Created by:

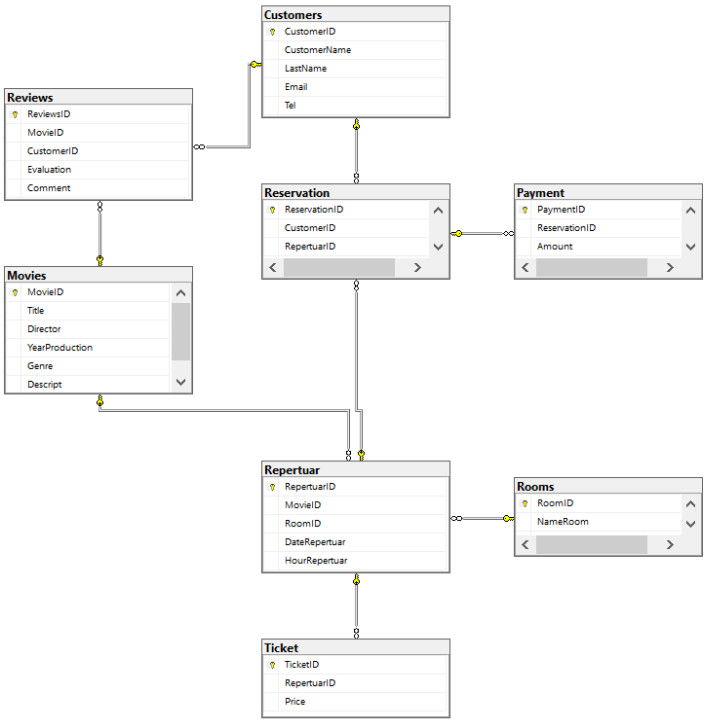
Sebastian Kościółek
Katarzyna Achramowicz

Projekt 1.

Temat:

Baza danych multikina. W bazie powinny być przechowywane informacje o filmach, salach, seansach, sprzedawanych biletach.

Diagram:



Treść zapytań SQL:

```
-- Na początek tworzymy bazę danych oraz nadajemy jej nazwę

CREATE DATABASE MyMultikino;
```

```
-- Używamy stworzonej bazy danych
USE MyMultikino;

-- Tworzymy tabele

-- Tabela Filmów zawierające podstawowe parametry
CREATE TABLE Movies (
      MovieID INT PRIMARY KEY, -- Unikalny identyfikator filmu
      Title VARCHAR(100),
      Director VARCHAR(100),
      YearProduction INT,
      Genre VARCHAR(100),
      Descript TEXT -- Opis filmu
);

-- Tabela z salami kina
CREATE TABLE Rooms (
      RoomID INT PRIMARY KEY, -- Unikalny identyfikator sali kinowej
     NameRoom VARCHAR(100),
     NumberSeats INT
);

-- Tabela repertuaru
```

```
CREATE TABLE Repertuar (

     RepertuarID INT PRIMARY KEY, -- Unikalny identyfikator seansu
w repertuarze

    MovieID INT,

    RoomID INT,

    DateRepertuar DATE,

    HourRepertuar TIME,

    FOREIGN KEY (MovieID) REFERENCES Movies(MovieID), -- *Węzły
klucza obcego*

    FOREIGN KEY (RoomID) REFERENCES Rooms(RoomID) -- *Węzły klucza
obcego*

);



-- Tabela z biletami

CREATE TABLE Ticket (

     TicketID INT PRIMARY KEY, -- Unikalny identyfikator biletu

    RepertuarID INT,

    Price DECIMAL(8,2),

    FOREIGN KEY (RepertuarID) REFERENCES Repertuar(RepertuarID) --
*Węzły klucza obcego*

);



-- Tabela gromadząca klientów

CREATE TABLE Customers (

     CustomerID INT PRIMARY KEY, -- Unikalny identyfikator klienta

    CustomerName VARCHAR(100),
```

```
     LastName VARCHAR(100),
    Email VARCHAR(100),
    Tel VARCHAR(20)
);

-- Tabela z rezerwacjami
CREATE TABLE Reservation (
     ReservationID INT PRIMARY KEY, -- Unikalny identyfikator
rezerwacji
    CustomerID INT,
    RepertuarID INT,
    NumberOfSeats INT,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID), --
*Węzły klucza obcego*
    FOREIGN KEY (RepertuarID) REFERENCES Repertuar(RepertuarID) --
*Węzły klucza obcego*
);

-- Tabela z płatnościami
CREATE TABLE Payment (
     PaymentID INT PRIMARY KEY, -- Unikalny identyfikator płatności
    ReservationID INT,
    Amount DECIMAL(8,2),
    PaymentDate DATE,
    FOREIGN KEY (ReservationID) REFERENCES
Reservation(ReservationID) -- *Węzły klucza obcego*
```

```
);


-- Tabela z recenzjami
CREATE TABLE Reviews (
     ReviewsID INT PRIMARY KEY, -- Unikalny identyfikator recenzji
    MovieID INT,
    CustomerID INT,
    Evaluation INT,
    Comment TEXT,
    FOREIGN KEY (MovieID) REFERENCES Movies(MovieID), -- *Węzły
klucza obcego*
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID) --
*Węzły klucza obcego*
);


-- Tworzymy ograniczenia CHECK
-- Ograniczenie sprawdzające rok produkcji filmu
ALTER TABLE Movies
ADD CONSTRAINT CHK_YearProduction CHECK (YearProduction >= 1970 AND
YearProduction <= YEAR(GETDATE()));
-- Ograniczenie sprawdzające format numeru telefonu klienta
ALTER TABLE Customers
ADD CONSTRAINT CHK_Tel_Format CHECK (Tel LIKE '[0-9]%');
-- Ograniczenie sprawdzające gatunek filmu
ALTER TABLE Movies
```

```
ADD CONSTRAINT CHK_Genre CHECK (Genre IN ('Action', 'Comedy',
'Drama', 'Science Fiction', 'Horror', 'Romance', 'Thriller',
'Adventure', 'Fantasy', 'Mystery', 'Crime', 'Animation', 'Family',
'Documentary', 'Musical', 'Historical', 'War', 'Biography',
'Western', 'Sports'));

-- Ograniczenie sprawdzające format adresu email klienta

ALTER TABLE Customers

ADD CONSTRAINT CHK_Email_Format CHECK (Email LIKE '%@%');



-- Wstawianie danych do tabeli Movies

INSERT INTO Movies (MovieID, Title, Director, YearProduction, Genre,
Descript)

VALUES

    (1, 'Inception', 'Christopher Nolan', 2010, 'Science Fiction',
'A thief enters the dreams of others to steal their secrets.'),

    (2, 'The Shawshank Redemption', 'Frank Darabont', 1994, 'Drama',
'Two imprisoned men bond over several years, finding solace and
eventual redemption through acts of common decency.'),

    (3, 'Pulp Fiction', 'Quentin Tarantino', 1994, 'Crime', 'Various
interconnected stories of criminals in Los Angeles.'),

     (4, 'The Dark Knight', 'Christopher Nolan', 2008, 'Action', 'A
vigilante known as Batman tries to save Gotham City from the
Joker.'),

    (5, 'Forrest Gump', 'Robert Zemeckis', 1994, 'Drama', 'The life
journey of a man with a low IQ but good intentions.'),

    (6, 'The Matrix', 'Lana Wachowski, Lilly Wachowski', 1999,
'Action', 'A computer hacker learns about the true nature of his
reality.'),

    (7, 'Schindler''s List', 'Steven Spielberg', 1993, 'Biography',
'In German-occupied Poland during World War II, Oskar Schindler
gradually becomes concerned for his Jewish workforce.'),
```

```
    (8, 'The Godfather', 'Francis Ford Coppola', 1972, 'Crime', 'The aging patriarch of an organized crime dynasty transfers control of his clandestine empire to his reluctant son.'),

    (9, 'Titanic', 'James Cameron', 1997, 'Romance', 'A seventeen-year-old aristocrat falls in love with a kind but poor artist aboard the luxurious, ill-fated R.M.S. Titanic.'),

    (10, 'Avatar', 'James Cameron', 2009, 'Adventure', 'A paraplegic marine dispatched to the moon Pandora on a unique mission becomes torn between following his orders and protecting the world he feels is his home.');



-- Wstawianie danych do tabeli Rooms

INSERT INTO Rooms (RoomID, NameRoom, NumberSeats)

VALUES

    (1, 'Room 1', 100),

    (2, 'Room 2', 80),

    (3, 'Room 3', 30),

     (4, 'Room 4', 110),

     (5, 'Room 5', 50),

     (6, 'Room 6', 150),

     (7, 'Room 7', 70),

     (8, 'Room 8', 90);



-- Wstawianie danych do tabeli Repertuar

INSERT INTO Repertuar (RepertuarID, MovieID, RoomID, DateRepertuar, HourRepertuar)

VALUES

    (1, 1, 1, '2023-12-31', '18:00:00'),
```

```
    (2, 2, 2, '2023-12-29', '20:00:00'),
    (3, 3, 3, '2024-01-03', '15:30:00'),
     (4, 4, 4, '2024-01-04', '19:30:00'),
     (5, 5, 5, '2024-01-05', '20:30:00'),
     (6, 6, 6, '2024-01-06', '22:30:00'),
     (7, 7, 7, '2024-01-07', '10:00:00');

-- Wstawianie danych do tabeli Ticket
INSERT INTO Ticket (TicketID, RepertuarID, Price)
VALUES
    (1, 1, 15.99),
    (2, 2, 12.50),
    (3, 3, 18.00);

-- Wstawianie danych do tabeli Customers
INSERT INTO Customers (CustomerID, CustomerName, LastName, Email,
Tel)
VALUES
    (1, 'John', 'Doe', 'john.doe@email.com', '123456789'),
    (2, 'Jane', 'Smith', 'jane.smith@email.com', '987654321'),
    (3, 'Alice', 'Johnson', 'alice.johnson@email.com', '456789123'),
     (4, 'Emily', 'Johnson', 'emily.j@example.com', '55581234'),
    (5, 'Michael', 'Williams', 'michael.w@example.com', '55515678'),
    (6, 'Sophia', 'Smith', 'sophia.s@example.com', '55569876'),
    (7, 'Daniel', 'Johnson', 'daniel.j@example.com', '55584321'),
```

```
    (8, 'Emma', 'Brown', 'emma.b@example.com', '55571111'),
    (9, 'Matthew', 'Miller', 'matthew.m@example.com', '55592222'),
    (10, 'Olivia', 'Davis', 'olivia.d@example.com', '55503333');

-- Wstawianie danych do tabeli Reservation
INSERT INTO Reservation (ReservationID, CustomerID, RepertuarID, NumberOfSeats)
VALUES
    (1, 1, 1, 2),
    (2, 2, 2, 3),
    (3, 3, 3, 1),
     (4, 4, 4, 2),
    (5, 5, 5, 3),
    (6, 6, 6, 1),
    (7, 7, 7, 4);

-- Wstawianie danych do tabeli Payment
INSERT INTO Payment (PaymentID, ReservationID, Amount, PaymentDate)
VALUES
    (1, 1, 31.98, '2023-01-01'),
    (2, 2, 37.50, '2023-01-02'),
    (3, 3, 18.00, '2023-01-03'),
     (4, 4, 25.00, '2023-01-10'),
    (5, 5, 30.00, '2023-01-11'),
    (6, 6, 15.00, '2023-01-12'),
```

```
    (7, 7, 34.00, '2023-01-13');


-- Wstawianie danych do tabeli Reviews

INSERT INTO Reviews (ReviewsID, MovieID, CustomerID, Evaluation,
Comment)

VALUES

    (1, 1, 1, 5, 'Amazing movie!'),

    (2, 2, 2, 4, 'A classic!'),

    (3, 3, 3, 3, 'Decent storyline.'),

     (4, 4, 4, 4, 'Great movie!'),

    (5, 5, 5, 5, 'Mind-bending!'),

    (6, 6, 6, 4, 'Classic masterpiece'),

    (7, 7, 7, 3, 'Good, but a bit long'),

    (8, 8, 8, 5, 'Awesome!');


-- Przy wypełnianiu tabel danymi trzeba bardzo uważać na to aby
posiadały one spójność warto dlatego też sprawdzać je poleceniami
SELECT * FROM "nazwa tabeli" jeżeli pojawi się błąd aby upewnić się
czy rekrody już nie istnieją w naszej bazie danych.




-- Tworzymy widoki:


-- Widok 1: Repertuar z informacją o liczbie dostępnych miejsc w
sali
```

```
CREATE VIEW ViewFreeSeats AS

SELECT

    Repertuar.RepertuarID,

    Movies.Title AS MovieTitle,

    Rooms.NameRoom AS RoomName,

    Repertuar.DateRepertuar,

    Repertuar.HourRepertuar,

    Rooms.NumberSeats - COUNT(Reservation.ReservationID) AS
AvailableSeats

FROM

    Repertuar

JOIN Movies ON Repertuar.MovieID = Movies.MovieID

JOIN Rooms ON Repertuar.RoomID = Rooms.RoomID

LEFT JOIN Reservation ON Repertuar.RepertuarID =
Reservation.RepertuarID

GROUP BY

    Repertuar.RepertuarID, Movies.Title, Rooms.NameRoom,
Repertuar.DateRepertuar, Repertuar.HourRepertuar, Rooms.NumberSeats;



-- Przypadek użycia:


 SELECT * FROM ViewFreeSeats

 WHERE DateRepertuar = '2023-12-31' AND HourRepertuar = '18:00:00';
```

	RepertuarID	MovieTitle	RoomName	DateRepertuar	HourRepertuar	AvailableSeats
1	1	Inception	Room 1	2023-12-31	18:00:00.0000000	119

```
-- Widok 2: Ten prezentuje średnią ocenę filmów wraz z ilością recenzji


CREATE VIEW ViewAverageMovie AS

SELECT

    Movies.MovieID,

    Movies.Title,

    AVG(Reviews.Evaluation) AS AverageRating,

    COUNT(Reviews.ReviewsID) AS NumberReviews

FROM

    Movies

LEFT JOIN Reviews ON Movies.MovieID = Reviews.MovieID

GROUP BY

    Movies.MovieID, Movies.Title

HAVING

   COUNT(Reviews.ReviewsID) > 0;


-- Przypadek użycia:


 SELECT * FROM ViewAverageMovie;
```

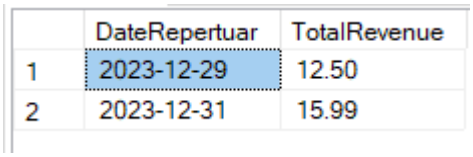
	MovieID	Title	AverageRating	NumberReviews
1	1	Inception	5	1
2	2	The Shawshank Redemption	4	1
3	3	Pulp Fiction	3	1
4	4	The Dark Knight	4	1
5	5	Forrest Gump	5	1
6	6	The Matrix	4	1
7	7	Schindler's List	3	1
8	8	The Godfather	5	1

```
-- Widok 3: Prezentuje listę najczęściej rezerwowanych filmów


CREATE VIEW ViewTopMovies AS

SELECT

    Movies.MovieID,

    Movies.Title,

    COUNT(Reservation.ReservationID) AS ReservationCount

FROM

    Movies

LEFT JOIN Repertuar ON Movies.MovieID = Repertuar.MovieID

LEFT JOIN Reservation ON Repertuar.RepertuarID =
Reservation.RepertuarID

GROUP BY

    Movies.MovieID, Movies.Title;


-- Przypadek użycia:
```

	MovieID	Title	ReservationCount
1	1	Inception	1
2	2	The Shawshank Redemption	1
3	3	Pulp Fiction	1
4	4	The Dark Knight	1
5	5	Forrest Gump	1
6	6	The Matrix	1
7	7	Schindler's List	1
8	8	The Godfather	0
9	9	Titanic	0
10	10	Avatar	0
11	11	New Movie	0

```
 SELECT * FROM ViewTopMovies;


-- Widok 4: Ten widok prezentuje sumę zarobków z biletów w danym
okresie czasu


CREATE VIEW ViewEarnings AS

SELECT

    Repertuar.DateRepertuar,

    SUM(Ticket.Price) AS TotalRevenue

FROM

    Repertuar

JOIN Ticket ON Repertuar.RepertuarID = Ticket.RepertuarID

GROUP BY

    Repertuar.DateRepertuar;


-- Przypadek użycia:
```

```
SELECT * FROM ViewEarnings
WHERE DateRepertuar BETWEEN '2023-01-01' AND '2023-12-31';
```



	DateRepertuar	TotalRevenue
1	2023-12-29	12.50
2	2023-12-31	15.99

Projekt 2.

```
-- Funkcje
-- Tworzymy funkcję która zwraca listę filmów danego reżysera

CREATE FUNCTION GetMoviesByDirector(@DirectorName VARCHAR(100))
RETURNS TABLE
AS
RETURN (
    SELECT * FROM Movies WHERE Director = @DirectorName
);

-- Przykład użycia funkcji GetMoviesByDirector
SELECT * FROM GetMoviesByDirector('James Cameron');
```

```
-- Tworzymy funkcję która zwraca listę klientów którzy dokonali płatności powyżej x kwoty


CREATE FUNCTION GetHighSpendingCustomers(@MinAmount DECIMAL(8,2))

RETURNS TABLE

AS

RETURN (

    SELECT DISTINCT C.* FROM Customers C

    JOIN Reservation R ON C.CustomerID = R.CustomerID

    JOIN Payment P ON R.ReservationID = P.ReservationID

    WHERE P.Amount >= @MinAmount

);


-- Przykład użycia funkcji GetHighSpendingCustomers

DECLARE @MinAmount DECIMAL(8,2);

SET @MinAmount = 5.00;  -- Ustaw dowolną minimalną kwotę płatności


SELECT * FROM GetHighSpendingCustomers(@MinAmount);




-- Procedury:

-- Tworzymy procedurę która aktualizuje opis filmu i zwiększa ilość dostępnych miejsc w sali o określoną wartość


-- W tym miejscu popełniłem błąd i musiałem usunąć procedure UpdateMovie. Użyłem do tego polecenia:
```

```
IF OBJECT_ID('UpdateMovie', 'P') IS NOT NULL
    DROP PROCEDURE UpdateMovie;
GO

-- Następnie utworzyłem ponownie procedurę:

CREATE PROCEDURE UpdateMovie(
    @MovieID INT,
    @NewDescription VARCHAR(MAX),
    @IncreaseSeatsBy INT
)
AS
BEGIN
    UPDATE Movies SET Descript = @NewDescription WHERE MovieID =
@MovieID;

    UPDATE Rooms
    SET NumberSeats = NumberSeats + @IncreaseSeatsBy
    WHERE RoomID IN (SELECT RoomID FROM Repertuar WHERE MovieID =
@MovieID);
END;

-- Przykład użycia procedury UpdateMovie
```

```
DECLARE @MovieIDToUpdate INT;

DECLARE @NewDescriptionToUpdate VARCHAR(MAX);

DECLARE @IncreaseSeatsBy INT;


SET @MovieIDToUpdate = 1;

SET @NewDescriptionToUpdate = 'New movie description';

SET @IncreaseSeatsBy = 20;


EXEC UpdateMovie

    @MovieID = @MovieIDToUpdate,

    @NewDescription = @NewDescriptionToUpdate,

    @IncreaseSeatsBy = @IncreaseSeatsBy;

-- Wyświetl zaktualizowane informacje

SELECT * FROM Movies WHERE MovieID = @MovieIDToUpdate;

SELECT * FROM Rooms WHERE RoomID IN (SELECT RoomID FROM Repertuar
WHERE MovieID = @MovieIDToUpdate);


-- Procedura dodaje nowego klienta i rezerwuje miejsca na wybrany
seans


CREATE PROCEDURE AddCustomerAndReservation(

    @CustomerName VARCHAR(255),

    @LastName VARCHAR(255),

    @Email VARCHAR(255),
```

```
    @Tel VARCHAR(20),
    @RepertuarID INT,
    @NumberOfSeats INT
)
AS
BEGIN
    INSERT INTO Customers (CustomerName, LastName, Email, Tel)
    VALUES (@CustomerName, @LastName, @Email, @Tel);
    DECLARE @CustomerID INT = SCOPE_IDENTITY();
    IF @CustomerID IS NOT NULL
    BEGIN
        INSERT INTO Reservation (CustomerID, RepertuarID,
NumberOfSeats)
        VALUES (@CustomerID, @RepertuarID, @NumberOfSeats);
    END
    ELSE
    BEGIN
        PRINT 'Błąd dodawania klienta: Nie udało się pobrać
identyfikatora klienta.';
        RETURN;
    END
END;


-- Wyzwalacze
-- Wyzwalacz do śledzenia zmian w tabeli Movies
```

```
CREATE TRIGGER MoviesTrigger

ON Movies

AFTER INSERT, UPDATE, DELETE

AS

BEGIN

    PRINT 'Zmiany w tabeli Movies zostały zarejestrowane.';

END;


-- Dodawanie nowego filmu - przypadek użycia

INSERT INTO Movies (MovieID, Title, Director, YearProduction, Genre,
Descript)

VALUES (11, 'New Movie', 'New Director', 2023, 'Action',
'Description of the new movie.');




-- Wyzwalacz do śledzenia dodawania recenzji


CREATE TRIGGER ReviewsTrigger

ON Reviews

AFTER INSERT

AS

BEGIN


    UPDATE ViewAverageMovie
```

```
    SET AverageRating = (
        SELECT AVG(Evaluation)
        FROM Reviews
        WHERE Reviews.MovieID = ViewAverageMovie.MovieID
        GROUP BY MovieID
    ),
    NumberReviews = (
        SELECT COUNT(ReviewsID)
        FROM Reviews
        WHERE Reviews.MovieID = ViewAverageMovie.MovieID
        GROUP BY MovieID
    )
    WHERE EXISTS (
        SELECT 1
        FROM inserted
        WHERE inserted.MovieID = ViewAverageMovie.MovieID
    );
END;

-- Dodawanie nowej recenzji - przypadek użycia
INSERT INTO Reviews (ReviewsID, MovieID, CustomerID, Evaluation,
Comment)
VALUES (9, 11, 1, 4, 'Enjoyed the new movie!');
```