Bazy danych

Autorzy: Mikołaj Pacek, Miłosz Gaszyna, Bartosz Grzybowski

1. Wymagania i funkcje systemu

a) Rezerwacja biletów na wycieczkę.

- Można zarezerwować bilety jedynie na wycieczki, które są w sprzedaży.
- Można zarezerwować maksymalnie tyle biletów na wycieczkę, ile jest aktualnie dostępnych.
- Ceny wycieczki mogą się zmieniać w czasie, lecz na rezerwacji cena się nie zmienia.
- Można zmieniać ilość biletów zarezerwowanych na wycieczkę.
- Bilety na wycieczkę są imienne, lecz nie trzeba od razu podawać listy osób.

b) Rezerwacja atrakcji na wycieczkę.

- Można zarezerwować bilety na atrakcję jedynie przeznaczoną na daną wycieczkę.
- Można zarezerwować maksymalnie tyle biletów, ile jest dostępnych na danej atrakcji.
- Ceny atrakcji mogą się zmieniać w czasie, lecz na rezerwacji cena się nie zmienia.
- Można zmieniać ilość biletów zarezerwowanych na atrakcję.
- Bilety na atrakcję są imienne, lecz nie trzeba od razu podawać listy osób.

c) Opłata rezerwacji.

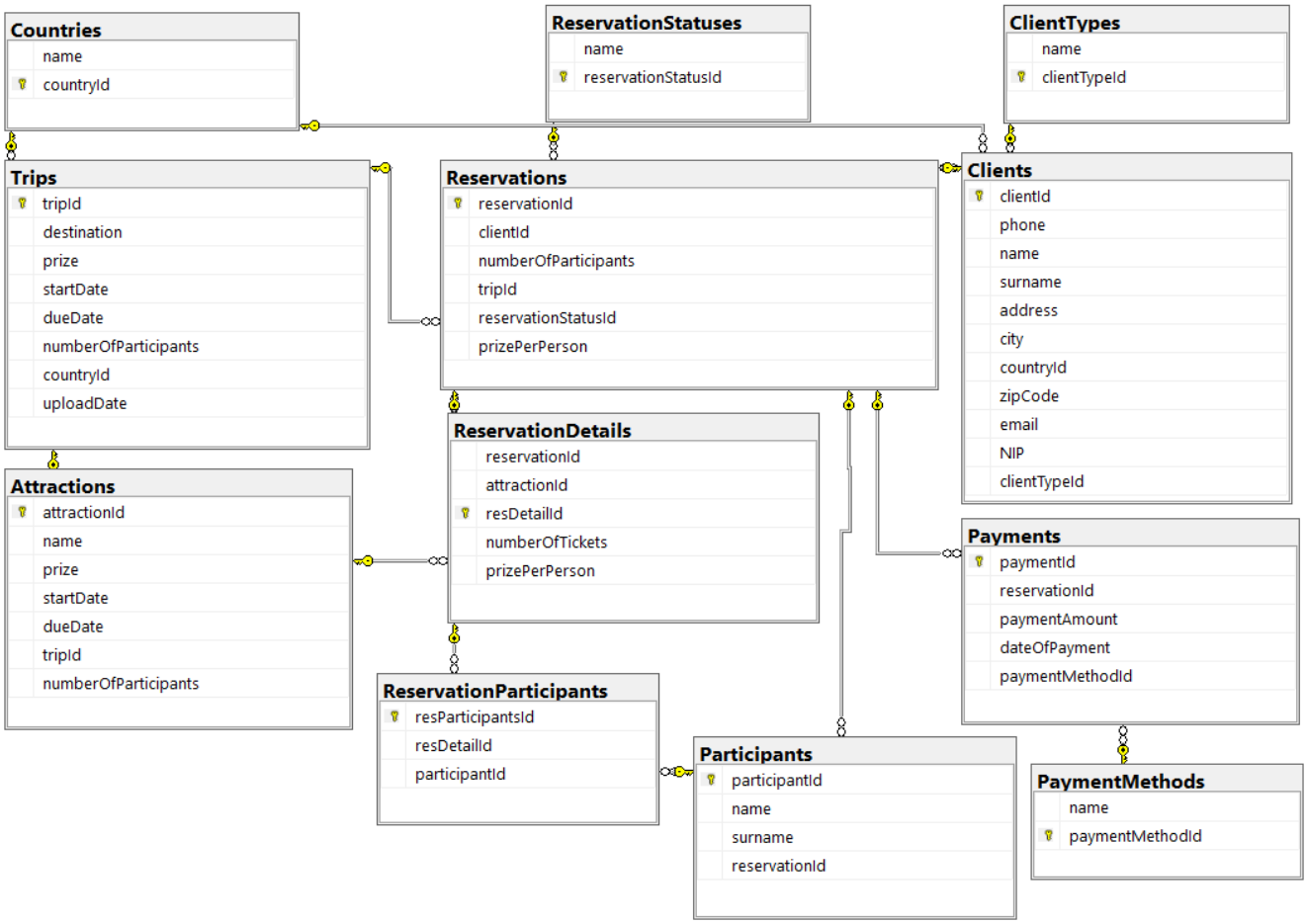
- Możliwość opłacenia rezerwacji 'na raty' - nie ma konieczności opłacenia całej wycieczki naraz.
- Różne sposoby płatności.

d) Tworzenie wycieczki.

- Możliwość wprowadzenia daty 'udostępnienia wycieczki' - przed tą datą wycieczka nie będzie dostępna do zakupienia.

2. Baza danych

Schemat bazy danych



Opis poszczególnych tabel

Attractions

Dostępne atrakcje dla każdej z wycieczek.

Nazwa atrybutu	Typ	Opis/Uwagi
attractionId	int	klucz główny tabeli
name	varchar(50)	imię uczestnika atrkacji
price	money	cena atrakcji
startDate	date	data początku atrakcji
dueDate	date	data końca atrakcji
tripId	int	klucz obcy łączący z tabelą Trips
numberOfParticipants	int	liczba uczestników wycieczki

	attractionId	name	price	startDate	dueDate	tripId	numberOfParticipants
1	1	Wieza Eiffla	150,00	2024-07-05	2024-07-15	2	20
2	2	Luwr	200,00	2024-07-10	2024-07-18	2	10
3	3	Musee dOrsay	100,00	2024-07-11	2024-07-15	2	8
4	4	Buckingham Palace	500,00	2024-07-05	2024-07-05	1	6
5	5	London Eye	150,00	2024-07-01	2024-07-07	1	16
6	6	Tower Bridge	75,00	2024-07-01	2024-07-07	1	16
7	7	Sagrada Familia	150,00	2024-07-20	2024-07-27	3	12
8	8	Camp Nou	200,00	2024-07-22	2024-07-22	3	12
9	9	Zamek sw. Jerzego	100,00	2024-07-20	2024-07-28	4	10
10	10	Wycieczka do Fatimy	800,00	2024-07-27	2024-07-28	4	20

```
CREATE TABLE [dbo].[Attractions] (
    [attractionId]         INT           IDENTITY (1, 1) NOT NULL,
    [name]                 VARCHAR (50) NOT NULL,
    [price]                MONEY         NOT NULL,
    [startDate]            DATE          NOT NULL,
    [dueDate]              DATE          NOT NULL,
    [tripId]               INT           NOT NULL,
    [numberOfParticipants] INT           CONSTRAINT
[DF_Attractions_numberOfParticipants] DEFAULT ((0)) NULL,
    CONSTRAINT [PK_Attractions] PRIMARY KEY CLUSTERED ([attractionId] ASC),
    CONSTRAINT [FK_Attractions_Trips] FOREIGN KEY ([tripId]) REFERENCES [dbo].
[Trips] ([tripId])
);
```

Clients

Klienci zamawiający bilety na wycieczkę.

Nazwa atrybutu	Typ	Opis/Uwagi
clientId	int	klucz główny tabeli
phone	varchar(50)	telefon kontaktowy do klienta biura podróży
name	varchar(50)	imię klienta
surname	varchar(50)	nazwisko klienta
address	varchar(50)	adres klienta
city	varchar(50)	miasto zamieszkania klienta
country	varchar(50)	kraj
zipCode	varchar(50)	kod pocztowy
email	varchar(50)	adres email klienta
NIP	varchar(50)	numer NIP klienta
clientTypeId	int	osoba prywatna/firma

```
CREATE TABLE [dbo].[Clients] (
    [clientId]     INT          IDENTITY (1, 1) NOT NULL,
    [phone]        VARCHAR (50) NOT NULL,
    [name]         VARCHAR (50) NOT NULL,
    [surname]      VARCHAR (50) NOT NULL,
    [address]      VARCHAR (50) NOT NULL,
    [city]         VARCHAR (50) NOT NULL,
    [country]      VARCHAR (50) NOT NULL,
    [zipCode]      VARCHAR (50) NOT NULL,
    [email]        VARCHAR (50) NULL,
    [NIP]          VARCHAR (50) NULL,
    [clientTypeId] INT          NOT NULL,
    CONSTRAINT [PK_Clients] PRIMARY KEY CLUSTERED ([clientId] ASC),
    CONSTRAINT [FK_Clients_ClientTypes] FOREIGN KEY ([clientTypeId]) REFERENCES
[dbo].[ClientTypes] ([clientTypeId])
);
```

	clientId	phone	name	surname	address	city	country	zipCode	email	NIP	clientTypeId
1	1	645221389	Wiktor	Wojtas	ul. Sucha 34	Warszawa	Polska	00-007	wwojtas@email.pl	1676283499	2
2	2	696234112	Filip	Kubski	ul. Wesola 2	Krakow	Polska	30-001	filipo@email.pl	NULL	1
3	3	752334121	Jaroslaw	Jarzabkowski	ul. Slowackiego 220	Poznan	Polska	60-002	jarekjar@email.pl	NULL	1
4	4	828828098	Janusz	Pogorzelski	ul. Wielka 34 m. 10	Wroclaw	Polska	50-009	janekpog@email.pl	1617234788	2
5	5	623466273	Pawel	Bielinski	ul. Bacha 1 m. 2	Gdansk	Polska	80-000	pawelek99@email.pl	NULL	1

ClientTypes

Typy klientów (osoba prywatna/firma).

Nazwa atrybutu	Typ	Opis/Uwagi
name	varchar(50)	typ klienta (osoba prywatna/firma)
clientTypeId	int	klucz główny tabeli

```
CREATE TABLE [dbo].[ClientTypes] (
    [name]         VARCHAR (50) NULL,
    [clientTypeId] INT          IDENTITY (1, 1) NOT NULL,
    CONSTRAINT [PK_ClientTypes] PRIMARY KEY CLUSTERED ([clientTypeId] ASC)
);
```

	name	clientTypeId
1	Osoba Prywatna	1
2	Firma	2

Countries

Kraje

Nazwa atrybutu	Typ	Opis/Uwagi

Nazwa atrybutu	Typ	Opis/Uwagi
name	varchar(50)	nazwa kraju
countryId	int	klucz główny tabeli

```
CREATE TABLE [dbo].[Countries] (
    [name]      VARCHAR (50) NOT NULL,
    [countryId] INT          IDENTITY (1, 1) NOT NULL,
    CONSTRAINT [PK_Countries] PRIMARY KEY CLUSTERED ([countryId] ASC)
);
```

	name	countryId
1	Stany Zjednoczone Ameryki	1
2	Wielka Brytania	2
3	Francja	3
4	Hiszpania	4
5	Portugalia	5
6	Polska	6

Participants

Uczestnicy wycieczki, przypisani do danej rezerwacji.

Nazwa atrybutu	Typ	Opis/Uwagi
participantId	int	klucz główny tabeli
name	varchar(50)	imię uczestnika
surname	varchar(50)	nazwisko uczestnika
reservationId	int	klucz obcy tabeli Reservations

```
CREATE TABLE [dbo].[Participants] (
    [participantId] INT          NOT NULL,
    [name]          VARCHAR (50) NOT NULL,
    [surname]       VARCHAR (50) NOT NULL,
    [reservationId] INT          IDENTITY (1, 1) NOT NULL,
    CONSTRAINT [PK_Participants] PRIMARY KEY CLUSTERED ([participantId] ASC),
    CONSTRAINT [FK_Participants_Reservations] FOREIGN KEY ([reservationId])
REFERENCES [dbo].[Reservations] ([reservationId])
);
```

	participantId	name	surname	reservationId
1	1	Miroslaw	Karabin	1
2	2	Karol	Sochocki	1
3	3	Kaja	Pedrycz	1
4	4	Michalina	Tworkowska	1
5	5	Benedykt	Lasak	1
6	6	Mikolaj	Ciepielewski	1
7	7	Konrad	Daniluk	1
8	8	Emilia	Rycerz	1
9	9	Lidia	Kontny	2
10	10	Leopold	Lapka	2
11	11	Florian	Szczepanowski	2
12	12	Angelika	Czupryna	2
13	13	Oskar	Pasieczny	2
14	14	Kazimiera	Chrustowska	2
15	15	Czeslaw	Niedzialek	3
16	16	Hanna	Smolarska	3

PaymentMethods

Metody płatności za rezerwację.

Nazwa atrybutu	Typ	Opis/Uwagi
name	varchar(50)	metoda płatności
paymentMethodId	int	klucz główny tabeli

```
CREATE TABLE [dbo].[PaymentMethods] (
    [name]            VARCHAR (50) NULL,
    [paymentMethodId] INT          IDENTITY (1, 1) NOT NULL,
    CONSTRAINT [PK_paymentMethods] PRIMARY KEY CLUSTERED ([paymentMethodId] ASC)
);
```

	name	paymentMethodId
1	przelew bankowy	1
2	platnosc gotowka	2

Payments

Historia płatności dla każdej z rezerwacji.

Nazwa atrybutu	Typ	Opis/Uwagi
paymentId	int	klucz główny tabeli
reservationId	itn	klucz obcy z łączący z tabelą Reservations
paymentAmount	money	cena wycieczki

Nazwa atrybutu	Typ	Opis/Uwagi
dateOfPayment	date	data płatności
paymentMethodId	int	klucz obcy łączący z tabelą PaymentMethods

```
CREATE TABLE [dbo].[Payments] (
    [paymentId]       INT   IDENTITY (1, 1) NOT NULL,
    [reservationId]   INT   NOT NULL,
    [paymentAmount]   MONEY NOT NULL,
    [dateOfPayment]   DATE  NOT NULL,
    [paymentMethodId] INT   NOT NULL,
    CONSTRAINT [PK_Payments] PRIMARY KEY CLUSTERED ([paymentId] ASC),
    CONSTRAINT [FK_Payments_paymentMethods] FOREIGN KEY ([paymentMethodId])
REFERENCES [dbo].[PaymentMethods] ([paymentMethodId]),
    CONSTRAINT [FK_Payments_Reservations] FOREIGN KEY ([reservationId]) REFERENCES
[dbo].[Reservations] ([reservationId])
);
```

	paymentId	reservationId	paymentAmount	dateOfPayment	paymentMethodId
1	1	1	44200,00	2024-06-14	1
2	2	2	30900,00	2024-06-10	1
3	3	3	41750,00	2024-06-22	1
4	4	4	56000,00	2024-06-20	1
5	5	5	31600,00	2024-06-30	1

ReservationDetails

Szczegóły rezerwacji na temat wykupionych atrakcji.

Nazwa atrybutu	Typ	Opis/Uwagi
reservatationId	int	klucz główny tabeli
attractionId	int	klucz obcy z tabeli Attractions
resDetailId	int	klucz obcy z tabeli ReservationParticipants
numberOfTickets	int	liczba rezerwowanych miejsc
pricePerPerson	money	cena atrakcji za osobę

```
CREATE TABLE [dbo].[ReservationDetails] (
    [reservationId]   INT   NOT NULL,
    [attractionId]    INT   NOT NULL,
    [resDetailId]     INT   IDENTITY (1, 1) NOT NULL,
    [numberOfTickets] INT   CONSTRAINT [DF_ReservationDetails_numberOfTickets]
DEFAULT ((0)) NOT NULL,
    [prizePerPerson]  MONEY NOT NULL,
    CONSTRAINT [PK_ReservationDetails] PRIMARY KEY CLUSTERED ([resDetailId] ASC),
```

```
    CONSTRAINT [FK_ReservationDetails_Attractions] FOREIGN KEY ([attractionId])
REFERENCES [dbo].[Attractions] ([attractionId]),
    CONSTRAINT [FK_ReservationDetails_Reservations] FOREIGN KEY ([reservationId])
REFERENCES [dbo].[Reservations] ([reservationId])
);
```

	reservationId	attractionId	resDetailId	numberOfTickets	pricePerPerson
1	1	4	1	6	500,00
2	1	5	2	8	150,00
3	2	6	3	4	75,00
4	2	5	4	6	150,00
5	3	1	5	5	150,00
6	3	2	6	5	200,00
7	4	8	7	10	200,00
8	5	9	8	4	100,00
9	5	10	9	4	800,00

ReservationParticipants

Tabela przypisująca uczestnikowi wycieczki wykupione atrakcje.

Nazwa atrybutu	Typ	Opis/Uwagi
resParticipantsId	int	klucz główny tabeli
resDetailId	int	klucz obcy łączący z tabelą ReservationDetails
participantId	int	klucz obcy łączący z tabelą Participants

```
CREATE TABLE [dbo].[ReservationParticipants] (
    [resParticipantsId] INT NOT NULL,
    [resDetailId]       INT NOT NULL,
    [participantId]     INT IDENTITY (1, 1) NOT NULL,
    CONSTRAINT [PK_ReservationParticipants] PRIMARY KEY CLUSTERED
([resParticipantsId] ASC),
    CONSTRAINT [FK_ReservationParticipants_Participants] FOREIGN KEY
([participantId]) REFERENCES [dbo].[Participants] ([participantId]),
    CONSTRAINT [FK_ReservationParticipants_ReservationDetails] FOREIGN KEY
([resDetailId]) REFERENCES [dbo].[ReservationDetails] ([resDetailId])
);
```

	resParticipantsId	resDetailId	participantId
1	1	1	1
2	2	1	2
3	3	1	3
4	4	1	4
5	5	1	5
6	6	1	6
7	7	2	1
8	8	2	2
9	9	2	3
10	10	2	4
11	11	2	5
12	12	2	6
13	13	2	7
14	14	2	8

Reservations

Podstawowe informacje o rezerwacji.

Nazwa atrybutu	Typ	Opis/Uwagi
reservationId	int	klucz główny tabeli
clientId	int	klucz obcy z tabeli Clients
numberOfParticipants	int	liczba uczestników
tripId	int	klucz obcy z tabeli Trips
reservationStatusId	int	klucz obcy łączący z tabelą ReservationStatuses
pricePerPerson	money	cena wycieczki za osobę

```
CREATE TABLE [dbo].[Reservations] (
    [reservationId]        INT   IDENTITY (1, 1) NOT NULL,
    [clientId]             INT   NOT NULL,
    [numberOfParticipants] INT   CONSTRAINT [DF_Reservations_numberOfParticipants]
DEFAULT ((0)) NOT NULL,
    [tripId]               INT   NOT NULL,
    [reservationStatusId]  INT   NOT NULL,
    [pricePerPerson]       MONEY CONSTRAINT [DF_Reservations_prizePerPerson]
DEFAULT ((0)) NOT NULL,
    CONSTRAINT [PK_Reservations] PRIMARY KEY CLUSTERED ([reservationId] ASC),
    CONSTRAINT [FK_Reservations_Clients] FOREIGN KEY ([clientId]) REFERENCES
[dbo].[Clients] ([clientId]),
    CONSTRAINT [FK_Reservations_ReservationStatuses] FOREIGN KEY
([reservationStatusId]) REFERENCES [dbo].[ReservationStatuses]
([reservationStatusId]),
    CONSTRAINT [FK_Reservations_Trips] FOREIGN KEY ([tripId]) REFERENCES [dbo].
[Trips] ([tripId])
);
```

	reservationId	clientId	numberOfParticipants	tripId	reservationStatusId	pricePerPerson
1	1	1	8	1	1	5000,00
2	2	2	6	1	2	5000,00
3	3	3	5	2	3	8000,00
4	4	4	12	3	1	4500,00
5	5	5	4	4	2	7000,00

ReservationStatuses

Aktualny status rezerwacji.

Nazwa atrybutu	Typ	Opis/Uwagi
name	varchar(50)	status rezerwacji
reservationStatusId	int	klucz główny

```
CREATE TABLE [dbo].[ReservationStatuses] (
    [name]                VARCHAR (50) NULL,
    [reservationStatusId] INT          IDENTITY (1, 1) NOT NULL,
    CONSTRAINT [PK_ReservationStatus] PRIMARY KEY CLUSTERED ([reservationStatusId]
ASC)
);
```

	name	reservationStatusId
1	nowa	1
2	potwierdzona	2
3	oplacona	3
4	anulowana	4
5	zrealizowana	5
6	nieobecnosc	6

Trips

Dostępne wycieczki.

Nazwa atrybutu	Typ	Opis/Uwagi
tripId	int	klucz główny tabeli
destination	varchar(50)	cel wycieczki (kraj/miasto/opis)
price	money	cena wycieczki
startDate	date	początek wycieczki
dueDate	date	koniec wycieczki
numberOfParticipants	int	liczba dostępnych biletów na wycieczkę

Nazwa atrybutu	Typ	Opis/Uwagi
countryId	int	klucz obcy łączący z tabelą Countries
uploadDate	datetime	data dodania oferty

```
CREATE TABLE [dbo].[Trips] (
    [tripId]               INT          IDENTITY (1, 1) NOT NULL,
    [destination]          VARCHAR (50) NULL,
    [price]                MONEY        CONSTRAINT [DF_Trips_prize] DEFAULT ((0))
NOT NULL,
    [startDate]            DATE         NULL,
    [dueDate]              DATE         NULL,
    [numberOfParticipants] INT          CONSTRAINT [DF_Trips_numberOfParticipants]
DEFAULT ((0)) NULL,
    [countryId]            INT          NULL,
    [uploadDate]           DATETIME     NULL,
    CONSTRAINT [PK_Trips] PRIMARY KEY CLUSTERED ([tripId] ASC),
    CONSTRAINT [FK_Trips_Countries] FOREIGN KEY ([countryId]) REFERENCES [dbo].
[Countries] ([countryId])
);
```

	tripId	destination	price	startDate	dueDate	numberOfParticipants	countryId	uploadDate
1	1	Londyn	5000,00	2024-07-01	2024-07-07	16	2	2024-06-01 00:00:00.000
2	2	Paryz	8000,00	2024-07-05	2024-07-15	20	3	2024-06-01 00:00:00.000
3	3	Barcelona	4500,00	2024-07-20	2024-07-27	24	4	2024-06-14 00:00:00.000
4	4	Lizbona	7000,00	2024-07-20	2024-07-31	20	5	2024-06-14 00:00:00.000

3. Widoki, procedury/funkcje, triggery

Widoki

AttractionsTickets

Informacje o biletach dla atrakcji.

```
CREATE VIEW [dbo].[AttractionsTickets] AS (
select Attractions.attractionId,
COALESCE(sum(ReservationDetails.numberOfTickets),0) as TicketsReserved,
Attractions.numberOfParticipants as numberOfParticipants,
Attractions.numberOfParticipants -
COALESCE(sum(ReservationDetails.numberOfTickets),0) as TicketsAvailable
FROM ReservationDetails
RIGHT OUTER JOIN Attractions on Attractions.attractionId =
ReservationDetails.attractionId
GROUP BY Attractions.attractionId, ReservationDetails.numberOfTickets,
Attractions.numberOfParticipants
)
```

attractionId	TicketsReserved	numberOfParticipants	TicketsAvailable
3	0	8	8
7	0	12	12
6	4	16	12
9	4	10	6
10	4	20	16
1	5	20	15
2	5	10	5
4	6	6	0
5	6	16	10
5	8	16	8
8	10	12	2

ClientsPayments

Dane o płatnościach dokonanych przez klientów

```
CREATE VIEW ClientsPayments AS
SELECT Clients.name,
Clients.surname,
Clients.address + ', ' + Clients.city + ', ' + Clients.zipCode + ', ' +
Clients.country as address,
Clients.phone,
Clients.email,
Clients.NIP,
ClientTypes.name as 'type',
Payments.paymentAmount,
Payments.dateOfPayment,
PaymentMethods.name AS 'payment type'
FROM Clients
JOIN ClientTypes ON ClientTypes.clientTypeId = Clients.clientTypeId
JOIN Reservations ON Reservations.clientId = Clients.clientId
JOIN Payments ON Payments.reservationId = Reservations.reservationId
JOIN PaymentMethods ON PaymentMethods.paymentMethodId = Payments.paymentMethodId;
```

	name	surname	address	phone	email	NIP	type	paymentAmount	dateOfPayment	payment type
1	Wiktor	Wojtas	ul. Sucha 34, Warszawa, 00-007, Polska	645221389	wwojtas@email.pl	1676283499	Firma	44200,00	2024-06-14	przelew bankowy
2	Filip	Kubski	ul. Wesola 2, Krakow, 30-001, Polska	696234112	filipo@email.pl	NULL	Osoba Prywatna	30900,00	2024-06-10	przelew bankowy
3	Jaroslaw	Jarzabkowski	ul. Slowackiego 220, Poznan, 60-002, Polska	752334121	jarekjar@email.pl	NULL	Osoba Prywatna	41750,00	2024-06-22	przelew bankowy
4	Janusz	Pogorzelski	ul. Wielka 34 m. 10, Wroclaw, 50-009, Polska	828828098	janekpog@email.pl	1617234788	Firma	56000,00	2024-06-20	przelew bankowy
5	Pawel	Bielinski	ul. Bacha 1 m. 2, Gdansk, 80-000, Polska	623466273	pawelek99@email.pl	NULL	Osoba Prywatna	31600,00	2024-06-30	przelew bankowy

ExtendedTripsInfo

Szczegółowe dane o wycieczkach.

```
CREATE VIEW ExtendedTripsInfo AS
WITH ReservationSums AS (
    SELECT tripId, SUM(numberOfParticipants) AS totalParticipants
    FROM Reservations
```

```
    WHERE reservationStatusId == 5
    GROUP BY tripId
)
SELECT Trips.tripId,
Trips.destination,
Trips.price,
Trips.startDate,
Trips.dueDate,
Trips.numberOfParticipants,
ReservationSums.totalParticipants AS participants,
STRING_AGG(Attractions.name, ', ') AS Attractions
FROM Trips
LEFT JOIN ReservationSums ON ReservationSums.tripId = Trips.tripId
LEFT JOIN Attractions ON Attractions.tripId = Trips.tripId
GROUP BY Trips.tripId,
Trips.destination,
Trips.price,
Trips.startDate,
Trips.dueDate,
Trips.numberOfParticipants,
ReservationSums.totalParticipants;
```

tripId	destination	price	startDate	dueDate	numberOfParticipants	participants	Attractions
1	Londyn	5000,00	2024-07-01	2024-07-07	16	14	Buckingham Palace, London Eye, Tower Bridge
2	Paryz	8000,00	2024-07-05	2024-07-15	20	5	Wieza Eiffla, Luwr, Musee dOrsay
3	Barcelona	4500,00	2024-07-20	2024-07-27	24	12	Sagrada Familia, Camp Nou
4	Lizbona	7000,00	2024-07-20	2024-07-31	20	4	Zamek sw. Jerzego, Wycieczka do Fatimy
5	Miami	2000,00	2024-07-01	2024-07-10	20	NULL	NULL

AttractionsForTrip

Dane o Atrakcjach dla danej wycieczki

```
CREATE VIEW AttractionsForTrip AS
SELECT Attractions.attractionId,
Attractions.name, Attractions.price,
Attractions.startDate, Attractions.dueDate,
Attractions.numberOfParticipants,
( Attractions.numberOfParticipants -
SUM(COALESCE(ReservationDetails.numberOfTickets,0)) )
as TicketsAvailable FROM Attractions LEFT JOIN ReservationDetails on
ReservationDetails.attractionId = Attractions.attractionId
GROUP BY Attractions.attractionId,
Attractions.name,
Attractions.price,
Attractions.startDate,
Attractions.dueDate,
Attractions.numberOfParticipants
```

	attractionId	name	price	startDate	dueDate	numberOfParticipants	TicketsAvailable
1	1	Wieza Eiffla	150,00	2024-07-05	2024-07-15	20	15
2	2	Luwr	200,00	2024-07-10	2024-07-18	10	5
3	3	Musee dOrsay	100,00	2024-07-11	2024-07-15	8	8
4	4	Buckingham Palace	500,00	2024-07-05	2024-07-05	6	0
5	5	London Eye	150,00	2024-07-01	2024-07-07	16	2
6	6	Tower Bridge	75,00	2024-07-01	2024-07-07	16	12
7	7	Sagrada Familia	150,00	2024-07-20	2024-07-27	12	12
8	8	Camp Nou	200,00	2024-07-22	2024-07-22	12	2
9	9	Zamek sw. Jerzego	100,00	2024-07-20	2024-07-28	10	6
10	10	Wycieczka do Fatimy	800,00	2024-07-27	2024-07-28	20	16

ClientsReservations

Informacje o rezerwacjach dla klientów

```
CREATE VIEW ClientsReservations AS
SELECT Clients.clientId,
Clients.name as ClientName,
ClientTypes.name as ClientType,
Reservations.numberOfParticipants,
ReservationStatuses.name as ReservationStatus,
Countries.name as Country,
Trips.destination,
Trips.startDate,
Trips.dueDate FROM Reservations
INNER JOIN Trips on Trips.tripId = Reservations.tripId
INNER JOIN Clients on Clients.clientId = Reservations.clientId
INNER JOIN ReservationStatuses on ReservationStatuses.reservationStatusId =
Reservations.reservationStatusId
INNER JOIN Countries on Countries.countryId = Trips.countryId
INNER JOIN ClientTypes on ClientTypes.clientTypeId = Clients.clientTypeId
```

	clientId	ClientName	ClientType	numberOfParticipants	ReservationStatus	Country	destination	startDate	dueDate
1	1	Wiktor	Firma	8	nowa	Wielka Brytania	Londyn	2024-07-01	2024-07-07
2	2	Filip	Osoba Prywatna	6	potwierdzona	Wielka Brytania	Londyn	2024-07-01	2024-07-07
3	3	Jaroslaw	Osoba Prywatna	5	oplacona	Francja	Paryz	2024-07-05	2024-07-15
4	4	Janusz	Firma	12	nowa	Hiszpania	Barcelona	2024-07-20	2024-07-27
5	5	Pawel	Osoba Prywatna	4	potwierdzona	Portugalia	Lizbona	2024-07-20	2024-07-31

ReservationDetailsPrize

Zsumowana cena dla każdych szczegółów rezerwacji

```
CREATE VIEW ReservationDetailsPrize as
SELECT ReservationDetails.resDetailId,
ReservationDetails.reservationId,
SUM(ReservationDetails.numberOfTickets*ReservationDetails.pricePerPerson) as
ReservationDetailsPrize FROM ReservationDetails
INNER JOIN Attractions ON ReservationDetails.attractionId =
```

```
Attractions.attractionId
GROUP BY ReservationDetails.resDetailId, ReservationDetails.reservationId
```

	resDetailId	reservationId	ReservationDetailsPrize
1	1	1	3000,00
2	2	1	1200,00
3	3	2	300,00
4	4	2	900,00
5	5	3	750,00
6	6	3	1000,00
7	7	4	2000,00
8	8	5	400,00
9	9	5	3200,00

ReservationAttractionsDetails

Szczegóły na temat zarezerowanych atrakcji

```
CREATE VIEW ReservationAttractionsDetails as
SELECT ReservationDetails.reservationId, ReservationDetails.resDetailId,
ReservationDetailsPrize.ReservationDetailsPrize, Attractions.name,
ReservationDetails.numberOfTickets,
COUNT(ReservationParticipants.resParticipantsId) as ParticipantsAssigned FROM
ReservationDetails
INNER JOIN Attractions on Attractions.attractionId =
ReservationDetails.attractionId
INNER JOIN ReservationDetailsPrize on ReservationDetailsPrize.resDetailId =
ReservationDetails.resDetailId
INNER JOIN ReservationParticipants on ReservationDetails.resDetailId =
ReservationParticipants.resDetailId
GROUP BY ReservationDetails.reservationId, ReservationDetails.resDetailId,
ReservationDetailsPrize.ReservationDetailsPrize, Attractions.name,
ReservationDetails.numberOfTickets
```

	reservationId	resDetailId	ReservationDetailsPrize	name	numberOfTickets	ParticipantsAssigned
1	1	1	3000,00	Buckingham Palace	6	6
2	1	2	1200,00	London Eye	8	8
3	2	3	300,00	Tower Bridge	4	4
4	2	4	900,00	London Eye	6	6
5	3	5	750,00	Wieza Eiffla	5	5
6	3	6	1000,00	Luwr	5	5
7	4	7	2000,00	Camp Nou	10	10
8	5	8	400,00	Zamek sw. Jerzego	4	4
9	5	9	3200,00	Wycieczka do Fatimy	4	4

ReservationParticipantsFullNames

Imiona i nazwiska uczestników rezerwacji

```
CREATE VIEW ReservationParticipantsFullNames as
SELECT Reservations.reservationId, (Participants.surname + ' ' +
Participants.name) as FullParticipantName
FROM Reservations
INNER JOIN Participants on Reservations.reservationId  =
Participants.reservationId
```

	reservationId	FullParticipantName
1	1	Karabin Miroslaw
2	1	Sochocki Karol
3	1	Pedrycz Kaja
4	1	Tworkowska Michalina
5	1	Lasak Benedykt
6	1	Ciepielewski Mikolaj
7	1	Daniluk Konrad
8	1	Rycerz Emilia
9	2	Kontny Lidia
10	2	Lapka Leopold
11	2	Szczepanowski Florian
12	2	Czupryna Angelika
13	2	Pasieczny Oskar
14	2	Chrustowska Kazimiera

TotalReservationsPrize

Cena za całą rezerwacje

```
CREATE VIEW TotalReservationsPrize as
SELECT Reservations.reservationId,
(SUM(ReservationDetailsPrize.ReservationDetailsPrize) +
FullTripPrize.FullTripPrize) as TotalPrize FROM Reservations
INNER JOIN ReservationDetailsPrize on ReservationDetailsPrize.reservationId =
Reservations.reservationId
INNER JOIN FullTripPrize on FullTripPrize.reservationId =
Reservations.reservationId
GROUP BY Reservations.reservationId, FullTripPrize.FullTripPrize
```

	reservationId	TotalPrize
1	1	44200,00
2	2	31200,00
3	3	41750,00
4	4	56000,00
5	5	31600,00

ReservationsPayments

Informacje na temat opłacenia rezerwacji

```
CREATE VIEW ReservationsPayments as
SELECT TotalReservationsPrize.reservationId, TotalReservationsPrize.TotalPrize,
SUM(Payments.paymentAmount) as TotalPaymentAmount,
(TotalReservationsPrize.TotalPrize - SUM(Payments.paymentAmount)) as
AmountLeftToPay  FROM TotalReservationsPrize
INNER JOIN Reservations on Reservations.reservationId =
TotalReservationsPrize.reservationId
INNER JOIN Payments on Payments.reservationId = Reservations.reservationId
GROUP BY TotalReservationsPrize.reservationId, TotalReservationsPrize.TotalPrize
```

	reservationId	TotalPrize	TotalPaymentAmount	AmountLeftToPay
1	2	31200,00	30900,00	300,00
2	5	31600,00	31600,00	0,00
3	3	41750,00	41750,00	0,00
4	1	44200,00	44200,00	0,00
5	4	56000,00	56000,00	0,00

TripsTickets

Informacje o biletach dla wycieczek.

```
CREATE VIEW [dbo].[TripsTickets] AS (
SELECT Trips.tripId, Trips.NumberOfParticipants as numbersOfParticipants,
COALESCE(sum(Reservations.numberOfParticipants),0) as TicketsReserved,
(Trips.numberOfParticipants - COALESCE(sum(Reservations.numberOfParticipants),0))
as TicketsAvailable FROM Reservations
RIGHT OUTER JOIN Trips on Trips.tripId = Reservations.tripId
GROUP BY Trips.tripId, Trips.numberOfParticipants
)
```

tripId	numbersOfParticipants	TicketsReserved	TicketsAvailable
1	16	14	2
2	20	5	15
3	24	12	12
4	20	4	16
5	20	0	20

Procedury/funkcje

GetClientReservations

Funkcja zwracająca szczegóły rezerwacji dla danego klienta

```
CREATE FUNCTION dbo.GetClientReservations (@clientId INT)
RETURNS TABLE
AS
```

```
RETURN (
    SELECT
        r.reservationId,
        r.tripId,
        t.destination,
        t.startDate,
        t.dueDate,
        rd.attractionId,
        a.name AS attractionName,
        rd.numberOfTickets,
        rd.prizePerPerson
    FROM
        Reservations r
        JOIN ReservationDetails rd ON r.reservationId = rd.reservationId
        JOIN Attractions a ON rd.attractionId = a.attractionId
        JOIN Trips t ON r.tripId = t.tripId
    WHERE
        r.clientId = @clientId
);
```

usage: select * from dbo.GetClientReservations(1)

Results Messages

reservationId	tripId	destination	startDate	dueDate	attractionId	attractionName	numberOfTickets	pricePerPerson
1	1	Londyn	2024-07-01	2024-07-07	4	Buckingham Palace	6	500,00
1	1	Londyn	2024-07-01	2024-07-07	5	London Eye	8	150,00

GetReservationsByCountry

Funkcja zwracająca listę rezerwacji dla danego kraju w określonym przedziale czasowym

```
CREATE FUNCTION dbo.GetReservationsByCountry (@countryId INT, @startDate DATE,
@endDate DATE)
RETURNS TABLE
AS
RETURN (
    SELECT
        r.reservationId,
        r.clientId,
        c.name AS clientName,
        t.destination,
        t.startDate,
        t.dueDate
    FROM
        Reservations r
        JOIN Clients c ON r.clientId = c.clientId
        JOIN Trips t ON r.tripId = t.tripId
    WHERE
        t.countryId = @countryId
        AND t.startDate >= @startDate
```

```
        AND t.dueDate <= @endDate
);
```

usage: select * from dbo.GetReservationsByCcountry(2, '2024-01-01', '2024-12-31');

reservationId	clientId	clientName	destination	startDate	dueDate
1	1	Wiktor	Londyn	2024-07-01	2024-07-07
2	2	Filip	Londyn	2024-07-01	2024-07-07

GetTopAttractions

Funkcja zwracająca najpopularniejsze atrakcje w zadanym przedziale czasowym

```
CREATE FUNCTION dbo.GetTopAttractions (@startDate DATE, @endDate DATE)
RETURNS TABLE
AS
RETURN (
    SELECT TOP (100) PERCENT
        a.attractionId,
        a.name AS attractionName,
        a.numberOfParticipants
    FROM
        Attractions a
    WHERE
        a.startDate >= @startDate
        AND a.dueDate <= @endDate
    ORDER BY
        a.numberOfParticipants DESC
);
```

select * from dbo.GetTopAttractions('2024-01-01', '2024-12-31');

attractionId	attractionName	numberOfParticipants
1	Wieza Eiffla	20
2	Luwr	10
3	Musee dOrsay	8
4	Buckingham Palace	6
5	London Eye	16
6	Tower Bridge	16
7	Sagrada Familia	12
8	Camp Nou	12
9	Zamek sw. Jerzego	10
10	Wycieczka do Fatimy	20

GetClientsByType

Funkcja zwracająca klientów według typu

```
CREATE FUNCTION dbo.GetClientsByType (@clientTypeId INT)
RETURNS TABLE
AS
RETURN (
    SELECT
        c.clientId,
        c.name,
        c.surname,
        c.address,
        c.city,
        c.country,
        c.zipCode,
        c.email,
        c.NIP
    FROM
        Clients c
    WHERE
        c.clientTypeId = @clientTypeId
);
```

select * from dbo.GetClientsByType(1)

clientId	name	surname	address	city	country	zipCode	email	NIP
2	Filip	Kubski	ul. Wesola 2	Krakow	Polska	30-001	filipo@email.pl	NULL
3	Jaroslaw	Jarzabkowski	ul. Slowackiego 220	Poznan	Polska	60-002	jarekjar@email.pl	NULL
5	Pawel	Bielinski	ul. Bacha 1 m. 2	Gdansk	Polska	80-000	pawelek99@email.pl	NULL

GetReservationsCountByMonth

Funkcja zwracająca liczbę rezerwacji na dany miesiąc

```
CREATE FUNCTION dbo.GetReservationsCountByMonth (@year INT, @month INT)
RETURNS INT
AS
BEGIN
    DECLARE @reservationCount INT;

    SELECT @reservationCount = COUNT(*)
    FROM
        Reservations
    WHERE
        YEAR(startDate) = @year AND MONTH(startDate) = @month;

    RETURN @reservationCount;
END;
```

usage: select dbo.GetReservationsCountByMonth(2024, 7) LiczbaRezerwacjiWLipcu

LiczbaRezerwacjiWLipcu
5

GetReservationStatus

Funkcja zwracająca status rezerwacji

```
CREATE FUNCTION dbo.GetReservationStatus (@reservationId INT)
RETURNS TABLE
AS
RETURN (
    SELECT
        r.reservationId,
        rs.name AS statusName
    FROM
        Reservations r
        JOIN ReservationStatuses rs ON r.reservationStatusId =
rs.reservationStatusId
    WHERE
        r.reservationId = @reservationId
);
```

usege: select * from dbo.GetReservationStatus(1)

reservationId	statusName
1	nowa

GetClientTrips

Funkcja zwracająca listę wycieczek dla danego klienta

```
CREATE FUNCTION dbo.GetClientTrips (@clientId INT)
RETURNS TABLE
AS
RETURN (
    SELECT
        r.reservationId,
        t.tripId,
        t.destination,
        t.startDate,
        t.dueDate
    FROM
        Reservations r
        JOIN Trips t ON r.tripId = t.tripId
    WHERE
        r.clientId = @clientId
);
```

usage: select * from dbo.GetClientTrips(1)

reservationId	tripId	destination	startDate	dueDate
1	1	Londyn	2024-07-01	2024-07-07

Procedury

AddClient

Dodaje klienta do bazy danych

```
CREATE PROCEDURE [dbo].[AddClient]
    @Phone VARCHAR(50),
    @Name VARCHAR(50),
    @Surname VARCHAR(50),
    @Address VARCHAR(50),
    @City VARCHAR(50),
    @Country VARCHAR(50),
    @ZipCode VARCHAR(50),
    @ClientTypeId INT,
    @Email VARCHAR(50) = NULL,
    @NIP VARCHAR(50) = NULL
AS
BEGIN
    INSERT INTO dbo.Clients (phone, name, surname, address, city, country,
zipCode, email, NIP, clientTypeId)
    VALUES (@Phone, @Name, @Surname, @Address, @City, @Country, @ZipCode, @Email,
@NIP, @ClientTypeId);

    SELECT SCOPE_IDENTITY() AS NewClientId;
END;
```

usage: EXEC AddClient '234234234', 'Sam', 'Thomson', 'os. Kopernikowe 111/2', 'Warszawa', 'Poland', '12-000', 1, 'samt@mail.com';

AddParticipant

Dodaje uczestnika do rezerwacji

```
CREATE PROCEDURE [dbo].[AddParticipant]
    @ParticipantId INT,
    @Name VARCHAR(50),
    @Surname VARCHAR(50),
    @ReservationId INT
AS
BEGIN
    INSERT INTO dbo.Participants (participantId, name, surname, reservationId)
    VALUES (@ParticipantId, @Name, @Surname, @ReservationId);
```

```
    SELECT SCOPE_IDENTITY() AS NewParticipantId;
END;
```

usage: EXEC AddParticipant 'Aleksander', 'Nowak', 2;

AddReservation

Dodaje rezerwację. Procedura sprawdza, czy istnieje klient z podanym id, oraz czy wycieczka na która chcemy stworzyć rezerwacje istnieje. Sprawdza również, czy liczba biletów na rezerwacje nie przekracza dostępnej liczby biletów.

```
CREATE PROCEDURE [dbo].[AddReservation]
    @ClientId INT,
    @NumberOfParticipants INT = 0,
    @TripId INT,
    @ReservationStatusId INT,
    @PricePerPerson MONEY = 0
AS
BEGIN
    IF NOT EXISTS (SELECT 1 FROM dbo.Clients WHERE clientId = @ClientId)
    BEGIN
        -- Rzucenie błędu, jeśli podany ClientId nie istnieje
        RAISERROR('Invalid ClientId. The provided ClientId does not exist.', 16,
1);
        RETURN;
    END

    IF NOT EXISTS (SELECT 1 FROM dbo.Trips WHERE tripId = @TripId)
    BEGIN
        RAISERROR('Invalid TripId. The provided TripId does not exist.', 16, 1);
        RETURN;
    END

    DECLARE @TicketsAvailable INT;

    SELECT @TicketsAvailable = TicketsAvailable
    FROM TripsTickets
    WHERE tripId = @TripId;

    IF @TicketsAvailable IS NULL
    BEGIN
        RAISERROR('Invalid TripId. No tickets available for the provided TripId.',
16, 1);
        RETURN;
    END

    IF @NumberOfParticipants > @TicketsAvailable
    BEGIN
        RAISERROR('Not enough tickets available. The number of participants
exceeds the available tickets.', 16, 1);
        RETURN;
```

```
    END

    INSERT INTO dbo.Reservations (clientId, numberOfParticipants, tripId,
reservationStatusId, pricePerPerson)
    VALUES (@ClientId, @NumberOfParticipants, @TripId, @ReservationStatusId,
@PricePerPerson);

    SELECT SCOPE_IDENTITY() AS NewReservationId;
END;
```

usage: EXEC AddReservation 1, 6, 3, 1, 700;

AddTrip

Dodaje wycieczkę do bazy danych.

```
CREATE PROCEDURE [dbo].[AddTrip]
    @Destination VARCHAR(50) = NULL,
    @Price MONEY = 0,
    @StartDate DATE = NULL,
    @DueDate DATE = NULL,
    @NumberOfParticipants INT = 0,
    @CountryId INT = NULL,
    @UploadDate DATETIME = NULL
AS
BEGIN
    INSERT INTO dbo.Trips (destination, price, startDate, dueDate,
numberOfParticipants, countryId, uploadDate)
    VALUES (@Destination, @Price, @StartDate, @DueDate, @NumberOfParticipants,
@CountryId, @UploadDate);

    SELECT SCOPE_IDENTITY() AS NewTripId;
END;
```

usage: EXEC AddTrip 'Gdańsk', 100, '2024-02-26', '2024-03-03', 40;

UpdateClient

Aktualizuje dane o kliencie.

```
CREATE PROCEDURE [dbo].[UpdateClient]
    @ClientId INT,
    @Phone VARCHAR(50) = NULL,
    @Name VARCHAR(50) = NULL,
    @Surname VARCHAR(50) = NULL,
    @Address VARCHAR(50) = NULL,
    @City VARCHAR(50) = NULL,
    @Country VARCHAR(50) = NULL,
    @ZipCode VARCHAR(50) = NULL,
```

```
    @ClientTypeId INT = NULL,
    @Email VARCHAR(50) = NULL,
    @NIP VARCHAR(50) = NULL
AS
BEGIN
    IF NOT EXISTS (SELECT 1 FROM dbo.Clients WHERE clientId = @ClientId)
    BEGIN
        RAISERROR('Invalid ClientId. The provided ClientId does not exist.', 16,
1);
        RETURN;
    END

    -- Aktualizacja danych klienta
    UPDATE dbo.Clients
    SET
        phone = COALESCE(@Phone, phone),
        name = COALESCE(@Name, name),
        surname = COALESCE(@Surname, surname),
        address = COALESCE(@Address, address),
        city = COALESCE(@City, city),
        country = COALESCE(@Country, country),
        zipCode = COALESCE(@ZipCode, zipCode),
        email = COALESCE(@Email, email),
        NIP = COALESCE(@NIP, NIP),
        clientTypeId = COALESCE(@ClientTypeId, clientTypeId)
    WHERE clientId = @ClientId;
END;
```

usage: EXEC UpdateClient '234234234', 'Sam', 'Thomason', 'os. Kopernikowe 111/2', 'Kraków', 'Poland', '12-000', 1, 'samt@mail.com';

UpdateReservation

Aktualizuje informację o rezerwacji. Sprawdza, czy reserwacja o podanym id istnieje. Sprawdza, czy podana liczba biletów nie przekracza liczby dostępnych biletów, a także, czy liczba przypisanych uczestników wycieczki nie jest większa od nowej podanej wartości.

```
CREATE PROCEDURE [dbo].[UpdateReservation]
    @ReservationId INT,
    @NumberOfParticipants INT = NULL,
    @PricePerPerson MONEY = NULL
AS
BEGIN
    IF NOT EXISTS (SELECT 1 FROM dbo.Reservations WHERE reservationId =
@ReservationId)
    BEGIN
        RAISERROR('Invalid ReservationId. The provided ReservationId does not
exist in the Reservations table.', 16, 1);
        RETURN;
    END
```

```
    IF @NumberOfParticipants IS NOT NULL
    BEGIN
        DECLARE @CurrentParticipants INT;

        SELECT @CurrentParticipants = COUNT(participantId)
        FROM Participants
        WHERE reservationId = @ReservationId;

        IF @NumberOfParticipants < @CurrentParticipants
        BEGIN
            RAISERROR('Cannot update reservation. The specified number of
participants is less than the current number of assigned participants.', 16, 1);
            RETURN;
        END
    END

    IF @NumberOfParticipants IS NOT NULL
    BEGIN
        DECLARE @TicketsTaken INT;
        DECLARE @TicketsAvailable INT;

        SELECT @TicketsTaken = numberOfParticipants
        FROM Reservations
        WHERE reservationId = @ReservationId;

        SELECT @TicketsAvailable = (tt.TicketsAvailable - @TicketsTaken)
        FROM TripsTickets tt
        WHERE tt.tripId = (SELECT tripId FROM Reservations WHERE reservationId =
@ReservationId);

        IF @NumberOfParticipants > @TicketsAvailable
        BEGIN
            RAISERROR('Cannot update reservation. The number of participants
exceeds the available tickets.', 16, 1);
            RETURN;
        END
    END

    UPDATE dbo.Reservations
    SET
        numberOfParticipants = COALESCE(@NumberOfParticipants,
numberOfParticipants),
        pricePerPerson = COALESCE(@PricePerPerson, pricePerPerson)
    WHERE reservationId = @ReservationId;
END;
```

usage: EXEC UpdateReservation 3, 20, 700;

UpdateReservationDetails

Aktualizuje szczegóły rezerwacji. Procedura sprawdza, czy resDetailId istnieje, a także, czy liczba biletów przypisanych do reservationDetails nie przekracza liczby dostępnych biletów na atrakcję. Sprawdza również,

czy liczba biletów nie jest mniejsza, niż liczba uczestników przypisanych do reservationDetails.

```
CREATE PROCEDURE [dbo].[UpdateReservationDetails]
    @ResDetailId INT,
    @AttractionId INT = NULL,
    @NumberOfTickets INT = NULL,
    @PricePerPerson MONEY = NULL
AS
BEGIN
    IF NOT EXISTS (SELECT 1 FROM dbo.ReservationDetails WHERE resDetailId =
@ResDetailId)
    BEGIN
        RAISERROR('Invalid ResDetailId. The provided ResDetailId does not exist in
the ReservationDetails table.', 16, 1);
        RETURN;
    END

    IF @NumberOfTickets IS NOT NULL
    BEGIN
        DECLARE @TicketsAvailableForAttraction INT;
        DECLARE @CurrentAssignedParticipants INT;

        SELECT @TicketsAvailableForAttraction = at.TicketsAvailable
        FROM AttractionsTickets at
        WHERE at.attractionId = (SELECT attractionId FROM ReservationDetails WHERE
resDetailId = @ResDetailId);

        SELECT @CurrentAssignedParticipants = COUNT(participantId)
        FROM ReservationParticipants
        WHERE resDetailId = @ResDetailId;

        IF @NumberOfTickets < @CurrentAssignedParticipants
        BEGIN
            RAISERROR('Cannot update reservation details. The specified number of
tickets is less than the current number of assigned participants.', 16, 1);
            RETURN;
        END

        IF @NumberOfTickets > @TicketsAvailableForAttraction
        BEGIN
            RAISERROR('Cannot update reservation details. The number of tickets
for the attraction exceeds the available capacity.', 16, 1);
            RETURN;
        END
    END

    UPDATE dbo.ReservationDetails
    SET
        attractionId = COALESCE(@AttractionId, attractionId),
        numberOfTickets = COALESCE(@NumberOfTickets, numberOfTickets),
        pricePerPerson = COALESCE(@PricePerPerson, pricePerPerson)
    WHERE resDetailId = @ResDetailId;
END;
```

usage: EXEC UpdateReservationDetails 3, 3, 2, 130;

UpdateTrip

Aktualizuje wycieczkę, sprawdza, czy wycieczka istnieje w bazie danych.

```
CREATE PROCEDURE [dbo].[UpdateTrip]
    @TripId INT,
    @Destination VARCHAR(50) = NULL,
    @Price MONEY = NULL,
    @StartDate DATE = NULL,
    @DueDate DATE = NULL,
    @NumberOfParticipants INT = NULL,
    @CountryId INT = NULL,
    @UploadDate DATETIME = NULL
AS
BEGIN
    IF NOT EXISTS (SELECT 1 FROM dbo.Trips WHERE tripId = @TripId)
    BEGIN
        RAISERROR('Invalid TripId. The provided TripId does not exist.', 16, 1);
        RETURN;
    END

    UPDATE dbo.Trips
    SET
        destination = COALESCE(@Destination, destination),
        price = COALESCE(@Price, price),
        startDate = COALESCE(@StartDate, startDate),
        dueDate = COALESCE(@DueDate, dueDate),
        numberOfParticipants = COALESCE(@NumberOfParticipants,
numberOfParticipants),
        countryId = COALESCE(@CountryId, countryId),
        uploadDate = COALESCE(@UploadDate, uploadDate)
    WHERE tripId = @TripId;
END;
```

usage: EXEC UpdateTrip 4, 'Kraków', 20, '2024-01-04', '2024-01-04', 30, 1, '2023-11-30';

4. Inne

Wygenerowane losowo dane testowe (imiona i nazwiska uczestników) ręcznie wprowadzono do bazy danych.