE356832' , 'Emiliano', 'del Pozo', 'Hiszpania'),
(25, 'KE895398' , 'Petr', 'Vasylev', 'Rosja'),
(26, 'KD493920' , 'Nikolay', 'Tarnowski', 'Bialorus'),
(27, 'PR324500' , 'Aleks', 'Mederski', 'Bialorus'),
(28, 'AD204893' , 'Ku', 'Mur', 'Chiny'),
(29, 'DP299211' , 'Kin', 'Sin', 'Korea Poludniowa'),
(30, 'XPL93484' , 'Caton', 'Orevue', 'Francja'),
(31, 'AD204894' , 'Piotr', 'Kotowski', 'Polska')

```

INSERT INTO [Klasa podrozy] VALUES

```

(1, 0.15), (2, 0.10) ,
(3, 0.05) , (4, 0.0)

```

INSERT INTO REZERWACJE VALUES

```

(0, 1000, 1, 1),
(1, 1001, 1, 2),
(2, 1001, 2, 4),
(3, 1034, 21, 1),
(4, 1005, 12, 3),
(5, 1000, 17, 2),
(6, 1001, 3, 2),
(7, 1001, 4, 4),
(8, 1023, 12, 3),
(9, 1031, 12, 3),
(10, 1007, 5, 1),
(11, 1008, 6, 2),
(12, 1009, 7, 4),
(13, 1010, 22, 1),
(14, 1011, 24, 3),
(15, 1012, 8, 2),
(16, 1011, 9, 2),
(17, 1013, 9, 4),
(18, 1012, 10, 3),
(19, 1017, 11, 3),
(20, 1000, 14, 1),
(21, 1001, 15, 2),
(22, 1001, 26, 4),
(23, 1034, 20, 1),
(24, 1005, 19, 3),
(25, 1031, 17, 2),
(26, 1001, 18, 2),
(27, 1001, 30, 4),
(28, 1023, 10, 3),
(29, 1005, 16, 3),
(30, 1007, 18, 1),
(31, 1009, 22, 2),

```



```

(32, 1001, 23, 4),
(33, 1033, 28, 1),
(34, 1015, 20, 3),
(35, 1016, 29, 2),
(36, 1018, 30, 2),
(37, 1013, 14, 4),
(38, 1010, 17, 3),
(39, 1023, 7, 3),
(40, 1034, 13, 1),
(41, 1030, 25, 3),
(42, 1000, 31, 1),
(43, 1032, 27, 3)

```

INSERT INTO BAGAZ VALUES

```

(0, 0, 5.34),
(1, 0, 12.4),
(1, 1, 2),
(3, 0, 10.6),
(4, 0, 23.12),
(5, 0, 1.56),
(5, 1, 10.3),
(5, 2, 12.4),
(7, 0, 5.3),
(9, 0, 6.4),
(10, 0, 2.2),
(11, 0, 2.2),
(12, 0, 3.4),
(12, 1, 4.0),
(14, 0, 7.8),
(15, 0, 10.1),
(16, 0, 9.8),
(16, 1, 23.45),
(18, 0, 24.8),
(19, 0, 11.0),
(20, 0, 9.7),
(21, 0, 18.9),
(21, 1, 78),
(23, 0, 5.5),
(24, 0, 3.9),
(25, 0, 4.5),
(25, 1, 4.5),
(25, 2, 5.6),
(25, 3, 6.1),
(29, 0, 5.9),
(30, 0, 4.9),
(31, 0, 7.8),
(31, 1, 8.1),
(33, 0, 7),
(34, 0, 6.7),
(35, 0, 8),
(35, 1, 3.4),
(37, 0, 13.9),
(37, 1, 2.1),
(39, 0, 23.2),

```

```
(40, 0, 44.77),  
(41, 0, 7.9),  
(41, 1, 8.0),  
(43, 0, 1.2)
```

```
--Powyżej 100 nie obsługujemy w pasazerskich samolotach
```

```
INSERT INTO CENNIK_BAGAZY VALUES
```

```
(1, 15) , (5, 20),  
(10, 30) , (15, 45),  
(20, 50) , (25, 75),  
(30, 100), (35, 125),  
(45, 150), (60, 175),  
(80, 200), (100, 250)
```

Funkcje

Funkcja dbo.F_gr_obsługi_lotu

Funkcja typu INLINE przyjmująca w argumencie id lotu zwraca tabelę, która zawiera informacje o osobach obsługujących dany lot. Czyli informacje o tym, kto jest pilotem, stewardessą/stewardem. Wyświetlane są dane: id lotu, id pracownika, stanowisko jakie zajmuje dany pracownik oraz jego imię i nazwisko.

```
IF OBJECT_ID('dbo.F_gr_obsługi_lotu', 'IF') IS NOT NULL
    DROP FUNCTION dbo.F_gr_obsługi_lotu
GO
CREATE FUNCTION F_gr_obsługi_lotu (@Id_lotu INT)
RETURNS TABLE
AS
RETURN (
    SELECT GR_Obsługi_lotu.*, Stanowisko, Imie, Nazwisko
    FROM GR_Obsługi_lotu
    INNER JOIN Pracownicy
    ON Pracownicy.ID_Pracownika = GR_Obsługi_lotu.Id_Pracownika
    WHERE Id_lotu = @Id_lotu
)
GO
-----
SELECT * FROM F_gr_obsługi_lotu(1001)
```

Funkcja dbo.F_polaczenia_przewoźnika

Funkcja typu INLINE przyjmująca w argumencie nazwę przewoźnika zwraca tabelę, która zawiera informacje o połączeniach przewoźnika danego argumentem. Czyli zwraca lotniska do których lata dany przewoźnik. W kolumnach tabeli jest odpowiednio nazwa lotniska, państwo, miasto w którym znajduje się wspomniane lotnisko oraz cena lotu.

```
IF OBJECT_ID('dbo.F_polaczenia_przewoźnika', 'IF') IS NOT NULL
    DROP FUNCTION dbo.F_polaczenia_przewoźnika
GO
CREATE FUNCTION F_polaczenia_przewoźnika (@Nazwa NVARCHAR(50))
RETURNS TABLE
AS
RETURN (
    SELECT Nazwa_lotniska, Polaczenia_z_lotniskami.Panstwo,
    Miasto, Cena FROM Cennik_lotow
```

```

        INNER JOIN Przewoznicy ON Przewoznicy.Id_przewoznika =
        Cennik_lotow.Id_przewoznika
        INNER JOIN Polaczenia_z_lotniskami ON
        Polaczenia_z_lotniskami.Id_lotniska = Cennik_lotow.Dokad
        WHERE Przewoznicy.Nazwa LIKE @Nazwa
    )
GO
-----
SELECT * FROM F_polaczenia_przewoznika('Ryanair')

```

Funkcja dbo.F_Polaczenia_do_lotniska

Funkcja typu INLINE, przyjmująca w argumencie id lotniska. Efektem działania funkcji jest zwrócenie zestawu wierszy, zawierających nazwy przewoźnika oraz ceny danego przewoźnika w połączeniach do lotniska podanego w argumencie.

```

IF OBJECT_ID('dbo.F_Polaczenia_do_lotniska', 'IF') IS NOT NULL
    DROP FUNCTION dbo.F_Polaczenia_do_lotniska
GO
CREATE FUNCTION F_Polaczenia_do_lotniska (@Id_lotniska INT)
RETURNS TABLE
AS
RETURN (
    SELECT Nazwa, Cena FROM Cennik_lotow
    INNER JOIN Przewoznicy
    ON Przewoznicy.Id_przewoznika = Cennik_lotow.Id_przewoznika
    WHERE Dokad = @Id_lotniska
)
GO
-----
SELECT * FROM F_Polaczenia_do_lotniska(1)

```

Funkcja dbo.F_Samoloty_przewoznika

Funkcja typu INLINE, przyjmująca w argumencie id przewoźnika. Efektem działania funkcji jest zwrócenie zestawu wierszy, zawierających numery i modele samolotów wraz z ich pojemnością (liczbą dostępnych miejsc) danego przewoźnika.

```

IF OBJECT_ID('dbo.F_Samoloty_przewoznika', 'IF') IS NOT NULL
DROP FUNCTION dbo.F_Samoloty_przewoznika
GO
CREATE FUNCTION dbo.F_Samoloty_przewoznika (@Id_przewoznika INT)

```

```

RETURNS TABLE
AS
RETURN (
    SELECT Id_samolotu, Samoloty.Marka, Samoloty.Model,
           Samoloty_szczegoly.Ilosc_miejsc
    FROM Samoloty
    JOIN Samoloty_szczegoly
    ON (Samoloty_szczegoly.Model = Samoloty.Model
        AND Samoloty_szczegoly.Marka = Samoloty.Marka)
    WHERE Id_przewoznika = @Id_przewoznika
)
GO
-----
SELECT * FROM F_Samoloty_przewoznika(19)

```

Funkcja skalarna (pomocnicza) dbo.Czy_samolot_na_lotnisku

Funkcja przyjmuje w argumentach numer samolotu oraz datę. Funkcja zwraca 'NIE' jeśli danego samolotu o danej dacie nie ma na lotnisku. Jeśli samolot przebywa na lotnisku o danej porze to funkcja zwraca 'TAK'.

```

IF OBJECT_ID('dbo.Czy_samolot_na_lotnisku', 'FN') IS NOT NULL
    DROP FUNCTION dbo.Czy_samolot_na_lotnisku
GO

CREATE FUNCTION Czy_samolot_na_lotnisku (@Id_samolotu INT,
                                         @Data DATETIME)
RETURNS VARCHAR(3)
AS
BEGIN
    DECLARE @Ilosc_lotow INT
    DECLARE @Out VARCHAR(3)
    SET @Ilosc_lotow = (SELECT COUNT(*) FROM Loty
                        WHERE Id_samolotu = @Id_samolotu
                           AND Data <= @Data)

    IF @Ilosc_lotow = 0
        SET @Out = 'TAK'
    ELSE
        BEGIN
            IF (SELECT TOP 1 Typ_lotu FROM Loty
                WHERE Id_samolotu = @Id_samolotu
                   AND Data <= @Data
                   ORDER BY Data DESC) = 'Przylot'
                SET @Out = 'TAK'
        END
    END

```

```

ELSE
    SET @Out = 'NIE'
END
RETURN @Out
END
GO
-----
SELECT dbo.Czy_samolot_na_lotnisku(36, '2021-01-15 13:30')
SELECT dbo.Czy_samolot_na_lotnisku(134, '2021-01-15 13:30')

```

Funkcja skalarna dbo.Ilosc_rezerwacji_na_lot

Funkcja przyjmuje jako argument identyfikator lotu i dla tego lotu zwraca ilość wykonanych na niego rezerwacji.

```

IF Object_ID('dbo.Ilosc_rezerwacji_na_lot' , 'FN') is not null
Drop function    dbo.Ilosc_rezerwacji_na_lot

Go
Create Function Ilosc_rezerwacji_na_lot (@Lot INT)
Returns INT
As
BEGIN
    DECLARE @Ilosc_rezerwacji INT
    Set @Ilosc_rezerwacji =(Select Count(R.Id_Rezerwacji) from
                           Rezerwacje R where R.Id_lotu = @Lot)

    Return @Ilosc_rezerwacji
END
GO
Select dbo.Ilosc_rezerwacji_na_lot(1000)

```

Funkcja skalarna dbo.Cena_sumaryczna_rezerwacji

Funkcja przyjmuje jako argument identyfikator rezerwacji i dla podanej rezerwacji wylicza jej sumaryczną cenę. W cenę sumaryczną rezerwacji wlicza się cena za lot, czyli cena za połączenie wystawiona przez linię lotniczą, która wykonuje dany lot, oraz cena za przewóz bagażu przypisanego do danej rezerwacji. Ponadto cena za połączenie zostaje zwiększona na odpowiednią ilość procent w zależności od klasy podróży wskazanej w rezerwacji. Cena za bagaż jest wystawiana na podstawie cennika bagażu. Za wagę bagażu przyjmujemy sumaryczną wagę wszystkich bagażu, przypisanych do danej rezerwacji.

Jeżeli rezerwacja ukazana w argumencie nie istnieje lub jest to rezerwacja na przylot do naszego lotniska zwracana jest wartość 0.00.

```
IF Object_ID('dbo.Cena_sumaryczna_rezerwacji' , 'FN') is not null
Drop function    dbo.Cena_sumaryczna_rezerwacji

GO
Create Function Cena_sumaryczna_rezerwacji (@rezerwacja INT)
Returns MONEY
AS
BEGIN
    DECLARE @CENA MONEY
    DECLARE @Typ_lotu NVARCHAR(10)
    SET @Typ_lotu = (Select L.Typ_lotu from Rezerwacje R join Loty L on
                        R.Id_lotu = L.Id_lotu
                        where R.Id_Rezerwacji = @rezerwacja)
    IF @Typ_lotu <> 'Odlot' or @Typ_lotu is NULL
        Set @CENA = 0
    ELSE
        BEGIN
            DECLARE @CENA_polaczenia MONEY
            DECLARE @Przewoznik INT
            DECLARE @Dokad INT
            SET @Przewoznik = (Select S.Id_przewoznika  from Rezerwacje R
                                join Loty L on R.Id_lotu = L.Id_lotu
                                join Samoloty S on L.Id_samolotu = S.Id_samolotu
                                where R.Id_Rezerwacji = @rezerwacja)

            SET @Dokad = (Select L.Dokad  from Rezerwacje R join Loty L on
                            R.Id_lotu = L.Id_lotu
                            where R.Id_Rezerwacji = @rezerwacja)

            SET @CENA_polaczenia = (Select C.Cena from Cennik_lotow C where
                                    C.Dokad = @Dokad and C.Id_przewoznika = @Przewoznik)

            DECLARE @CENA_bagazy MONEY
            DECLARE @WAGA real
            SET @Waga = (Select Sum(B.Waga) from Bagaz B where
                            B.Id_Rezerwacji = @rezerwacja)

            SET @Cena_bagazy = (Select MIN(C.Cena) from Cennik_Bagazy C where
                                C.Waga > @WAGA)

            DECLARE @Prowizja real
            SET @Prowizja = (Select K.Prowizja from Rezerwacje R join
                            [Klasa podrozy] K on R.Klasa_podrozy = K.Klasa
```

```

        where R.Id_Rezerwacji = @rezerwacja)

SET @CENA = (@CENA_polaczenia*(1+@Prowizja) + @CENA_bagazy)
END

RETURN @CENA
END
GO

Select dbo.Cena_sumaryczna_rezerwacji(1)
Select dbo.Cena_sumaryczna_rezerwacji(13)
Select dbo.Cena_sumaryczna_rezerwacji(15)

```


Procedury

Procedura [Przesun_czas_lotu(o minut)]

Procedura przesuująca czas lotu o id podanym w argumencie, o liczbę minut podaną w argumencie. Procedura sprawdza ponadto czy numer lotu podanego w argumencie istnieje w tabeli Loty, oraz czy liczba minut jest liczbą dodatnią. Sprawdzana jest również informacja czy dana bramka będzie wolna o nowym czasie odlotu/przylotu, oraz zostanie odpowiednia ilość czasu na odprawienie samolotu. Zakładamy, że czas potrzebny na obsługę samolotu przy bramce wynosi 30 minut.

```
IF OBJECT_ID('Przesun_czas_lotu(o minut)', 'P') IS NOT NULL
    DROP PROC [Przesun_czas_lotu(o minut) ]
GO

CREATE PROC [Przesun_czas_lotu(o minut)] (
    @Id_lotu INT,
    @Ilosc_minut INT
)
AS
    BEGIN TRY
        IF(NOT EXISTS(SELECT * FROM Loty WHERE Id_lotu = @Id_lotu))
            BEGIN
                RAISERROR('Nie istnieje lot o nr %i', 16, 1, @Id_lotu)
            END

        IF (@Ilosc_minut <= 0) BEGIN
            RAISERROR('Ilosc minut nie jest liczba dodatnia', 16,
                1)
        END

        DECLARE @New_date DATETIME = DATEADD(MINUTE, @Ilosc_minut,
            (SELECT Data FROM Loty WHERE Id_lotu = @Id_lotu))

        --sprawdzanie czy bramka nie bedzie zajeta w tym czasie
        --czas potrzebny na obsluge samolotu przy bramce wynosi 30 minut

        DECLARE @Id_terminalu NVARCHAR(1) = (SELECT Id_terminalu
            FROM Loty WHERE Id_lotu = @Id_lotu)

        DECLARE @Nr_bramki INT = (SELECT Nr_bramki FROM Loty WHERE
            Id_lotu = @Id_lotu)

        DECLARE @Potential_flights INT = (SELECT COUNT(*)
```

```

FROM Loty
WHERE Id_terminalu = @Id_terminalu
AND Nr_bramki = @Nr_bramki AND Id_lotu <> @Id_lotu
AND Data > DATEADD(MINUTE, -30, @New_date) AND Data <
DATEADD(MINUTE, 30, @New_date))

IF (@Potential_flights > 0) BEGIN
    RAISERROR('Kolizja z bramkami, poprzedni samolot nie
    zdazy opuscic bramki', 16, 1)
END

UPDATE Loty
SET Data = @New_date
WHERE Id_lotu = @Id_lotu

END TRY
BEGIN CATCH
    SELECT
        ERROR_NUMBER() AS ErrorNumber,
        ERROR_SEVERITY() AS ErrorSeverity,
        ERROR_STATE() AS ErrorState,
        ERROR_PROCEDURE() AS ErrorProcedure,
        ERROR_LINE() AS ErrorLine,
        ERROR_MESSAGE() AS ErrorMessage;
END CATCH

EXEC [Przesun_czas_lotu(o minut)]
    @Id_lotu = 1021,
    @Ilosc_minut = 2880

```

Procedura Skasuj_bagaz

Procedura przyjmuje dwie wartości podane w argumentach id rezerwacji oraz id bagażu. Efektem działania procedury jest usunięcie podanego w argumencie bagażu z danej rezerwacji. W ciele programu sprawdzamy czy istnieje bagaż o podanym numerze.

```

IF OBJECT_ID('Skasuj_bagaz', 'P') IS NOT NULL
    DROP PROC Skasuj_bagaz
GO

CREATE PROC Skasuj_bagaz (
    @Id_rezerwacji INT,

```

```

        @Id_bagazu INT
    )
AS
BEGIN TRY
    IF(NOT EXISTS(SELECT * FROM Bagaz
    WHERE Id_rezerwacji = @Id_rezerwacji
    AND Id_bagazu = @Id_bagazu)) BEGIN
        RAISERROR('Nie ma bagazu nr %i w rezerwacji nr %i', 16,
        1, @Id_bagazu, @Id_rezerwacji)
    END

    DELETE FROM Bagaz
    WHERE Id_rezerwacji = @Id_rezerwacji
    AND Id_bagazu = @Id_bagazu
END TRY
BEGIN CATCH
    SELECT
        ERROR_NUMBER() AS ErrorNumber,
        ERROR_SEVERITY() AS ErrorSeverity,
        ERROR_STATE() AS ErrorState,
        ERROR_PROCEDURE() AS ErrorProcedure,
        ERROR_LINE() AS ErrorLine,
        ERROR_MESSAGE() AS ErrorMessage;
END CATCH

EXEC Skasuj_bagaz
    @Id_rezerwacji = 5,
    @Id_bagazu = 1

```

Procedura Zmien_samolot_obslugujacy_lot

Procedura przyjmuje w argumentach id lotu, oraz id nowego samolotu. Efektem działania procedury jest zmiana samolotu na id nowego samolotu w locie o numerze podanym argumentem. W ciele procedury sprawdzane są następujące warunki:

- czy nowy samolot jest tej samej linii lotniczej co stary. Nie może być sytuacji, że nowy samolot jest innej linii lotniczej, gdyż każdy przewoźnik ma własną cenę za lot do danego miejsca. Poza tym nie każdy przewoźnik ma połączenia z każdym lotniskiem,
- czy nowy samolot w chwili wykonania lotu będzie przebywał na lotnisku. Nie może być sytuacji, że samolot nie przebywa na lotnisku (wykonał odlot), a mimo to zatwierdziliśmy go w naszej procedurze jako samolot zastępczy,

- czy nowy samolot będzie miał wystarczająco czasu by wrócić na swój zaplanowany lot. Zakładamy, że samolot od odlotu potrzebuje równo 2 dni, by znów wrócić na lotnisko i być gotowym do ponownego startu.

Ponadto w chwili zmiany samolotu na nowy. Stary samolot jeśli posiada zaplanowany przylot w tabeli Loty, to automatycznie ten przylot zostaje skasowany, gdyż samolot nie odleciał z lotniska więc nie ma podstaw do tego by mógł na nie przylecieć. Natomiast w tabeli Loty, pojawia się przylot nowego samolotu z miejsca w które poleciał.

```
IF OBJECT_ID('Zmien_samolot_obsługujacy_lot', 'P') IS NOT NULL
    DROP PROC Zmien_samolot_obsługujacy_lot
GO

CREATE PROC Zmien_samolot_obsługujacy_lot (
    @Id_lotu INT,
    @Id_nowego_samolotu INT
)
AS
    BEGIN TRY
        DECLARE @Id_linia_lotnicza INT
        --pobranie daty lotu nad którym pracujemy
        DECLARE @Data_lotu DATETIME = (SELECT Data FROM Loty
        WHERE Id_lotu = @Id_lotu)

        SET @Id_linia_lotnicza = (SELECT Id_przewoźnika FROM Loty
        INNER JOIN Samoloty
        ON Samoloty.Id_samolotu = Loty.Id_samolotu
        WHERE Id_lotu = @Id_lotu)

        IF (NOT EXISTS(SELECT * FROM Samoloty
        WHERE Id_samolotu = @Id_nowego_samolotu
        AND Id_przewoźnika = @Id_linia_lotnicza)) BEGIN
            RAISERROR('Nowy samolot nie jest tej samej linii
            lotniczej co stary', 16,1)
        END

        DECLARE @Date VARCHAR(30) = CAST(@Data_lotu AS
        VARCHAR)

        IF
            ((SELECT dbo.Czy_samolot_na_lotnisku(@Id_nowego_samolotu,
            @Data_lotu)) = 'NIE') BEGIN
```

```

        RAISERROR('Samolot o id %i, nie przebywa na naszym
lotnisku w dniu %s', 16, 1, @Id_nowego_samolotu,
@DateVARCHAR)
    END

--usuniecie ewentualnego przylotu zmienianego(starego) samolotu

    DECLARE @Id_starego_samolotu INT = (SELECT Id_samolotu FROM
Loty WHERE Id_lotu = @Id_lotu)

    DECLARE @Data_przylotu DATETIME = (SELECT Data FROM Loty
WHERE Id_lotu = @Id_lotu)

    DECLARE @Nr_lotu_przylotu INT = (SELECT TOP 1 Id_lotu
FROM Loty WHERE Id_samolotu = @Id_starego_samolotu
AND Data > @Data_przylotu)

--Sprawdzenie czy lot jest naprawde przylotem(gdyby byl odlot to cos
poszlo nie tak przy dodawaniu lotow)

    IF ((SELECT Typ_lotu FROM Loty WHERE Id_lotu =
@Nr_lotu_przylotu) <> 'Przylot') BEGIN
        RAISERROR('Lot nr %i powinien byc Przylotem a nim nie
jest', 16, 1, @Nr_lotu_przylotu)
    END

--nastepny lot nowego samolotu, musimy sprawdzic czy samolot bedzie
mial czas by wrócic na swój zaplanowany lot
--zakladamy, ze samolot bedzie potrzebował pełnych 2 dni, czyli 48
godzin

    DECLARE @Data_nastepnego_lot_nowego DATETIME = (SELECT TOP 1
Data FROM Loty WHERE Id_samolotu = @Id_nowego_samolotu
AND Data > @Data_lotu ORDER BY Data ASC)

    IF ((SELECT DATEDIFF(HOUR, @Data_lotu,
@Data_nastepnego_lot_nowego)) < 48) BEGIN
        RAISERROR('Nowy samolot nie bedzie mial wystarczająco
czasu by wrócic na swój zaplanowany lot.', 16, 1)
    END

--Kasowanie tego przylotu, skoro samolot nie polecil to nie może
przyleciec bo będzie na lotnisku cały czas

    IF (@Nr_lotu_przylotu IS NOT NULL) BEGIN
        DELETE FROM Loty
        WHERE Id_lotu = @Nr_lotu_przylotu
    END

```

```

END

--update na nowy samolot zmienianego lotu
UPDATE Loty
SET Id_samolotu = @Id_nowego_samolotu
WHERE Id_lotu = @Id_lotu

--dodanie przylotu nowego samolotu z lotu nad którym pracowalismy, do
tabeli loty

DECLARE @Last_flight_nr INT = (SELECT MAX(Id_lotu)
FROM Loty)

DECLARE @Skad INT = (SELECT Dokad FROM Loty
WHERE Id_lotu = @Id_lotu)

DECLARE @Arrival_time DATETIME = DATEADD(HOUR, 48,
@Data_lotu)

INSERT INTO Loty VALUES
(@Last_flight_nr+1, @Id_nowego_samolotu, NULL, @Skad, 'D',
6, @Arrival_time, 'Przylot')

END TRY
BEGIN CATCH
SELECT
ERROR_NUMBER() AS ErrorNumber,
ERROR_SEVERITY() AS ErrorSeverity,
ERROR_STATE() AS ErrorState,
ERROR_PROCEDURE() AS ErrorProcedure,
ERROR_LINE() AS ErrorLine,
ERROR_MESSAGE() AS ErrorMessage;
END CATCH

```

Procedura Wypowiedz_umowe_z_przewoźnikiem

Procedura w argumencie przyjmuje id przewoźnika. Efektem działania procedury jest zerwanie umowy z danym przewoźnikiem z chwilą natychmiastową. W tabeli Umowy_z_przewoźnikami we wierszu odpowiadającym id przewoźnika w kolumnie Do_kiedy wpisywana jest aktualna data. Z chwilą wypowiedzenia umowy, automatycznie z tabeli Loty kasowane są wszystkie zaplanowane loty, które miał obsługiwać dany przewoźnik.

```

IF OBJECT_ID('Wypowiedz_umowe_z_przewoznikiem', 'P') IS NOT NULL
    DROP PROC Wypowiedz_umowe_z_przewoznikiem
GO

--wypowiedzenia następuje z chwilą natychmiastową
--w chwili wypowiedzenia zaplanowane loty zostają automatycznie
wykasowane

CREATE PROC Wypowiedz_umowe_z_przewoznikiem (
    @Id_przewoznika INT
)
AS
    BEGIN TRY
        IF(NOT EXISTS(SELECT * FROM Przewoznicy
            WHERE Id_przewoznika = @Id_przewoznika)) BEGIN
            RAISERROR('Nie istnieje przewoźnik o id %i', 16, 1,
                @Id_przewoznika)
        END

        --wpisanie aktualnej daty do tabeli z umowami w miejscu daty do kiedy
        przy odpowiednim przewoźniku

        UPDATE Umowy_z_przewoznikami
        SET Do_kiedy = CURRENT_TIMESTAMP
        WHERE Id_przewoznika = @Id_przewoznika

        --wykasowanie lotów przeprowadzanych przez danego przewoźnika z tabeli
        loty

        DELETE FROM Loty
        WHERE Id_samolotu IN (SELECT Id_samolotu FROM Samoloty WHERE
            Id_przewoznika = @Id_przewoznika)
        AND Data > CURRENT_TIMESTAMP

    END TRY
    BEGIN CATCH
        SELECT
            ERROR_NUMBER() AS ErrorNumber,
            ERROR_SEVERITY() AS ErrorSeverity,
            ERROR_STATE() AS ErrorState,
            ERROR_PROCEDURE() AS ErrorProcedure,
            ERROR_LINE() AS ErrorLine,
            ERROR_MESSAGE() AS ErrorMessage;
    END CATCH

EXEC Wypowiedz_umowe_z_przewoznikiem
    @Id_przewoznika = 1

```

Procedura [Przedluz_umowe_z_przewoznikiem(o miesiecy)]

Procedura w argumentach przyjmuje id przewoźnika, oraz ilość miesięcy. Efektem działania procedury jest przedłużenie umowy z danym przewoźnikiem o zadaną ilość miesięcy. Standardowo w ciele procedury sprawdzane są warunki czy dany przewoźnik istnieje oraz czy liczba miesięcy jest liczbą dodatnią.

```
IF OBJECT_ID('Przedluz_umowe_z_przewoznikiem(o miesiecy)', 'P') IS NOT
NULL
    DROP PROC [Przedluz_umowe_z_przewoznikiem(o miesiecy)]
GO

--przedłużenie następuje o podaną w argumencie ilość miesięcy
CREATE PROC [Przedluz_umowe_z_przewoznikiem(o miesiecy)] (
    @Id_przewoznika INT,
    @Ilosc_miesiecy INT
)
AS
    BEGIN TRY
        IF(NOT EXISTS(SELECT * FROM Przewoznicy WHERE Id_przewoznika
        = @Id_przewoznika)) BEGIN
            RAISERROR('Nie istnieje przewoznik o id %i', 16, 1,
                @Id_przewoznika)
        END

        IF (@Ilosc_miesiecy <= 0) BEGIN
            RAISERROR('Ilosc miesiecy nie jest liczba dodatnia',
                16, 1)
        END

        UPDATE Umowy_z_przewoznikami
        SET Do_kiedy = DATEADD(MONTH, 12, Do_kiedy)
        WHERE Id_przewoznika = @Id_przewoznika

    END TRY
    BEGIN CATCH
        SELECT
            ERROR_NUMBER() AS ErrorNumber,
            ERROR_SEVERITY() AS ErrorSeverity,
            ERROR_STATE() AS ErrorState,
            ERROR_PROCEDURE() AS ErrorProcedure,
            ERROR_LINE() AS ErrorLine,
            ERROR_MESSAGE() AS ErrorMessage;
    END CATCH
```



```
EXEC [Przedluz_umowe_z_przewoznikiem(o miesiecy)]
    @Id_przewoznika = 1,
    @Ilosc_miesiecy = 12
```

Procedura Usun_rezerwacje

Procedura usuwa rezerwację o podanym identyfikatorze, który przyjmuje jako argument. Obsługa błędów jest organizowana przy pomocy konstrukcji TRY-CATCH. Sprawdzane są dwa warunki :

- istnienie rezerwacji o podanym identyfikatorze,
- czy lot, na który została złożona rezerwacja, którą się usuwa, jeszcze się nie odbył (nie wolno usuwać informacji o rezerwacjach na loty, które już się odbyły).

W przypadku niespełnienia chociaż jednego z tych warunków jest wyświetlany komunikat o błędzie oraz żadne konsekwencje działania procedury nie następują. W przeciwnym przypadku zostaje usuwana informacja o rezerwacji o podanym identyfikatorze oraz kaskadowo o bagażu do niej przypisanym.

```
IF OBJECT_ID('Usun_rezerwacje', 'P') is not null
DROP PROC Usun_rezerwacje

GO
CREATE PROCEDURE Usun_rezerwacje (
    @ID_Rezerwacji INT
)
AS
BEGIN TRY
    IF NOT EXISTS (SELECT ID_rezerwacji From Rezerwacje where
                    ID_rezerwacji = @ID_Rezerwacji)
        RAISERROR('Rezerwacja jaką należy usunąć ( o id %i ) nie
istnieje', 16, 1, @ID_Rezerwacji)

    DECLARE @DATA DATETIME
    SET @DATA = (SELECT DATA FROM Rezerwacje R join Loty L on
                    R.Id_lotu = L.Id_lotu
                    where R.Id_Rezerwacji = @ID_Rezerwacji)

    IF @DATA < CURRENT_TIMESTAMP
        RAISERROR('Nie można usuwać rezerwacji na lot który już się
odbył', 16, 1)
```

```

        DELETE FROM Rezerwacje
        where Id_Rezerwacji = @ID_Rezerwacji
    END TRY
    BEGIN CATCH
    DECLARE @ErrorMessage VARCHAR(4000)
    DECLARE @ErrorSeverity INT
    DECLARE @ErrorState INT
    SET @ErrorMessage = ERROR_MESSAGE()
    SET @ErrorSeverity = ERROR_SEVERITY()
    SET @ErrorState = ERROR_STATE()
    RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState)
    END CATCH
GO

```

Procedura Zmien_bramke

Procedura zmiany bramki i/lub terminalu do której przylatuje / od której odlatuje lot o podanym identyfikatorze. Argumentami procedury są id Lotu, id nowego terminalu przylotu/odlotu oraz numer nowej bramki przylotu/odlotu. Przed zmianą bramki i/lub terminalu dla danego lotu są sprawdzane następujące warunki :

- istnienie wskazanego terminalu oraz wskazanej bramki w tym terminale,
- istnienie wskazanego lotu,
- czy wskazany lot jeszcze się nie odbył ,
- jeżeli lot o podanym jako argument procedury id jest odlotem, to sprawdzamy czy nowa bramka jest wolna 2 godziny przed, oraz 30 minut po czasie odlotu.
- jeżeli lot o podanym jako argument procedury id jest przylotem, to sprawdzamy czy nowa bramka jest wolna 30 minut do oraz 1 godzinę po czasie przylotu.

W przypadku niespełnienia chociaż jednego z tych warunków jest wyświetlany komunikat o błędzie oraz żadne konsekwencje działania procedury nie następują. Obsługa błędów jest organizowana przy pomocy konstrukcji TRY-CATCH. W przeciwnym razie zostaje zmieniona informacja o bramce i/lub terminalu odlotu/przylotu.

```

IF OBJECT_ID('Zmien_bramke', 'P') is not null
DROP PROC Zmien_bramke

GO
CREATE PROCEDURE Zmien_bramke (

```

```

@ID_lotu INT,
@Nowy_terminal VARCHAR(1),
@Nowa_bramka INT
)
AS
BEGIN TRY
    IF NOT EXISTS (SELECT Id_lotu FROM LOTY WHERE Id_lotu = @ID_lotu)
        RAISERROR ('Nie istnieje Lot o id %i', 16, 1, @ID_lotu)

    IF NOT EXISTS (SELECT Id_terminalu FROM Terminale
        WHERE Id_terminalu = @Nowy_terminal)
        RAISERROR ('Nie istnieje Terminal o id %s', 16, 1, @Nowy_terminal)

    IF NOT EXISTS (SELECT Nr_bramki FROM Bramki WHERE Id_terminalu =
        @Nowy_terminal and Nr_bramki = @Nowa_bramka)
        RAISERROR ('Nie istnieje Bramka o numerze %i w terminale %s', 16, 1,
            @Nowa_bramka, @Nowy_terminal)

    DECLARE @DATA DATETIME
    SET @DATA = (SELECT DATA from Loty where Id_lotu = @ID_lotu)

    IF @DATA < CURRENT_TIMESTAMP
        RAISERROR('Lot o id %i juz się odbył, nie można zmieniać dane archiwum',
            16, 1 , @ID_lotu)

    DECLARE @Typ_lotu VARCHAR(12)
    SET @Typ_lotu = (Select Typ_lotu from Loty where Id_lotu = @ID_lotu)

    IF @Typ_lotu = 'Odlot'
    BEGIN
        IF EXISTS (SELECT ID_Lotu FROM LOTY where ID_terminalu =
            @Nowy_terminal and Nr_bramki = @Nowa_bramka
            and (Data BETWEEN DATEADD(HOUR, -2, @DATA) and
                DATEADD(MINUTE, 30, @DATA)))
            RAISERROR ('Bramka o numerze %i w terminale %s jest zajęta w czasie
                potrzebnym dla obsługi lotu o id %i', 16, 1,
                    @Nowa_bramka, @Nowy_terminal, @Id_lotu)
    END
    ELSE
    BEGIN
        IF EXISTS (SELECT ID_Lotu FROM LOTY where ID_terminalu =
            @Nowy_terminal and Nr_bramki = @Nowa_bramka
            and (Data BETWEEN DATEADD(MINUTE, -30, @DATA) and
                DATEADD(HOUR, 1, @DATA)))
            RAISERROR ('Bramka o numerze %i w terminale %s jest zajęta w czasie
                potrzebnym dla obsługi lotu o id %i', 16, 1,
                    @Nowa_bramka, @Nowy_terminal, @Id_lotu)
    END
    UPDATE LOTY
    SET Nr_Bramki = @Nowa_bramka, ID_terminalu = @Nowy_terminal
    where Id_lotu = @ID_lotu
END TRY
BEGIN CATCH
    DECLARE @ErrorMessage VARCHAR(4000)
    DECLARE @ErrorSeverity INT

```

```

        DECLARE @ErrorState INT
        SET @ErrorMessage = ERROR_MESSAGE()
        SET @ErrorSeverity = ERROR_SEVERITY()
        SET @ErrorState = ERROR_STATE()
        RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState)
    END CATCH
GO

EXEC Zmien_bramke @ID_Lotu = 1067 , @Nowy_Terminal = 'A', @Nowa_bramka = 2
EXEC Zmien_bramke @ID_Lotu = 1002 , @Nowy_Terminal = 'F', @Nowa_bramka = 2
EXEC Zmien_bramke @ID_Lotu = 1002 , @Nowy_Terminal = 'B', @Nowa_bramka = 8
EXEC Zmien_bramke @ID_Lotu = 1067 , @Nowy_Terminal = 'A', @Nowa_bramka = 2

INSERT INTO LOTY VALUES
(1035, 81, 9, null, 'A', 3, '2021-01-31 10:20:00.000' , 'Odlot')

EXEC Zmien_bramke @ID_Lotu = 1035 , @Nowy_Terminal = 'C', @Nowa_bramka = 1
EXEC Zmien_bramke @ID_Lotu = 1034 , @Nowy_Terminal = 'D', @Nowa_bramka = 5

SELECT * FROM LOTY
DELETE FROM LOTY where Id_lotu = 1035

```

Wyzwalacze

Trigger TR_AU_rezerwacja

Trigger typu AFTER UPDATE zdefiniowany dla tabeli Rezerwacje. Zadaniem triggera jest niedopuszczanie do zaktualizowania rezerwacji danego pasażera, na sytuację taką w której dany pasażer miałby zrobione rezerwacje na dwa loty odbywające się w tym samym czasie.

```
IF OBJECT_ID('Tr_AU_rezerwacja', 'TR') IS NOT NULL
DROP TRIGGER Tr_AU_rezerwacja
GO
CREATE TRIGGER Tr_AU_rezerwacja ON Rezerwacje
AFTER UPDATE
AS
    DECLARE @Id_pasazera INT = (SELECT Id_pasazera FROM Deleted)

    DECLARE @Id_rezerwacji INT = (SELECT Id_rezerwacji
    FROM Deleted)

    DECLARE @Id_starego_lotu INT = (SELECT Id_lotu FROM Deleted)

--dane zaktualizowanej rezerwacji

    DECLARE @Id_nowego_lotu INT = (SELECT Id_lotu FROM Rezerwacje
    WHERE Id_Rezerwacji = @Id_rezerwacji)

    DECLARE @Data_nowego_lotu DATETIME = (SELECT Data FROM Loty
    JOIN Rezerwacje ON Rezerwacje.Id_lotu = Loty.Id_lotu
    WHERE Rezerwacje.Id_lotu = @Id_nowego_lotu)

--sprawdzenie czy data nowego lotu nie koliduje z innymi rezerwacjami
danego pasażera przyjmujemy, że data nie koliduje gdy odstępy między
lotami wynoszą przynajmniej 4 godziny

    DECLARE @Ilosc_lotow INT -- ilosc lotow danego pasazera w
    przedziale 4 godzinny względem zaktualizowanej rezerwacji

    SET @Ilosc_lotow = (SELECT COUNT(*) FROM Rezerwacje
    JOIN Loty
    ON Loty.Id_lotu = Rezerwacje.Id_lotu
    WHERE Id_pasazera = @Id_pasazera
    AND Data < DATEADD(HOUR, 4, @Data_nowego_lotu)
    AND Data > DATEADD(HOUR, -4, @Data_nowego_lotu))
```

```

IF (@Ilosc_lotow <> 1) BEGIN
    PRINT('Dany pasazer ma juz rezerwacje na lot w tym samym
        czasie')

    --zatem przywracamy rezerwacje
    UPDATE Rezerwacje
    SET Id_lotu = @Id_starego_lotu
    WHERE Id_Rezerwacji = @Id_rezerwacji
END
GO
---
```

```

INSERT INTO Rezerwacje VALUES
(42, 1005, 1, 1)

UPDATE Rezerwacje
SET Id_lotu = 1002
WHERE Id_rezerwacji = 42

DELETE FROM Rezerwacje
WHERE Id_rezerwacji = 42

```

Trigger TR_dodaj_lot

Trigger typu INSTEAD OF INSERT zdefiniowany dla tabeli Loty. Zadaniem triggera jest sprawdzanie poprawności wprowadzanych danych do tabeli Loty. Trigger sprawdza odpowiednio:

- czy numer lotu nie jest liczbą ujemną,
- czy użytkownik nie chce wprowadzić lotu o numerze, który już istnieje w tabeli Loty,
- czy samolot o podanym id istnieje,
- czy umowa z przewoźnikiem danego samolotu jest ważna,
- czy w danym terminalu istnieje podana bramka,
- czy bramka w danym terminalu nie jest zajęta na ten czas, czyli czy nie obsługuje w tym momencie innego samolotu,
- czy miasto wprowadzone przez użytkownika istnieje w połączeniach lotniska,

- czy dany samolot może polecieć na danego lotnisko, czyli czy dana linia lotnicza posiadająca ten samolot wykonuje połączenia do danego lotniska,
- czy data lotu już minęła, czyli czy użytkownik nie chce dodać lotu „do przeszłości”,
- czy dany samolot o podanej dacie przebywa wtedy na lotnisku. Samolot nie może wykonać lotu skoro wcześniej wykonał odlot, czyli nie przebywa aktualnie na lotnisku, musi najpierw przylecieć, by móc zrobić kolejny kurs.

W przypadku wprowadzenia poprawnych danych na koniec wyświetla się informacja o dodaniu lotu.

```

IF OBJECT_ID('TR_dodaj_lot', 'TR') IS NOT NULL
    DROP TRIGGER TR_dodaj_lot
GO

CREATE TRIGGER TR_dodaj_lot ON Loty
INSTEAD OF INSERT
AS
BEGIN TRY
    DECLARE @Id_lotu INT
    DECLARE @Id_samolotu INT
    DECLARE @Dokad INT = NULL
    DECLARE @Skad INT = NULL
    DECLARE @Id_terminalu NVARCHAR(1)
    DECLARE @Nr_bramki INT
    DECLARE @Data DATETIME
    DECLARE @Typ_lotu NVARCHAR(10)

    DECLARE @Counter INT = 0
    DECLARE @RowsAmount INT = (SELECT COUNT(*) FROM INSERTED)

    WHILE @Counter < @RowsAmount BEGIN
        SET @Id_lotu = (SELECT Id_lotu FROM INSERTED ORDER BY Id_lotu OFFSET
            @Counter ROWS FETCH NEXT 1 ROWS ONLY)

        SET @Id_samolotu = (SELECT Id_samolotu FROM INSERTED ORDER BY Id_lotu OFFSET
            @Counter ROWS FETCH NEXT 1 ROWS ONLY)

        SET @Dokad = (SELECT Dokad FROM INSERTED ORDER BY Id_lotu OFFSET @Counter
            ROWS FETCH NEXT 1 ROWS ONLY)

        SET @Skad = (SELECT Skad FROM INSERTED ORDER BY Id_lotu OFFSET @Counter ROWS
            FETCH NEXT 1 ROWS ONLY)

        SET @Id_terminalu = (SELECT Id_terminalu FROM INSERTED ORDER BY Id_lotu
            OFFSET @Counter ROWS FETCH NEXT 1 ROWS ONLY)

        SET @Nr_bramki = (SELECT Nr_bramki FROM INSERTED ORDER BY Id_lotu OFFSET
            @Counter ROWS FETCH NEXT 1 ROWS ONLY)

        SET @Data = (SELECT Data FROM INSERTED ORDER BY Id_lotu OFFSET @Counter ROWS
            FETCH NEXT 1 ROWS ONLY)
    
```

```

SET @Typ_lotu = (SELECT Typ_lotu FROM INSERTED ORDER BY Id_lotu OFFSET
@Counter ROWS FETCH NEXT 1 ROWS ONLY)

IF (@Id_lotu < 0) BEGIN
    RAISERROR('Id_lotu jest liczba ujemna. Nie dodano lotu', 16, 1)
END

IF (EXISTS (SELECT Id_lotu FROM Loty WHERE Id_lotu = @Id_lotu)) BEGIN
    RAISERROR('Lot o id %i juz istnieje.', 16,1,@Id_lotu)
END

IF (NOT EXISTS (SELECT Id_samolotu FROM Samoloty
WHERE Id_samolotu = @Id_samolotu)) BEGIN
    RAISERROR('Samolot o id %i nie istnieje', 16,1,@Id_samolotu)
END

IF (@Data > (SELECT Do_kiedy FROM Umowy_z_przewoznikami
INNER JOIN Samoloty
ON Samoloty.Id_przewoznika = Umowy_z_przewoznikami.Id_przewoznika
WHERE Id_samolotu = @Id_samolotu)) BEGIN
    RAISERROR('Umowa z przewoznikiem danego samolotu
jest juz niewazna', 16, 1)
END

IF (NOT EXISTS (SELECT * FROM Bramki WHERE Id_terminalu = @Id_terminalu
AND Nr_bramki = @Nr_bramki)) BEGIN
    RAISERROR('Nie ma bramki nr %i w terminalu %s', 16, 1,
@Nr_bramki, @Id_terminalu)
END

--sprawdzanie czy bramka nie bedzie zajeta w tym czasie
--czas potrzebny na obsluge samolotu przy bramce wynosi 30 minut
DECLARE @Potential_flights INT = (SELECT COUNT(*)
FROM Loty
WHERE Id_terminalu = @Id_terminalu
AND Nr_bramki = @Nr_bramki AND Id_lotu <> @Id_lotu
AND Data > DATEADD(MINUTE, -30, @Data)
AND Data < DATEADD(MINUTE, 30, @Data))
IF (@Potential_flights > 0) BEGIN
    RAISERROR('Kolizja z bramkami. Nie wystarczajaca ilosc czasu na obsluge
samolotu przy bramce', 16, 1)
END

IF (@Dokad IS NOT NULL) BEGIN
    IF (@Dokad NOT BETWEEN 1 AND 22) BEGIN
        RAISERROR('Miasto o id %i nie istnieje', 16, 1, @Dokad)
    END
    --teraz sprawdzanie czy dany samolot moze leciec do danego panstwa
    IF(NOT EXISTS (SELECT * FROM Cennik_lotow
INNER JOIN Samoloty
ON (Samoloty.Id_przewoznika = Cennik_lotow.Id_przewoznika)
WHERE (Dokad = @Dokad AND Id_samolotu = @Id_samolotu)) ) BEGIN
        RAISERROR('Samolot o id %i nie lata do lotniska o id %i.', 16,
1, @Id_samolotu, @Dokad)
    END
END

IF (@Skad IS NOT NULL) BEGIN

```



```

        IF (@Skad NOT BETWEEN 1 AND 22) BEGIN
            RAISERROR('Miasto o id %i nie istnieje', 16, 1, @Skad)
        END
    END

    IF (@Data < CURRENT_TIMESTAMP) BEGIN
        RAISERROR('Wprowadzona data juz minela, nie mozna dodac lotu "do
przeszlosci"', 16, 1)
    END

    DECLARE @DateVARCHAR VARCHAR(30) = CAST(@Data AS VARCHAR)

    IF (SELECT dbo.Czy_samolot_na_lotnisku(@Id_samolotu, @Data)) = 'NIE' BEGIN
        RAISERROR('Samolot o id %i, nie przebywa na naszym lotnisku w dniu
%s', 16, 1, @Id_samolotu, @DateVARCHAR)
    END

    SET @Counter += 1

    INSERT INTO Loty VALUES
    (@Id_lotu, @Id_samolotu, @Dokad, @Skad, @Id_terminalu, @Nr_bramki, @Data,
@Typ_lotu)

    PRINT('Dodano lot nr' + CONVERT(VARCHAR(10), @Id_lotu))

    END
END TRY
BEGIN CATCH
    SELECT
        ERROR_NUMBER() AS ErrorNumber,
        ERROR_SEVERITY() AS ErrorSeverity,
        ERROR_STATE() AS ErrorState,
        ERROR_PROCEDURE() AS ErrorProcedure,
        ERROR_LINE() AS ErrorLine,
        ERROR_MESSAGE() AS ErrorMessage;
END CATCH

-----
SELECT * FROM Loty

INSERT INTO Loty VALUES
(1035, 97, 7, NULL, 'A', 2, '2021-03-10 10:00', 'Odlot'),
(1036, 4, 1, NULL, 'A', 2, '2021-02-10 12:00', 'Odlot')

INSERT INTO Loty VALUES
(1035, 97, 6, NULL, 'A', 2, '2021-03-10 10:00', 'Odlot'),
(1036, 4, 1, NULL, 'A', 2, '2021-02-10 12:00', 'Odlot')

DELETE FROM Loty
WHERE Id_lotu = 1035

DELETE FROM Loty

```

Trigger TR_Bagaz

Trigger typu AFTER INSERT, UPDATE zdefiniowany dla tabeli Bagaz kontroluje, żeby sumaryczna waga bagaży, przypisanych do jednej rezerwacji nie przekraczała lub nie była równa 100 kg, ponieważ bagaż o takiej wadze nie jest obsługiwany na lotach pasażerskich. W przypadku, gdy zostaje to wykryte, wszystkie rejestracje bagażu odpowiedniej rezerwacji zostają usunięte oraz jest wyświetlany komunikat o przekroczeniu limitu wagi bagażu, co skutkuje koniecznością zmniejszenia wagi bagażu przypisanego do tej rezerwacji i jego ponownego zarejestrowania.

```
IF OBJECT_ID('Tr_bagaz','TR') IS NOT NULL
DROP TRIGGER Tr_Bagaz

GO
CREATE TRIGGER Tr_Bagaz ON BAGAZ
AFTER INSERT, UPDATE
AS
    DECLARE @Tab TABLE (ID_rezerwacji INT, Waga_sumaryczna REAL)

    INSERT INTO @Tab
    Select B.Id_rezerwacji, SUM(B.Waga) from Bagaz B
    Group by B.Id_rezerwacji

    IF EXISTS(SELECT * from @Tab T where Waga_sumaryczna >= 100)
    Begin
        Delete from Bagaz
        where ID_rezerwacji in (SELECT T.ID_rezerwacji from @Tab T
                                where Waga_sumaryczna >= 100)

        SELECT T.ID_rezerwacji [Bagaż następujących rezerwacji
                                przekroczył limit] from @Tab T
        where Waga_sumaryczna >= 100
    END

GO
```

Trigger TR_gr_obsługi

Trigger typu INSTEAD OF INSERT zdefiniowany dla tabeli GR_Obsluga_lotu. Głównym celem tego wyzwalacza jest kontrola, by żaden pracownik nie musiał wykonywać dwóch loty w jeden i ten sam dzień, ponieważ byłoby to niemożliwe fizycznie dla niego. Istotnym tutaj jest założenie, że w następnym dniu pracownicy

już będą w stanie wykonać inny lot. W implementacji tego wyzwalacza korzystamy z kursora K_pracownik, żeby móc po kolei sprawdzać dane każdego pracownika oraz wprowadzać dane do tabeli GR_Obslugi_lotu osobno dla każdego pracownika. Przed wprowadzeniem danych pracownika i lotu, który będzie on obsługiwał, w tabeli GR_Obslugi_lotu są sprawdzane następujące warunki:

- istnienie lotu o podanym w insert id,
- czy występuje informacja o pracowniku o podanym w insert id w rejestrze pracowników lotniska,
- czy ma pracownik o podanym w insert id uprawnienia do obsługi lotu,
- czy w dniu, kiedy odbywa się lot o id podanym w insert, pracownik nie obsługuje innego lotu,

W przypadku niespełnienia chociaż jednego z tych warunków jest wyświetlany odpowiedni komunikat, oraz pracownik nie zostaje przypisany do odpowiedniego lotu. W przeciwnym razie pracownik zostaje przypisany do odpowiedniego lotu.

```
IF OBJECT_ID('Tr_gr_obsługi','TR') IS NOT NULL
DROP TRIGGER Tr_gr_obsługi

GO
CREATE TRIGGER TR_gr_obsługi on Gr_Obslugi_lotu
Instead of INSERT
As
    DECLARE K_Pracownik CURSOR
    FOR SELECT I.Id_lotu, I.Id_Pracownika FROM inserted I
    FOR READ ONLY

    DECLARE @Lot INT
    DECLARE @Pracownik INT

    OPEN K_Pracownik
    FETCH K_Pracownik INTO @Lot, @Pracownik
    WHILE @@FETCH_STATUS <> -1
    BEGIN
        IF not exists(Select * from LOTY where Id_lotu = @LOT)
        BEGIN
            PRINT 'Nie istnieje Lot o ID '+CONVERT(VARCHAR(10), @LOT)
            FETCH K_Pracownik INTO @Lot, @Pracownik
            CONTINUE
        END

        IF not exists(SELECT * FROM Pracownicy
            where ID_Pracownika = @Pracownik)
        BEGIN
```

```

        PRINT 'Nie występuje informacja o pracowniku o ID'
              +CONVERT(VARCHAR(10), @Pracownik)
        FETCH K_Pracownik INTO @Lot, @Pracownik
        CONTINUE
    END

    DECLARE @Stanowisko NVARCHAR(50)
    SET @Stanowisko = (SELECT Stanowisko from Pracownicy where
                        ID_Pracownika = @Pracownik)
    IF @Stanowisko = 'celnik' OR @Stanowisko = 'straznik'
    BEGIN
        PRINT 'Pracownikowi o ID ' +CONVERT(VARCHAR(10), @Pracownik) +
              ' nie ma uprawnień dla obsługi lotu'
        FETCH K_Pracownik INTO @Lot, @Pracownik
        CONTINUE
    END

    DECLARE @DATA DATE
    SET @DATA = (SELECT CAST(DATA as DATE) from Loty
                 where Id_lotu = @Lot)

    IF exists (Select * from GR_Obslugi_lotu Gr join Loty L on
               Gr.Id_lotu = L.Id_lotu
               where Gr.Id_Pracownika = @Pracownik and
                     Cast(L.Data as Date) = @DATA)
    BEGIN
        DECLARE @LOT2 INT
        SET @LOT2 = (Select L.Id_lotu from GR_Obslugi_lotu Gr join
                        Loty L on Gr.Id_lotu = L.Id_lotu
                        where Gr.Id_Pracownika = @Pracownik and
                              Cast(L.Data as Date) = @DATA)
        PRINT 'Pracownikowi o ID ' + CONVERT(VARCHAR(10), @Pracownik) +
              ' LOT ' +CONVERT(VARCHAR(10), @Lot)
              + ' koliduje się z lotem ' +CONVERT(VARCHAR(10), @Lot2)
        FETCH K_Pracownik INTO @Lot, @Pracownik
        CONTINUE
    END

    INSERT INTO GR_Obslugi_lotu VALUES
        (@Lot, @Pracownik)
    FETCH K_Pracownik INTO @Lot, @Pracownik

    END
    CLOSE K_Pracownik
    DEALLOCATE K_Pracownik

GO

INSERT INTO GR_Obslugi_lotu VALUES
(1009, 34),
(1040, 29),
(1004, 12),
(1028, 9),
(1013, 13)

DELETE FROM GR_Obslugi_lotu
where Id_lotu = 1013 and Id_Pracownika = 13

```

Trigger TR_rezerwacje

Trigger typu INSTEAD OF INSERT zdefiniowany dla tabeli Rezerwacje. Głównym zadaniem triggera jest kontrola za poprawnością wykonania rezerwacji oraz zabezpieczenie bezpieczeństwa lotu. Przed wprowadzeniem danych o pasażerze, oraz locie na który pasażer wykonuje rezerwację są sprawdzane następujące warunki :

- czy osoba wykonująca rezerwację na lot nie jest na liście osób niebezpiecznych. Jeżeli okaże się, że jest to osoba niebezpieczna, jest wyświetlany odpowiedni komunikat i odmawia się w rezerwacji nie tylko tej osobie, ale i wszystkim innym (z tej samej instrukcji insert),
- istnienie wskazanej w insert klasy podróży,
- czy jest osoba wykonująca rezerwację jest zidentyfikowana w bazie pasażerów lotniska,
- istnienie lotu o id wskazanym w insert,
- czy pasażer nie ma rejestracji na loty, które wykonują się w ciągu 4 godzin do lub 4 godzin po locie o id wskazanym w insert,
- czy jest dostateczna ilość miejsc w samolocie, uwzględniając już wykonane rezerwacje.

W przypadku niespełnienia chociaż jednego z powyższych warunków pasażerowi odmawia się w rezerwacji oraz wyświetlane są wtedy odpowiednie komunikaty. W przeciwnym razie rezerwacja zostaje dodana do tabeli Rezerwacje.

W implementacji tego wyzwalacza korzystamy z kursora K_Rezerwacja, żeby móc po kolei sprawdzać dane każdej nowej wprowadzonej rezerwacji z pseudotabeli INSERTED, oraz po kolei niezależnie jedna od drugiej móc je zatwierdzać lub odrzucać pasażerom.

```
IF OBJECT_ID('Tr_rezerwacje', 'TR') IS NOT NULL
DROP TRIGGER Tr_rezerwacje

GO
Create TRIGGER Tr_rezerwacje ON Rezerwacje
Instead of insert
AS
```

```

IF EXISTS(SELECT * FROM inserted I join Pasazer P on I.Id_pasazera =
        P.Id_Pasazera where EXISTS (Select * from [Osoby niebezpieczne] O
        where O.NR_Paszportu = P.NR_Paszportu))
    BEGIN
        DECLARE @Imie NVARCHAR(50)
        DECLARE @Nazwisko NVARCHAR(50)
        DECLARE @Paszport VARCHAR(15)
        DECLARE @KOMUNIKAT NVARCHAR(100)
        SET @Imie = (SELECT P.Imie FROM inserted I join Pasazer P
            on I.Id_pasazera = P.Id_Pasazera where EXISTS
            (Select * from [Osoby niebezpieczne] O where
            O.NR_Paszportu = P.NR_Paszportu))
        SET @Nazwisko = (SELECT P.Nazwisko FROM inserted I join
            Pasazer P on I.Id_pasazera = P.Id_Pasazera
            where EXISTS (Select * from [Osoby
            niebezpieczne] O where O.NR_Paszportu =
            P.NR_Paszportu))
        SET @Paszport = (SELECT P.NR_Paszportu FROM inserted I join
            Pasazer P on I.Id_pasazera = P.Id_Pasazera
            where EXISTS (Select * from [Osoby
            niebezpieczne] O where O.NR_Paszportu =
            P.NR_Paszportu))
        SET @Komunikat = 'Pasażer '+@Imie + ' '+@Nazwisko + ' o
            numerze paszportu '+@Paszport+
            ' jest osobą niebezpieczną.'
        PRINT @KOMUNIKAT
        PRINT 'Odmowa w całej grupie rezerwacji. Proszę służbę
            bezpieczeństwa sprawdzić wszystkich pasażerów'
    END
ELSE
    BEGIN
        DECLARE K_Rezerwacja CURSOR
        FOR SELECT I.Id_Rezerwacji, I.Id_lotu, I.Id_pasazera, I.Klasa_podrozy
            FROM inserted i
        FOR READ ONLY
        DECLARE @ID_rezerwacji INT
        DECLARE @Id_lotu INT
        DECLARE @ID_pasazera INT
        DECLARE @Klasa INT
        OPEN K_Rezerwacja
        FETCH K_Rezerwacja INTO @ID_rezerwacji, @Id_lotu, @ID_pasazera, @Klasa
        WHILE @@FETCH_STATUS <> -1
        BEGIN
            IF not EXISTS (SELECT * from [Klasa podrozy] K
                where K.Klasa = @Klasa)
            BEGIN
                PRINT 'Odmowa rezerwacji o id '+CONVERT(VARCHAR(10),
                    @ID_rezerwacji) + ' z powodu nie istnienia wskazanej klasy
                    podróży'
                FETCH K_Rezerwacja INTO @ID_rezerwacji, @Id_lotu,
                    @ID_pasazera, @Klasa
                CONTINUE
            END
            IF not EXISTS(Select * from Pasazer P where P.Id_Pasazera =

```

```

                                @ID_pasazera)
BEGIN
    PRINT 'Pasazer o id '+CONVERT(VARCHAR(10), @ID_pasazera)+
        ' nie zidentyfikowany w bazie pasażerow. Odmowa w
        rezerwacji o id '+CONVERT(VARCHAR(10),
            @ID_rezerwacji)
    FETCH K_Rezerwacja INTO @ID_rezerwacji, @Id_lotu,
                            @ID_pasazera, @Klasa
    CONTINUE
END

IF EXISTS(Select * from Loty L where L.Id_lotu = @Id_Lotu)
BEGIN
    DECLARE @Data_lotu DATETIME
    SET @Data_lotu = (Select DATA from LOTY where Id_lotu =
                        @Id_lotu)

    DECLARE @Ilosc_lotow INT
    SET @Ilosc_lotow = (SELECT COUNT(*) FROM Rezerwacje
                        JOIN Loty ON Loty.Id_lotu = Rezerwacje.Id_lotu
                        WHERE Id_pasazera = @ID_pasazera
                        AND Data < DATEADD(HOUR, 4, @Data_lotu)
                        AND Data > DATEADD(HOUR, -4, @Data_lotu))

    IF @Ilosc_lotow <> 0
    BEGIN
        PRINT('Pasazer o id ' +CONVERT(VARCHAR(10),
            @ID_pasazera)+ ' ma juz rezerwacje na lot w tym samym
            czasie')
        FETCH K_Rezerwacja INTO @ID_rezerwacji, @Id_lotu,
                                @ID_pasazera, @Klasa
        CONTINUE
    END

    DECLARE @Ilosc_miejsc_w_samolocie INT
    SET @Ilosc_miejsc_w_samolocie = (Select Ss.Ilosc_miejsc
        from Loty L join Samoloty S on L.Id_samolotu =
        S.Id_samolotu join Samoloty_szczegoly Ss on S.Marka =
        Ss.Marka and S.Model = Ss.Model where L.Id_lotu =
        @Id_lotu)

    DECLARE @Ilosc_rezerwacji INT
    SET @Ilosc_rezerwacji = (Select
        dbo.Ilosc_rezerwacji_na_lot(@Id_lotu))

    DECLARE @Ilosc_miejsc INT
    SET @Ilosc_miejsc = @Ilosc_miejsc_w_samolocie -
        @Ilosc_rezerwacji

    IF @Ilosc_miejsc <= 0
        PRINT 'Odmowa w rezerwacji o id '+
            CONVERT(VARCHAR(10), @ID_rezerwacji)+
            ' z powodu braku miejsc w samolocie'
    ELSE
    BEGIN

```

```

        INSERT INTO Rezerwacje Values
        (@ID_rezerwacji, @id_lotu, @ID_pasazera, @Klasa)
    END
END
ELSE
    PRINT 'Odmowa w rezerwacji o id ' + CONVERT(VARCHAR(10),
        @ID_rezerwacji) + ' z powodu nie istnienia lotu o id ' +
        CONVERT(VARCHAR(10), @Id_lotu)

    FETCH K_Rezerwacja INTO @ID_rezerwacji, @Id_lotu, @ID_pasazera,
        @Klasa
END
CLOSE K_rezerwacja
DEALLOCATE K_rezerwacja
END
Go

INSERT INTO Rezerwacje Values
(44, 1022, 27 , 1)

INSERT INTO Rezerwacje Values
(44, 1022, 23 , 5),
(45, 1050, 23, 1),
(46, 1023, 99, 2),
(47, 1015, 18, 3)

Select * from Rezerwacje
Delete from Rezerwacje where Id_Rezerwacji = 47

```


Widoki

Widok dbo.vw_Szczegoly_lotow

Wyświetlane są wszystkie loty zawarte w tabeli Loty. Dla każdego lotu podawana jest data wykonania podróży, typ podróży (czy jest to odlot z lotniska czy przylot na lotnisko), lotnisko do którego leci samolot lub z którego przylatuje, dodatkowo miasto oraz państwo danego lotniska. Kolejno podawane są informacje o samolocie, ilość wykonanych rezerwacji na dany lot, ilość wszystkich miejsc w samolocie, numer samolotu, model, marka oraz nazwa linii lotniczej, która wykonuje lot. Na samym końcu podawane są informacje na temat miejsca na lotnisku z którego samolot odlatuje/przylatuje, czyli id terminalu oraz numer bramki.

```
IF OBJECT_ID('dbo.vw_Szczegoly_lotow', 'V') IS NOT NULL
    DROP VIEW dbo.vw_Szczegoly_lotow
GO
CREATE VIEW vw_Szczegoly_lotow AS
SELECT Loty.Id_lotu, Loty.Data, Loty.Typ_lotu,
Polaczenia_z_lotniskami.Nazwa_lotniska AS [Lotnisko],
Polaczenia_z_lotniskami.Miasto,
Polaczenia_z_lotniskami.Panstwo, dbo.Ilosc_rezerwacji_na_lot(Id_lotu)
AS Rezerwacje, Samoloty_szczegoly.Ilosc_miejsc, Loty.Id_samolotu,
Samoloty_szczegoly.Model, Samoloty_szczegoly.Marka,
Przewoznicy.Nazwa AS [Linia lotnicza], Loty.Id_terminalu,
Loty.Nr_bramki
FROM Loty
INNER JOIN Samoloty ON Loty.Id_samolotu = Samoloty.Id_samolotu
INNER JOIN Przewoznicy
ON Przewoznicy.Id_przewoznika = Samoloty.Id_przewoznika
INNER JOIN Samoloty_szczegoly
ON (Samoloty_szczegoly.Marka = Samoloty.Marka
AND Samoloty_szczegoly.Model = Samoloty.Model)
INNER JOIN Polaczenia_z_lotniskami
ON (Loty.Dokad = Polaczenia_z_lotniskami.Id_lotniska
OR Loty.Skad = Polaczenia_z_lotniskami.Id_lotniska)
GO
-----
SELECT * FROM dbo.vw_Szczegoly_lotow
```

Widok dbo.vw_przebieg_samolotow

Widok, który zawiera informacje o przebiegu każdego samolotu. Czyli ilość lotów obsługanych przez nasze lotnisko danego samolotu. W kolumnach są odpowiednio id samolotu, przebieg, marka, model oraz nazwa linii lotniczej do której należy dany samolot.

```
IF OBJECT_ID('dbo.vw_przebieg_samolotow', 'V') IS NOT NULL
    DROP VIEW dbo.vw_przebieg_samolotow
GO
CREATE VIEW vw_przebieg_samolotow AS
    SELECT Samoloty.Id_samolotu, COUNT(Id_lotu) AS Przebieg, Marka,
        Model, Nazwa
    FROM Samoloty
    LEFT JOIN Loty ON Loty.Id_samolotu = Samoloty.Id_samolotu
    INNER JOIN Przewoźnicy
    ON Przewoźnicy.Id_przewoźnika = Samoloty.Id_przewoźnika
    GROUP BY Samoloty.Id_samolotu, Marka, Model, Nazwa
GO
-----
SELECT * FROM dbo.vw_przebieg_samolotow
```

Widok dbo.vw_BRAMKI

Dla każdej bramki wyświetlana jest następująca informacja : jej numer, id jej terminalu, położenie tego terminalu, id lotu do niej przypisanego, oraz typ tego lotu, skąd/dokąd przylatuje/odlatuje (miasto i państwo), data i czas odlotu/przylotu, marka oraz model samolotu wykonującego lot, i przewoźnik.

```
IF OBJECT_ID('dbo.vw_Bramki', 'V') is not null DROP VIEW dbo.vw_Bramki
GO
CREATE VIEW vw_BRAMKI AS
SELECT T.Id_terminalu, T.Polozenie, B.Nr_bramki,
    L.Id_lotu, L.Typ_lotu, S.Marka [Marka samolotu], S.Model
    [Model samolotu], L.Data,
    Pl.Miasto, Pl.Panstwo, P.Nazwa [Przewoźnik]
FROM BRAMKI B join Terminale T on B.Id_terminalu = T.Id_terminalu
JOIN LOTY L on L.Id_terminalu = B.Id_terminalu
and L.Nr_bramki = B.Nr_bramki
JOIN Polaczenia_z_lotniskami Pl on (L.Dokad = Pl.Id_lotniska OR
    L.Skad=Pl.Id_lotniska)
JOIN Samoloty S on L.Id_samolotu = S.Id_samolotu
```

```
JOIN Przewoźnicy P on S.Id_przewoźnika =P.Id_przewoźnika
GO
```

```
Select * from dbo.vw_BRAMKI
```

Widok dbo.vw_Odloty

Dla każdego odlotu wyświetlana jest następująca informacja: id lotu, numer bramki oraz id terminalu od którego odlatuje samolot, czas i data odlotu, miejsce docelowe lotu (miasto, państwo i nazwa lotniska), marka i model samolotu wykonującego lot, oraz przewoźnik.

```
IF Object_ID('dbo.vw_Odloty' , 'V') is not null
Drop view    dbo.vw_Odloty
```

```
GO
```

```
CREATE VIEW vw_Odloty AS
Select L.Id_lotu, L.Id_terminalu, L.Nr_bramki, L.Data, L.Typ_lotu,
Pl.Miasto, Pl.Nazwa_lotniska, Pl.Panstwo,
P.Nazwa [Przewoźnik],
S.Marka [Marka samolotu], S.Model [Model samolotu]
from Loty L join Polaczenia_z_lotniskami Pl on L.Dokad = Pl.Id_lotniska
join Samoloty S on L.Id_samolotu = S.Id_samolotu
join Przewoźnicy P on S.Id_przewoźnika = P.Id_przewoźnika
where Typ_lotu = 'Odlot'
GO
```

```
SELECT * from dbo.vw_Odloty
```

Widok dbo.vw_Przyloty

Dla każdego przylotu wyświetlana jest następująca informacja: id lotu, numer bramki oraz id terminalu do którego przylatuje samolot, czas i data przylotu, skąd przylatuje (miasto, państwo i nazwa lotniska), marka i model samolotu, wykonującego lot, oraz przewoźnik.

```
IF Object_ID('dbo.vw_Przyloty' , 'V') is not null
Drop view    dbo.vw_Przyloty
```

```
GO
```

```
CREATE VIEW vw_Przyloty AS
Select L.Id_lotu, L.Id_terminalu, L.Nr_bramki, L.Data, L.Typ_lotu,
```

```

Pl.Miasto, Pl.Nazwa_lotniska, Pl.Panstwo,
P.Nazwa [Przewoznik],
S.Marka [Marka samolotu], S.Model [Model samolotu]
from Loty L join Polaczenia_z_lotniskami Pl on L.Skad = Pl.Id_lotniska
join Samoloty S on L.Id_samolotu = S.Id_samolotu
join Przewoznicy P on S.Id_przewoznika = P.Id_przewoznika
where Typ_lotu = 'Przylot'
GO

SELECT * from dbo.vw_Przyloty

```

Widok dbo.vw_Szczegoly_rezerwacji_pasazerow

Dla każdego pasażera wyświetlana jest następująca informacja: jego id, numer paszportu, imię, nazwisko, obywatelstwo, id wykonanych przez niego rezerwacji. Z kolei dla każdej takiej rezerwacji wyświetlamy klasę podróży, id lotu na który ona była wykonana, typ lotu, data i czas odlotu/przylotu, miasto oraz państwo skąd/dokąd był/będzie wykonany lot, ilość bagaży przypisanych do tej rezerwacji, ich waga sumaryczna, oraz cena sumaryczna za te rezerwacje.

```

IF Object_ID('dbo.vw_Szczegoly_rezerwacji_pasazerow' , 'V') is not null
Drop view    dbo.vw_Szczegoly_rezerwacji_pasazerow

GO
CREATE VIEW vw_Szczegoly_rezerwacji_pasazerow As
SELECT P.Id_Pasazera, P.NR_Paszportu, P.Imie, P.Nazwisko, P.Panstwo
[Obywatelstwo],
R.Id_Rezerwacji, R.Id_lotu, R.Klasa_podrozy,
L1.Miasto, L1.Panstwo, L1.Data, L1.Typ_lotu,
Count(B.Id_bagazu) [Ilosc bagazy],
Round(Sum(B.Waga), 2) [Waga sumaryczna bagazy],
ROUND(dbo.Cena_sumaryczna_rezerwacji(R.Id_Rezerwacji), 2) [Cena rezerwacji]
From Pasazer P join Rezerwacje R on P.Id_Pasazera = R.Id_pasazera
join (Select Loty.Data, Loty.Id_lotu, Loty.Id_samolotu, Loty.Typ_lotu,
Pl.Miasto, Pl.Panstwo from Loty
join Polaczenia_z_lotniskami Pl on Loty.Dokad = Pl.Id_lotniska Or
Loty.Skad = Pl.Id_lotniska) L1 on R.Id_lotu = L1.Id_lotu
join Bagaz B on R.Id_Rezerwacji = B.Id_rezerwacji
Group by P.Id_Pasazera, P.NR_Paszportu, P.Imie, P.Nazwisko, P.Panstwo,
R.Id_Rezerwacji,
R.Id_lotu, R.Klasa_podrozy, L1.Miasto, L1.Panstwo, L1.Data, L1.Typ_lotu
Go

Select * from dbo.vw_Szczegoly_rezerwacji_pasazerow

```