DBMS

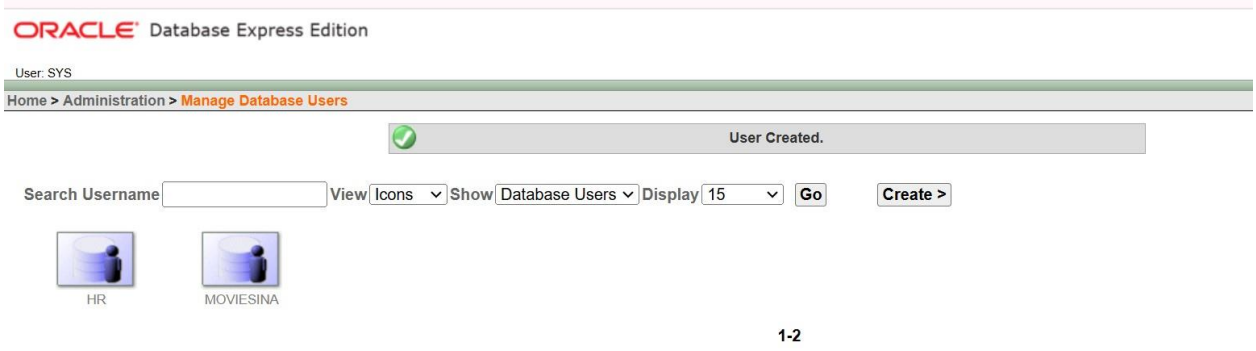
MOVIE RENTAL SYSTEM

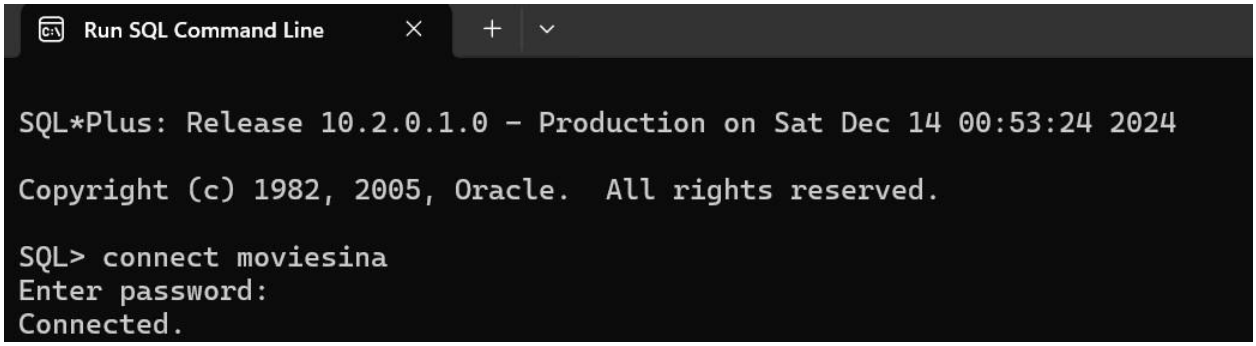
Inayat Arshad
BS-22-IB-105732
SUBMITTED TO: DR IRFAN HAMEED

CREATING DATABASE:

LOGIN:

Login to oracle server and create a user in sys named moviesina set password and go to run sql command line connect **moviesina** and now we create db





CREATE TABLES:

//command

CREATE TABLE Customer_T (

customer_id INT PRIMARY KEY,

customer_name VARCHAR(100),

phone VARCHAR(15),

membership VARCHAR(20)

);

CREATE TABLE Movie_T (

movie_id INT PRIMARY KEY,

title VARCHAR(100), genre

VARCHAR(50),

```
    language VARCHAR(30),
    duration INT,
rating VARCHAR(5)
);
CREATE TABLE Rental_T (
rental_id INT PRIMARY KEY,
rental_date DATE,
customer_id INT,     movie_id
INT,     late_fee DECIMAL(5,2),
    FOREIGN KEY (customer_id) REFERENCES Customer_T(customer_id),
    FOREIGN KEY (movie_id) REFERENCES Movie_T(movie_id)
);
CREATE   TABLE   Payment_T   (
payment_id   INT   PRIMARY   KEY,
amount                  DECIMAL(7,2),
payment_method VARCHAR(20),
    rental_id INT,
    FOREIGN KEY (rental_id) REFERENCES Rental_T(rental_id)
);
CREATE TABLE Staff_T (
staff_id INT PRIMARY KEY,
name VARCHAR(100),
position VARCHAR(50),
email VARCHAR(100),     phone
VARCHAR(15)
);
```

```
SQL> CREATE TABLE Customer_T (
  2      customer_id INT PRIMARY KEY,
  3      customer_name VARCHAR(100),
  4      phone VARCHAR(15),
  5      membership VARCHAR(20)
  6  );

Table created.

SQL>
SQL> CREATE TABLE Movie_T (
  2      movie_id INT PRIMARY KEY,
  3      title VARCHAR(100),
  4      genre VARCHAR(50),
  5      language VARCHAR(30),
  6      duration INT,
  7      rating VARCHAR(5)
  8  );

Table created.
```

Run SQL Command Line

```
SQL>
SQL> CREATE TABLE Rental_T (
  2      rental_id INT PRIMARY KEY,
  3      rental_date DATE,
  4      customer_id INT,
  5      movie_id INT,
  6      late_fee DECIMAL(5,2),
  7      FOREIGN KEY (customer_id) REFERENCES Customer_T(customer_id),
  8      FOREIGN KEY (movie_id) REFERENCES Movie_T(movie_id)
  9  );

Table created.

SQL>
SQL> CREATE TABLE Payment_T (
  2      payment_id INT PRIMARY KEY,
  3      amount DECIMAL(7,2),
  4      payment_method VARCHAR(20),
  5      rental_id INT,
  6      FOREIGN KEY (rental_id) REFERENCES Rental_T(rental_id)
  7  );

Table created.

SQL>
SQL> CREATE TABLE Staff_T (
  2      staff_id INT PRIMARY KEY,
  3      name VARCHAR(100),
  4      position VARCHAR(50),
  5      email VARCHAR(100),
  6      phone VARCHAR(15)
  7  );
```

//INSERT 20 RECORDS

//command

INSERT INTO CUSTOMER_T

INSERT INTO Customer_T (customer_id, customer_name, phone) VALUES (1, 'John Smith', '123-4567890');

INSERT INTO Customer_T (customer_id, customer_name, phone) VALUES (2, 'Alice Johnson', '321654-9870');

INSERT INTO Customer_T (customer_id, customer_name, phone) VALUES (3, 'Bob Williams', '555123-4567');

INSERT INTO Customer_T (customer_id, customer_name, phone) VALUES (4, 'Emma Brown', '123234-5678');

INSERT INTO Customer_T (customer_id, customer_name, phone) VALUES (5, 'Michael Davis', '987654-3210');

INSERT INTO Customer_T (customer_id, customer_name, phone) VALUES (6, 'Sophia Martinez', '555678-1234');

INSERT INTO Customer_T (customer_id, customer_name, phone) VALUES (7, 'James Wilson', '444555-6666');

INSERT INTO Customer_T (customer_id, customer_name, phone) VALUES (8, 'Olivia Moore', '222333-4444');

INSERT INTO Customer_T (customer_id, customer_name, phone) VALUES (9, 'William Taylor', '111222-3333');

INSERT INTO Customer_T (customer_id, customer_name, phone) VALUES (10, 'Isabella Anderson', '666-777-8888');

INSERT INTO Customer_T (customer_id, customer_name, phone) VALUES (11, 'Liam Thomas', '777888-9999');

INSERT INTO Customer_T (customer_id, customer_name, phone) VALUES (12, 'Mia Jackson', '888999-0000');

INSERT INTO Customer_T (customer_id, customer_name, phone) VALUES (13, 'Noah White', '999000-1111');

INSERT INTO Customer_T (customer_id, customer_name, phone) VALUES (14, 'Ava Harris', '222-4446666');

INSERT INTO Customer_T (customer_id, customer_name, phone) VALUES (15, 'Ethan Martin', '333555-7777');

INSERT INTO Customer_T (customer_id, customer_name, phone) VALUES (16, 'Charlotte Thompson', '444-666-8888');

INSERT INTO Customer_T (customer_id, customer_name, phone) VALUES (17, 'Lucas Garcia', '555777-9999');

INSERT INTO Customer_T (customer_id, customer_name, phone) VALUES (18, 'Amelia Martinez', '666-888-0000');

INSERT INTO Customer_T (customer_id, customer_name, phone) VALUES (19, 'Mason Robinson', '777-999-1111');

INSERT INTO Customer_T (customer_id, customer_name, phone) VALUES (20, 'Harper Clark', '888000-2222');

INSERT INTO Rental_T (rental_id, rental_date, customer_id, movie_id, late_fee) VALUES (1, '2024-0901', 1, 1, 2.50);

VERIFY ENTRIES ADDED:

```
--------------------------------------------------------------------------------
PHONE           MEMBERSHIP
--------------- --------------------
         1
John Smith
123-456-7890

         2
Alice Johnson
321-654-9870

CUSTOMER_ID
-----------
CUSTOMER_NAME
--------------------------------------------------------------------------------
PHONE           MEMBERSHIP
--------------- --------------------

         3
Bob Williams
555-123-4567

         4
Emma Brown

CUSTOMER_ID
-----------
CUSTOMER_NAME
--------------------------------------------------------------------------------
PHONE           MEMBERSHIP
--------------- --------------------
123-234-5678
```

//INSERT INTO MOVIE_T

INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (1, 'Inception', 'SciFi', 'English', 148, 'PG-13');

INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (2, 'Parasite', 'Thriller', 'Korean', 132, 'R');

INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (3, 'The Matrix', 'Action', 'English', 136, 'R');

INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (4, 'The Shawshank Redemption', 'Drama', 'English', 142, 'R');

INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (5, 'The Godfather', 'Crime', 'English', 175, 'R');

INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (6, 'Schindler\'s List', 'Biography', 'English', 195, 'R');

INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (7, 'Pulp Fiction', 'Crime', 'English', 154, 'R');

INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (8, 'The Dark Knight', 'Action', 'English', 152, 'PG-13');

INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (9, 'Forrest Gump', 'Drama', 'English', 142, 'PG-13');

INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (10, 'Fight Club', 'Drama', 'English', 139, 'R');

INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (11, 'The Lord of the Rings: The Return of the King', 'Fantasy', 'English', 201, 'PG-13');

INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (12, 'Interstellar', 'Sci-Fi', 'English', 169, 'PG-13');

INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (13, 'The Silence of the Lambs', 'Thriller', 'English', 118, 'R');

INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (14, 'Gladiator', 'Action', 'English', 155, 'R');

INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (15, 'The Avengers', 'Action', 'English', 143, 'PG-13');

INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (16, 'Titanic', 'Romance', 'English', 195, 'PG-13');

INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (17, 'Avatar', 'SciFi', 'English', 162, 'PG-13');

INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (18, 'Spirited Away', 'Animation', 'Japanese', 125, 'PG');

INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (19, 'The Lion King', 'Animation', 'English', 88, 'G');

INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (20, 'Your Name', 'Animation', 'Japanese', 106, 'PG');

```
Run SQL Command Line
1 row created.

SQL> INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (1, 'Inception', 'Sci-Fi', 'English', 148, 'PG-1
3');

1 row created.

SQL> INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (2, 'Parasite', 'Thriller', 'Korean', 132, 'R');

1 row created.

SQL> INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (3, 'The Matrix', 'Action', 'English', 136, 'R')
;

1 row created.

SQL> INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (4, 'The Shawshank Redemption', 'Drama', 'Englis
h', 142, 'R');

1 row created.

SQL> INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (5, 'The Godfather', 'Crime', 'English', 175, 'R
');

1 row created.

SQL> INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (6, 'Schindler\'s List', 'Biography', 'English',
 195, 'R');
ERROR:
ORA-01756: quoted string not properly terminated


SQL> INSERT INTO Movie_T (movie_id, title, genre, language, duration, rating) VALUES (7, 'Pulp Fiction', 'Crime', 'English', 154, 'R'
```

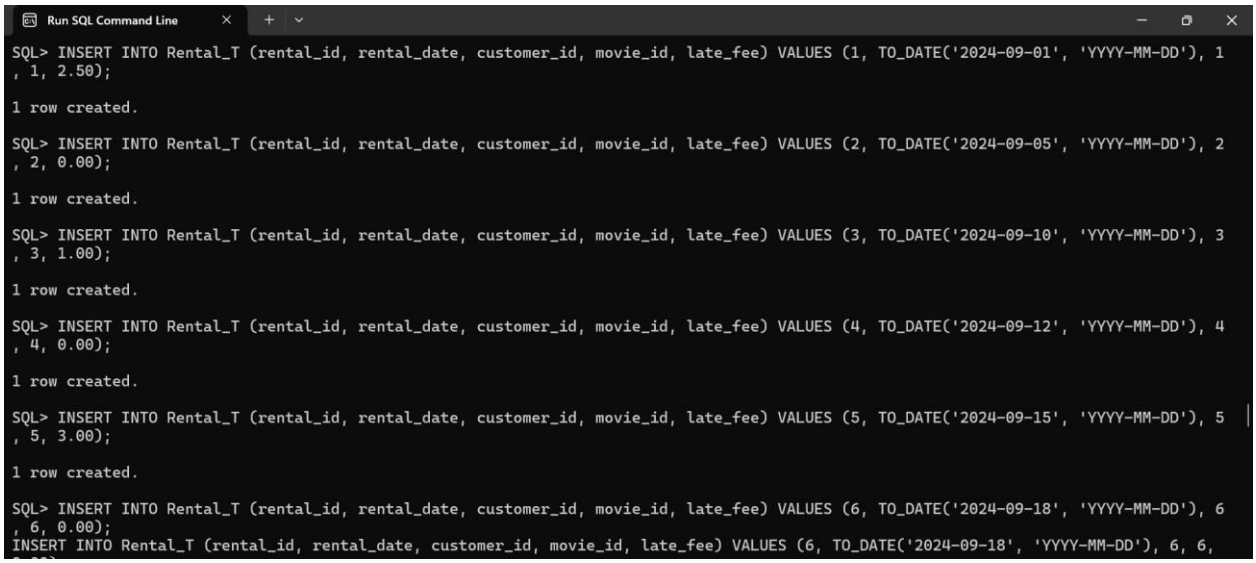
VERIFYING ENTRIES IN MOVIE_T:

```
SQL> select * from movie_t;

  MOVIE_ID
----------
TITLE
--------------------------------------------------------------------------------
GENRE
--------------------------------------------------
LANGUAGE                       DURATION RATIN
------------------------------ ---------- -----
         1
Inception
Sci-Fi
English                               148 PG-13


  MOVIE_ID
----------
TITLE
--------------------------------------------------------------------------------
GENRE
--------------------------------------------------
LANGUAGE                       DURATION RATIN
------------------------------ ---------- -----
         2
Parasite
Thriller
Korean                                132 R


  MOVIE_ID
----------
TITLE
```

//INSERT INTO RENTAL_ID

```
INSERT INTO Rental_T (rental_id, rental_date, customer_id, movie_id, late_fee) VALUES (2, '2024-09-05', 2, 2, 0.00);

INSERT INTO Rental_T (rental_id, rental_date, customer_id, movie_id, late_fee) VALUES (3, '2024-09-10', 3, 3, 1.00);

INSERT INTO Rental_T (rental_id, rental_date, customer_id, movie_id, late_fee) VALUES (4, '2024-09-12', 4, 4, 0.00);

INSERT INTO Rental_T (rental_id, rental_date, customer_id, movie_id, late_fee) VALUES (5, '2024-09-15', 5, 5, 3.00);

INSERT INTO Rental_T (rental_id, rental_date, customer_id, movie_id, late_fee) VALUES (6, '2024-09-18', 6, 6, 0.00);

INSERT INTO Rental_T (rental_id, rental_date, customer_id, movie_id, late_fee) VALUES (7, '2024-09-20', 7, 7, 2.00);

INSERT INTO Rental_T (rental_id, rental_date, customer_id, movie_id, late_fee) VALUES (8, '2024-09-22', 8, 8, 4.50);

INSERT INTO Rental_T (rental_id, rental_date, customer_id, movie_id, late_fee) VALUES (9, '2024-0925', 9, 9, 1.50);

INSERT INTO Rental_T (rental_id, rental_date, customer_id, movie_id, late_fee) VALUES (10, '202409-28', 10, 10, 0.00);

INSERT INTO Rental_T (rental_id, rental_date, customer_id, movie_id, late_fee) VALUES (11, '202410-01', 11, 11, 2.75);

INSERT INTO Rental_T (rental_id, rental_date, customer_id, movie_id, late_fee) VALUES (12, '202410-03', 12, 12, 0.00);

INSERT INTO Rental_T (rental_id, rental_date, customer_id, movie_id, late_fee) VALUES (13, '202410-05', 13, 13, 3.25);

INSERT INTO Rental_T (rental_id, rental_date, customer_id, movie_id, late_fee) VALUES (14, '202410-07', 14, 14, 1.00);

INSERT INTO Rental_T (rental_id, rental_date, customer_id, movie_id, late_fee) VALUES (15, '202410-10', 15, 15, 0.00);

INSERT INTO Rental_T (rental_id, rental_date, customer_id, movie_id, late_fee) VALUES (16, '202410-12', 16, 16, 2.50);

INSERT INTO Rental_T (rental_id, rental_date, customer_id, movie_id, late_fee) VALUES (17, '202410-15', 17, 17, 0.00);

INSERT INTO Rental_T (rental_id, rental_date, customer_id, movie_id, late_fee) VALUES (18, '202410-18', 18, 18, 1.75);

INSERT INTO Rental_T (rental_id, rental_date, customer_id, movie_id, late_fee) VALUES (19, '202410-20', 19, 19, 0.50);
```

INSERT INTO Rental_T (rental_id, rental_date, customer_id, movie_id, late_fee) VALUES (20, '202410-22', 20, 20, 2.00);



```
SQL> select * from rental_t;

 RENTAL_ID RENTAL_DA CUSTOMER_ID   MOVIE_ID   LATE_FEE
---------- --------- ----------- ---------- ----------
         1 01-SEP-24           1          1        2.5
         2 05-SEP-24           2          2          0
         3 10-SEP-24           3          3          1
         4 12-SEP-24           4          4          0
         5 15-SEP-24           5          5          3
         6 18-SEP-24           6          6          0
         7 20-SEP-24           7          7          2
         8 22-SEP-24           8          8        4.5
         9 25-SEP-24           9          9        1.5
        10 28-SEP-24          10         10          0
        11 01-OCT-24          11         11       2.75

 RENTAL_ID RENTAL_DA CUSTOMER_ID   MOVIE_ID   LATE_FEE
---------- --------- ----------- ---------- ----------
        12 03-OCT-24          12         12          0
        13 05-OCT-24          13         13       3.25
        14 07-OCT-24          14         14          1
        15 10-OCT-24          15         15          0
        16 12-OCT-24          16         16        2.5
        17 15-OCT-24          17         17          1
        18 18-OCT-24          18         18       1.75
        19 20-OCT-24          19         19         .5
        20 22-OCT-24          20         20          2

20 rows selected.
```

//INSERT INTO PAYMENT_T:

INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (1, 15.00, 'Credit Card', 1);

INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (2, 20.00, 'Cash', 2);

INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (3, 10.00, 'Debit Card', 3);

INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (4, 25.00, 'Credit Card', 4);

INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (5, 30.00, 'Cash', 5);

INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (6, 5.00, 'Debit Card', 6);

```
INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (7, 12.50, 'Credit Card', 7);

INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (8, 22.00, 'Cash', 8);

INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (9, 18.00, 'Debit Card', 9);

INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (10, 11.00, 'Credit Card', 10);

INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (11, 14.00, 'Cash', 11);

INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (12, 16.00, 'Debit Card', 12);

INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (13, 24.00, 'Credit Card', 13);

INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (14, 28.00, 'Cash', 14);

INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (15, 19.00, 'Debit Card', 15);

INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (16, 21.00, 'Credit Card', 16);

INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (17, 17.00, 'Cash', 17);

INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (18, 13.00, 'Debit Card', 18);

INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (19, 23.00, 'Credit Card', 19);

INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (20, 29.00, 'Cash', 20);
```

```
SQL> INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (1, 15.00, 'Credit Card', 1);
1 row created.
SQL> INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (2, 20.00, 'Cash', 2);
1 row created.
SQL> INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (3, 10.00, 'Debit Card', 3);
1 row created.
SQL> INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (4, 25.00, 'Credit Card', 4);
1 row created.
SQL> INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (5, 30.00, 'Cash', 5);
1 row created.
SQL> INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (6, 5.00, 'Debit Card', 6);
1 row created.
SQL> INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (7, 12.50, 'Credit Card', 7);
1 row created.
SQL> INSERT INTO Payment_T (payment_id, amount, payment_method, rental_id) VALUES (8, 22.00, 'Cash', 8);
1 row created.
```

VERIFY PAYMENT_T ENTRIES:

```
SQL> select * from payment_t;

PAYMENT_ID     AMOUNT PAYMENT_METHOD        RENTAL_ID
---------- ---------- -------------------- ----------
         1         15 Credit Card                   1
         2         20 Cash                          2
         3         10 Debit Card                    3
         4         25 Credit Card                   4
         5         30 Cash                          5
         6          5 Debit Card                    6
         7       12.5 Credit Card                   7
         8         22 Cash                          8
         9         18 Debit Card                    9
        10         11 Credit Card                  10
        11         14 Cash                         11

PAYMENT_ID     AMOUNT PAYMENT_METHOD        RENTAL_ID
---------- ---------- -------------------- ----------
        12         16 Debit Card                   12
        13         24 Credit Card                  13
        14         28 Cash                         14
        15         19 Debit Card                   15
        16         21 Credit Card                  16
        17         17 Cash                         17
        18         13 Debit Card                   18
        19         23 Credit Card                  19
        20         29 Cash                         20

20 rows selected.
```

INSERT INTO STAFF_T

//COMMAND

INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (1, 'Mike Brown', 'Manager', 'mike.brown@example.com', '444-555-6666');

INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (2, 'Sarah Davis', 'Cashier', 'sarah.davis@example.com', '777-888-9999');

INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (3, 'Tom Wilson', 'Technician', 'tom.wilson@example.com', '222-333-4444');

INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (4, 'Lisa Smith', 'Manager', 'lisa.smith@example.com', '111-222-3333');

INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (5, 'James Johnson', 'Cashier', 'james.johnson@example.com', '444-777-8888');

INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (6, 'Emily Garcia', 'Technician', 'emily.garcia@example.com', '555-666-7777');

INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (7, 'David Martinez', 'Manager', 'david.martinez@example.com', '888-999-0000');

INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (8, 'Sophia Rodriguez', 'Cashier', 'sophia.rodriguez@example.com', '222-444-6666');

INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (9, 'Michael Lee', 'Technician', 'michael.lee@example.com', '333-555-7777');

INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (10, 'Olivia Perez', 'Manager', 'olivia.perez@example.com', '444-888-0000');

INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (11, 'William Wilson', 'Cashier', 'william.wilson@example.com', '111-999-2222');

INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (12, 'Isabella Anderson', 'Technician', 'isabella.anderson@example.com', '222-333-8888');

INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (13, 'Daniel Thomas', 'Manager', 'daniel.thomas@example.com', '333-444-5555');

INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (14, 'Mia Moore', 'Cashier', 'mia.moore@example.com', '111-222-4444');

INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (15, 'Matthew Taylor', 'Technician', 'matthew.taylor@example.com', '555-222-7777');

INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (16, 'Ava Jackson', 'Manager', 'ava.jackson@example.com', '777-555-1111');

INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (17, 'Ethan Harris', 'Cashier', 'ethan.harris@example.com', '888-333-2222');

INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (18, 'Charlotte Clark', 'Technician', 'charlotte.clark@example.com', '999-444-3333');

INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (19, 'James Lewis', 'Manager', 'james.lewis@example.com', '111-000-2222');

INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (20, 'Grace Robinson', 'Cashier', 'grace.robinson@example.com', '444-555-8888');

```
Run SQL Command Line

SQL> INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (1, 'Mike Brown', 'Manager', 'mike.brown@example.com', '444-
555-6666');

1 row created.

SQL> INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (2, 'Sarah Davis', 'Cashier', 'sarah.davis@example.com', '77
7-888-9999');

1 row created.

SQL> INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (3, 'Tom Wilson', 'Technician', 'tom.wilson@example.com', '2
22-333-4444');

1 row created.

SQL> INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (4, 'Lisa Smith', 'Manager', 'lisa.smith@example.com', '111-
222-3333');

1 row created.

SQL> INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (5, 'James Johnson', 'Cashier', 'james.johnson@example.com',
 '444-777-8888');

1 row created.

SQL> INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (6, 'Emily Garcia', 'Technician', 'emily.garcia@example.com'
, '555-666-7777');

1 row created.

SQL> INSERT INTO Staff_T (staff_id, name, position, email, phone) VALUES (7, 'David Martinez', 'Manager', 'david.martinez@example.com
', '888-999-0000');
```

VERIFY STAFF_T:

```
  STAFF_ID
----------
NAME
--------------------------------------------------------------------------------
POSITION
--------------------------------------------------
EMAIL
--------------------------------------------------------------------------------
PHONE
---------------
         1
Mike Brown
Manager

  STAFF_ID
----------
NAME
--------------------------------------------------------------------------------
POSITION
--------------------------------------------------
EMAIL
--------------------------------------------------------------------------------
PHONE
---------------
mike.brown@example.com
444-555-6666


  STAFF_ID
----------
NAME
```

STEP 2:

APPLYING DIFFERENT QUERIES:

Find Staff by Position: Retrieves all staff members who hold the position of "Manager"

```
SQL> SELECT * FROM Staff_T WHERE position = 'Manager';

  STAFF_ID
----------
NAME
--------------------------------------------------------------------------------
POSITION
--------------------------------------------------
EMAIL
--------------------------------------------------------------------------------
PHONE
---------------
         1
Mike Brown
Manager

  STAFF_ID
----------
NAME
--------------------------------------------------------------------------------
POSITION
--------------------------------------------------
EMAIL
--------------------------------------------------------------------------------
PHONE
---------------
mike.brown@example.com
444-555-6666


  STAFF_ID
```

```
mike.brown@example.com
444-555-6666


  STAFF_ID
----------
NAME
--------------------------------------------------------------------------------
POSITION
--------------------------------------------------
EMAIL
--------------------------------------------------------------------------------
PHONE
---------------
         4
Lisa Smith
Manager

  STAFF_ID
----------
NAME
--------------------------------------------------------------------------------
POSITION
--------------------------------------------------
EMAIL
--------------------------------------------------------------------------------
PHONE
---------------
lisa.smith@example.com
111-222-3333
```

2) Count Total Staff Members: Returns the total number of staff members in the Staff_T table.

```
SQL> SELECT COUNT(*) FROM Staff_T;

  COUNT(*)
----------
        20
```

3)Retrieve All Payments with Staff Names

Retrieves payment details along with the name of the staff member who processed the payment.

```
SQL> SELECT p.payment_id, p.amount, s.name
  2  FROM Payment_T p
  3  JOIN Staff_T s ON p.rental_id = s.staff_id;

PAYMENT_ID     AMOUNT
---------- ----------
NAME
--------------------------------------------------------------------------------
         1         15
Mike Brown

         2         20
Sarah Davis

         3         10
Tom Wilson


PAYMENT_ID     AMOUNT
---------- ----------
NAME
--------------------------------------------------------------------------------
         4         25
Lisa Smith

         5         30
James Johnson

         6          5
Emily Garcia
```

```
        15         19
Matthew Taylor


PAYMENT_ID     AMOUNT
---------- ----------
NAME
--------------------------------------------------------------------------------
        16         21
Ava Jackson

        17         17
Ethan Harris

        18         13
Charlotte Clark


PAYMENT_ID     AMOUNT
---------- ----------
NAME
--------------------------------------------------------------------------------
        19         23
James Lewis

        20         29
Grace Robinson


20 rows selected.
```

4) Count Rentals by Movie

Counts the number of rentals for each movie by grouping the results by movie title.

```
SQL> SELECT m.title, COUNT(r.rental_id) AS rental_count
  2  FROM Movie_T m
  3  JOIN Rental_T r ON m.movie_id = r.movie_id
  4  GROUP BY m.title;

TITLE
--------------------------------------------------------------------------------
RENTAL_COUNT
------------
Inception
           1

The Matrix
           1

Titanic
           1


TITLE
--------------------------------------------------------------------------------
RENTAL_COUNT
------------
The Silence of the Lambs
           1

The Avengers
           1

The Lion King
           1


TITLE
```

```
The Dark Knight
           1

Pulp Fiction
           1


TITLE
--------------------------------------------------------------------------------
RENTAL_COUNT
------------
Forrest Gump
           1

Interstellar
           1

Avatar
           1


TITLE
--------------------------------------------------------------------------------
RENTAL_COUNT
------------
Parasite
           1

Spirited Away
           1


20 rows selected.
```

5) Find Total Payments per Staff Member

Calculates the total payments processed by each staff member.

```
SQL> SELECT s.name, SUM(p.amount) AS total_payments
  2  FROM Staff_T s
  3  JOIN Payment_T p ON s.staff_id = p.rental_id
  4  GROUP BY s.name;

NAME
--------------------------------------------------------------------------------
TOTAL_PAYMENTS
--------------
Sarah Davis
            20

Emily Garcia
             5

Daniel Thomas
            24


NAME
--------------------------------------------------------------------------------
TOTAL_PAYMENTS
--------------
Ethan Harris
            17

Grace Robinson
            29

Olivia Perez
            11


NAME
--------------------------------------------------------------------------------
```

6) Calculate Average Payment Amount

Calculates the average payment amount across all entries in the Payment_T table.

```
SQL> SELECT AVG(amount) AS average_payment FROM Payment_T;

AVERAGE_PAYMENT
---------------
         18.625
```

7) Retrieve All Rentals with Movie Titles

Retrieves rental IDs along with the titles of the movies rented.

```
SQL> SELECT r.rental_id, m.title
  2  FROM Rental_t r
  3  JOIN Movie_t m ON r.movie_id = m.movie_id;

 RENTAL_ID
----------
TITLE
--------------------------------------------------------------------------------
         1
Inception

         2
Parasite

         3
The Matrix


 RENTAL_ID
----------
TITLE
--------------------------------------------------------------------------------
         4
The Shawshank Redemption

         5
The Godfather

         7
Pulp Fiction


 RENTAL_ID
----------
```

8) Find Movies with Rentals Over a Certain Amount

Retrieves titles of movies that have been rented with payments over $20.00.

```
SQL> SELECT m.title
  2  FROM Movie_t m
  3  JOIN Rental_t r ON m.movie_id = r.movie_id
  4  JOIN Payment_T p ON r.rental_id = p.rental_id
  5  WHERE p.amount > 20.00;

TITLE
--------------------------------------------------------------------------------
The Shawshank Redemption
The Godfather
The Dark Knight
The Silence of the Lambs
Gladiator
Titanic
The Lion King
Your Name

8 rows selected.
```

9) Retrieve Movies with Payment Information

Retrieves movie titles along with the corresponding payment amounts.

```
SQL> SELECT m.title, p.amount
  2  FROM Movie_t m
  3  JOIN Rental_t r ON m.movie_id = r.movie_id
  4  JOIN Payment_T p ON r.rental_id = p.rental_id;

TITLE
--------------------------------------------------------------------------------
    AMOUNT
----------
Inception
        15

Parasite
        20

The Matrix
        10


TITLE
--------------------------------------------------------------------------------
    AMOUNT
----------
The Shawshank Redemption
        25

The Godfather
        30
```

10) Retrieve Payments Made for a Specific Movie

Retrieves payment amounts for rentals of the movie "Schindler's List".

```
SQL> SELECT p.amount
  2  FROM Payment_T p
  3  JOIN Rental_t r ON p.rental_id = r.rental_id
  4  JOIN Movie_t m ON r.movie_id = m.movie_id
  5  WHERE m.title = 'Schindler''s List';

    AMOUNT
----------
         5
```

TRIGGER:

1)Update Rental Status on Payment

//command

```
CREATE TRIGGER UpdateRentalStatusOnPayment

AFTER INSERT ON Payment_T

FOR EACH ROW

BEGIN

   UPDATE Rental_T
```

```
    SET status = 'returned'
    WHERE rental_id = NEW.rental_id;
END;
```

WORKING OF TRIGGER:

When a new payment is inserted into the Payment_T table, the trigger updates the corresponding rental's status in the Rental_T table to 'returned'.

It uses NEW.rental_id to reference the rental_id of the newly inserted payment.

After creating the trigger, you can test it by inserting a payment:

INSERT INTO Payment_T (payment_id, rental_id, amount, payment_method, payment_date)
VALUES (1, 1, 15.00, 'Credit Card', NOW());

This insert will trigger the UpdateRentalStatusOnPayment trigger, updating the status of the rental with rental_id 1 to 'returned'.

2) Trigger to Prevent Overdue Rentals from Being Returned

This trigger prevents the return of rentals that are overdue.

```
//command
DELIMITER //

CREATE TRIGGER PreventOverdueReturn
BEFORE UPDATE ON Rental_T
FOR EACH ROW
BEGIN
    IF NEW.status = 'returned' AND DATEDIFF(NOW(), NEW.return_date) > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot return overdue rental.';
    END IF;
END;
//

DELIMITER ;
```

3) Trigger to Automatically Set Payment Date

This trigger automatically sets the payment date to the current date when a new payment is made.

```
//command
DELIMITER //

CREATE TRIGGER SetPaymentDate
BEFORE INSERT ON Payment_T
FOR EACH ROW
BEGIN
    SET NEW.payment_date = NOW();
END;
//

DELIMITER ;
```

CREATING ROLES:

```
CREATE ROLE Admin_Role;
CREATE ROLE Staff_Role;
CREATE ROLE Customer_Role;
```

GRANT PRIVILEGES TO ROLES:

```
GRANT ALL PRIVILEGES ON *.* TO Admin_Role;
 GRANT SELECT, INSERT, UPDATE, DELETE ON Movies_T TO Staff_Role;
GRANT SELECT, INSERT, UPDATE, DELETE ON Rental_T TO Staff_Role;
GRANT SELECT ON Movies_T TO Customer_Role;
GRANT SELECT, INSERT ON Rental_T TO Customer_Role;
```

REVOKE PRIVILEGES:

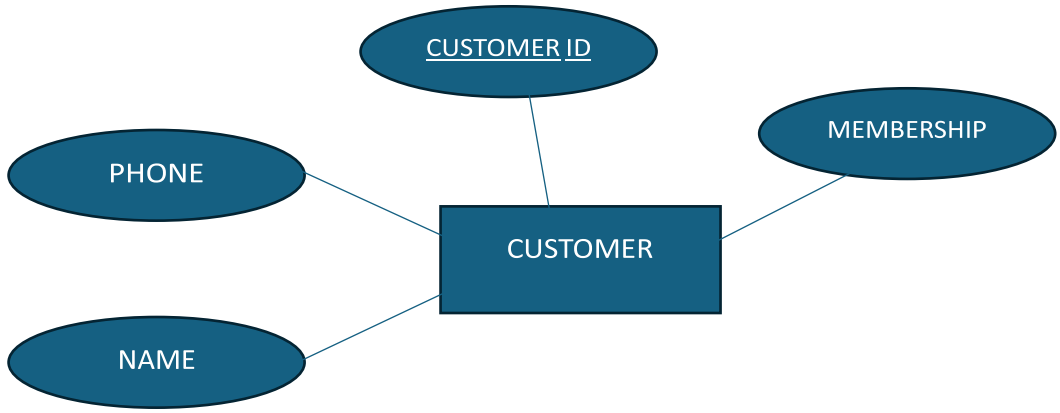
```
REVOKE INSERT ON Movies_T FROM Staff_Role;
```

DROP ROLES:

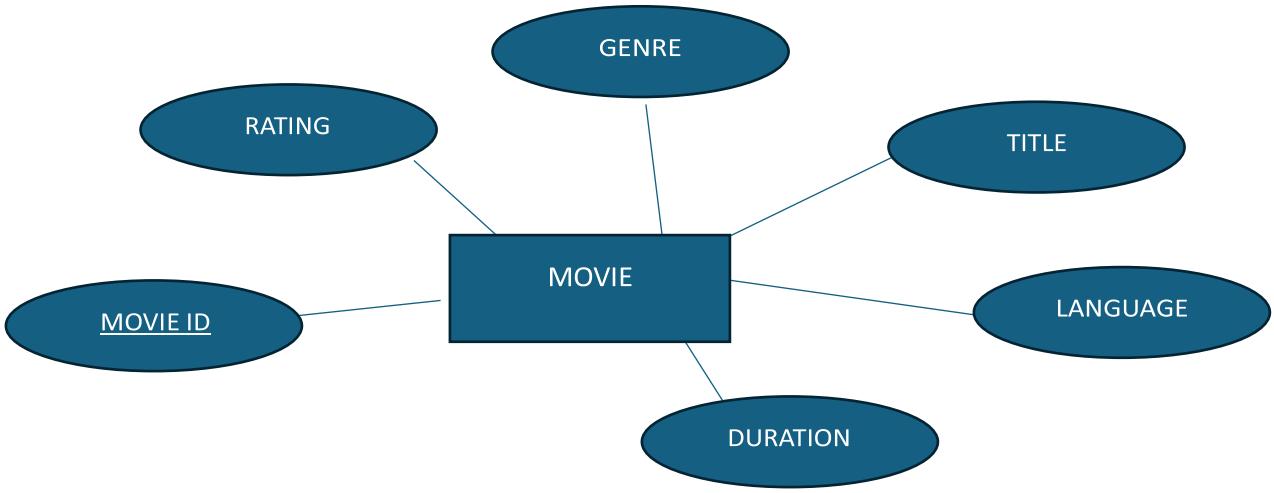
```
DROP ROLE Staff_Role;
```

ENTERPRISE DATA MODEL FOR MOVIE RENTAL SYSTEM

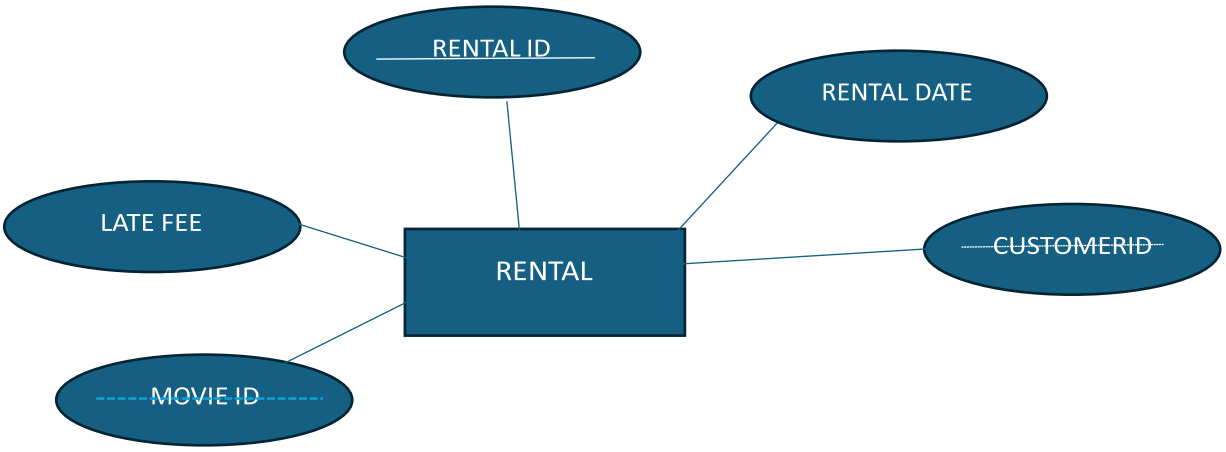
FOR CUSTOMER:



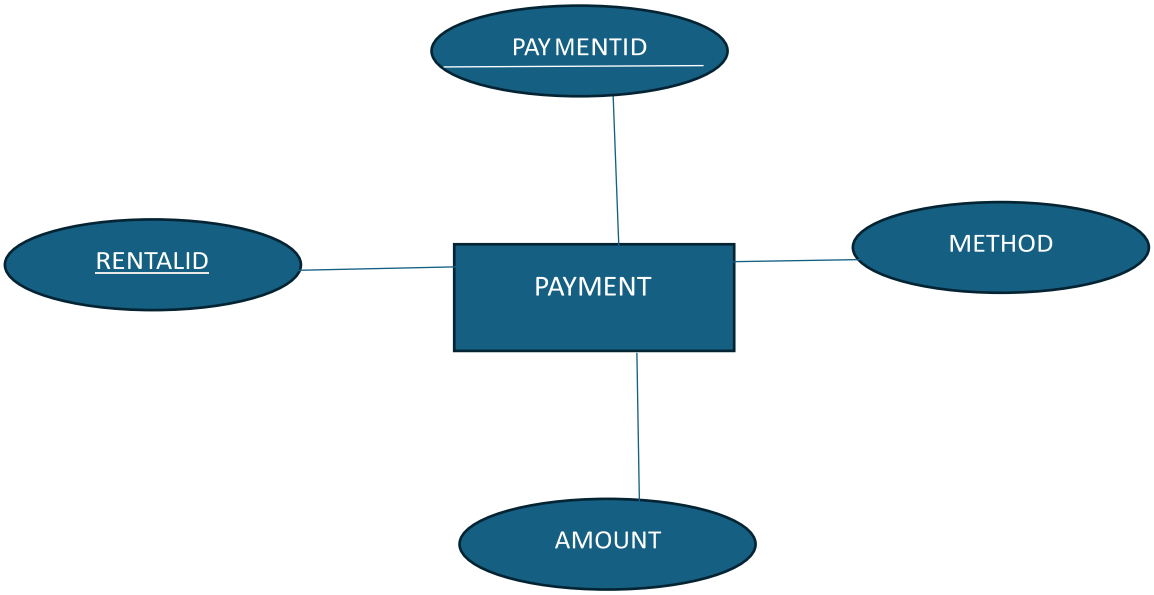
FOR MOVIE:



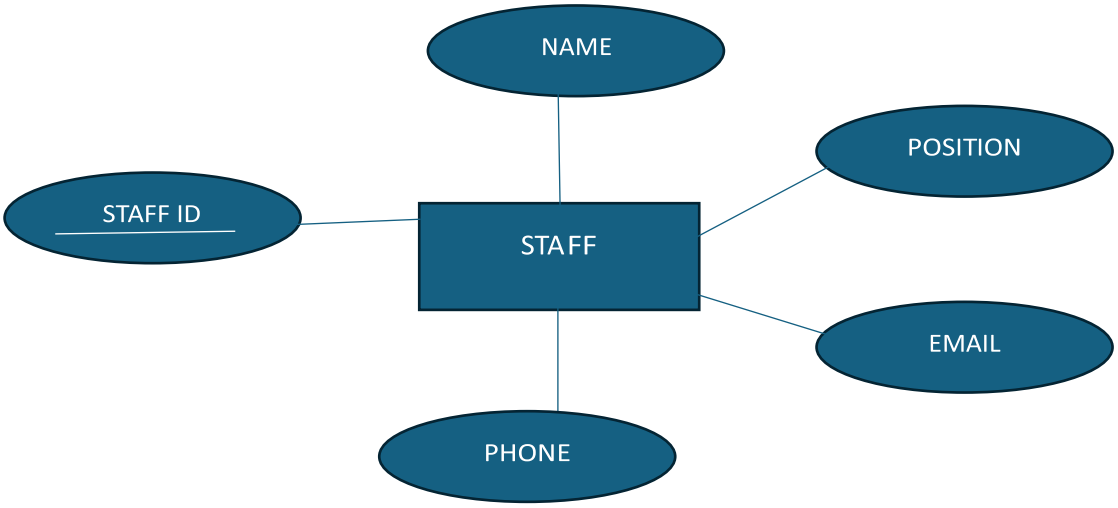
FOR RENTAL:



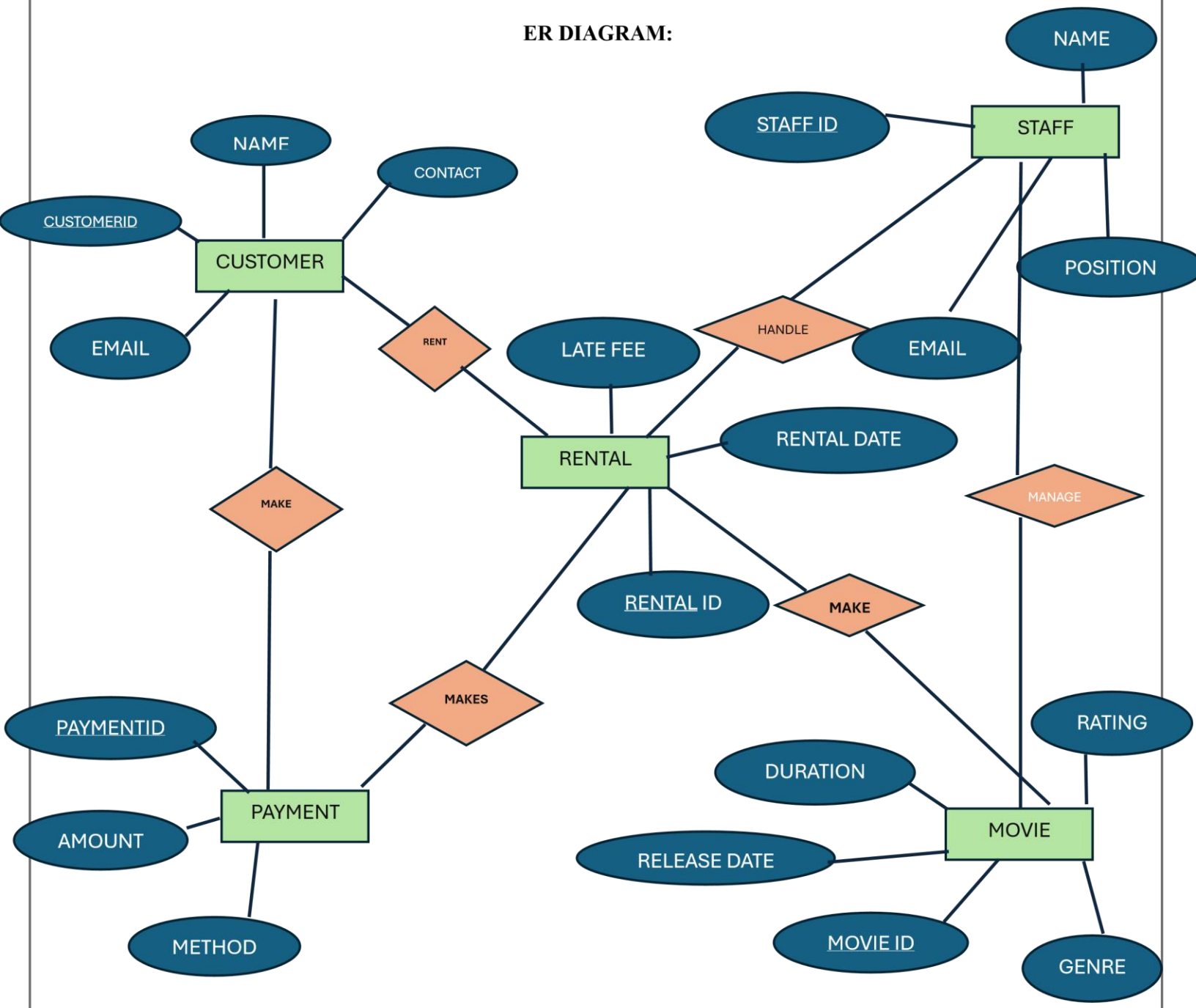
FOR PAYMENT:



FOR STAFF:


NAME
POSITION
STAFF ID
STAFF
EMAIL
PHONE

ER DIAGRAM:



ENTERPRISE RELATIONAL MODEL

SCHEMAS:

FOR CUSTOMER:

CUSTOMER ID	NAME	PHONE	MEMBERSHIP

FOR MOVIE:

MOVIE ID	TITLE	GENRE	RATING	LANGUAGE	DURATION

FOR RENTAL:

RENTAL ID	RENTAL DATE	CUSTOMER ID	MOVIE ID	STAFF ID	LATE FEE

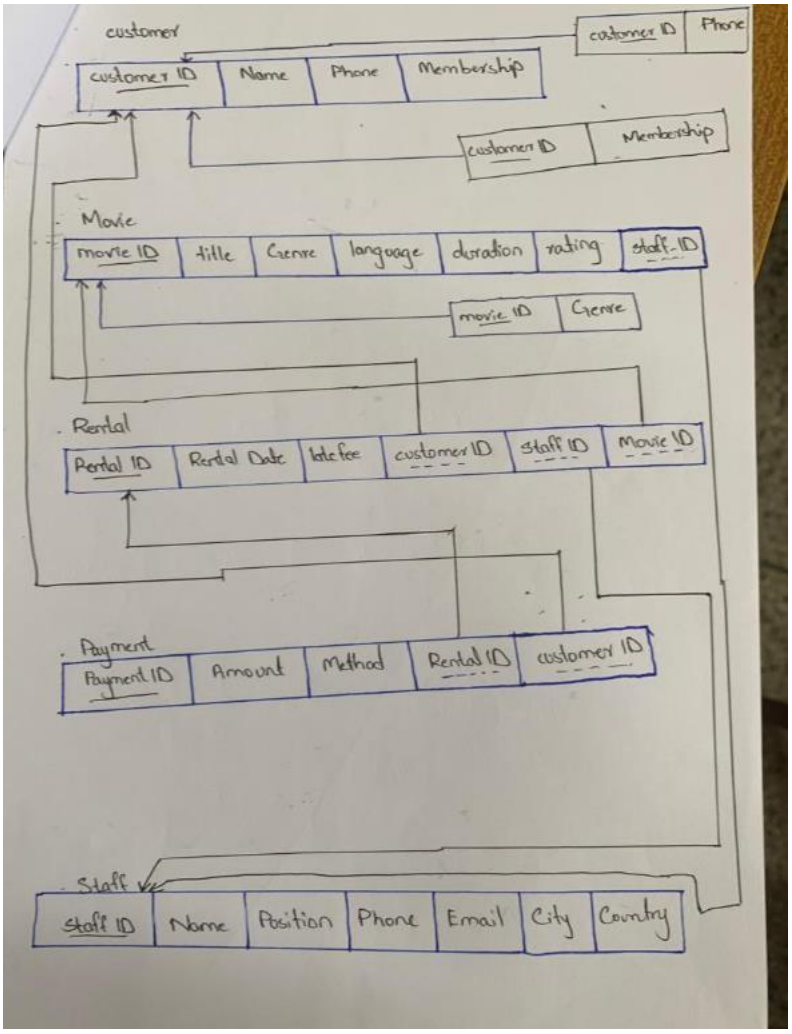
FOR PAYMENT:

PAYMENT ID	CUSTOMER ID	RENTAL ID	METHOD	AMOUNT

FOR STAFF:

STAFF ID	NAME	PHONE	POSITION	EMAIL

COMBINED RELATIONAL MODEL



NORMALIZATION:

CUSTOMER_t

CUSTOMER ID	NAME	PHONE	MEMBERSHIP
1	John	+9278844748	Gold
2	sarah	+9274784939	silver

CONVERTING TO 1 NF

CUSTOMER ID	NAME	MEMBERSHIP
1	John	gold
2	sarah	Silver

CUSTOMER ID	PHONE
1	+9278844748
2	+9274784939

CONVERTING TO 2 NF:

Already in 2 NF

CONVERTING TO 3 NF:

Already in 3 nf(No transitive dependencies)

MOVIE_t:

MOVIE ID	TITLE	DURATION	RATING	YEAR	RENTALID	GENRE	LANGUAGE
1	Avengers	2 hr	4	2019	R1	Scifi,comedy	Eng
2	revenge	3 hr	3.5	2020	R2	Romcom,thriller	Eng,urdu

CONVERTING TO 1 NF:

MOVIE ID	GENRE
1	Scifi
1	Comedy
2	Romcom
2	thriller

MOVIE ID	LANGUAGE
1	Eng
1	Eng
2	Eng
2	urdu

MOVIE ID	TITLE	DURATION	RATING	YEAR	RENTAL ID
1	Avengers	2 hr	4	2019	R1
2	Revenge	3 hr	3.5	2020	R2

CONVERTING TO 2 NF:

MOVIE ID	RENTAL ID
1	R1
2	R2

MOVIE ID	TITLE	DURATION	RATING	YEAR
1	Avengers	2 hr	4	2019
2	revenge	3 hr	3.5	2020

CONVERTING TO 3 NF:

Already in 3 nf

RENTAL_t:

Rental id	date	Late fee	Customer id	Payment id
R1	1-2-24	5000	1	P1
R2	4-2-24	2000	2	P2

Already in 1 nf

Already in 2 nf

Already in 3 NF(No transitive dependencies)

PAYMENT_t:

Payment id	amount	method	Customer id
P1	5000	Cash	1
P2	2000	online	2

Already in 1 nf

Already in 2 nf

Already in 3Nf(No transitive dependencies) **STAFF:**

Staff id	name	phone	email	position	Rental id	Movie id
S1	**Ali**	**+09133930**	**ali@gmail.com**	**Employee**	**R1**	**1**
S2	**Ahmed**	**+0918383**	**ahmed@gmail.com**	**head**	**R2**	**2**

CONVERTING TO 1 NF:

STAFF ID	NAME	POSTION	RENTAL ID	MOVIE ID
S1	Ali	Employee	R1	1
S2	ahmed	Head	R2	2

STAFF ID	PHONE

S1	+09133930
S2	+0918383

STAFF ID	EMAIL
S1	ali@gmail.com
S2	ahmed@gmail.com

CONVERTING TO 2NF:

STAFF ID	NAME	POSITION
S1	Ali	Employee
S2	Ahmed	Head

STAFF ID	PHONE
S1	+09133930
S2	+0918383

STAFF ID	RENTAL ID	MOVIE ID
S1	R1	1
S2	R2	2

(COMPOSITE KEY: staff id , rental id , movie id)

CONVERTING TO 3NF:

Already in 3 NF(No transitive dependencies)

BUSINESS RULES FOR THE MOVIE RENTAL SYSYTEM

1. Customer Rules

Customer Registration:

A customer must be registered with the system before renting any movies.

Required fields: Name, Phone, Email, and Membership Type.

Unique Customer ID:

Each customer must have a unique CustomerID.

Membership Types:

Customers can have different membership levels (e.g., Regular, Premium).

Membership level affects rental rates, late fees, and borrowing limits.

Contact Information:

Customers must provide a valid phone number and email address.

2. Movie Rules

Unique Movie ID:

Each movie must have a unique MovieID.

Movie Availability:

A movie can be rented only if it is currently available in stock.

Genres and Ratings:

Each movie must have a defined Genre and Rating (e.g., PG, R).

Movie Duration:

The Duration field must be a positive integer (in minutes).

Staff Association:

Each movie record must include the StaffID of the staff member who manages the movie inventory. **3.**

Rental Rules

Rental Eligibility:

Only registered customers can rent movies.

Rental Limits:

Regular members can rent up to 3 movies at a time.

Premium members can rent up to 5 movies at a time.

Rental Period:

Standard rental period is 7 days. Premium members may have extended periods (e.g., 14 days).

Late Fees:

Late fees apply for movies returned after the due date.

Fee structure: $1.50 per day late (Regular), $1.00 per day late (Premium).

Rental ID:

Each rental transaction must have a unique RentalID.

Foreign Key Constraints:

CustomerID in the Rental table references CustomerID in the Customer table.

MovieID in the Rental table references MovieID in the Movie table.

StaffID in the Rental table references StaffID in the Staff table.

4. Payment Rules

Payment Requirement:

Payments must be made for each rental transaction before the due date or upon return.

Payment Methods:

Accepted methods: Cash, Credit Card, Debit Card, Mobile Payment.

Payment Amount:

Payment amount includes rental fees and any applicable late fees.

Unique Payment ID:

Each payment must have a unique PaymentID.

Foreign Key Constraints:

RentalID in the Payment table references RentalID in the Rental table.

CustomerID in the Payment table references CustomerID in the Customer table.

5. Staff Rules

Staff Registration:

All staff members must be registered in the system with details like Name, Position, Email, and Phone.

Unique Staff ID:

Each staff member must have a unique StaffID. ***Roles***

and Responsibilities:

Staff roles may include Manager, Cashier, and Inventory Manager.

Only authorized staff (e.g., Inventory Managers) can add or remove movies.

6. Delivery (If Applicable) Home Delivery Option:

Customers can opt for home delivery for an additional fee.

Delivery Tracking:

Each delivery must have a unique DeliveryID.

Delivery Status:

Status values: Pending, Dispatched, Delivered, Returned.

Departure and Arrival Times:

Delivery records must include Departure and Arrival timestamps.

7. System Constraints

Data Integrity:

Ensure foreign keys are enforced to maintain relationships between tables.

Data Validation:

Phone numbers must follow a valid format.

Email addresses must be unique and properly formatted.

Mandatory Fields:

Certain fields (e.g., Customer Name, Movie Title, Rental Date) cannot be left blank.

Sample Business Scenarios **Renting**

a Movie:

When a customer rents a movie, a new Rental record is created.

The movie's availability status is updated.

Late Returns:

If a movie is returned late, a late fee is calculated and added to the payment amount.

Payments:

A customer can make partial or full payment for their rentals and any incurred late fees.

Inventory Management:

Staff members add new movies, update movie details, and remove old movies from the catalog.

By clearly defining these business rules, you ensure that the system meets operational requirements and maintains data integrity.

----------THE END----------