Bazy danych

Autorzy: Jakub Skwarczek, Tymoteusz Szwech, Jakub Warchoł

1. Wymagania i funkcje systemu

System wspomaga działalność firmy świadczącej usługi turystyczne. Oferuje on rezerwację różnorodnych wycieczek z określoną datą, limitem miejsc i ceną. Klienci mogą dodatkowo rezerwować usługi i atrakcje związane z daną wycieczką, które również mają swoje limity miejsc i ceny.

Klientami są zarówno osoby prywatne, jak i firmy, które dokonują rezerwacji i płatności za uczestników wycieczki. Przy rezerwacji klient podaje liczbę miejsc oraz wybiera dodatkowe usługi, a najpóźniej na tydzień przed wyjazdem musi podać imiona i nazwiska uczestników. Brak tych danych lub pełnej wpłaty skutkuje anulowaniem zamówienia.

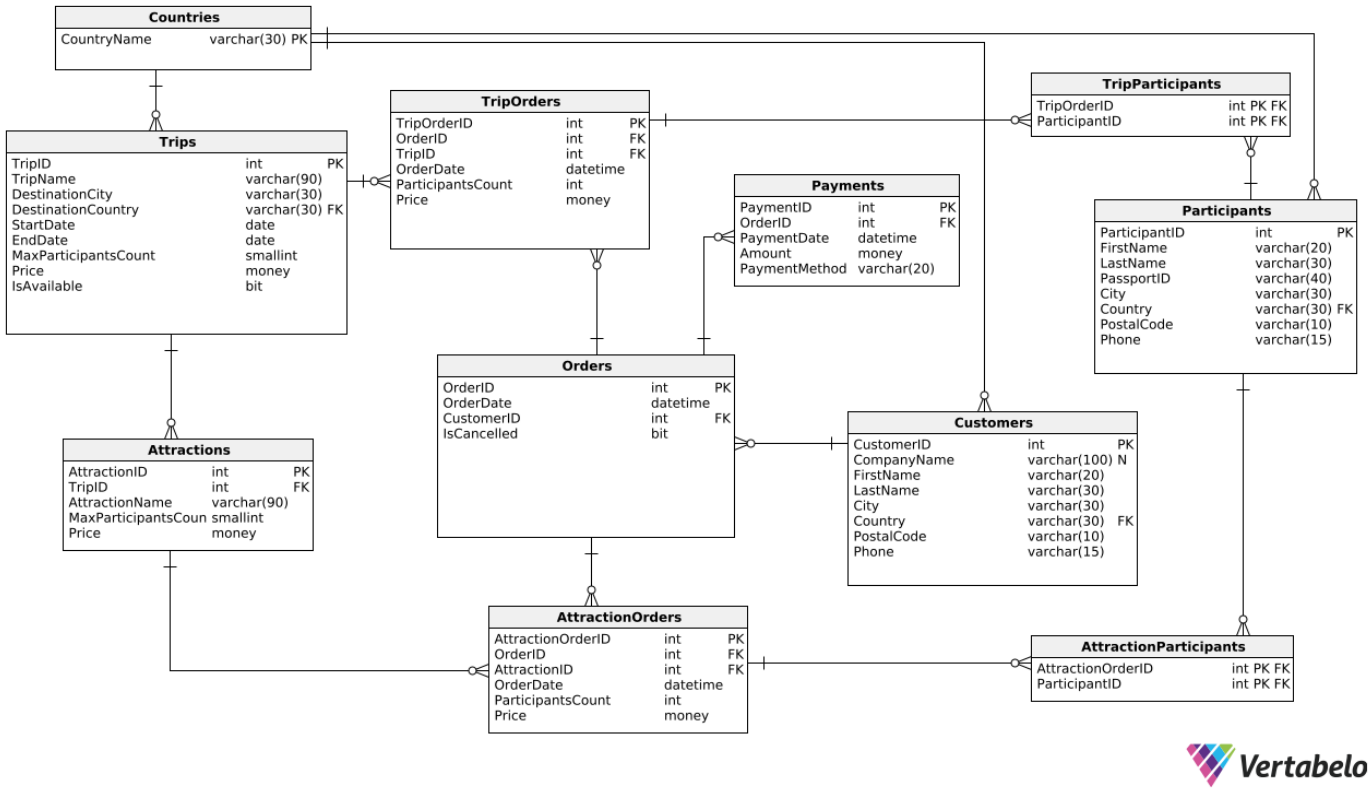
Rezerwacje dodatkowych usług są możliwe tylko wraz z rezerwacją wycieczki. Zmiany w rezerwacji można wprowadzać do tygodnia przed wyjazdem. Po tym terminie zamówienie musi być w pełni opłacone i nie można wprowadzać żadnych zmian. System zapewnia przejrzystość i wygodę obsługi, wspierając efektywne zarządzanie wycieczkami i usługami dodatkowymi.

Lista funkcji jakie użytkownik może wykonywać w systemie.

1. Uzyskanie informacji na temat dostępnej oferty wraz z ilością miejsc oraz ceną.
2. Dodanie rezerwacji wraz z wymaganymi danymi.
3. Zmiana informacji na temat rezerwacji.
4. Anulowanie rezerwacji.
5. Rezerwacja usług/atrakcji w ramach jednej wycieczki.
6. Dodanie informacji na temat płatności.
7. Rejestracja i modyfikacja danych uczestników wycieczki.

2. Baza danych

Schemat bazy danych



Opis poszczególnych tabel

Nazwa tabeli: **Countries**

- Opis: Tabela słownikowa zawierająca nazwy państw.

Nazwa atrybutu	Typ	Opis/Uwagi
CountryName	varchar(30)	Nazwa państwa (**PK, FK**)

- kod DDL

```
CREATE TABLE Countries (
    CountryName varchar(30)  NOT NULL,
    CONSTRAINT Countries_pk PRIMARY KEY  (CountryName)
);
```

Nazwa tabeli: **Trips**

- Opis: Tabela zawierająca informacje dotyczące dostępnych do zamówienia wycieczek.

Nazwa atrybutu	Typ	Opis/Uwagi
TripID	int	Identyfikator wycieczki (**PK**)
TripName	varchar(90)	Nazwa wycieczki
DestinationCity	varchar(30)	Miasto, do którego jest wycieczka
DestinationCountry	varchar(30)	Kraj, do którego jest wycieczka (**FK**)
StartDate	date	Początek wycieczki; **StartDate < EndDate** - data początku jest przed datą końca
EndDate	date	Koniec wycieczki
MaxParticipantsCount	smallint	Maksymalna liczba osób, które mogą uczestniczyć; **MaxParticipantsCount > 0**
Price	money	Koszt wycieczki; **Price >= 0**
IsAvailable	bit	Czy wycieczka jest dostępna do zamówienia (0 - nie, 1 - tak); **DEFAULT - 0**

- kod DDL

```
CREATE TABLE Trips (
    TripID int  NOT NULL,
    TripName varchar(90)  NOT NULL,
    DestinationCity varchar(30)  NOT NULL,
    DestinationCountry varchar(30)  NOT NULL,
    StartDate date  NOT NULL,
    EndDate date  NOT NULL,
    MaxParticipantsCount smallint  NOT NULL,
    Price money  NOT NULL,
    IsAvailable bit  NOT NULL DEFAULT 0,
    CONSTRAINT Trips_DateCheck CHECK (StartDate < EndDate),
    CONSTRAINT Trips_PriceCheck CHECK (Price >= 0),
    CONSTRAINT Trips_MPCheck CHECK (MaxParticipantsCount > 0),
    CONSTRAINT Trips_pk PRIMARY KEY  (TripID)
);

ALTER TABLE Trips ADD CONSTRAINT Trips_Countries
    FOREIGN KEY (DestinationCountry)
    REFERENCES Countries (CountryName);
```

Nazwa tabeli: **Attractions**

- Opis: Tabela zawierająca listę dostępnych atrakcji dla wycieczek.

Nazwa atrybutu	Typ	Opis/Uwagi
AttractionID	int	Identyfikator atrakcji (**PK**)
TripID	int	Identyfikator wycieczki (**FK**)
AttractionName	varchar(90)	Nazwa atrakcji
MaxParticipantsCount	smallint	Maksymalna ilość uczestników; **MaxParticipantsCount > 0**
Price	money	Koszt atrakcji; **Price >= 0**

- kod DDL

```
CREATE TABLE Attractions (
    AttractionID int  NOT NULL,
    TripID int  NOT NULL,
    AttractionName varchar(90)  NOT NULL,
    MaxParticipantsCount smallint  NOT NULL,
    Price money  NOT NULL,
    CONSTRAINT Attractions_PriceCheck CHECK (Price >= 0),
    CONSTRAINT Attractions_MPCheck CHECK (MaxParticipantsCount > 0),
    CONSTRAINT Attractions_pk PRIMARY KEY  (AttractionID)
);

ALTER TABLE Attractions ADD CONSTRAINT Attractions_Trips
    FOREIGN KEY (TripID)
    REFERENCES Trips (TripID);
```

Nazwa tabeli: **Orders**

- Opis: Tabela zawierająca najważniejsze informacje dotyczące głównego zamówienia tj. datę oraz identyfikator klienta.

Nazwa atrybutu	Typ	Opis/Uwagi
OrderID	int	Identyfikator zamówienia (**PK**)
OrderDate	int	Data złożenia zamówienia
CustomerID	datetime	Identyfikator klienta, który złożył zamówienie (**FK**)

Nazwa atrybutu	Typ	Opis/Uwagi
IsCancelled	bit	Czy zamówienie zostało anulowane (0 - nie, 1 - tak); **DEFAULT - 0**

- kod DDL

```
CREATE TABLE Orders (
    OrderID int  NOT NULL,
    OrderDate datetime  NOT NULL,
    CustomerID int  NOT NULL,
    IsCancelled bit  NOT NULL DEFAULT 0,
    CONSTRAINT Orders_pk PRIMARY KEY  (OrderID)
);

ALTER TABLE Orders ADD CONSTRAINT Orders_Customers
    FOREIGN KEY (CustomerID)
    REFERENCES Customers (CustomerID);
```

Nazwa tabeli: **Payments**

- Opis: Tabela zawierająca informacje dotyczące opłat: daty ich wykonania, kwoty, oraz tego jakiego zamówienia dotyczą.

Nazwa atrybutu	Typ	Opis/Uwagi
PaymentID	int	Identyfikator płatności (**PK**)
OrderID	int	Identyfikator zamówienia, które jest opłacane (**FK**)
PaymentDate	datetime	Data dokonania płatności
Amount	money	Kwota płatności; **Amount >= 0**
PaymentMethod	varchar(20)	Metoda płatnośc; **PaymentMethod IN ('Cash', 'Card', 'Check')**

- kod DDL

```
CREATE TABLE Payments (
    PaymentID int  NOT NULL,
    OrderID int  NOT NULL,
    PaymentDate datetime  NOT NULL,
    Amount money  NOT NULL,
    PaymentMethod varchar(20) NOT NULL,
    CONSTRAINT Payments_AmountCheck CHECK (Amount >= 0),
    CONSTRAINT Payments_PaymentMethod CHECK (PaymentMethod IN ('Cash', 'Card', 'Check')),
    CONSTRAINT Payments_pk PRIMARY KEY  (PaymentID)
);

ALTER TABLE Payments ADD CONSTRAINT Payments_Orders
    FOREIGN KEY (OrderID)
    REFERENCES Orders (OrderID);
```

Nazwa tabeli: **TripOrders**

- Opis: Tabela z zamówieniami wycieczek, zawierająca informacje między innymi na temat daty złożenia zamówienia.

Nazwa atrybutu	Typ	Opis/Uwagi
TripOrderID	int	Identyfikator zamówienia wycieczki (**PK**)
OrderID	int	Identyfikator zamówienia (**FK**)
TripID	int	Identyfikator wycieczki, która została zamówiona (**FK**)
OrderDate	datetime	Data, kiedy zostało złożone zamówienie
ParticipantsCount	int	Liczba uczestników zamówionej wycieczki; **ParticipantsCount > 0**
Price	money	Cena zamówienia; **Price >= 0**

- kod DDL

```
CREATE TABLE TripOrders (
    TripOrderID int  NOT NULL,
    OrderID int  NOT NULL,
    TripID int  NOT NULL,
    OrderDate datetime  NOT NULL,
    ParticipantsCount int  NOT NULL,
    Price money  NOT NULL,
    CONSTRAINT TripOrders_PriceCheck CHECK (Price >= 0),
    CONSTRAINT TripOrders_ParticipantCountCheck CHECK (ParticipantsCount > 0),
    CONSTRAINT OrderID PRIMARY KEY  (TripOrderID)
);

ALTER TABLE TripOrders ADD CONSTRAINT TripOrders_Orders
    FOREIGN KEY (OrderID)
    REFERENCES Orders (OrderID);
```

```
ALTER TABLE TripOrders ADD CONSTRAINT TripOrders_Trips
    FOREIGN KEY (TripID)
    REFERENCES Trips (TripID);
```

Nazwa tabeli: **TripParticipants**

- Opis: Tabela zawierająca identyfikatory uczestników powiązane z konkretnymi zamówieniami wycieczek. Powiązani uczestnicy są na nie zapisani.

Nazwa atrybutu	Typ	Opis/Uwagi
TripOrderID	int	Identyfikator zamówienia wycieczki (**PK, FK**)
ParticipantID	int	Identyfikator uczestnika (**PK, FK**)

- kod DDL

```
CREATE TABLE TripParticipants (
    TripOrderID int  NOT NULL,
    ParticipantID int  NOT NULL,
    CONSTRAINT TripParticipants_pk PRIMARY KEY  (TripOrderID,ParticipantID)
);

ALTER TABLE TripParticipants ADD CONSTRAINT TripParticipants_Participants
    FOREIGN KEY (ParticipantID)
    REFERENCES Participants (ParticipantID);

ALTER TABLE TripParticipants ADD CONSTRAINT TripParticipants_TripOrders
    FOREIGN KEY (TripOrderID)
    REFERENCES TripOrders (TripOrderID);
```

Nazwa tabeli: **AttractionOrders**

- Opis: Dodatkowe zamówienia atrakcji podpięte pod zamówienie.

Nazwa atrybutu	Typ	Opis/Uwagi
AttractionOrderID	int	Identyfikator zamówienia atrakcji (**PK**)
OrderID	int	Identyfikator zamówienia wycieczki (**FK**)
AttractionID	int	Identyfikator atrakcji (**FK**)
OrderDate	datetime	Data, kiedy zostało złożone zamówienie
ParticipantsCount	int	Liczba uczestników; **ParticipantsCount > 0**
Price	money	Koszt zamówienia; **Price >= 0**

- kod DDL

```
CREATE TABLE AttractionOrders (
    AttractionOrderID int  NOT NULL,
    OrderID int  NOT NULL,
    AttractionID int  NOT NULL,
    OrderDate datetime  NOT NULL,
    ParticipantsCount int  NOT NULL,
    Price money  NOT NULL,
    CONSTRAINT AttractionOrders_PriceCheck CHECK (Price >= 0),
    CONSTRAINT AttractionOrders_PCCheck CHECK (ParticipantsCount > 0),
    CONSTRAINT AttractionOrders_pk PRIMARY KEY  (AttractionOrderID)
);

ALTER TABLE AttractionOrders ADD CONSTRAINT AttractionOrders_Attractions
    FOREIGN KEY (AttractionID)
    REFERENCES Attractions (AttractionID);

ALTER TABLE AttractionOrders ADD CONSTRAINT AttractionOrders_Orders
    FOREIGN KEY (OrderID)
    REFERENCES Orders (OrderID);
```

Nazwa tabeli: **AttractionParticipants**

- Opis: Tabela zawierająca identyfikatory uczestników powiązane z konkretnymi zamówieniami atrakcji. Powiązani uczestnicy są na nie zapisani.

Nazwa atrybutu	Typ	Opis/Uwagi
AttractionOrderID	int	Identyfikator zamówienia atrakcji (**PK, FK**)
ParticipantID	int	Identyfikator uczestnika (**PK, FK**)

- kod DDL

```
CREATE TABLE AttractionParticipants (
    AttractionOrderID int  NOT NULL,
    ParticipantID int  NOT NULL,
```

```
    CONSTRAINT AttractionParticipants_pk PRIMARY KEY  (AttractionOrderID,ParticipantID)
);

ALTER TABLE AttractionParticipants ADD CONSTRAINT do_nazwania_AttractionOrders
    FOREIGN KEY (AttractionOrderID)
    REFERENCES AttractionOrders (AttractionOrderID);

ALTER TABLE AttractionParticipants ADD CONSTRAINT do_nazwania_Participants
    FOREIGN KEY (ParticipantID)
    REFERENCES Participants (ParticipantID);
```

Nazwa tabeli: **Participants**

- Opis: Tabela zawierająca informacje na temat uczestników.

Nazwa atrybutu	Typ	Opis/Uwagi
ParticipantID	int	Identyfikator uczestnika (**PK**)
FirstName	varchar(20)	Imię uczestnika
LastName	varchar(30)	Nazwisko uczestnika
PassportID	varchar(40)	Identyfikator paszportu uczesnika
City	varchar(30)	Miasto zamieszkania uczestnika (**FK**)
Country	varchar(30)	Kraj, z którego pochodzi uczestnik
PostalCode	varchar(10)	Kod pocztowy
Phone	varchar(15)	Telefon kontaktowy do uczestnika

- kod DDL

```
CREATE TABLE Participants (
    ParticipantID int  NOT NULL,
    FirstName varchar(20)  NOT NULL,
    LastName varchar(30)  NOT NULL,
    PassportID varchar(40)  NOT NULL,
    City varchar(30)  NOT NULL,
    Country varchar(30)  NOT NULL,
    PostalCode varchar(10)  NOT NULL,
    Phone varchar(15)  NOT NULL,
    CONSTRAINT Participants_pk PRIMARY KEY  (ParticipantID)
);

ALTER TABLE Participants ADD CONSTRAINT Participants_Countries
    FOREIGN KEY (Country)
    REFERENCES Countries (CountryName);
```

Nazwa tabeli: **Customers**

- Opis: Tabela z listą klientów oraz ich danymi.

Nazwa atrybutu	Typ	Opis/Uwagi
CustomerID	int	Identyfikator klienta (**PK**)
CompanyName	varchar(100)	Nazwa firmy klienta
FirstName	varchar(20)	Imię klienta / reprezentanta firmy
LastName	varchar(30)	Nazwisko klienta / reprezentanta firmy
City	varchar(30)	Miasto, w którym znajduje się firma
Country	varchar(30)	Kraj, w którym znajduje się firma (**FK**)
PostalCode	varchar(10)	Kod pocztowy
Phone	varchar(15)	Telefon kontaktowy do klienta

- kod DDL

```
CREATE TABLE Customers (
    CustomerID int  NOT NULL,
    CompanyName varchar(100)  NULL,
    FirstName varchar(20)  NOT NULL,
    LastName varchar(30)  NOT NULL,
    City varchar(30)  NOT NULL,
    Country varchar(30)  NOT NULL,
    PostalCode varchar(10)  NOT NULL,
    Phone varchar(15)  NOT NULL,
    CONSTRAINT Customers_pk PRIMARY KEY  (CustomerID)
);

ALTER TABLE Customers ADD CONSTRAINT Customers_Countries
```

```
    FOREIGN KEY (Country)
    REFERENCES Countries (CountryName);
```

3. Widoki, procedury/funkcje, triggery

Widoki

Nazwa widoku: **TripParticipantsCount**

- Opis: Widok ten wyświetla sumę uczestników, która jest zapisana na konkretną wycieczkę. Oprócz tego podaje maksymalną liczbę uczestników na tę wycieczkę oraz ilość wolnych miejsc.

```
CREATE VIEW TripParticipantsCount
AS
SELECT Trips.TripID, StartDate AS TripDate, ISNULL(SUM(ParticipantsCount), 0) AS SumParticipants,
       MaxParticipantsCount, MaxParticipantsCount - ISNULL(SUM(ParticipantsCount), 0) as SlotsLeft
FROM TripOrders
RIGHT JOIN Trips ON Trips.TripID = TripOrders.TripID
WHERE IsAvailable = 1
GROUP BY Trips.TripID, StartDate, MaxParticipantsCount;
```

TripID	TripDate	SumParticipants	MaxParticipantsCount	SlotsLeft
1	2024-06-15	7	30	23
2	2024-07-10	2	25	23
3	2024-08-05	1	20	19
4	2024-09-01	1	35	34
5	2024-10-01	1	40	39
6	2025-05-14	1	15	14
7	2024-11-15	1	10	9
8	2024-12-01	0	50	50
9	2024-04-10	0	30	30
10	2025-03-14	3	25	22

Nazwa widoku: **AttractionParticipantsCount**

- Opis: Widok ten wyświetla sumę uczestników, która jest zapisana na konkretną atrakcję. Oprócz tego podaje maksymalną liczbę uczestników atrakcji, numer wycieczki, do której ta atrakcja jest przypisana, jak i liczbę pozostałych miejsc.

```
CREATE VIEW AttractionParticipantsCount
AS
SELECT Trips.TripID, Attractions.AttractionID, Trips.StartDate AS TripDate,
       ISNULL(SUM(ParticipantsCount), 0) AS SumParticipants, Attractions.MaxParticipantsCount,
       Attractions.MaxParticipantsCount - ISNULL(SUM(ParticipantsCount), 0) as SlotsLeft
FROM AttractionOrders
RIGHT JOIN Attractions ON Attractions.AttractionID = AttractionOrders.AttractionID
JOIN Trips ON Trips.TripID = Attractions.TripID
WHERE IsAvailable = 1
GROUP BY Trips.TripID, Attractions.AttractionID, Trips.StartDate, Attractions.MaxParticipantsCount;
```

TripID	AttractionID	TripDate	SumParticipants	MaxParticipantsCount	SlotsLeft
1	1	2024-06-15	0	30	30
1	2	2024-06-15	0	30	30
1	3	2024-06-15	0	30	30
2	4	2024-07-10	0	25	25
2	5	2024-07-10	1	25	24
2	6	2024-07-10	0	25	25
3	7	2024-08-05	0	20	20
3	8	2024-08-05	0	20	20
3	9	2024-08-05	0	20	20
4	10	2024-09-01	0	35	35
4	11	2024-09-01	0	35	35
4	12	2024-09-01	0	35	35
5	13	2024-10-01	1	40	39
5	14	2024-10-01	0	40	40

TripID	AttractionID	TripDate	SumParticipants	MaxParticipantsCount	SlotsLeft
5	15	2024-10-01	0	40	40
6	16	2025-05-14	1	15	14
6	17	2025-05-14	0	15	15
6	18	2025-05-14	0	15	15
7	19	2024-11-15	0	10	10
7	20	2024-11-15	0	10	10
7	21	2024-11-15	0	10	10
8	22	2024-12-01	0	50	50
8	23	2024-12-01	0	50	50
8	24	2024-12-01	0	50	50
9	25	2024-04-10	0	30	30
9	26	2024-04-10	0	30	30
9	27	2024-04-10	0	30	30
10	28	2025-03-14	0	25	25
10	29	2025-03-14	0	25	25
10	30	2025-03-14	0	25	25
11	31	2025-02-05	0	45	45
11	32	2025-02-05	0	45	45
11	33	2025-02-05	0	45	45
12	34	2025-01-01	0	20	20
12	35	2025-01-01	0	20	20
12	36	2025-01-01	0	20	20

Nazwa widoku: **TotalPrice**

- Opis: Widok ten wyświetla sumę kosztów wszystkich zamówionych wycieczek oraz atrakcji dla konkretnego zamówienia. Wyświetla również sumę wszelkich opłat wykonanych w ramach tego zamówienie.

```
CREATE VIEW TotalPrice
AS
SELECT OrderID,
        (SELECT ISNULL(SUM(Price), 0)
         FROM TripOrders
         WHERE Orders.OrderID = TripOrders.OrderID) AS TripPrice,
        (SELECT ISNULL(SUM(Price), 0)
         FROM AttractionOrders
         WHERE Orders.OrderID = AttractionOrders.OrderID) AS AttractionPrice,
        (SELECT ISNULL(SUM(Amount), 0)
         FROM Payments
         WHERE Orders.OrderID = Payments.OrderID) AS Amount
FROM Orders;
```

OrderID	OrderDate	TripPrice	AttractionPrice	Amount
1	2024-05-01 00:00:00.000	1000.0000	0.0000	300.0000
2	2024-05-02 00:00:00.000	900.0000	25.0000	0.0000
3	2024-05-03 00:00:00.000	600.0000	0.0000	0.0000
4	2024-05-04 00:00:00.000	550.0000	0.0000	0.0000
5	2024-05-05 00:00:00.000	650.0000	60.0000	0.0000
6	2024-05-06 00:00:00.000	400.0000	20.0000	0.0000
7	2024-05-07 00:00:00.000	700.0000	0.0000	0.0000
8	2024-05-08 00:00:00.000	550.0000	0.0000	0.0000
9	2024-05-09 00:00:00.000	550.0000	0.0000	0.0000
10	2024-05-10 00:00:00.000	550.0000	0.0000	0.0000

Nazwa widoku: **SumCustomerOrders**

- Opis: Widok ten wyświetla liczbę zamówień złożonych przez każdego klienta.

```
CREATE VIEW SumCustomerOrders
AS
SELECT Customers.CustomerID, COUNT(OrderID) AS AllOrders
```

```
FROM Customers
JOIN Orders ON Orders.CustomerID = Customers.CustomerID
GROUP BY Customers.CustomerID;
```

CustomerID	AllOrders
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1

Nazwa widoku: **UnpaidOrders**

- Opis: Wyświetla numery wszystkich zamówień (oraz identyfikator klienta, który je złożył), które nie zostały jeszcze w pełni opłacone oraz brakującą sumę do zapłacenia.

```
CREATE VIEW UnpaidOrders
AS
SELECT TotalPrice.OrderID, Orders.OrderDate, Orders.CustomerID,
       TripPrice + AttractionPrice - Amount AS LeftToPay
FROM TotalPrice
JOIN Orders ON Orders.OrderID = TotalPrice.OrderID
WHERE TripPrice + AttractionPrice - Amount > 0
```

OrderID	OrderDate	CustomerID	LeftToPay
1	2024-05-01 00:00:00.000	1	700.0000
2	2024-05-02 00:00:00.000	2	925.0000
3	2024-05-03 00:00:00.000	3	600.0000
4	2024-05-04 00:00:00.000	4	550.0000
5	2024-05-05 00:00:00.000	5	710.0000
6	2024-05-06 00:00:00.000	6	420.0000
7	2024-05-07 00:00:00.000	7	700.0000
8	2024-05-08 00:00:00.000	8	550.0000
9	2024-05-09 00:00:00.000	9	550.0000
10	2024-05-10 00:00:00.000	10	550.0000

Nazwa widoku: **CustomerParticipantList**

- Opis: Wyświetla listę wszystkich dodanych uczestników przez danego klienta w ramach danego zamówienia.

```
CREATE VIEW CustomerParticipantList
AS
SELECT Customers.CustomerID, Participants.ParticipantID, Orders.OrderID, TripOrders.TripOrderID, Orders.OrderDate
FROM Customers
JOIN Orders on Customers.CustomerID = Orders.CustomerID
JOIN TripOrders on Orders.OrderID = TripOrders.OrderID
JOIN TripParticipants on TripOrders.TripOrderID = TripParticipants.TripOrderID
JOIN Participants on TripParticipants.ParticipantID = Participants.ParticipantID;
```

CustomerID	ParticipantID	OrderID	TripOrderID	OrderDate
1	1	1	1	2024-05-01 00:00:00.000
1	2	1	1	2024-05-01 00:00:00.000
2	3	2	2	2024-05-02 00:00:00.000
2	4	2	2	2024-05-02 00:00:00.000
3	5	3	3	2024-05-03 00:00:00.000
4	6	4	4	2024-05-04 00:00:00.000
5	7	5	5	2024-05-05 00:00:00.000
6	8	6	6	2024-05-06 00:00:00.000

CustomerID	ParticipantID	OrderID	TripOrderID	OrderDate
7	9	7	7	2024-05-07 00:00:00.000
8	10	8	8	2024-05-08 00:00:00.000
9	11	9	9	2024-05-09 00:00:00.000
10	12	10	10	2024-05-10 00:00:00.000

Procedury

Nazwa procedury: **AllUnpaidByCustomer**

- Opis: Wyświetla wszystkie numery wycieczek, które nie zostały opłacone przez danego klienta, wraz z brakującą kwotą.

```
CREATE PROCEDURE AllUnpaidByCustomer @CustomerID int
AS
SELECT OrderID, LeftToPay
FROM UnpaidOrders
WHERE CustomerID = @CustomerID
```

Dla *CustomerID* równego 1:

OrderID	LeftToPay
1	700.0000

Nazwa procedury: **ListOrderParticipants**

- Opis: Wylistowuje dane wszystkich uczestników, którzy są zapisani do konkretnego zamówienia.

```
CREATE PROCEDURE ListOrderParticipants @OrderID int
AS
SELECT OrderID, TripOrderID, ParticipantID
FROM CustomerParticipantList
WHERE OrderID = @OrderID;
```

Dla *OrderID* równego 1:

OrderID	ParticipantID
1	1
1	2

Nazwa procedury: **ListTripOrderParticipants**

- Opis: Wylistowuje dane wszystkich uczestników, którzy są zapisani do konkretnego zamówienia wycieczki.

```
CREATE PROCEDURE ListTripOrderParticipants @TripOrderID int
AS
SELECT TripOrderID, ParticipantID
FROM CustomerParticipantList
WHERE TripOrderID = @TripOrderID;
```

Dla *TripOrderID* równego 1:

TripOrderID	ParticipantID
1	1
1	2

Nazwa procedury: **ListAttractionParticipants**

- Opis: Wylistowuje dane wszystkich uczestników, którzy są zapisani do konkretnego zamówienia atrakcji.

```
CREATE PROCEDURE ListAttractionParticipants @AttractionOrderID int
AS
SELECT AttractionOrders.AttractionOrderID, ParticipantID
FROM AttractionOrders
JOIN AttractionParticipants ON AttractionOrders.AttractionOrderID = AttractionParticipants.AttractionOrderID
WHERE AttractionOrders.AttractionOrderID = @AttractionOrderID;
```

Dla *AttractionOrderID* równego 1:

AttractionOrderID	ParticipantID
1	8

Nazwa procedury: **TripsWithXSlotsLeft**

- Opis: Wyświetla wszystkie dostępne wycieczki, w ramach których jest conajmniej podana liczba wolnych miejsc.

```
CREATE PROCEDURE TripsWithXSlotsLeft @SlotsLeft int
AS
SELECT Trips.TripID, Trips.TripName, TripParticipantsCount.TripDate, SlotsLeft
FROM TripParticipantsCount
JOIN Trips ON TripParticipantsCount.TripID = Trips.TripID
WHERE SlotsLeft >= @SlotsLeft;
```

Dla *SlotsLeft* równego 10:

TripID	TripName	TripDate	SlotsLeft
1	Vienna City Tour	2024-06-15	23
2	Prague Historic Walk	2024-07-10	23
3	Paris Museum Excursion	2024-08-05	19
4	Berlin Wall Experience	2024-09-01	34
5	Athens Ancient Sites	2024-10-01	39
6	Dublin Literary Tour	2025-05-14	14
8	Tokyo Technology Tour	2024-12-01	50
9	Krakow Cultural Exploration	2024-04-10	30
10	Barcelona Art Journey	2025-03-14	22
11	Bangkok Temple Tour	2025-02-05	45
12	London Royal Sights	2025-01-01	20

Nazwa procedury: **TripsTo**

- Opis: Wylistowuje numery oraz nazwy wycieczek odbywających się w danym kraju.

```
CREATE PROCEDURE TripsTo @Country varchar(30)
AS
SELECT TripID, TripName, StartDate, EndDate
FROM Trips
WHERE DestinationCountry = @Country;
```

Dla *Country* równego 'Poland':

TripID	TripName	StartDate	EndDate
9	Krakow Cultural Exploration	2024-04-10	2024-04-15

Nazwa procedury: **BuyTrip**

- Opis: Umożliwia zakupienie wybranej wycieczki dla podanej liczby osób. Cena zostaje automatycznie wyliczana.

```
CREATE PROCEDURE BuyTrip @OrderID int, @TripID int, @ParticipantsCount int, @CustomerID int
AS
BEGIN
    IF ((SELECT SlotsLeft FROM TripParticipantsCount
        WHERE TripID = @TripID) < @ParticipantsCount)
        OR
        (DATEADD(day, -7, (SELECT StartDate FROM Trips WHERE TripID = @TripID)) < GETDATE())
        OR
        ((SELECT IsAvailable FROM Trips WHERE TripID = @TripID) <> 1)
        OR
        ((SELECT IsCancelled FROM Orders WHERE OrderID = @OrderID) <> 0)
    BEGIN
        THROW 50001, 'You cannot buy this trip.', 1
    END
    IF (@OrderID NOT IN (SELECT OrderID FROM Orders))
        INSERT INTO Orders(OrderID, OrderDate, CustomerID, IsCancelled)
        VALUES
            ((SELECT MAX(OrderID) + 1 FROM Orders), GETDATE(), @CustomerID, 0)

    INSERT INTO TripOrders(TripOrderID, OrderID, TripID, OrderDate, ParticipantsCount, Price)
    VALUES
        ((SELECT MAX(TripOrderID) + 1 FROM TripOrders), (SELECT MAX(OrderID) FROM Orders), @TripID, GETDATE(),
        @ParticipantsCount, CAST((SELECT Price * @ParticipantsCount FROM Trips WHERE TripID = @TripID) as money))
END;
```

Nazwa procedury: **BuyAttraction**

- Opis: Umożliwia zakupienie wybranej atrakcji dla podanej liczby osób. Cena zostaje automatycznie wyliczana.

```
CREATE PROCEDURE BuyAttraction @OrderID int, @AttractionID int, @ParticipantsCount int
AS
BEGIN
    IF ((SELECT SlotsLeft FROM AttractionParticipantsCount
        WHERE AttractionID = @AttractionID) < @ParticipantsCount)
        OR
        (DATEADD(day, -7, (SELECT StartDate FROM Trips WHERE TripID =
        (SELECT TripID FROM Attractions WHERE AttractionID = @AttractionID))) < GETDATE())
        OR
        ((SELECT IsAvailable FROM Trips WHERE TripID = (SELECT TripID FROM Attractions WHERE AttractionID = @AttractionID)) <> 1)
        OR
        ((SELECT IsCancelled FROM Orders WHERE OrderID = @OrderID) <> 0)
    BEGIN
        THROW 50001, 'You cannot buy this attraction.', 1
    END
    INSERT INTO AttractionOrders(AttractionOrderID, OrderID, AttractionID, OrderDate, ParticipantsCount, Price)
    VALUES
        ((SELECT MAX(AttractionOrderID) + 1 FROM AttractionOrders), @OrderID, @AttractionID, GETDATE(), @ParticipantsCount,
        CAST((SELECT Price * @ParticipantsCount FROM Attractions WHERE AttractionID = @AttractionID) as money))
END;
```

Nazwa procedury: **AssociateParticipantWithTrip**

- Opis: Umożliwia powiązanie uczestnika z konkretną zamówioną wycieczką.

```
CREATE PROCEDURE AssociateParticipantWithTrip @ParticipantID int, @TripOrderID int
AS
BEGIN
    IF (((SELECT COUNT(*) AS cnt FROM CustomerParticipantList WHERE TripOrderID = @TripOrderID)
        >= (SELECT ParticipantsCount FROM TripOrders WHERE TripOrderID = @TripOrderID))
        OR
        (DATEADD(day, -7, (SELECT StartDate FROM Trips WHERE TripID = (SELECT TripID FROM TripOrders WHERE TripOrderID = @TripOrderID)))
        < GETDATE())
        OR
        (@ParticipantID NOT IN (SELECT ParticipantID FROM Participants))
    )
    BEGIN
        THROW 50001, 'You cannot associate the participant with this trip order.', 1
    END
    INSERT INTO TripParticipants(TripOrderID, ParticipantID)
    VALUES
        (@TripOrderID, @ParticipantID)
END;
```

Nazwa procedury: **AssociateParticipantWithAttraction**

- Opis: Umożliwia powiązanie uczestnika z konkretną zamówioną atrakcją.

```
CREATE PROCEDURE AssociateParticipantWithAttraction @ParticipantID int, @AttractionOrderID int
AS
BEGIN
    IF (((SELECT COUNT(*) AS cnt
        FROM AttractionOrders
        JOIN AttractionParticipants ON AttractionOrders.AttractionOrderID = AttractionParticipants.AttractionOrderID
        WHERE AttractionOrders.AttractionOrderID = @AttractionOrderID)
        >= (SELECT ParticipantsCount FROM AttractionOrders WHERE AttractionOrderID = @AttractionOrderID))
        OR
        ((DATEADD(day, -7, (SELECT StartDate FROM Trips WHERE TripID =
        (SELECT TripID FROM AttractionOrders JOIN Attractions ON Attractions.AttractionID = AttractionOrders.AttractionID
        WHERE AttractionOrderID = @AttractionOrderID))) < GETDATE()))
        OR
        (@ParticipantID NOT IN (SELECT ParticipantID FROM Participants))
    )
    BEGIN
        THROW 50001, 'You cannot associate the participant with this attraction order.', 1
    END
    INSERT INTO AttractionParticipants(AttractionOrderID, ParticipantID)
    VALUES
        (@AttractionOrderID, @ParticipantID)
END;
```

Triggery

Nazwa triggera: **ParticipantTripAssociationCheck**

- Opis: Przy dodawaniu uczestnika do atrakcji sprawdza czy dany uczestnik jest już powiązany z wycieczką, do której ta atrakcja jest przypisana.

```
CREATE TRIGGER ParticipantTripAssociationCheck
    ON AttractionParticipants
    AFTER INSERT, UPDATE
AS
BEGIN
    IF NOT EXISTS(SELECT 1
```

```
        FROM inserted
        JOIN AttractionOrders ON inserted.AttractionOrderID = AttractionOrders.AttractionOrderID
        JOIN TripParticipants ON inserted.ParticipantID = TripParticipants.ParticipantID
        JOIN TripOrders ON TripParticipants.TripOrderID = TripOrders.TripOrderID
        WHERE TripOrders.OrderID = AttractionOrders.OrderID
    )
    BEGIN
        THROW 50001, 'Participant is not associated with the trip for this attraction.', 1
    END
END;
```

Nazwa triggera: **AttractionOrderCheck**

- Opis: Przy składaniu zamówienia na daną atrakcję, sprawdza czy została wykupiona wycieczka, która ją oferuje.

```
CREATE TRIGGER AttractionOrderCheck
    ON AttractionOrders
    AFTER INSERT, UPDATE
AS
BEGIN
    IF NOT EXISTS(SELECT TripOrderID
              FROM TripOrders
              WHERE TripOrders.OrderID = (SELECT OrderID
                                          FROM inserted)
                AND TripID = (SELECT TripID
                              FROM Attractions
                              WHERE Attractions.AttractionID = (SELECT AttractionID
                                                                FROM inserted)))
    BEGIN
        THROW 50001, 'You cannot add an attraction if the related trip has not been purchased.', 1
    END
END;
```

4. Inne

Role

Nazwa roli: **User**

- Opis: Rola, która jest nadawana domyślnie każdemu.

```
CREATE ROLE DefaultUser

GRANT SELECT ON dbo.Trips TO DefaultUser
GRANT SELECT ON dbo.Attractions TO DefaultUser
```