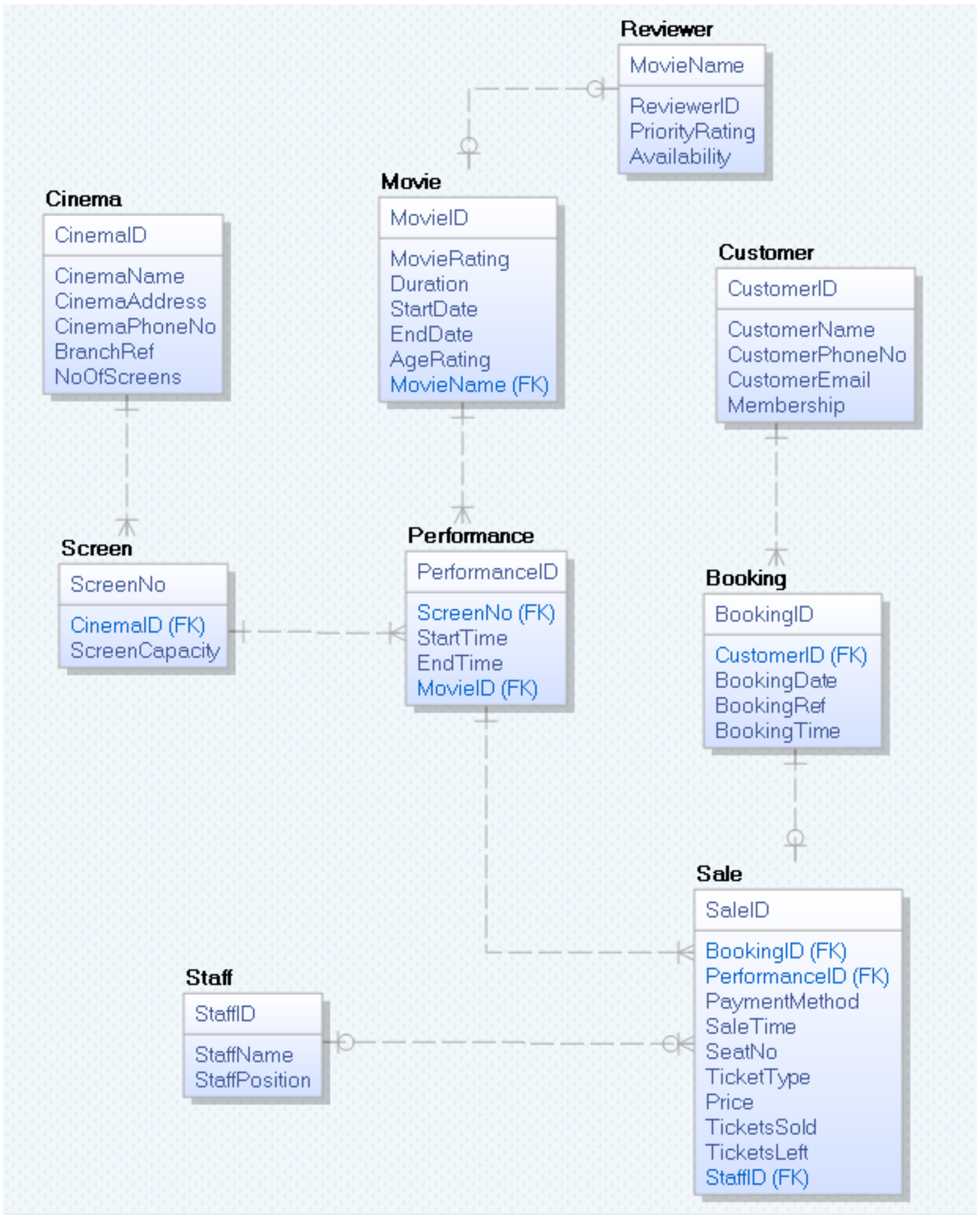
ERD



Reviewer
MovieName
ReviewerID
PriorityRating
Availability
Movie
MovieID
MovieRating
Duration
StartDate
EndDate
AgeRating
MovieName (FK)
Cinema
CinemaID
CinemaName
CinemaAddress
CinemaPhoneNo
BranchRef
NoOfScreens
Customer
CustomerID
CustomerName
CustomerPhoneNo
CustomerEmail
Membership
Screen
ScreenNo
CinemaID (FK)
ScreenCapacity
Performance
PerformanceID
ScreenNo (FK)
StartTime
EndTime
MovieID (FK)
Booking
BookingID
CustomerID (FK)
BookingDate
BookingRef
BookingTime
Sale
SaleID
BookingID (FK)
PerformanceID (FK)
PaymentMethod
SaleTime
SeatNo
TicketType
Price
TicketsSold
TicketsLeft
StaffID (FK)
Staff
StaffID
StaffName
StaffPosition

SQL

```
-- Cinema System SQL to create tables

/* Drop Table statements are included for each table to ensure that
when you first create the tables none with the same name exist. */

DROP TABLE Cinema CASCADE CONSTRAINTS PURGE;
DROP TABLE Reviewer CASCADE CONSTRAINTS PURGE;
DROP TABLE Staff CASCADE CONSTRAINTS PURGE;
DROP TABLE Customer CASCADE CONSTRAINTS PURGE;
DROP TABLE Movie CASCADE CONSTRAINTS PURGE;
DROP TABLE Screen CASCADE CONSTRAINTS PURGE;
DROP TABLE Performances CASCADE CONSTRAINTS PURGE;
DROP TABLE Booking CASCADE CONSTRAINTS PURGE;
DROP TABLE Sale CASCADE CONSTRAINTS PURGE;

/* Create table statements
Tables are created in 3 layers following a given order:
1st: Those with no foreign keys are created first
2nd: Tables depending only on these tables i.e. have foreign key relationships to 1st layer
3rd: Tables depending on 2nd layer or combination of 1st and 2nd layer
*/

-- Create table Cinema- holds data on the different branches for the company
CREATE TABLE Cinema
(
   CinemaID                NUMBER(6) NOT NULL,
   CinemaName              VARCHAR2(30) NULL,
   CinemaAddress           VARCHAR2(30) NULL,
   CinemaPhoneNo           VARCHAR2(10) NULL,
   BranchRef               Number(2) NULL,
   NoOfScreens             Number(2) NULL,
CONSTRAINT Cinema_PK PRIMARY KEY (CinemaID)
);

CREATE TABLE Reviewer
(
   MovieName               VARCHAR2(50) NOT NULL,
   ReviewerID              NUMBER(6) NOT NULL,
   PriorityRating          VARCHAR2(2) NULL,
   Starts                  DATE NULL,
CONSTRAINT Reviewer_PK PRIMARY KEY (MovieName)
);

CREATE TABLE Staff
(
   StaffID         NUMBER(6) NOT NULL,
   StaffName               VARCHAR2(30) NULL,
   StaffPosition           VARCHAR2(10) NULL,
CONSTRAINT Staff_PK PRIMARY KEY (StaffID)
);
```

```
-- Create table Customer - holds data on the cinemas customers
CREATE TABLE Customer
(
   CustomerID              NUMBER(6) NOT NULL,
   CustomerName            VARCHAR2(30) NULL,
   CustomerPhoneNo         VARCHAR2(10) NULL,
   CustomerEmail           VARCHAR2(30) NULL,
   Membership              VARCHAR2(1) NULL,
CONSTRAINT Customer_PK PRIMARY KEY (CustomerID)
);

-- Create table Movies - holds details of all movies that the cinema is showing
CREATE TABLE Movie
(
   MovieID                 NUMBER(6) NOT NULL,
   MovieRating             VARCHAR2(10) NULL,
   RunTime                 NUMBER(3) NULL,
   StartDate               DATE NULL,
   EndDate                 DATE NULL,
   AgeRating               VARCHAR2(4) NULL,
   MovieName               VARCHAR2(50) NULL,
CONSTRAINT Movie_PK PRIMARY KEY (MovieID),
CONSTRAINT Movie_Reviewer_FK FOREIGN KEY (MovieName) REFERENCES
Reviewer (MovieName)
);

-- Create table Screen - holds details on each screen in each cinema
CREATE TABLE Screen
(
   ScreenNo                NUMBER(2) NOT NULL,
   ScreenCapacity          NUMBER(3) NOT NULL,
   CinemaID                NUMBER(6) NOT NULL,
CONSTRAINT Screen_PK PRIMARY KEY (ScreenNo),
CONSTRAINT Cinema_Screen_FK FOREIGN KEY (CinemaID) REFERENCES Cinema
(CinemaID)
);

-- Create table Performance - holds details of all performances each week in the cinema
CREATE TABLE Performances
(
   PerformanceID           NUMBER(6) NOT NULL,
   StartTime               VARCHAR2(10) NULL,
   EndTime                 VARCHAR2(10) NULL,
   ScreenNo                NUMBER(2) NULL,
   MovieID                 NUMBER(6) NOT NULL,
CONSTRAINT Performances_PK PRIMARY KEY (PerformanceID),
CONSTRAINT Screen_Performances_FK FOREIGN KEY (ScreenNo) REFERENCES
Screen (ScreenNo),
CONSTRAINT Movie_Performances_FK FOREIGN KEY (MovieID) REFERENCES Movie
(MovieID)
);
```

```
-- Create table Bookings - holds details of all bookings made over the internet/phone
CREATE TABLE Booking
(
   BookingID                NUMBER(6) NOT NULL,
   BookingDate              DATE NULL,
   BookingReference         NUMBER(6) NOT NULL,
   BookingTime              VARCHAR2(10) NULL,
   CustomerID               NUMBER(6) NULL,
CONSTRAINT Booking_PK PRIMARY KEY (BookingID),
CONSTRAINT Customer_Booking_FK FOREIGN KEY (CustomerID) REFERENCES
Customer (CustomerID)
);

CREATE TABLE Sale
(
        SaleID              NUMBER(6) NOT NULL,
        SaleTime            VARCHAR2(10) NULL,
        PaymentMethod       VARCHAR2(4) NULL,
        SeatNo              VARCHAR2(2) NULL,
        TicketType          VARCHAR2(10) NULL,
        Price               NUMBER(2,2) NULL,
        BookingID           NUMBER(6) NOT NULL,
        PerformanceID        NUMBER(6) NOT NULL,
CONSTRAINT Sale_PK PRIMARY KEY (SaleID),
CONSTRAINT Booking_Sale_FK FOREIGN KEY (BookingID) REFERENCES Booking
(BookingID),
CONSTRAINT Performances_Sale_FK FOREIGN KEY (PerformanceID) REFERENCES
Performances (PerformanceID)
);


-- Insert statements to populate the tables
--Cinema Table
INSERT INTO Cinema (CinemaID, CinemaName, CinemaAddress, CinemaPhoneNo,
BranchRef, NoOfScreens)
VALUES (1000, 'International Cinema', 'WestEnd complex D19', 83223332, 4, 10);

--Reviewer Table
INSERT INTO Reviewer (MovieName, ReviewerID, PriorityRating, Starts)
VALUES ('Pan', 14378, 6, '2015-09-20');
INSERT INTO Reviewer (MovieName, ReviewerID, PriorityRating, Starts)
VALUES ('The Martian', 15378, 9, '2015-09-11');
INSERT INTO Reviewer (MovieName, ReviewerID, PriorityRating, Starts)
VALUES ('Sicario', 178, 8, '2015-09-19');
INSERT INTO Reviewer (MovieName, ReviewerID, PriorityRating, Starts)
VALUES ('Inside Out', 12658, 10, '2015-09-24');
INSERT INTO Reviewer (MovieName, ReviewerID, PriorityRating, Starts)
VALUES ('The Last Witch Hunter', 58, 6, '2015-09-18');

--Staff Table
INSERT INTO Staff (StaffID , StaffName, StaffPosition)
VALUES (2000, 'Ben' , 'Cinema Operative');
```

```sql
INSERT INTO Staff (StaffID , StaffName, StaffPosition)
VALUES (2001, 'Orla' , 'Cinema Operative');
INSERT INTO Staff (StaffID , StaffName, StaffPosition)
VALUES (2002, 'Saira' , 'Cinema Operative');
INSERT INTO Staff (StaffID , StaffName, StaffPosition)
VALUES (2003, 'Patricia' , 'cinema Manager');
INSERT INTO Staff (StaffID , StaffName, StaffPosition)
VALUES (2004, 'Bob' , 'Supervisor');
INSERT INTO Staff (StaffID , StaffName, StaffPosition)
VALUES (2005, 'Deirdre' , 'Cinema Operative');

--Customer Table
INSERT INTO Customer (CustomerID, CustomerName, CustomerPhoneNo,
CustomerEmail, Membership)
VALUES (900, 'Andrew Pembroke', '0863734009', 'AndrewPembroke@gmail.com', '1');
INSERT INTO Customer (CustomerID, CustomerName, CustomerPhoneNo,
CustomerEmail, Membership)
VALUES (901, 'Helena Smith', '0873754398', 'Helena.Smith@gmail.com', '0');
INSERT INTO Customer (CustomerID, CustomerName, CustomerPhoneNo,
CustomerEmail, Membership)
VALUES (902, 'John Finn', '0851454076', 'JohnMFinn@gmail.com', '0');
INSERT INTO Customer (CustomerID, CustomerName, CustomerPhoneNo,
CustomerEmail, Membership)
VALUES (903, 'Aoife Green', '0863140497', 'AoifeGreen123@gmail.com', '0');
INSERT INTO Customer (CustomerID, CustomerName, CustomerPhoneNo,
CustomerEmail, Membership)
VALUES (904, 'Gavin Morris', '0879874501', 'GavinMorris@gmail.com', '0');

--Movie Table
INSERT INTO Movie (MovieID, MovieRating, RunTime, StartDate, EndDate, MovieName)
VALUES (601, 'PG', 111, '2015-09-20', '2015-10-20', 'Pan');
INSERT INTO Movie (MovieID, MovieRating, Duration, StartDate, EndDate, MovieName)
VALUES (602, 'PG13', 141, '2015-09-11', '2015-10-11', 'The Martian');
INSERT INTO Movie (MovieID, MovieRating, Duration, StartDate, EndDate, MovieName)
VALUES (603, '15', 121, '2015-09-19', '2015-10-19', 'Sicario');
INSERT INTO Movie (MovieID, MovieRating, Duration, StartDate, EndDate, MovieName)
VALUES (604, 'G', 94, '2015-09-24', '2015-10-24', 'Inside Out');
INSERT INTO Movie (MovieID, MovieRating, Duration, StartDate, EndDate, MovieName)
VALUES (605, '12A', 106, '2015-09-18', '2015-10-18', 'The Last Witch Hunter');

--Screen table
INSERT INTO Screen (ScreenNo, CinemaId, ScreenCapacity)
VALUES (1, 1000, 250);
INSERT INTO Screen (ScreenNo, CinemaId, ScreenCapacity)
VALUES (2, 1000, 150);
INSERT INTO Screen (ScreenNo, CinemaId, ScreenCapacity)
VALUES (3, 1000, 75);
INSERT INTO Screen (ScreenNo, CinemaId, ScreenCapacity)
VALUES (4, 1000, 200);
INSERT INTO Screen (ScreenNo, CinemaId, ScreenCapacity)
VALUES (5, 1000, 60);
```

```
--Performance Table
INSERT INTO Performances (PerformanceID, StartTime, EndTime, ScreenNo, MovieID)
VALUES (6723, '16:00:00', '17:51:00', 1, 601);
INSERT INTO Performances (PerformanceID, StartTime, EndTime, ScreenNo, MovieID)
VALUES (6724, '18:00:00', '20:22:00', 2, 602);
INSERT INTO Performances (PerformanceID, StartTime, EndTime, ScreenNo, MovieID)
VALUES (6725, '13:00:00', '15:01:00', 3, 603);
INSERT INTO Performances (PerformanceID, StartTime, EndTime, ScreenNo, MovieID)
VALUES (6726, '15:00:00', '16:42:00', 4, 604);
INSERT INTO Performances (PerformanceID, StartTime, EndTime, ScreenNo, MovieID)
VALUES (6727, '19:00:00', '13:46:00', 5, 605);

--Booking Table
INSERT INTO Booking (BookingID, BookingDate, BookingReference, BookingTime,
CustomerID)
VALUES (1000, '2015-09-22', 22345, '17:49:00', 901);
INSERT INTO Booking (BookingID, BookingDate, BookingReference, BookingTime,
CustomerID)
VALUES (1001, '2015-09-30', 22346, '12:00:00', 903);
INSERT INTO Booking (BookingID, BookingDate, BookingReference, BookingTime,
CustomerID)
VALUES (1002, '2015-10-07', 22347, '13:10:00', 900);
INSERT INTO Booking (BookingID, BookingDate, BookingReference, BookingTime,
CustomerID)
VALUES (1003, '2015-10-01', 22348, '13:11:00', 902);
INSERT INTO Booking (BookingID, BookingDate, BookingReference, BookingTime,
CustomerID)
VALUES (1004, '2015-10-11', 22349, '18:30:00', 904);

--Sales table
INSERT INTO Sale (SaleID, SaleTime, PaymentMethod, SeatNo, TicketType, Price,
BookingID, PerformanceID)
VALUES (1000, '13:24:00', 'cash', 'C7', 'Standard', '7.50', NULL, 6723);
INSERT INTO Sale (SaleID, SaleTime, PaymentMethod, SeatNo, TicketType, Price,
BookingID, PerformanceID)
VALUES (1001, '17:49:00', 'card', 'B6', 'Standard', '7.50', 1000, 6724);
INSERT INTO Sale (SaleID, SaleTime, PaymentMethod, SeatNo, TicketType, Price,
BookingID, PerformanceID)
VALUES (1002, '12:00:00', 'card', 'A10', 'Premier', '19.99', 1001, 6725);
INSERT INTO Sale (SaleID, SaleTime, PaymentMethod, SeatNo, TicketType, Price,
BookingID, PerformanceID)
VALUES (1004, '13:10:00', 'card', 'B9', 'Standard', '7.50', 1002, 6726);
INSERT INTO Sale (SaleID, SaleTime, PaymentMethod, SeatNo, TicketType, Price,
BookingID, PerformanceID)
VALUES (1005, '13:11:00', 'card', 'B10', 'Standard', '7.50', 1003, 6726);
INSERT INTO Sale (SaleID, SaleTime, PaymentMethod, SeatNo, TicketType, Price,
BookingID, PerformanceID)
VALUES (1005, '18:30:00', 'card', 'D6', 'Standard', '7.50', 1004, 6727);

-- Commit included to persist the data
commit;
```

<u>Grants</u>

```
GRANT DELETE, INSERT ON Movie TO ofahy;
GRANT DELETE, UPDATE, INSERT, JOIN ON Performance TO ofahy;
GRANT INSERT ON Reviewer TO sbarlas;
GRANT DELETE, UPDATE ON Performance TO bdann;
```

<u>Functional Specification</u>

<u>Reviewer - Saira</u>

The reviewer will be granted access to the Reviewer Table. Here they will be granted access to create inserts. They do not need to be able to edit the table in any other way, however they could also be given access to DELETE and UPDATE incase there was a mistake made they would be able to rectify it themselves. The main duties that the reviewers need to be able to do is to let management know what films are available in advance and from what date they are available from. As a whole the pool of reviewers decide what new films should be offered by distributors, give priority ratings, letting management know which movies they should show first, provide more screenings off ect. These are the group of people who try their best to cater to exactly what the public will want and inform the management of such.

<u>Management - Orla</u>

The management as a whole wish to be able to update their system every Thursday. For this they will need to be granted access to the Movie table. This means needing access to DELETE and INSERT, the user will be able to delete any movies that are no longer doing well, and enter in the new movies for these times that have been given to them by the reviewers. To be able to do this they must also have some form of guideline to go by to check if a movie is doing well or not. This would perhaps mean using the DELETE, but with a selective statement, like checking ticket sale percentages are above a certain amount. An example would be; Delete any movie from the system if their ticket sales have been less than 60%. If it is above this amount they have the option of keeping it running for an extra week. The manager also made it very clear that he would like to be able to see the percentage sales for movies, as well as the seats sold for each screening. This would also given him the need to have access to the performance table. Granting him access to SELECT and JOIN.

<u>Staff - Ben</u>

Staff will need access to the Performance table, and they will need to be able to use the UPDATE and possibly the DELETE method. This is because as stated in the case study by the cinema manager, counter staff can either change tickets or refund them, and sell tickets. This would mean every time a ticket is sold the system would update the seats available to be one less, but if refunding them again they would be able to update the seats available to be one more. They may need to change the performance or movie that the customer has chosen to see, which would involve updating the performance id, or perhaps even just their seat number if they are unhappy with where they have been placed in the screen.