ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

STAT311

Modern Database Systems

Term Project

DVD Rental Database Management System

by

Aişe Esra Gönen 2561272 - Ayşe Büşra Aktaş 2561017

Düzgün Ali Özdemir 2561454 - Melisa Çepni 2288132

Instructors

Prof. Dr. İhsan Tolga Medeni - Res.Assist. Mehmet Ali Erkan

JANUARY 2025

METU – ANKARA

Table of Contents

1. Introduction

Our project for the "STAT311 Modern Database Systems" course focuses on a "DVD Rental System". The purpose of this project is to design and implement a database system that manages the operations of a DVD rental business. The system includes several key entities that reflect the essential components of the business, such as films, rentals, customers, payments, and staff. These tables and their attributes ensure efficient data organization, management, and retrieval, facilitating smooth operations for the rental service.

We have 13 tables in total, including 3 relation tables and 2 of them are considered as weak entities. In addition, we have 4 other relations. We thought that 13 entities are enough and adding more entities was not necessary for our database and it can make our database system more complicated which might cause redundancy.

Here are the tables and their attributes in our database:

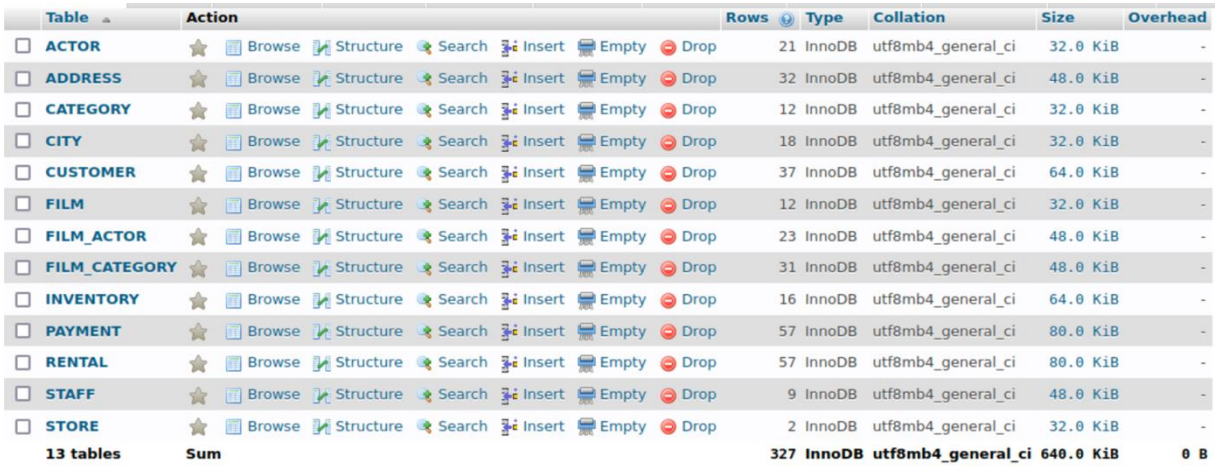


Table	Action	Rows	Type	Collation	Size	Overhead
ACTOR	Browse Structure Search Insert Empty Drop	21	InnoDB	utf8mb4_general_ci	32.0 KiB	-
ADDRESS	Browse Structure Search Insert Empty Drop	32	InnoDB	utf8mb4_general_ci	48.0 KiB	-
CATEGORY	Browse Structure Search Insert Empty Drop	12	InnoDB	utf8mb4_general_ci	32.0 KiB	-
CITY	Browse Structure Search Insert Empty Drop	18	InnoDB	utf8mb4_general_ci	32.0 KiB	-
CUSTOMER	Browse Structure Search Insert Empty Drop	37	InnoDB	utf8mb4_general_ci	64.0 KiB	-
FILM	Browse Structure Search Insert Empty Drop	12	InnoDB	utf8mb4_general_ci	32.0 KiB	-
FILM_ACTOR	Browse Structure Search Insert Empty Drop	23	InnoDB	utf8mb4_general_ci	48.0 KiB	-
FILM_CATEGORY	Browse Structure Search Insert Empty Drop	31	InnoDB	utf8mb4_general_ci	48.0 KiB	-
INVENTORY	Browse Structure Search Insert Empty Drop	16	InnoDB	utf8mb4_general_ci	64.0 KiB	-
PAYMENT	Browse Structure Search Insert Empty Drop	57	InnoDB	utf8mb4_general_ci	80.0 KiB	-
RENTAL	Browse Structure Search Insert Empty Drop	57	InnoDB	utf8mb4_general_ci	80.0 KiB	-
STAFF	Browse Structure Search Insert Empty Drop	9	InnoDB	utf8mb4_general_ci	48.0 KiB	-
STORE	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	32.0 KiB	-
13 tables	Sum	327	InnoDB	utf8mb4_general_ci	640.0 KiB	0 B

- actor = actor_id (PK), actor_name, actor_surname
- address = address_id (PK), address, district, phone, city_id (FK)
- category = category_id (PK), name
- city = city_id (PK), city
- customer = customer_id (PK), store_id (FK), first_name, last_name, email, address_id (FK), activebool, create_date
- film = film_id (PK), title
- film_actor (relationship between film and actor tables) = actor_id (FK), film_id (FK)
- film_category (relationship between film and category tables) = film_id (FK), category_id (FK)
- inventory (relationship between film and store tables) = inventory_id (PK), film_id (FK), store_id (FK)
- staff = staff_id (PK), staff_name, staff_surname, email, store_id (FK), username
- payment = payment_id (PK), customer_id (FK), staff_id (FK), rental_id (FK), amount, payment_date
- rental = rental_id (PK), inventory_id (FK), customer_id (FK), staff_id (FK), rental_date, return_date
- store = store_id (PK), manager_staff_id

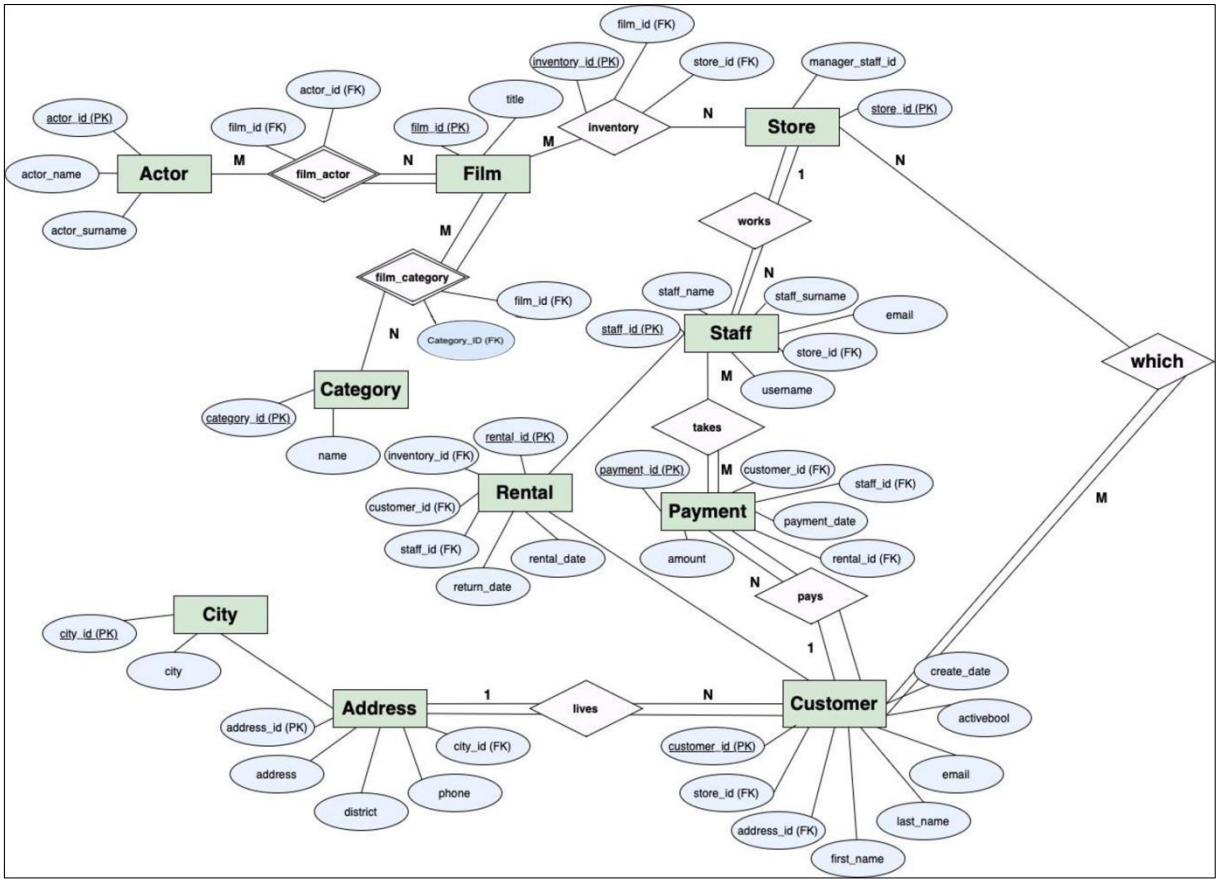
Key Instructions and Business Logic:

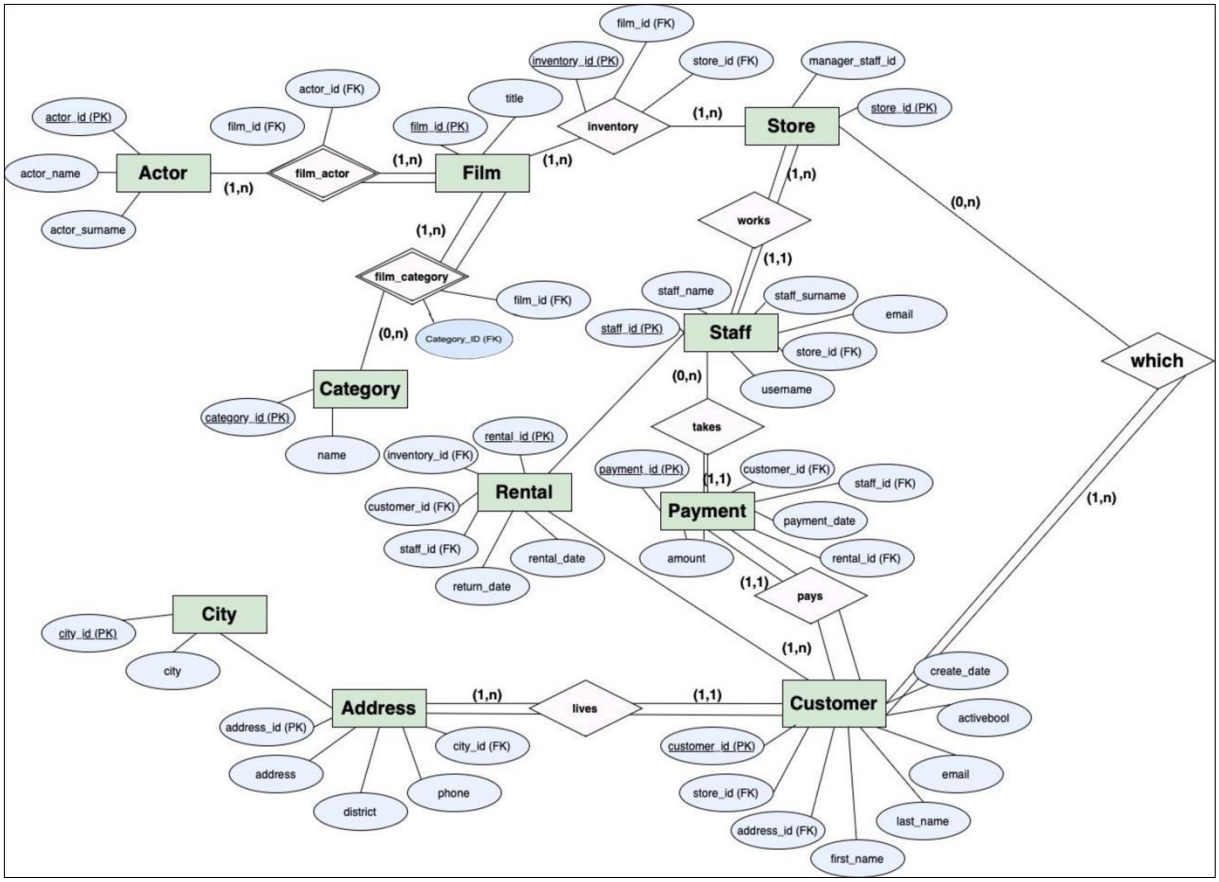
1. Customer and Address: Each customer is linked to a specific address. However, a single address can have multiple customers.
2. Customer and Store Relationship: Each customer can make purchases from multiple stores, and each store serves multiple customers.
3. Customer and Payment Relationship: A customer can make multiple payments, but each payment is associated with only one customer.
4. Staff and Payment Relationship: A single staff member can process multiple payments, but a payment can only be handled by one staff member.
5. Staff and Store Relationship: Each staff member works in one store only, while a store can have multiple staff members.
6. Store and Film Relationship: Each store can rent out multiple films, and each film can be available in multiple stores.
7. Film and Category Relationship: Every film must belong to at least one category. However, a category does not necessarily need to have a film assigned to it.
8. Film and Actor Relationship: An actor can appear in multiple films, and a film can feature multiple actors.

Implementation Details:

Our project is built on MySQL 8.0.40 as the database management system, and the development and testing were conducted using Apache 2.4.52 as the web server and PHP 8.1.2 The design includes relationships, constraints, and necessary functionalities to meet the requirements of a modern DVD rental business. This system efficiently manages customer data, rental operations, payment transactions, and inventory, providing a comprehensive solution for a rental service.

2. ER Diagrams





3. Schema Design

Here are our table design properties. We used and showed CREATE, UPDATE, DELETE and SELECT queries in the following parts.

Actor Schema

a. CREATE

```
CREATE TABLE ACTOR ( Actor_ID int NOT NULL, Actor_Name varchar(50) COLLATE utf8mb4_general_ci NOT NULL,
Actor_Surname varchar(50) COLLATE utf8mb4_general_ci NOT NULL, PRIMARY KEY (Actor_ID) ) ENGINE=InnoDB DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

#	Name	Type	Collation	Attributes	Null	Default
1	**Actor_ID**	int			No	*None*
2	**Actor_Name**	varchar(50)	utf8mb4_general_ci		No	*None*
3	**Actor_Surname**	varchar(50)	utf8mb4_general_ci		No	*None*

The actor schema stores Actor_ID, which is primary key, Actor_Name and Actor_Surname. Actor_Name is the name of the actor and the Actor_Surname is the surname of the actor. Actor_Name and Actor_Surname are variable characters which consist of maximum 50 characters.

b. UPDATE

	Actor_ID	Actor_Name	Actor_Surname
Edit Copy Delete	1	Al	Pacino
Edit Copy Delete	2	Marlon	Brando
Edit Copy Delete	3	Christian	Bale

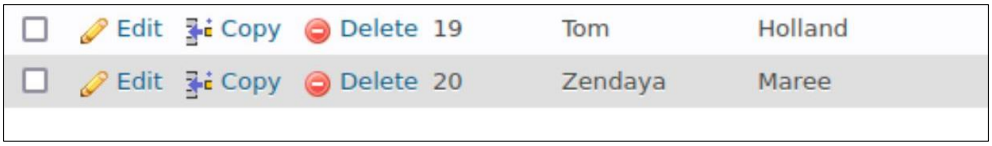
```
1 UPDATE ACTOR SET
2 Actor_Name = "Diane" Actor_Surname = "Keaton"
3 WHERE Actor_ID = 2
4
```

	Actor_ID	Actor_Name	Actor_Surname
Edit Copy Delete	1	Al	Pacino
Edit Copy Delete	2	Diane	Keaton
Edit Copy Delete	3	Christian	Bale

c. DELETE

Edit	Copy	Delete	19	Tom	Holland
Edit	Copy	Delete	20	Zendaya	Maree
Edit	Copy	Delete	21	Tobin	Bell

```
1 DELETE FROM ACTOR
2 WHERE Actor_Name = "Tobin"
```



Edit	Copy	Delete	19	Tom	Holland
Edit	Copy	Delete	20	Zendaya	Maree

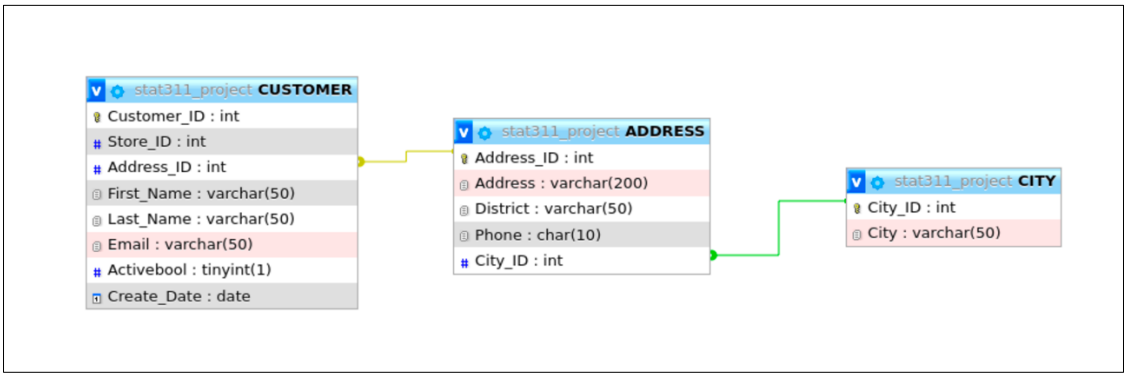
Address Schema

a. CREATE

```
CREATE TABLE ADDRESS ( Address_ID int NOT NULL, Address varchar(200) COLLATE utf8mb4_general_ci NOT NULL,
District varchar(50) COLLATE utf8mb4_general_ci NOT NULL, Phone char(10) COLLATE utf8mb4_general_ci NOT NULL,
City_ID int NOT NULL, PRIMARY KEY (Address_ID), FOREIGN KEY (City_ID) REFERENCES CITY (City_ID) ON DELETE
CASCADE ON UPDATE CASCADE ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

#	Name	Type	Collation	Attributes	Null	Default
1	**Address_ID**	int			No	*None*
2	**Address**	varchar(200)	utf8mb4_general_ci		No	*None*
3	**District**	varchar(50)	utf8mb4_general_ci		No	*None*
4	**Phone**	char(10)	utf8mb4_general_ci		No	*None*
5	**City_ID**	int			No	*None*

Address_ID is the primary key of this table. Address is the location record of the costumers. District is the area that addresses are affiliated. Costumers' phone numbers are recorded as Phone. City_ID is referenced from City table as a foreign key. Addresses and Districts are variable characters which may take up to 200 and 50 respectively, and Phone is the character that takes exactly 10 characters.



b. UPDATE

	Address_ID	Address	District	Phone	City_ID
Edit Copy Delete	101	İstiklal Caddesi No:45	Beyoğlu	2125551234	1

```
1 UPDATE ADDRESS SET Address = 'Abdi İpekçi Caddesi No:45'
2 WHERE Address_ID = 101
```

	Address_ID	Address	District	Phone	City_ID
Edit Copy Delete	101	Abdi İpekçi Caddesi No:45	Beyoğlu	2125551234	1

c. DELETE

Edit Copy Delete	114	Kayalıpark Mahallesi No:2	Selçuklu	3324567890	7
Edit Copy Delete	115	Piri Reis Mahallesi No:14	Yenişehir	3241234567	9
Edit Copy Delete	116	Mithatpaşa Caddesi No:55	Bornova	2323456789	4

```
1 DELETE FROM ADDRESS
2 WHERE Address = 'Piri Reis Mahallesi No:14'
```

Edit Copy Delete	114	Kayalıpark Mahallesi No:2	Selçuklu	3324567890	7
Edit Copy Delete	116	Mithatpaşa Caddesi No:55	Bornova	2323456789	4

Category Schema

a. CREATE

```
CREATE TABLE CATEGORY ( Category_ID int NOT NULL, Name varchar(50) COLLATE utf8mb4_general_ci NOT NULL )
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

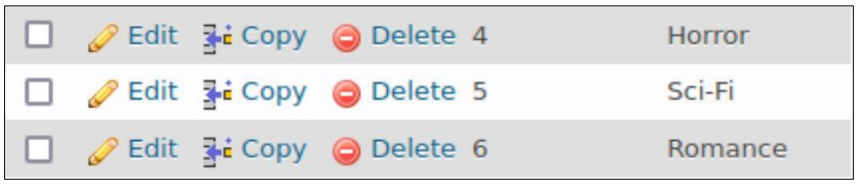
#	Name	Type	Collation	Attributes	Null	Default
1	**Category_ID**	int			No	*None*
2	**Name**	varchar(50)	utf8mb4_general_ci		No	*None*

Category_ID is the primary key of the table. Category names are shown as Name, and it can take maximum 50 characters.

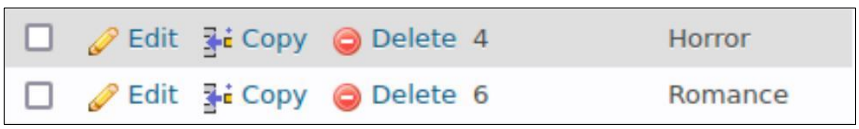
b. UPDATE


```
1 UPDATE CATEGORY SET
2 Name = "Adventure"
3 WHERE Category_ID = 10
```

c. DELETE




```
1 DELETE FROM CATEGORY
2 WHERE Name = "Sci-Fi"
```



City Schema

a. CREATE

```
CREATE TABLE CITY ( City_ID int NOT NULL, City varchar(50) COLLATE utf8mb4_general_ci NOT NULL ) ENGINE=InnoDB
DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

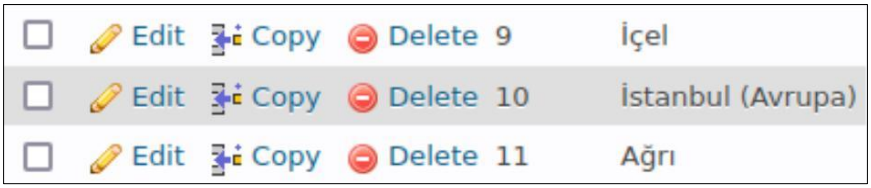
#	Name	Type	Collation	Attributes	Null	Default
1	City_ID	int			No	*None*
2	City	varchar(50)	utf8mb4_general_ci		No	*None*

The City schema has City_ID, which is the primary key, and City as attributes. City is a variable that refers the city for the addresses of costumers and its type is variable character that can contain maximum 50 characters.

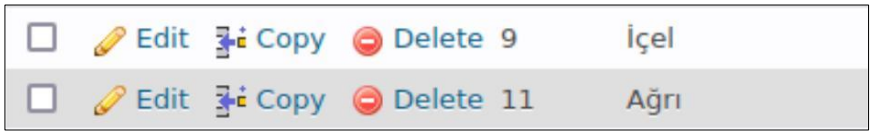
b. UPDATE


```
1 UPDATE CITY SET City= "İçel"
2 WHERE City_ID = 9
```

c. DELETE




```
1 DELETE FROM CITY
2 WHERE City = "İstanbul (Avrupa)"
```



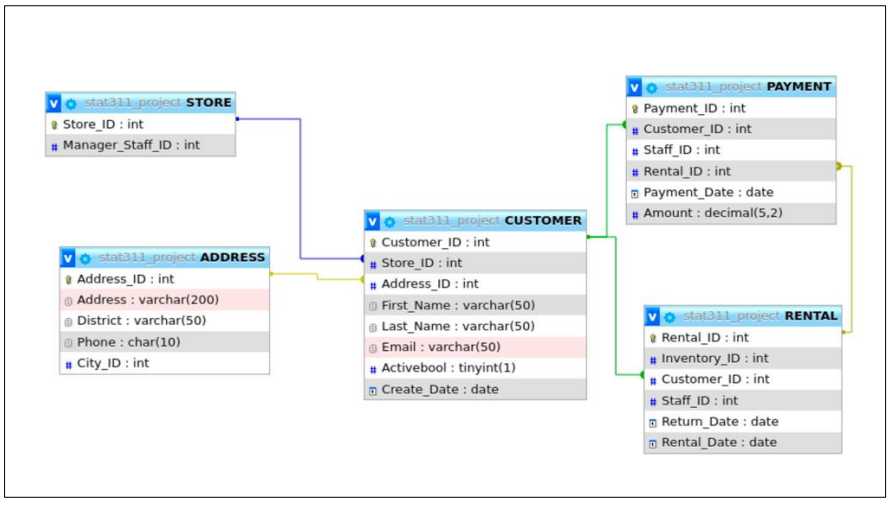
Customer Schema

a. CREATE

```
CREATE TABLE CUSTOMER ( Customer_ID int NOT NULL, Store_ID int NOT NULL, Address_ID int NOT NULL, First_Name
varchar(50) COLLATE utf8mb4_general_ci NOT NULL, Last_Name varchar(50) COLLATE utf8mb4_general_ci NOT NULL,
Email varchar(50) COLLATE utf8mb4_general_ci NOT NULL, Activebool tinyint(1) NOT NULL, Create_Date date NOT
NULL, PRIMARY KEY (Customer_ID), FOREIGN KEY (Store_ID) REFERENCES STORE (Store_ID) ON DELETE CASCADE ON UPDATE
CASCADE, FOREIGN KEY (Address_ID) REFERENCES ADDRESS (Address_ID) ON DELETE CASCADE ON UPDATE CASCADE )
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

#	Name	Type	Collation	Attributes	Null	Default
1	**Customer_ID**	int			No	*None*
2	**Store_ID**	int			No	*None*
3	**Address_ID**	int			No	*None*
4	**First_Name**	varchar(50)	utf8mb4_general_ci		No	*None*
5	**Last_Name**	varchar(50)	utf8mb4_general_ci		No	*None*
6	**Email**	varchar(50)	utf8mb4_general_ci		No	*None*
7	**Activebool**	tinyint(1)			No	*None*
8	**Create_Date**	date			No	*None*

Customer schema has 8 attributes which are Customer_ID, as primary key, Store_ID, Address_ID, First_Name, Last_Name, Email, Activebool, Create_Date. Store_ID which is taken from Store table is a foreign key in this table. Address_ID is a foreign key from Address table. First_Name is the names of the customers while Last_Name refers to surnames of the customers. Email is the address that we can reach to customers via mail. Activebool indicates that whether customers are active or not. Create_Date is the date that customers activated their memberships. First_Name, Last_Name and Email can take maximum 50 characters. Activebool is determined as tinyint(1) because it can take 0 for not active customers and 1 foractive ones.



b. UPDATE

Edit Copy Delete 7	1	107	Fatma	Aslan	fatma.aslan@hotmail.com	1	2023-06-18

```
UPDATE CUSTOMER SET
Email  = "aslan.fatma@hotmail.com"
WHERE Customer_ID = 7
```

Edit Copy Delete 7	1	107	Fatma	Aslan	aslan.fatma@hotmail.com	1	2023-06-18

c. DELETE

Edit Copy Delete 30	1	130	Yusuf	Taş	yusuf.tas@gmail.com	1	2024-02-01
Edit Copy Delete 31	1	131	Aylin	Demirtaş	aylin.demirtas@hotmail.com	1	2024-02-10
Edit Copy Delete 32	2	132	Sibel	Çiçek	sibel.cicek@gmail.com	1	2024-02-20

```
DELETE FROM CUSTOMER
WHERE Customer_ID = 31
```

Edit Copy Delete 30	1	130	Yusuf	Taş	yusuf.tas@gmail.com	1	2024-02-01
Edit Copy Delete 32	2	132	Sibel	Çiçek	sibel.cicek@gmail.com	1	2024-02-20

Film Schema

a. CREATE

```
CREATE TABLE FILM ( Film_ID int NOT NULL, Title varchar(100) COLLATE utf8mb4_general_ci NOT NULL, PRIMARY KEY (Film_ID) ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

#	Name	Type	Collation	Attributes	Null	Default
1	**Film_ID**	int			No	*None*
2	**Title**	varchar(100)	utf8mb4_general_ci		No	*None*

In film table, Film_ID is the primary key and Title refers to the title of the films. Title can take maximum of 100 variable characters.

b. UPDATE

Edit Copy Delete 11	Spiderman: Homecoming		

```
UPDATE FILM
SET Title = "Spiderman: No Way Home"
WHERE Film_ID = 11
```

Edit Copy Delete 11	Spiderman: No Way Home		

c. DELETE


```
DELETE FROM FILM
WHERE Title = "Catch Me If You Can"
```

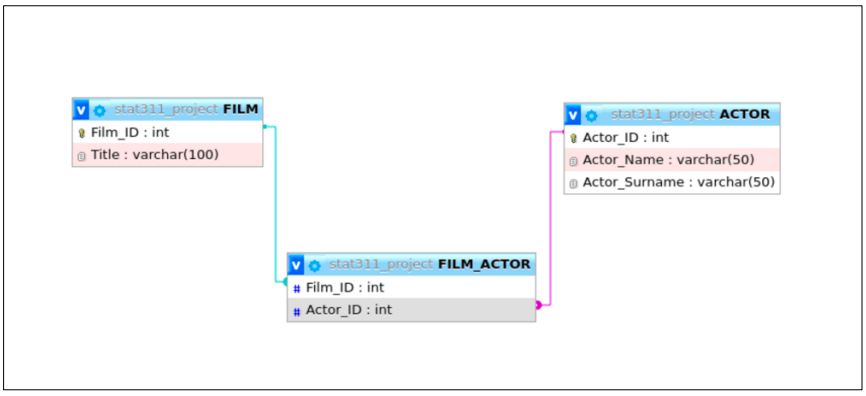
Film_Actor Schema

a. CREATE

```
CREATE TABLE FILM_ACTOR ( Film_ID int NOT NULL, Actor_ID int NOT NULL, FOREIGN KEY (Actor_ID) REFERENCES ACTOR (Actor_ID) ON DELETE CASCADE ON UPDATE CASCADE, FOREIGN KEY (Film_ID) REFERENCES FILM (Film_ID) ON DELETE CASCADE ON UPDATE CASCADE ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

#	Name	Type	Collation	Attributes	Null	Default
1	**Film_ID**	int			No	*None*
2	**Actor_ID**	int			No	*None*

This table, which is a weak entity, is only for showing the relationship between Film and Actor tables. Film_ID comes from Film table and Actor_ID comes from Actor table as foreign keys.



b. UPDATE

Since it is a relationship table, we did not update it individually.

c. DELETE

Here from the given tables, we can see the before-after part of deleting process. Since it is a relationshp table of FILM and ACTOR tables, any deleted records from these tables also will be deleted here. Two red highlighted records were deleted from FILM table, where yellow highlighted record was deleted from ACTOR table.

Film_ID	Actor_ID
1	1
1	2
2	3
2	4
3	5
4	6
4	7
5	8
2	9
10	9
5	10
6	11
6	12
7	13
2	14
7	14
8	15
9	16
10	17
10	18
11	19
11	20
12	21

Film_ID	Actor_ID
1	1
1	2
2	3
2	4
3	5
4	6
4	7
2	9
10	9
6	11
6	12
7	13
2	14
7	14
8	15
9	16
10	17
10	18
11	19
11	20

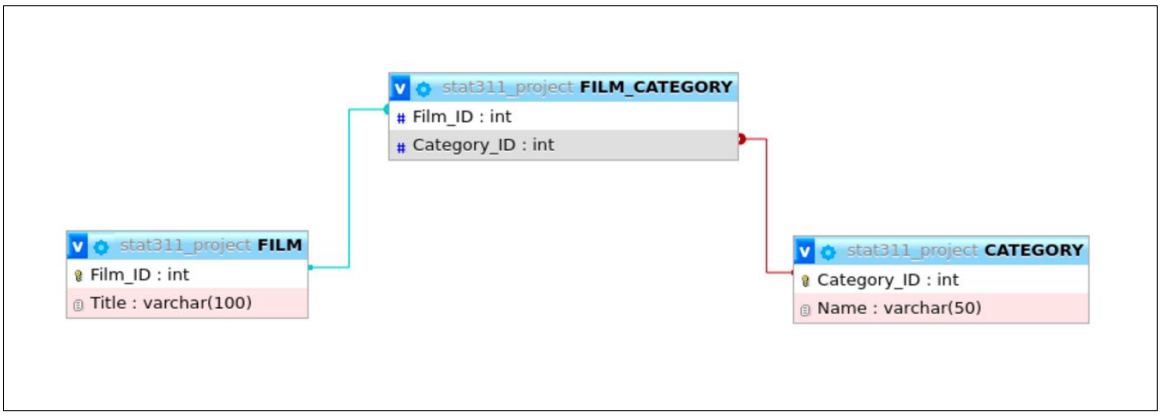
Film_Category Schema

a. CREATE

```
CREATE TABLE FILM_CATEGORY ( Film_ID int NOT NULL, Category_ID int NOT NULL, FOREIGN KEY (Category_ID)
REFERENCES CATEGORY (Category_ID) ON DELETE CASCADE ON UPDATE CASCADE, FOREIGN KEY (Film_ID)
REFERENCES FILM (Film_ID) ON DELETE CASCADE ON UPDATE CASCADE ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;
```

#	Name	Type	Collation	Attributes	Null	Default
1	**Film_ID**	int			No	*None*
2	**Category_ID**	int			No	*None*

This table, which is a weak entity, is only for showing the relationship between Film and Category tables. Film_ID comes from Film table and Category_ID comes from Category table as foreign keys.



b. UPDATE

Since it is a relationship table, we did not update it individually.

c. DELETE

Here from the given tables, we can see the before-after part of deleting process. Since it is a relationship table of FILM and CATEGORY tables, any deleted records from these tables also will be deleted here. Three red highlighted records were deleted from FILM table, where two yellow highlighted records were deleted from CATEGORY table.

Film_ID	Category_ID
1	1
1	3
1	12
2	1
2	5
2	7
3	9
3	10
4	2
4	6
5	1
5	2
5	10
6	3
6	6
7	1
7	5
7	10
7	12
8	4
8	7
9	4
9	7
10	3
10	9
10	10
11	1
11	2
11	9
12	4
12	7

Film_ID	Category_ID
1	1
1	3
1	12
2	1
2	7
3	9
3	10
4	2
4	6
6	3
6	6
7	1
7	10
7	12
8	4
8	7
9	4
9	7
10	3
10	9
10	10
11	1
11	2
11	9
12	4
12	7

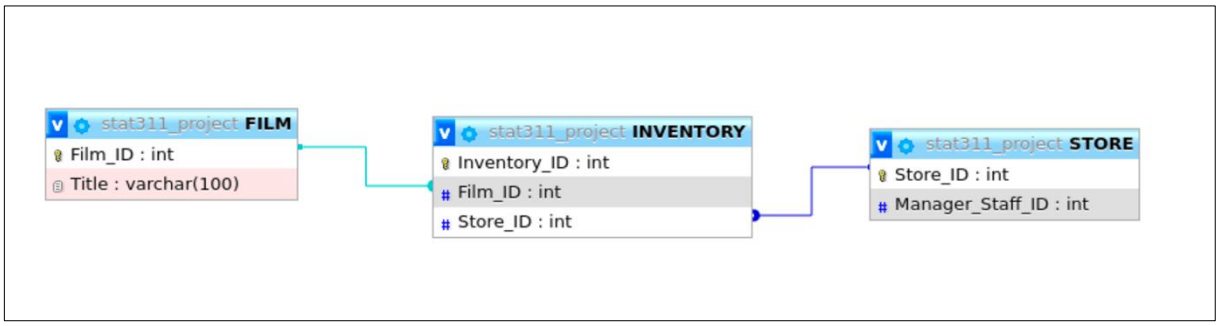
Inventory Schema

a. CREATE

```
CREATE TABLE INVENTORY ( Inventory_ID int NOT NULL, Film_ID int NOT NULL, Store_ID int NOT NULL, PRIMARY KEY (Inventory_ID), FOREIGN KEY (Film_ID) REFERENCES FILM (Film_ID) ON DELETE CASCADE ON UPDATE CASCADE, FOREIGN KEY (Store_ID) REFERENCES STORE (Store_ID) ON DELETE CASCADE ON UPDATE CASCADE ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

#	Name	Type	Collation	Attributes	Null	Default
1	**Inventory_ID**	int			No	*None*
2	**Film_ID**	int			No	*None*
3	**Store_ID**	int			No	*None*

Inventory table, which is a relationship table, has 3 attributes which are Inventory_ID as primary key, Film_ID and Store_ID as foreign keys refers to Film and Store tables respectively.

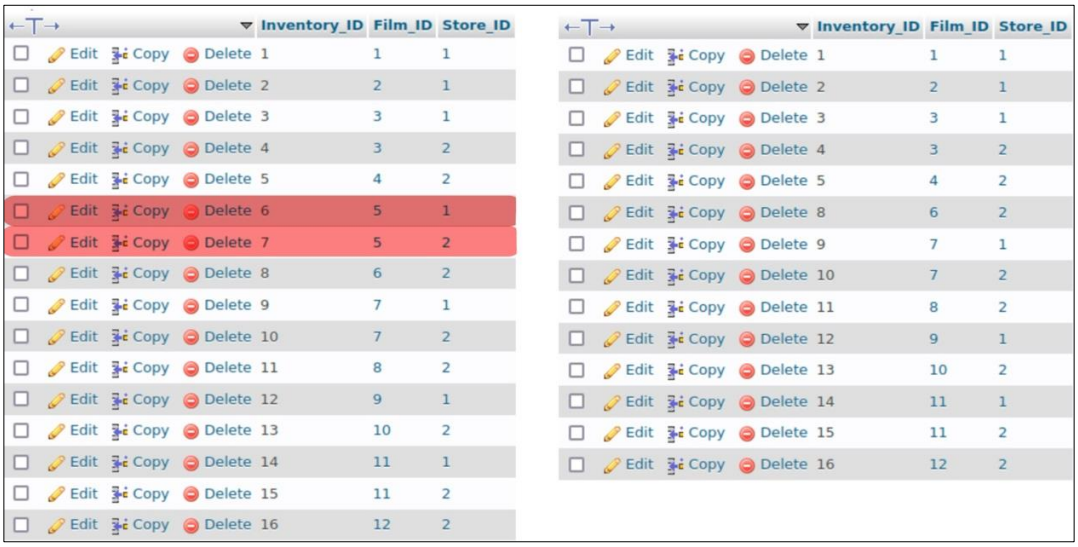


b. UPDATE


```
UPDATE INVENTORY
SET Store_ID = 1
WHERE Inventory_ID = 8
```

c. DELETE

Here from the given tables, we can see the before-after part of deleting process. Since it is a relationship table of FILM and STORE tables, any deleted records from these tables also will be deleted here. Two red highlighted records were deleted from FILM table.



Inventory_ID	Film_ID	Store_ID
1	1	1
2	1	1
3	1	1
4	2	2
5	2	2
6	1	1
7	2	2
8	2	2
9	1	1
10	2	2
11	2	2
12	1	1
13	2	2
14	1	1
15	2	2
16	2	2

Inventory_ID	Film_ID	Store_ID
1	1	1
2	1	1
3	1	1
4	2	2
5	2	2
8	2	2
9	1	1
10	2	2
11	2	2
12	1	1
13	2	2
14	1	1
15	2	2
16	2	2

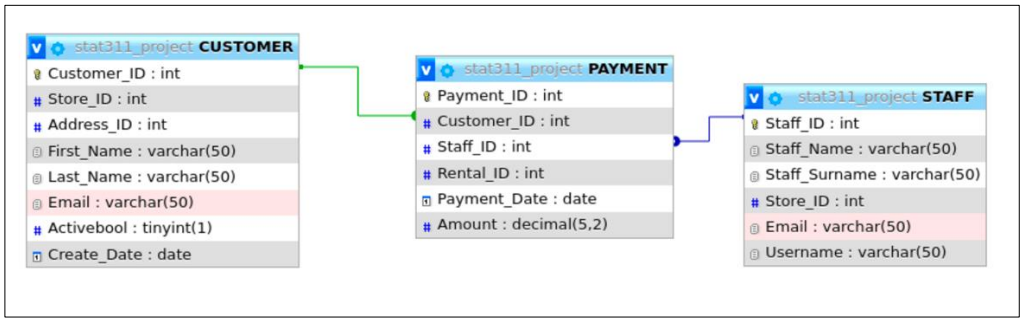
Payment Schema

a.CREATE

```
CREATE TABLE PAYMENT ( Payment_ID int NOT NULL, Customer_ID int NOT NULL, Staff_ID int NOT NULL,
Rental_ID int NOT NULL, Payment_Date date NOT NULL, Amount decimal(5,2) NOT NULL, PRIMARY KEY
(Payment_ID), FOREIGN KEY (Customer_ID) REFERENCES CUSTOMER (Customer_ID) ON DELETE CASCADE ON UPDATE
CASCADE, FOREIGN KEY (Staff_ID) REFERENCES STAFF (Staff_ID) ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (Rental_ID) REFERENCES RENTAL (Rental_ID) ON DELETE CASCADE ON UPDATE CASCADE )
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

#	Name	Type	Collation	Attributes	Null	Default
1	**Payment_ID**	int			No	*None*
2	**Customer_ID**	int			No	*None*
3	**Staff_ID**	int			No	*None*
4	**Rental_ID**	int			No	*None*
5	**Payment_Date**	date			No	*None*
6	**Amount**	decimal(5,2)			No	*None*

Payment schema has 6 attributes which are Payment_ID, as the primary key, Customer_ID, Staff_ID, Rental_ID, Payment_Date, and Amount. Customer_ID, Staff_ID, Rental_ID are foreign keys from Customer, Staff and Rental tables respectively. Amount can take up to 5 digits, including 2 decimal places.



b. UPDATE

	Payment_ID	Customer_ID	Staff_ID	Rental_ID	Payment_Date	Amount
Edit Copy Delete	1	1	1	1	2024-01-05	104.99

```
1 UPDATE PAYMENT SET
2 Amount = 138.50
3 WHERE Payment_ID = 1
```

	Payment_ID	Customer_ID	Staff_ID	Rental_ID	Payment_Date	Amount
Edit Copy Delete	1	1	1	1	2024-01-05	138.50

c. DELETE

Edit Copy Delete	5	5	2	5	2024-02-17	158.50
Edit Copy Delete	6	6	2	6	2024-02-28	102.50
Edit Copy Delete	7	7	3	7	2024-02-28	134.99

```
1 DELETE FROM PAYMENT
2 WHERE Payment_ID = 6
```

Edit Copy Delete	5	5	2	5	2024-02-17	158.50
Edit Copy Delete	7	7	3	7	2024-02-28	134.99

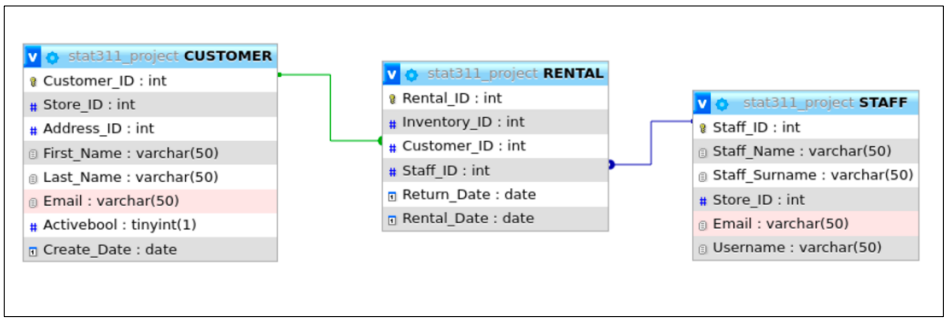
Rental Schema

a. CREATE

```
CREATE TABLE RENTAL ( Rental_ID int NOT NULL, Inventory_ID int NOT NULL, Customer_ID int NOT NULL,
Staff_ID int NOT NULL, Return_Date date NOT NULL, Rental_Date date NOT NULL, PRIMARY KEY (Rental_ID),
FOREIGN KEY (Inventory_ID) REFERENCES INVENTORY (Inventory_ID) ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (Customer_ID) REFERENCES CUSTOMER (Customer_ID) ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (Staff_ID) REFERENCES STAFF (Staff_ID) ON DELETE CASCADE ON UPDATE CASCADE ) ENGINE=InnoDB
DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

#	Name	Type	Collation	Attributes	Null	Default
1	**Rental_ID**	int			No	*None*
2	**Inventory_ID**	int			No	*None*
3	**Customer_ID**	int			No	*None*
4	**Staff_ID**	int			No	*None*
5	**Return_Date**	date			No	*None*
6	**Rental_Date**	date			No	*None*

In Rental Schema, Rental_ID is the primary key. Inventory_ID is the foreign key from Inventory table. Customer_ID is referenced from Customer table, and lastly Staff_ID comes from Staff table. Return_Date is the date that DVDs are returned while the Rental_Date is the date that customers take the DVD.



b. UPDATE

	Rental_ID	Inventory_ID	Customer_ID	Staff_ID	Return_Date	Rental_Date
Edit Copy Delete	1	1	1	1	2024-01-15	2024-01-05
Edit Copy Delete	2	4	2	2	2024-01-17	2024-01-14

```
1 UPDATE RENTAL SET
2 Return_Date = "2024-01-25"
3 WHERE Rental_ID = 1
```

	Rental_ID	Inventory_ID	Customer_ID	Staff_ID	Return_Date	Rental_Date
Edit Copy Delete	1	1	1	1	2024-01-25	2024-01-05
Edit Copy Delete	2	4	2	2	2024-01-17	2024-01-14

c. DELETE

	Rental_ID	Inventory_ID	Customer_ID	Staff_ID	Return_Date	Rental_Date
Edit Copy Delete	1	1	1	1	2024-01-25	2024-01-05
Edit Copy Delete	2	4	2	2	2024-01-17	2024-01-14

```
1 DELETE FROM RENTAL
2 WHERE Rental_ID = 1
```

	Rental_ID	Inventory_ID	Customer_ID	Staff_ID	Return_Date	Rental_Date
Edit Copy Delete	2	4	2	2	2024-01-17	2024-01-14
Edit Copy Delete	3	3	3	1	2024-02-19	2024-01-26

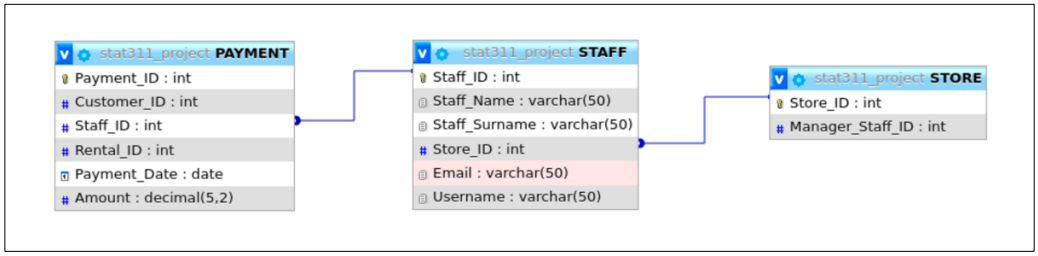
Staff Schema

a. CREATE

```
CREATE TABLE STAFF ( Staff_ID int NOT NULL, Staff_Name varchar(50) COLLATE utf8mb4_general_ci NOT NULL,
Staff_Surname varchar(50) COLLATE utf8mb4_general_ci NOT NULL, Store_ID int NOT NULL, Email varchar(50) COLLATE
utf8mb4_general_ci NOT NULL, Username varchar(50) COLLATE utf8mb4_general_ci NOT NULL, PRIMARY KEY (Staff_ID),
FOREIGN KEY (Store_ID) REFERENCES STORE (Store_ID) ON DELETE CASCADE ON UPDATE CASCADE ) ENGINE=InnoDB DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

#	Name	Type	Collation	Attributes	Null	Default
1	**Staff_ID**	int			No	*None*
2	**Staff_Name**	varchar(50)	utf8mb4_general_ci		No	*None*
3	**Staff_Surname**	varchar(50)	utf8mb4_general_ci		No	*None*
4	**Store_ID**	int			No	*None*
5	**Email**	varchar(50)	utf8mb4_general_ci		No	*None*
6	**Username**	varchar(50)	utf8mb4_general_ci		No	*None*

Staff_ID is the primary key. Staff_Name is the name and Staff_Surname is the last name for staffs. Store_ID is a foreign key from Store table. Email is the address that we can reach to staff via mail. Username is the nickname for the staffs.



b. UPDATE

	Staff_ID	Staff_Name	Staff_Surname	Store_ID	Email	Username
Edit Copy Delete	1	Ali	Özdemir	1	ali.özdemir@dvdrental.com	aliozdmr
Edit Copy Delete	2	Melisa	Çepni	2	melisa.cepni@dvdrental.com	melisacepni

```
UPDATE STAFF SET
Staff_Name  = "Düzgün Ali"
WHERE Staff_ID  = 1
```

	Staff_ID	Staff_Name	Staff_Surname	Store_ID	Email	Username
Edit Copy Delete	1	Düzgün Ali	Özdemir	1	ali.özdemir@dvdrental.com	aliozdmr
Edit Copy Delete	2	Melisa	Çepni	2	melisa.cepni@dvdrental.com	melisacepni

c. DELETE

Edit Copy Delete	4	Ayşe	Aktaş	2	ayse.aktas@dvdrental.com	ayseaktas
Edit Copy Delete	5	Büşra	Gökçen	1	busra.gokcen@dvdrental.com	busragkcn
Edit Copy Delete	6	Furkan	Karatoprak	2	furkan.karatoprak@dvdrental.com	frknkrtprk

```
DELETE FROM STAFF
WHERE Staff_ID = 5
```

Edit Copy Delete	4	Ayşe	Aktaş	2	ayse.aktas@dvdrental.com	ayseaktas
Edit Copy Delete	6	Furkan	Karatoprak	2	furkan.karatoprak@dvdrental.com	frknkrtprk

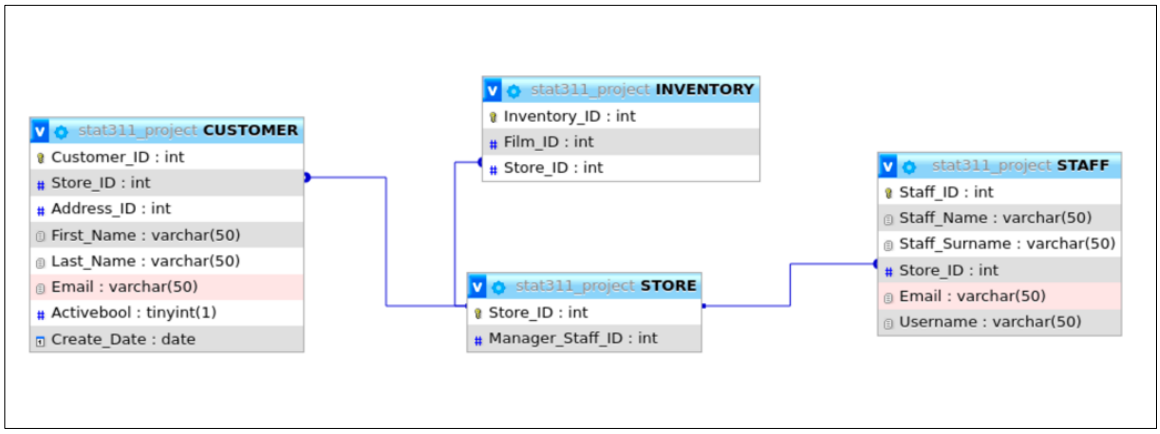
Store Schema

a. CREATE

```
CREATE TABLE STORE ( Store_ID int NOT NULL, Manager_Staff_ID int NOT NULL, PRIMARY KEY (Store_ID) )
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

#	Name	Type	Collation	Attributes	Null	Default
1	**Store_ID**	int			No	*None*
2	**Manager_Staff_ID**	int			No	*None*

In this table, Store_ID is the primary key that specifies each store and Manager_Staff_ID shows that which staff manages which store.



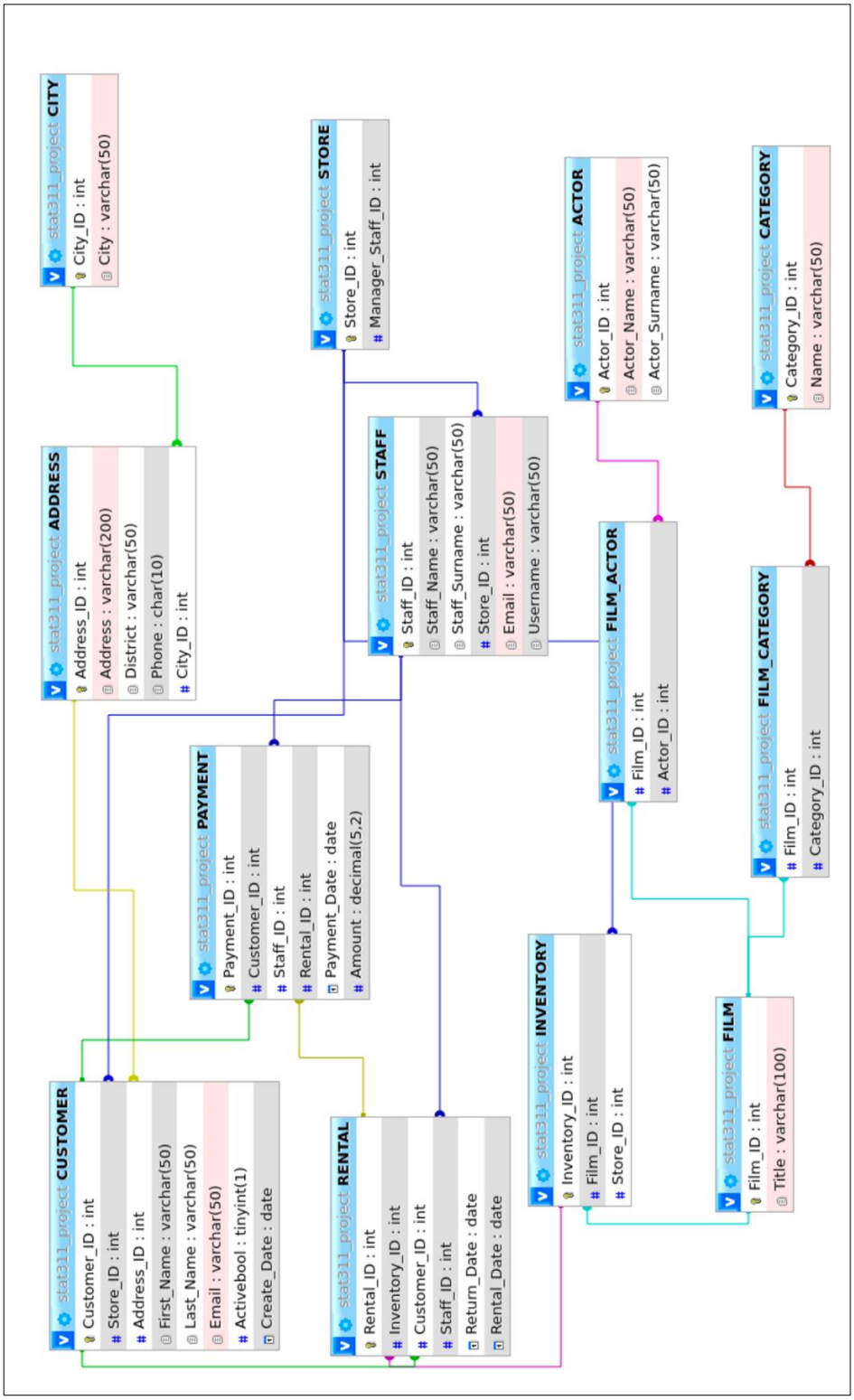
b. UPDATE - DELETE

Since we have only two stores, deleting or updating STORE table seems unnecessary. Hence, we leave our STORE table as it was. However, it is possible to apply DELETE and UPDATE on STORE table.

Overall, while applying DELETE and UPDATE process we did not encounter any problem. That because when we created the database, and its relationship we, used CASCADE to be consistent all over the table. When we delete the any primary key in any table. the foreign key that referenced from that primary key will be also deleted. It can be seen in the FILM_CATEGORY and FILM_ACTOR table.

4. Relationship Table

We can see the relationships between the schemas as follows;



5. Populated Tables

We populate the tables using test records, which are later updated during the development process. The frozen versions of these tables, along with their populated records (tuples), are provided in this section.

1.Actor Table

```
SELECT * FROM `ACTOR`
```

Actor_ID	Actor_Name	Actor_Surname
1	Al	Pacino
2	Marlon	Brando
3	Christian	Bale
4	Heath	Ledger
5	Elijah	Wood
6	Julie	Delpy
7	Ethan	Hawke
8	Tom	Hanks
9	Gary	Oldman
10	Leonardo	DiCaprio
11	Emilia	Clarke
12	Sam	Claflin
13	Robert	Pattinson
14	Michael	Caine
15	Bill	Skarsgard
16	Anthony	Hopkins
17	Daniel	Radcliffe
18	Emma	Watson
19	Tom	Holland
20	Zendaya	Maree
21	Tobin	Bell

2. Address Table

```
SELECT * FROM `ADDRESS`
```

Address_ID	Address	District	Phone	City_ID
101	İstiklal Caddesi No:45	Beyoğlu	2125551234	1
102	Bağdat Caddesi No:12	Kadıköy	2164567890	2
103	Cumhuriyet Mahallesi 123. Sokak No:5	Yenimahalle	3123456789	3
104	Atatürk Bulvarı No:18	Konak	2324567890	4
105	Sakarya Caddesi No:9	Odunpazarı	2225678901	5
106	Kazım Karabekir Caddesi No:3	Melikgazi	3527890123	6
107	Mevlana Caddesi No:22	Karatay	3325678901	7
108	Çırağan Caddesi No:15	Beşiktaş	2126789012	1
109	Alsancak Mahallesi Kordon Sokak No:7	Konak	2326789012	4

...

115	Piri Reis Mahallesi No:14	Yenişehir	3241234567	9
116	Mithatpaşa Caddesi No:55	Bornova	2323456789	4
117	Osmanlı Bulvarı No:18	Başakşehir	2125678901	10
118	Cumhuriyet Caddesi No:6	Talas	3523456789	6
119	Ağrı Dağı Caddesi No:9	Doğubayazıt	4721234567	11
120	Pamukkale Mahallesi No:10	Pamukkale	2585678901	12
121	Kaleiçi Sokak No:22	Muratpaşa	2427890123	13
122	Karşıyaka Mahallesi No:19	Karşıyaka	2326789012	4
123	Yalı Caddesi No:4	Dikili	2323456789	4
124	Huzur Mahallesi No:7	Nilüfer	2241234567	14
125	Fuar Caddesi No:2	Yenişehir	4125678901	15
126	Altınkum Mahallesi No:11	Didim	2561234567	16
127	Zeytinlik Caddesi No:29	Tarsus	3246789012	17
128	Kale Mahallesi No:6	Alanya	2421234567	13
129	Barbaros Bulvarı No:17	Kaş	2425678901	13
130	Güzeloba Mahallesi No:24	Lara	2427890123	13
131	Denizevleri Mahallesi No:7	Atakum	3624567890	18
132	Cumhuriyet Mahallesi No:10	Atakum	3621234567	18

3. Category Table

```
SELECT * FROM `CATEGORY`
```

Category_ID	Name
1	Action
2	Comedy
3	Drama
4	Horror
5	Sci-Fi
6	Romance
7	Thriller
8	Documentary
9	Fantasy
10	Adventure
11	Biography
12	Mystery

4. City Table

```
SELECT * FROM `CITY`
```

City_ID	City
1	İstanbul
2	Ankara
3	İzmir
4	Antalya
5	Eskişehir
6	Kayseri
7	Konya
8	Hatay
9	Mersin
10	İstanbul (Avrupa)
11	Ağrı
12	Denizli
13	Antalya (Merkez)
14	Bursa
15	Diyarbakır
16	Aydın
17	Tarsus
18	Samsun

5. Customer Table

```
SELECT * FROM `CUSTOMER`
```

Customer_ID	Store_ID	Address_ID	First_Name	Last_Name	Email	Activebool	Create_Date
1	1	101	Ahmet	Yılmaz	ahmet.yilmaz@hotmail.com	1	2023-01-15
2	1	102	Mehmet	Kaya	mehmet.kaya@gmail.com	1	2023-02-20
3	2	103	Ali	Demir	ali.demir@hotmail.com	0	2023-03-10
4	2	104	Elif	Çelik	elif.celik@gmail.com	1	2023-04-05
5	1	105	Ayşe	Duru	ayse.duru@hotmail.com	1	2023-05-14
6	2	106	Can	Yıldız	can.yildiz@gmail.com	1	2023-06-01
7	1	107	Fatma	Aslan	fatma.aslan@hotmail.com	1	2023-06-18
8	1	108	Veli	Çetin	veli.cetin@gmail.com	1	2023-07-10
9	2	109	Zeynep	Öztürk	zeynep.ozturk@hotmail.com	1	2023-07-25
10	1	110	Murat	Kurt	murat.kurt@gmail.com	0	2023-08-05
11	2	111	Büşra	Güzel	busra.guzel@hotmail.com	1	2023-08-20
12	1	112	Seda	Eren	seda.eren@gmail.com	1	2023-09-01
13	1	113	Burak	Sarı	burak.sari@hotmail.com	0	2023-09-10
14	2	114	Okan	Polat	okan.polat@gmail.com	1	2023-09-25
15	2	115	Aylin	Kara	aylin.kara@hotmail.com	1	2023-10-01
16	1	116	Emre	Büyükgüngör	emre.buyukgungor@gmail.com	1	2023-10-15
17	1	117	Derya	Yılmazer	derya.yilmazer@hotmail.com	1	2023-10-20
18	2	118	Burcu	Arslan	burcu.arslan@gmail.com	1	2023-11-01
19	1	119	Onur	Özdemir	onur.ozdemir@hotmail.com	1	2023-11-10

...

26	1	126	Merve	Sönmez	merve.sonmez@gmail.com	1	2024-01-01
27	1	127	Nihal	Bölükbaşı	nihal.bolukbasi@hotmail.com	1	2024-01-10
28	2	128	Tuncay	Erdoğan	tuncay.erdogan@gmail.com	1	2024-01-20
29	2	129	Serap	Tosun	serap.tosun@hotmail.com	1	2024-01-25
30	1	130	Yusuf	Taş	yusuf.tas@gmail.com	1	2024-02-01
31	1	131	Aylin	Demirtaş	aylin.demirtas@hotmail.com	1	2024-02-10
32	2	132	Sibel	Çiçek	sibel.cicek@gmail.com	1	2024-02-20
33	2	132	Zafer	Mert	zafer.mert@hotmail.com	1	2024-02-25
34	1	131	Elif	Köksal	elif.koksal@gmail.com	1	2024-03-01
35	1	127	Vildan	Tuna	vildan.tuna@hotmail.com	1	2024-03-10
36	2	120	Feyza	Yaralı	feyza.yarali@gmail.com	1	2024-03-15
37	2	101	Ozan	Şahin	ozan.sahin@hotmail.com	1	2024-03-20

6. Film Table

```
SELECT * FROM `FILM`
```

Film_ID	Title
1	The Godfather
2	The Dark Knight
3	The Lord of the Rings: The Return of the King
4	Before Sunrise
5	Catch Me If You Can
6	Me Before You
7	Tenet
8	IT
9	The Silence of the Lambs
10	Harry Potter and the Order of the Phoenix
11	Spiderman: Homecoming
12	Jigsaw X

7. Film_Actor Table

```
SELECT * FROM `FILM_ACTOR`
```

Film_ID	Actor_ID
1	1
1	2
2	3
2	4
3	5
4	6
4	7
5	8
2	9
10	9
5	10
6	11
6	12
7	13
2	14
7	14
8	15
9	16
10	17
10	18
11	19
11	20
12	21

8. Film_Category Table

```
SELECT * FROM `FILM_CATEGORY`
```

Film_ID	Category_ID
1	1
1	3
1	12
2	1
2	5
2	7
3	9
3	10
4	2
4	6
5	1
5	2
5	10
6	3
6	6
7	1
7	5
7	10
7	12
8	4
8	7
9	4
9	7
10	3
10	9

9. Inventory Table

```
SELECT * FROM `INVENTORY`
```

Inventory_ID	Film_ID	Store_ID
1	1	1
2	2	1
3	3	1
4	3	2
5	4	2
6	5	1
7	5	2
8	6	2
9	7	1
10	7	2
11	8	2
12	9	1
13	10	2
14	11	1
15	11	2
16	12	2

10. Payment Table

```
SELECT * FROM `PAYMENT`
```

Payment_ID	Customer_ID	Staff_ID	Rental_ID	Payment_Date	Amount
1	1	1	1	2024-01-05	104.99
2	2	2	2	2024-01-14	112.50
3	3	1	3	2024-01-26	176.50
4	4	1	4	2024-02-03	65.00
5	5	2	5	2024-02-17	158.50
6	6	2	6	2024-02-28	102.50
7	7	3	7	2024-02-28	134.99
8	8	3	8	2024-03-07	195.00
9	9	4	9	2024-03-15	88.40
10	10	4	10	2024-03-29	149.99
11	11	5	11	2024-03-29	171.50
12	12	6	12	2024-04-02	127.50
13	13	6	13	2024-04-27	105.00
14	14	5	14	2024-05-05	183.50
15	15	7	15	2024-05-30	92.50
16	15	4	16	2024-06-09	67.50
17	16	4	17	2024-06-18	163.49
18	17	5	18	2024-06-26	122.50
19	18	6	19	2024-07-07	110.00
20	18	6	20	2024-07-14	148.50
21	19	7	21	2024-07-29	94.99

...

49	19	3	49	2024-08-13	178.90
50	21	4	50	2024-09-01	162.25
51	3	2	51	2024-04-25	110.50
52	26	6	52	2024-02-03	140.99
53	30	1	53	2024-07-12	190.00
54	15	9	54	2024-06-11	120.80
55	9	7	55	2024-03-07	145.20
56	11	5	56	2024-01-12	135.65
57	17	8	57	2024-05-14	155.50

11. Rental Table

```
SELECT * FROM `RENTAL`
```

Rental_ID	Inventory_ID	Customer_ID	Staff_ID	Return_Date	Rental_Date
1	1	1	1	2024-01-15	2024-01-05
2	4	2	2	2024-01-17	2024-01-14
3	3	3	1	2024-02-19	2024-01-26
4	6	4	1	2024-02-15	2024-02-03
5	4	5	2	2024-03-15	2024-02-17
6	5	6	2	2024-03-05	2024-02-28
7	1	7	3	2024-03-10	2024-02-28
8	2	8	3	2024-03-14	2024-03-07
9	5	9	4	2024-03-21	2024-03-15
10	5	10	4	2024-04-07	2024-03-29
11	2	11	5	2024-04-11	2024-03-29
12	7	12	6	2024-04-09	2024-04-02
13	1	13	5	2024-05-02	2024-04-27
14	7	14	6	2024-05-11	2024-05-05
15	2	15	7	2024-06-18	2024-05-30
16	8	15	4	2024-06-17	2024-06-09
17	1	16	5	2024-06-21	2024-06-18
18	8	17	4	2024-06-28	2024-06-26
19	3	18	5	2024-07-16	2024-07-07

...

42	14	37	7	2024-12-30	2024-12-27
43	13	2	6	2024-06-18	2024-05-21
44	5	8	3	2024-02-05	2024-01-02
45	7	33	8	2024-05-01	2024-04-09
46	4	25	2	2024-03-12	2024-02-26
47	6	14	9	2024-07-30	2024-07-03
48	1	18	5	2024-07-14	2024-06-23
49	16	19	3	2024-09-12	2024-08-13
50	8	21	4	2024-09-15	2024-09-01
51	14	3	2	2024-05-14	2024-04-25
52	7	26	6	2024-02-28	2024-02-03
53	10	30	1	2024-08-05	2024-07-12
54	12	15	9	2024-06-25	2024-06-11
55	4	9	7	2024-04-01	2024-03-07
56	3	11	5	2024-02-10	2024-01-12
57	6	17	8	2024-06-10	2024-05-14

12. Staff Table

```
SELECT * FROM `STAFF`
```

Staff_ID	Staff_Name	Staff_Surname	Store_ID	Email	Username
1	Ali	Özdemir	1	ali.özdemir@dvdrental.com	aliozdmr
2	Melisa	Çepni	2	melisa.cepni@dvdrental.com	melisacepni
3	Esra	Gönen	1	esra.gonen@dvdrental.com	esragonen
4	Ayşe	Aktaş	2	ayse.aktas@dvdrental.com	ayseaktas
5	Büşra	Gökçen	1	busra.gokcen@dvdrental.com	busragkcn
6	Furkan	Karatoprak	2	furkan.karatoprak@dvdrental.com	frknkrtprk
7	Hande	Çepni	1	hande.cepni@dvdrental.com	handecepni
8	Aişe	Uygunol	2	aise.uygunol@dvdrental.com	aiseygnl
9	Düzgün	Bulut	1	düzgün.bulut@dvdrental.com	duzgunblt

13. Store Table

```
SELECT * FROM `STORE`
```

Store_ID	Manager_Staff_ID
1	1
2	2

6. Queries

We have 19 queries in total. While writing each query, we used the JOIN keyword at least once, leveraging the primary-foreign key relationships between tables to obtain desired information from different tables and generate outputs. Additionally, by using the GROUP BY keyword, we organized the outputs into specific groups. Moreover, we used ORDER BY to sort numerical data in the output. We applied several aggregate functions such as SUM, COUNT, MIN, and MAX to gain an overall perspective. When writing some queries, we used the WHERE keyword to retrieve specific information.

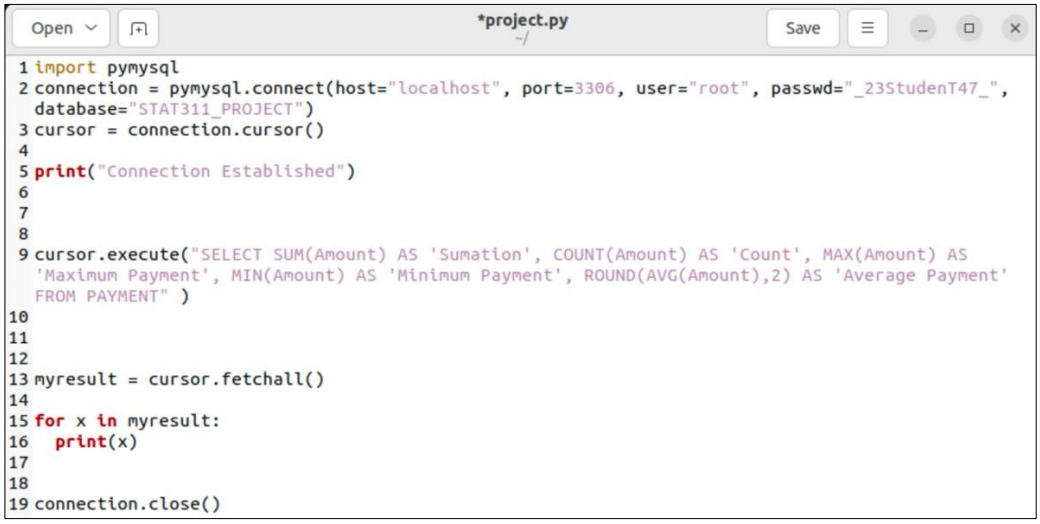
1. Summary statistics of Payment

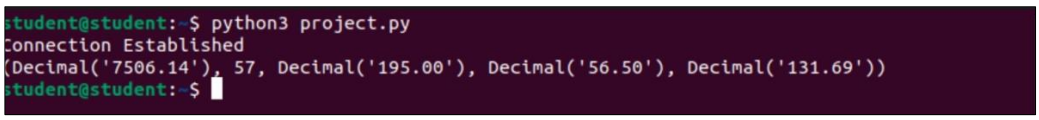
Before writing queries to answer our questions, we first created a query using aggregate functions (COUNT, MAX, MIN, AVG) to obtain summary statistics of the payments in our DVD rental stores. This helped us better understand the Payment records, as our main focus is on the sales of movies in our stores.

```sql
SELECT SUM(Amount) AS 'Sumation',
COUNT(Amount) AS 'Count',
MAX(Amount) AS 'Maximum Payment',
MIN(Amount) AS 'Minimum Payment',
ROUND(AVG(Amount),2) AS 'Average Payment' FROM PAYMENT;
```

Sumation	Count	Maximum Payment	Minimum Payment	Average Payment
7506.14	57	195.00	56.50	131.69

In the table, which is given above, we can see that our average payment is 131.69.





2. What is the total amount of spending by each customer?

We wrote a query to show the total amount of spending by each customer in our DVD Rental stores.

```
SELECT c.First_Name, c.Last_Name, sum(p.Amount) as Total_Amount
FROM CUSTOMER c
JOIN PAYMENT p on c.Customer_ID=p.Customer_ID
GROUP BY c.First_Name, c.Last_Name ORDER BY Total_Amount DESC;
```

First_Name	Last_Name	Total_Amount ▾ 1
Berk	Karaca	381.00
Burcu	Arslan	359.10
Tolga	Aydın	342.25
Veli	Çetin	340.99
Okan	Polat	338.90
Zafer	Mert	323.80
Büşra	Güzel	307.15
Merve	Sönmez	297.49
Ali	Demir	287.00
Aylin	Kara	280.80
Derya	Yılmazer	278.00
Onur	Özdemir	273.89
Yusuf	Taş	259.99
Mehmet	Kaya	235.00
Zeynep	Öztürk	233.60
Aylin	Demirtaş	221.49
Elif	Köksal	206.00
Serap	Tosun	191.50
Feyza	Yaralı	185.00

Open *project.py ~/ Save

```
1 import pymysql
2 connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
3 cursor = connection.cursor()
4
5 print("Connection Established")
6
7
8 import pymysql
9 connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
10 cursor = connection.cursor()
11
12 print("Connection Established")
13
14
15
16
17 cursor.execute("SELECT c.First_Name, c.Last_Name, sum(p.Amount) as Total_Amount FROM CUSTOMER c
   JOIN PAYMENT p on c.Customer_ID=p.Customer_ID GROUP BY c.First_Name, c.Last_Name ORDER BY
   Total_Amount DESC;")
18
19
20 myresult = cursor.fetchall()
21
22 for x in myresult:
23   print(x)
24
25
26 myresult2 = cursor.fetchall()
27
28
29
30 connection.close()
```

```
student@student:~$ python3 project.py
Connection Established
Connection Established
('Berk', 'Karaca', Decimal('381.00'))
('Burcu', 'Arslan', Decimal('359.10'))
('Tolga', 'Aydın', Decimal('342.25'))
('Veli', 'Çetin', Decimal('340.99'))
('Okan', 'Polat', Decimal('338.90'))
('Zafer', 'Mert', Decimal('323.80'))
('Büşra', 'Güzel', Decimal('307.15'))
('Merve', 'Sönmez', Decimal('297.49'))
('Ali', 'Demir', Decimal('287.00'))
('Aylin', 'Kara', Decimal('280.80'))
('Derya', 'Yılmazer', Decimal('278.00'))
('Onur', 'Özdemir', Decimal('273.89'))
('Yusuf', 'Taş', Decimal('259.99'))
('Mehmet', 'Kaya', Decimal('235.00'))
('Zeynep', 'Öztürk', Decimal('233.60'))
('Aylin', 'Demirtaş', Decimal('221.49'))
('Elif', 'Köksal', Decimal('206.00'))
('Serap', 'Tosun', Decimal('191.50'))
('Feyza', 'Yaralı', Decimal('185.00'))
('Hüseyin', 'Güler', Decimal('183.25'))
('Deniz', 'Kucuk', Decimal('176.50'))
('Nihal', 'Bölükbaşı', Decimal('165.00'))
('Emre', 'Büyükgüngör', Decimal('163.49'))
('Ayşe', 'Duru', Decimal('158.50'))
('Sibel', 'Çiçek', Decimal('150.00'))
('Murat', 'Kurt', Decimal('149.99'))
('Fatma', 'Aslan', Decimal('134.99'))
('Seda', 'Eren', Decimal('127.50'))
```

3. Which customer was served by which staff at which store?

We wrote a query to show which customer was served by which staff at which store. This provided us with a comprehensive overview of our sales.

```
SELECT c.First_Name, c.Last_Name, s.Store_ID , st.Staff_Name,
st.Staff_Surname
FROM CUSTOMER c
JOIN STORE s ON c.Store_ID=s.Store_ID
JOIN STAFF st ON s.Store_ID=st.Store_ID
```

First_Name	Last_Name	Store_ID	Staff_Name	Staff_Surname
Ahmet	Yılmaz	1	Ali	Özdemir
Mehmet	Kaya	1	Ali	Özdemir
Ayşe	Duru	1	Ali	Özdemir
Fatma	Aslan	1	Ali	Özdemir
Veli	Çetin	1	Ali	Özdemir
Murat	Kurt	1	Ali	Özdemir
Seda	Eren	1	Ali	Özdemir

...

Ahmet	Yılmaz	1	Esra	Gönen
Mehmet	Kaya	1	Esra	Gönen
Ayşe	Duru	1	Esra	Gönen
Fatma	Aslan	1	Esra	Gönen
Veli	Çetin	1	Esra	Gönen

...

Tuncay	Erdoğan	2	Melisa	Çepni
Serap	Tosun	2	Melisa	Çepni
Sibel	Çiçek	2	Melisa	Çepni
Zafer	Mert	2	Melisa	Çepni
Feyza	Yaralı	2	Melisa	Çepni
Ozan	Şahin	2	Melisa	Çepni
Ali	Demir	2	Ayşe	Aktaş
Elif	Çelik	2	Ayşe	Aktaş
Can	Yıldız	2	Ayşe	Aktaş
Zeynep	Öztürk	2	Ayşe	Aktaş
Büşra	Güzel	2	Ayşe	Aktaş

Open project.py Save

```python
import pymysql
connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
cursor = connection.cursor()


import pymysql
connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
cursor = connection.cursor()

print("Connection Established")


cursor.execute("SELECT c.First_Name, c.Last_Name, s.Store_ID , st.Staff_Name, st.Staff_Surname
  FROM CUSTOMER c  JOIN STORE s on c.Store_ID=s.Store_ID JOIN STAFF st on s.Store_ID=st.Store_ID")


myresult = cursor.fetchall()

for x in myresult:
  print(x)


myresult2 = cursor.fetchall()



connection.close()
```

```
student@student:~$ python3 project.py
Connection Established
('Ahmet', 'Yılmaz', 1, 'Ali', 'Özdemir')
('Mehmet', 'Kaya', 1, 'Ali', 'Özdemir')
('Ayşe', 'Duru', 1, 'Ali', 'Özdemir')
('Fatma', 'Aslan', 1, 'Ali', 'Özdemir')
('Veli', 'Çetin', 1, 'Ali', 'Özdemir')
('Murat', 'Kurt', 1, 'Ali', 'Özdemir')
('Seda', 'Eren', 1, 'Ali', 'Özdemir')
('Burak', 'Sarı', 1, 'Ali', 'Özdemir')
('Emre', 'Büyükgüngör', 1, 'Ali', 'Özdemir')
('Derya', 'Yılmazer', 1, 'Ali', 'Özdemir')
('Onur', 'Özdemir', 1, 'Ali', 'Özdemir')
('Selin', 'Yıldırım', 1, 'Ali', 'Özdemir')
('Kemal', 'Tuna', 1, 'Ali', 'Özdemir')
('Deniz', 'Kucuk', 1, 'Ali', 'Özdemir')
('Merve', 'Sönmez', 1, 'Ali', 'Özdemir')
('Nihal', 'Bölükbaşı', 1, 'Ali', 'Özdemir')
('Yusuf', 'Taş', 1, 'Ali', 'Özdemir')
('Aylin', 'Demirtaş', 1, 'Ali', 'Özdemir')
('Elif', 'Köksal', 1, 'Ali', 'Özdemir')
('Vildan', 'Tuna', 1, 'Ali', 'Özdemir')
('Ahmet', 'Yılmaz', 1, 'Esra', 'Gönen')
('Mehmet', 'Kaya', 1, 'Esra', 'Gönen')
('Ayşe', 'Duru', 1, 'Esra', 'Gönen')
('Fatma', 'Aslan', 1, 'Esra', 'Gönen')
('Veli', 'Çetin', 1, 'Esra', 'Gönen')
('Murat', 'Kurt', 1, 'Esra', 'Gönen')
('Seda', 'Eren', 1, 'Esra', 'Gönen')
('Burak', 'Sarı', 1, 'Esra', 'Gönen')
('Emre', 'Büyükgüngör', 1, 'Esra', 'Gönen')
```

4. Which movies were acted in by actors whose names start with ‘T’?

With this query we can see the movies played by actors whose names start with any letter we want. As an example, we choose the letter ‘T’.

```
SELECT a.Actor_Name, a.Actor_Surname, f.Title FROM ACTOR a
JOIN FILM_ACTOR fa ON a.Actor_ID = fa.Actor_ID
JOIN FILM f ON fa.Film_ID = f.Film_ID
WHERE a.Actor_Name LIKE 'T%';
```

Actor_Name	Actor_Surname	Title
Tom	Hanks	Catch Me If You Can
Tom	Holland	Spiderman: Homecoming
Tobin	Bell	Jigsaw X

Open project.py Save

```
import pymysql
connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
cursor = connection.cursor()


import pymysql
connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
cursor = connection.cursor()

print("Connection Established")


cursor.execute("SELECT a.Actor_Name, a.Actor_Surname, f.Title FROM ACTOR a JOIN FILM_ACTOR fa ON
  a.Actor_ID = fa.Actor_ID JOIN FILM f ON fa.Film_ID = f.Film_ID WHERE a.Actor_Name LIKE 'T%';")


myresult = cursor.fetchall()

for x in myresult:
  print(x)


myresult2 = cursor.fetchall()



connection.close()
```

```
student@student:~$ python3 project.py
Connection Established
('Tom', 'Hanks', 'Catch Me If You Can')
('Tom', 'Holland', 'Spiderman: Homecoming')
('Tobin', 'Bell', 'Jigsaw X')
```

5. Which customers rented ‘The Godfather’ movie?

This query shows which customers rented a specific movie. We chose ‘The Godfather’ as an example.

```sql
SELECT DISTINCT c.First_Name, c.Last_Name, f.Title AS Rented_Film
FROM CUSTOMER c
JOIN RENTAL r ON c.Customer_ID = r.Customer_ID
JOIN INVENTORY i ON r.Inventory_ID = i.Inventory_ID
JOIN FILM f ON i.Film_ID = f.Film_ID
WHERE f.Title = 'The Godfather';
```

First_Name	Last_Name	Rented_Film
Ahmet	Yılmaz	The Godfather
Fatma	Aslan	The Godfather
Burak	Sarı	The Godfather
Emre	Büyükgüngör	The Godfather
Burcu	Arslan	The Godfather

project.py

```python
import pymysql
connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_", database="STAT311_PROJECT")
cursor = connection.cursor()


import pymysql
connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_", database="STAT311_PROJECT")
cursor = connection.cursor()

print("Connection Established")


cursor.execute("SELECT DISTINCT c.First_Name, c.Last_Name, f.Title AS Rented_Film FROM CUSTOMER c JOIN RENTAL r ON c.Customer_ID = r.Customer_ID JOIN INVENTORY i ON r.Inventory_ID = i.Inventory_ID JOIN FILM f ON i.Film_ID = f.Film_ID WHERE f.Title = 'The Godfather';")


myresult = cursor.fetchall()

for x in myresult:
  print(x)


myresult2 = cursor.fetchall()



connection.close()
```

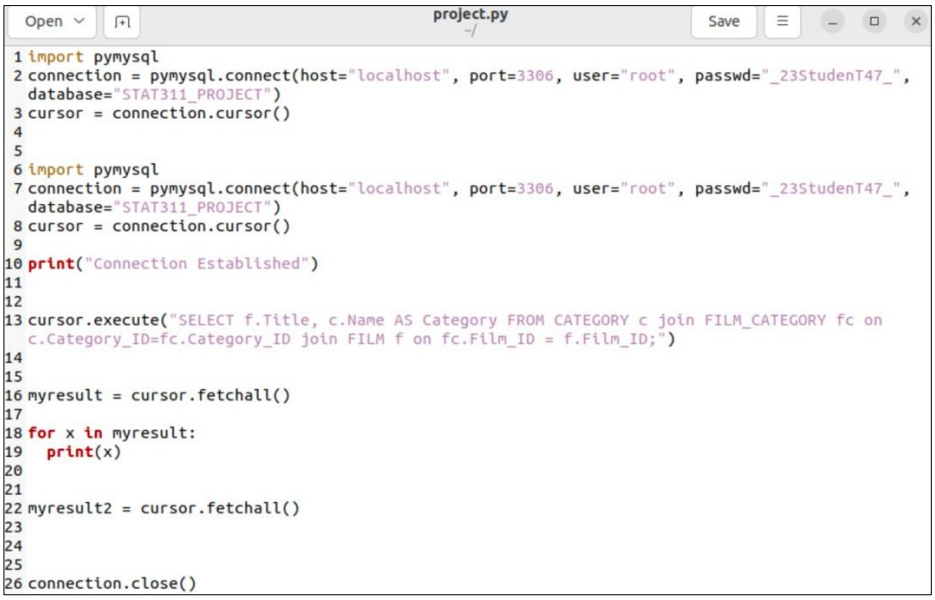
```
student@student:~$ python3 project.py
Connection Established
('Ahmet', 'Yılmaz', 'The Godfather')
('Fatma', 'Aslan', 'The Godfather')
('Burak', 'Sarı', 'The Godfather')
('Emre', 'Büyükgüngör', 'The Godfather')
('Burcu', 'Arslan', 'The Godfather')
```

6. What are the categories of each movie?

This query helps us to see which categories the movies belong to.

```sql
SELECT f.Title, c.Name AS Category FROM CATEGORY c
JOIN FILM_CATEGORY fc ON c.Category_ID=fc.Category_ID
JOIN FILM f ON fc.Film_ID = f.Film_ID;
```

Title	Category
The Godfather	Action
The Godfather	Drama
The Godfather	Mystery
The Dark Knight	Action
The Dark Knight	Sci-Fi
The Dark Knight	Thriller
The Lord of the Rings: The Return of the King	Fantasy
The Lord of the Rings: The Return of the King	Adventure
Before Sunrise	Comedy
Before Sunrise	Romance
Catch Me If You Can	Action
Catch Me If You Can	Comedy
Catch Me If You Can	Adventure
Me Before You	Drama



```
student@student:~$ python3 project.py
Connection Established
('The Godfather', 'Action')
('The Godfather', 'Drama')
('The Godfather', 'Mystery')
('The Dark Knight', 'Action')
('The Dark Knight', 'Sci-Fi')
('The Dark Knight', 'Thriller')
('The Lord of the Rings: The Return of the King', 'Fantasy')
('The Lord of the Rings: The Return of the King', 'Adventure')
('Before Sunrise', 'Comedy')
('Before Sunrise', 'Romance')
('Catch Me If You Can', 'Action')
('Catch Me If You Can', 'Comedy')
('Catch Me If You Can', 'Adventure')
('Me Before You', 'Drama')
('Me Before You', 'Romance')
('Tenet', 'Action')
('Tenet', 'Sci-Fi')
('Tenet', 'Adventure')
('Tenet', 'Mystery')
('IT', 'Horror')
('IT', 'Thriller')
('The Silence of the Lambs', 'Horror')
('The Silence of the Lambs', 'Thriller')
('Harry Potter and the Order of the Phoenix', 'Drama')
('Harry Potter and the Order of the Phoenix', 'Fantasy')
('Harry Potter and the Order of the Phoenix', 'Adventure')
('Spiderman: Homecoming', 'Action')
('Spiderman: Homecoming', 'Comedy')
```

7. Which movie was rented the most by our customers?

We wrote this query to find out which movie was rented the most. This allows us to understand which movie is the most popular in our stores.

```sql
SELECT f.Title AS Film_Title, COUNT(r.Rental_ID) AS Rental_Count
FROM FILM f
JOIN INVENTORY i ON f.Film_ID = i.Film_ID
JOIN RENTAL r ON i.Inventory_ID = r.Inventory_ID
GROUP BY f.Title
ORDER BY Rental_Count DESC
LIMIT 1;
```

Film_Title	Rental_Count
The Lord of the Rings: The Return of the King	10

Open *project.py Save

```python
import pymysql
connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_", database="STAT311_PROJECT")
cursor = connection.cursor()


import pymysql
connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_", database="STAT311_PROJECT")
cursor = connection.cursor()

print("Connection Established")


cursor.execute("SELECT f.Title AS Film_Title, COUNT(r.Rental_ID) AS Rental_Count FROM FILM f JOIN INVENTORY i ON f.Film_ID = i.Film_ID JOIN RENTAL r ON i.Inventory_ID = r.Inventory_ID GROUP BY f.Title ORDER BY Rental_Count DESC LIMIT 1;")


myresult = cursor.fetchall()

for x in myresult:
  print(x)


myresult2 = cursor.fetchall()



connection.close()
```

```
student@student:~$ python3 project.py
Connection Established
('The Lord of the Rings: The Return of the King', 10)
```

8. Which customer rented which movie?

With this query, we can see which customer rented which movie/s.

```sql
SELECT c.First_Name, c.Last_Name, f.Title
FROM INVENTORY i
JOIN RENTAL r ON i.Inventory_ID=r.Inventory_ID
JOIN CUSTOMER c ON r.Customer_ID=c.Customer_ID
JOIN FILM f ON i.Film_ID=f.Film_ID
```

First_Name	Last_Name	Title
Berk	Karaca	Catch Me If You Can
Okan	Polat	Catch Me If You Can
Derya	Yılmazer	Catch Me If You Can
Seda	Eren	Catch Me If You Can
Okan	Polat	Catch Me If You Can
Zafer	Mert	Catch Me If You Can
Merve	Sönmez	Catch Me If You Can
Aylin	Kara	Me Before You
Derya	Yılmazer	Me Before You
Burcu	Arslan	Me Before You
Hüseyin	Güler	Me Before You
Tolga	Aydın	Me Before You
Berk	Karaca	Tenet
Deniz	Kucuk	Tenet
Merve	Sönmez	Tenet
Yusuf	Taş	Tenet
Tuncay	Erdoğan	IT
Nihal	Bölükbaşı	The Silence of the Lambs
Serap	Tosun	The Silence of the Lambs
Yusuf	Taş	The Silence of the Lambs

Open project.py ~/ Save

```
1 import pymysql
2 connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
3 cursor = connection.cursor()
4
5
6 import pymysql
7 connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
8 cursor = connection.cursor()
9
10 print("Connection Established")
11
12
13 cursor.execute("SELECT c.First_Name, c.Last_Name, f.Title FROM INVENTORY i JOIN RENTAL r on
  i.Inventory_ID=r.Inventory_ID JOIN CUSTOMER c on r.Customer_ID=c.Customer_ID JOIN FILM f on
  i.Film_ID=f.Film_ID")
14
15
16 myresult = cursor.fetchall()
17
18 for x in myresult:
19   print(x)
20
21
22 myresult2 = cursor.fetchall()
23
24
25
26 connection.close()
```

```
student@student:~$ python3 project.py
Connection Established
('Ahmet', 'Yılmaz', 'The Godfather')
('Fatma', 'Aslan', 'The Godfather')
('Burak', 'Sarı', 'The Godfather')
('Emre', 'BÜyükgüngör', 'The Godfather')
('Burcu', 'Arslan', 'The Godfather')
('Veli', 'Çetin', 'The Dark Knight')
('Büşra', 'Güzel', 'The Dark Knight')
('Aylin', 'Kara', 'The Dark Knight')
('Ali', 'Demir', 'The Lord of the Rings: The Return of the King')
('Burcu', 'Arslan', 'The Lord of the Rings: The Return of the King')
('Onur', 'Özdemir', 'The Lord of the Rings: The Return of the King')
('Selin', 'Yıldırım', 'The Lord of the Rings: The Return of the King')
('Tolga', 'Aydın', 'The Lord of the Rings: The Return of the King')
('Büşra', 'Güzel', 'The Lord of the Rings: The Return of the King')
('Mehmet', 'Kaya', 'The Lord of the Rings: The Return of the King')
('Ayşe', 'Duru', 'The Lord of the Rings: The Return of the King')
('Hüseyin', 'Güler', 'The Lord of the Rings: The Return of the King')
('Zeynep', 'Öztürk', 'The Lord of the Rings: The Return of the King')
('Can', 'Yıldız', 'Before Sunrise')
('Zeynep', 'Öztürk', 'Before Sunrise')
('Murat', 'Kurt', 'Before Sunrise')
('Veli', 'Çetin', 'Before Sunrise')
('Elif', 'Çelik', 'Catch Me If You Can')
('Kemal', 'Tuna', 'Catch Me If You Can')
('Berk', 'Karaca', 'Catch Me If You Can')
('Berk', 'Karaca', 'Catch Me If You Can')
('Okan', 'Polat', 'Catch Me If You Can')
('Derya', 'Yılmazer', 'Catch Me If You Can')
```

9. How many times was each movie rented by customers?

With this query, we can see how many times each movie was rented. This allows us to understand which movie is more popular and rented more, or which one was preferred less compared to the others.

```sql
SELECT f.Title, COUNT(r.Inventory_ID) AS 'Rented Count'
FROM INVENTORY i
JOIN RENTAL r ON i.Inventory_ID = r.Inventory_ID
JOIN FILM f ON i.Film_ID=f.Film_ID
GROUP BY f.Title
```

Title	Rented Count
The Godfather	5
The Dark Knight	3
The Lord of the Rings: The Return of the King	10
Before Sunrise	4
Catch Me If You Can	10
Me Before You	5
Tenet	4
IT	1
The Silence of the Lambs	5
Harry Potter and the Order of the Phoenix	2
Spiderman: Homecoming	6
Jigsaw X	2

Open ▾ project.py ~/ Save

```
1 import pymysql
2 connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
3 cursor = connection.cursor()
4
5
6 import pymysql
7 connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
8 cursor = connection.cursor()
9
10 print("Connection Established")
11
12
13 cursor.execute("SELECT f.Title, COUNT(r.Inventory_ID) as 'Rented Count' FROM INVENTORY i join
  RENTAL r on i.Inventory_ID = r.Inventory_ID JOIN FILM f on i.Film_ID=f.Film_ID GROUP BY
  f.Title")
14
15
16 myresult = cursor.fetchall()
17
18 for x in myresult:
19   print(x)
20
21
22 myresult2 = cursor.fetchall()
23
24
25
26 connection.close()
```

```
student@student:~$ python3 project.py
Connection Established
('The Godfather', 5)
('The Dark Knight', 3)
('The Lord of the Rings: The Return of the King', 10)
('Before Sunrise', 4)
('Catch Me If You Can', 10)
('Me Before You', 5)
('Tenet', 4)
('IT', 1)
('The Silence of the Lambs', 5)
('Harry Potter and the Order of the Phoenix', 2)
('Spiderman: Homecoming', 6)
('Jigsaw X', 2)
```

10. What are the details of the movies rented in 2024?

With this query, we can see the details of the movies rented by year. We chose the current year, 2024, to get the most up-to-date details.

```sql
SELECT r.Rental_ID, f.Title AS Film_Title, c.First_Name,
c.Last_Name, r.Rental_Date, r.Return_Date
FROM RENTAL r
JOIN INVENTORY i ON r.Inventory_ID = i.Inventory_ID
JOIN FILM f ON i.Film_ID = f.Film_ID
JOIN CUSTOMER c ON r.Customer_ID = c.Customer_ID
WHERE r.Rental_Date BETWEEN '2024-01-01' AND '2024-12-31';
```

Rental_ID	Film_Title	First_Name	Last_Name	Rental_Date	Return_Date
1	The Godfather	Ahmet	Yılmaz	2024-01-05	2024-01-15
2	The Lord of the Rings: The Return of the King	Mehmet	Kaya	2024-01-14	2024-01-17
3	The Lord of the Rings: The Return of the King	Ali	Demir	2024-01-26	2024-02-19
4	Catch Me If You Can	Elif	Çelik	2024-02-03	2024-02-15
5	The Lord of the Rings: The Return of the King	Ayşe	Duru	2024-02-17	2024-03-15
6	Before Sunrise	Can	Yıldız	2024-02-28	2024-03-05
7	The Godfather	Fatma	Aslan	2024-02-28	2024-03-10
8	The Dark Knight	Veli	Çetin	2024-03-07	2024-03-14
9	Before Sunrise	Zeynep	Öztürk	2024-03-15	2024-03-21
10	Before Sunrise	Murat	Kurt	2024-03-29	2024-04-07
11	The Dark Knight	Büşra	Güzel	2024-03-29	2024-04-11
12	Catch Me If You Can	Seda	Eren	2024-04-02	2024-04-09
13	The Godfather	Burak	Sarı	2024-04-27	2024-05-02
14	Catch Me If You Can	Okan	Polat	2024-05-05	2024-05-11
15	The Dark Knight	Aylin	Kara	2024-05-30	2024-06-18
16	Me Before You	Aylin	Kara	2024-06-09	2024-06-17
17	The Godfather	Emre	Büyükgüngör	2024-06-18	2024-06-21
18	Me Before You	Derya	Yılmazer	2024-06-26	2024-06-28
19	The Lord of the Rings: The Return of the King	Burcu	Arslan	2024-07-07	2024-07-16
20	Me Before You	Burcu	Arslan	2024-07-14	2024-07-19

Open *project.py Save

```python
import pymysql
connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
cursor = connection.cursor()


import pymysql
connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
cursor = connection.cursor()

print("Connection Established")


cursor.execute("SELECT r.Rental_ID, f.Title AS Film_Title, c.First_Name, c.Last_Name,
  r.Rental_Date, r.Return_Date FROM RENTAL r JOIN INVENTORY i ON r.Inventory_ID = i.Inventory_ID
  JOIN FILM f ON i.Film_ID = f.Film_ID JOIN CUSTOMER c ON r.Customer_ID = c.Customer_ID WHERE
  r.Rental_Date BETWEEN '2024-01-01' AND '2024-12-31';")


myresult = cursor.fetchall()

for x in myresult:
  print(x)

myresult2 = cursor.fetchall()



connection.close()
```

```
student@student:~$ python3 project.py
Connection Established
(1, 'The Godfather', 'Ahmet', 'Yılmaz', datetime.date(2024, 1, 5), datetime.date(2024, 1, 15))
(2, 'The Lord of the Rings: The Return of the King', 'Mehmet', 'Kaya', datetime.date(2024, 1, 14), datetime.date(2024, 1, 17))
(3, 'The Lord of the Rings: The Return of the King', 'Ali', 'Demir', datetime.date(2024, 1, 26), datetime.date(2024, 2, 19))
(4, 'Catch Me If You Can', 'Elif', 'Çelik', datetime.date(2024, 2, 3), datetime.date(2024, 2, 15))
(5, 'The Lord of the Rings: The Return of the King', 'Ayşe', 'Duru', datetime.date(2024, 2, 17), datetime.date(2024, 3, 15))
(6, 'Before Sunrise', 'Can', 'Yıldız', datetime.date(2024, 2, 28), datetime.date(2024, 3, 5))
(7, 'The Godfather', 'Fatma', 'Aslan', datetime.date(2024, 2, 28), datetime.date(2024, 3, 10))
(8, 'The Dark Knight', 'Veli', 'Çetin', datetime.date(2024, 3, 7), datetime.date(2024, 3, 14))
(9, 'Before Sunrise', 'Zeynep', 'Öztürk', datetime.date(2024, 3, 15), datetime.date(2024, 3, 21))
(10, 'Before Sunrise', 'Murat', 'Kurt', datetime.date(2024, 3, 29), datetime.date(2024, 4, 7))
(11, 'The Dark Knight', 'Büşra', 'Güzel', datetime.date(2024, 3, 29), datetime.date(2024, 4, 11))
(12, 'Catch Me If You Can', 'Seda', 'Eren', datetime.date(2024, 4, 2), datetime.date(2024, 4, 9))
(13, 'The Godfather', 'Burak', 'Sarı', datetime.date(2024, 4, 27), datetime.date(2024, 5, 2))
(14, 'Catch Me If You Can', 'Okan', 'Polat', datetime.date(2024, 5, 5), datetime.date(2024, 5, 11))
(15, 'The Dark Knight', 'Aylin', 'Kara', datetime.date(2024, 5, 30), datetime.date(2024, 6, 18))
(16, 'Me Before You', 'Aylin', 'Kara', datetime.date(2024, 6, 9), datetime.date(2024, 6, 17))
(17, 'The Godfather', 'Emre', 'Büyükgüngör', datetime.date(2024, 6, 18), datetime.date(2024, 6, 21))
(18, 'Me Before You', 'Derya', 'Yılmazer', datetime.date(2024, 6, 26), datetime.date(2024, 6, 28))
(19, 'The Lord of the Rings: The Return of the King', 'Burcu', 'Arslan', datetime.date(2024, 7, 7), datetime.date(2024, 7, 16))
(20, 'Me Before You', 'Burcu', 'Arslan', datetime.date(2024, 7, 14), datetime.date(2024, 7, 19))
(21, 'The Lord of the Rings: The Return of the King', 'Onur', 'Özdemir', datetime.date(2024, 7, 29), datetime.date(2024, 8, 5))
(22, 'The Lord of the Rings: The Return of the King', 'Selin', 'Yıldırım', datetime.date(2024, 8, 5), datetime.date(2024, 8, 7))
(23, 'The Lord of the Rings: The Return of the King', 'Tolga', 'Aydın', datetime.date(2024, 8, 16), datetime.date(2024, 8, 23))
(24, 'Catch Me If You Can', 'Kemal', 'Tuna', datetime.date(2024, 8, 30), datetime.date(2024, 9, 4))
(25, 'Catch Me If You Can', 'Berk', 'Karaca', datetime.date(2024, 9, 3), datetime.date(2024, 9, 8))
(26, 'Catch Me If You Can', 'Berk', 'Karaca', datetime.date(2024, 9, 15), datetime.date(2024, 9, 25))
(27, 'Tenet', 'Berk', 'Karaca', datetime.date(2024, 9, 27), datetime.date(2024, 10, 2))
(28, 'Tenet', 'Deniz', 'Kucuk', datetime.date(2024, 10, 5), datetime.date(2024, 10, 10))
(29, 'Me Before You', 'Hüseyin', 'Güler', datetime.date(2024, 10, 19), datetime.date(2024, 10, 24))
(30, 'Tenet', 'Merve', 'Sönmez', datetime.date(2024, 10, 30), datetime.date(2024, 11, 5))
(31, 'The Silence of the Lambs', 'Nihal', 'Bölükbaşı', datetime.date(2024, 11, 4), datetime.date(2024, 11, 16))
(32, 'IT', 'Tuncay', 'Erdoğan', datetime.date(2024, 11, 16), datetime.date(2024, 11, 23))
(33, 'The Silence of the Lambs', 'Serap', 'Tosun', datetime.date(2024, 11, 16), datetime.date(2024, 11, 30))
(34, 'The Silence of the Lambs', 'Yusuf', 'Taş', datetime.date(2024, 11, 27), datetime.date(2024, 12, 4))
(35, 'The Silence of the Lambs', 'Aylin', 'Demirtaş', datetime.date(2024, 11, 27), datetime.date(2024, 12, 6))
(36, 'Spiderman: Homecoming', 'Aylin', 'Demirtaş', datetime.date(2024, 11, 27), datetime.date(2024, 12, 7))
(37, 'Harry Potter and the Order of the Phoenix', 'Sibel', 'Çiçek', datetime.date(2024, 12, 7), datetime.date(2024, 12, 11))
(38, 'Spiderman: Homecoming', 'Zafer', 'Mert', datetime.date(2024, 12, 7), datetime.date(2024, 12, 17))
```

11. How much sales did each staff member make, and how many sales did they complete in total?

With this query, we can observe the number of sales made by each staff member and the total payment generated from those sales. This allows us to identify the staff who made the most and the least sales. Additionally, we can see that some staff members, despite making fewer sales, generate higher payment. This gives us a better opportunity to evaluate the staff's performance.

```sql
SELECT s.Staff_Name, s.Staff_Surname,
SUM(p.Amount) AS Total_Sales_Payment,
COUNT(p.Amount) AS Total_Number_Of_Sales
FROM STAFF s
JOIN PAYMENT p ON s.Staff_ID = p.Staff_ID
GROUP BY s.Staff_ID
ORDER BY Total_Sales_Payment DESC;
```

Staff_Name	Staff_Surname	Total_Sales_Payment ▾ 1	Total_Number_Of_Sales
Hande	Çepni	1265.17	10
Büşra	Gökçen	955.25	7
Düzgün	Bulut	924.18	7
Melisa	Çepni	873.25	7
Furkan	Karatoprak	754.49	6
Ayşe	Aktaş	724.13	6
Aişe	Uygunol	704.80	5
Esra	Gönen	654.88	4
Ali	Özdemir	649.99	5

```
Open   project.py   Save
1 import pymysql
2 connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
3 cursor = connection.cursor()
4
5
6 import pymysql
7 connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
8 cursor = connection.cursor()
9
10 print("Connection Established")
11
12
13 cursor.execute("SELECT s.Staff_Name, s.Staff_Surname, SUM(p.Amount) AS Total_Sales_Payment,
  COUNT(p.Amount) AS Total_Number_Of_Sales FROM STAFF s JOIN PAYMENT p ON s.Staff_ID = p.Staff_ID
  GROUP BY s.Staff_ID ORDER BY Total_Sales_Payment DESC; ")
14
15
16 myresult = cursor.fetchall()
17
18 for x in myresult:
19   print(x)
20
21
22 myresult2 = cursor.fetchall()
23
24
25
26 connection.close()
```

```
student@student:~$ python3 project.py
Connection Established
('Hande', 'Çepni', Decimal('1265.17'), 10)
('Büşra', 'Gökçen', Decimal('955.25'), 7)
('Düzgün', 'Bulut', Decimal('924.18'), 7)
('Melisa', 'Çepni', Decimal('873.25'), 7)
('Furkan', 'Karatoprak', Decimal('754.49'), 6)
('Ayşe', 'Aktaş', Decimal('724.13'), 6)
('Aişe', 'Uygunol', Decimal('704.80'), 5)
('Esra', 'Gönen', Decimal('654.88'), 4)
('Ali', 'Özdemir', Decimal('649.99'), 5)
```

12. What is the total amount of payment of each store?

With this query, we can see the total amount of payment made in each store. This allows us to determine which store has made more sales.

```sql
SELECT c.Store_ID, SUM(p.Amount) AS Total_Sell
FROM CUSTOMER c
JOIN PAYMENT p ON p.Customer_ID = c.Customer_ID
GROUP BY c.Store_ID;
```

Store_ID	Total_Sell
1	3575.30
2	3930.84

```
Open   *project.py   Save
1 import pymysql
2 connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
3 cursor = connection.cursor()
4
5
6 import pymysql
7 connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
8 cursor = connection.cursor()
9
10 print("Connection Established")
11
12
13 cursor.execute("SELECT c.Store_ID, SUM(p.Amount) AS Total_Sell FROM CUSTOMER c JOIN PAYMENT p ON
  p.Customer_ID = c.Customer_ID GROUP BY c.Store_ID;")
14
15
16 myresult = cursor.fetchall()
17
18 for x in myresult:
19   print(x)
20
21
22 myresult2 = cursor.fetchall()
23
24
25
26 connection.close()
```

```
student@student:~$ python3 project.py
Connection Established
(1, Decimal('3575.30'))
(2, Decimal('3930.84'))
```

13. Which city has the most registered customers?

With this query, we can see which city most of our customers are registered in. Even if our stores are located in Ankara, we can see that where the customers who made purchases are from.

```
1 SELECT ci.City,
2 COUNT(c.Customer_ID) AS Total_Customers FROM CUSTOMER c
3 JOIN ADDRESS a ON c.Address_ID = a.Address_ID
4 JOIN CITY ci ON a.City_ID = ci.City_ID
5 GROUP BY ci.City_ID
6 ORDER BY Total_Customers DESC LIMIT 1;
```

City	Total_Customers
Antalya	5

```
Open   *project.py   Save
 1 import pymysql
 2 connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
   database="STAT311_PROJECT")
 3 cursor = connection.cursor()
 4
 5
 6 import pymysql
 7 connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
   database="STAT311_PROJECT")
 8 cursor = connection.cursor()
 9
10 print("Connection Established")
11
12
13 cursor.execute("SELECT ci.City, COUNT(c.Customer_ID) AS Total_Customers FROM CUSTOMER c JOIN
   ADDRESS a ON c.Address_ID = a.Address_ID JOIN CITY ci ON a.City_ID = ci.City_ID GROUP BY
   ci.City_ID ORDER BY Total_Customers DESC LIMIT 1;")
14
15
16 myresult = cursor.fetchall()
17
18 for x in myresult:
19   print(x)
20
21
22 myresult2 = cursor.fetchall()
23
24
25
26 connection.close()
```

```
student@student:~$ python3 project.py
Connection Established
('Antalya', 5)
```

14. What are the movies in the action category?

With this query, we can view the movies in any category we choose. This makes it easier to suggest or select films when customers request movies from a specific category. We used the action category as an example.

```sql
SELECT f.Title AS Film_Title, c.Name AS Category_Name
FROM FILM f
JOIN FILM_CATEGORY fc ON f.Film_ID = fc.Film_ID
JOIN CATEGORY c ON fc.Category_ID = c.Category_ID
WHERE c.Name = 'Action';
```

Film_Title	Category_Name
The Godfather	Action
The Dark Knight	Action
Catch Me If You Can	Action
Tenet	Action
Spiderman: Homecoming	Action

```
Open  *project.py  Save
1 import pymysql
2 connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
3 cursor = connection.cursor()
4
5
6 import pymysql
7 connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
8 cursor = connection.cursor()
9
10 print("Connection Established")
11
12
13 cursor.execute("SELECT f.Title AS Film_Title, c.Name AS Category_Name FROM FILM f JOIN
  FILM_CATEGORY fc ON f.Film_ID = fc.Film_ID JOIN CATEGORY c ON fc.Category_ID = c.Category_ID
  WHERE c.Name = 'Action';")
14
15
16 myresult = cursor.fetchall()
17
18 for x in myresult:
19   print(x)
20
21
22 myresult2 = cursor.fetchall()
23
24
25
26 connection.close()
```

```
student@student:~$ python3 project.py
Connection Established
('The Godfather', 'Action')
('The Dark Knight', 'Action')
('Catch Me If You Can', 'Action')
('Tenet', 'Action')
('Spiderman: Homecoming', 'Action')
```

15. Which customer rented which movie last, and what was the rental date?

With this query, we can see the details of the last rental, including which customer made it and which movie was rented. We can also determine when the last rental occurred.

```sql
SELECT c.First_Name, c.Last_Name, f.Title AS Film_Title,
r.Rental_Date
FROM RENTAL r
JOIN INVENTORY i ON r.Inventory_ID = i.Inventory_ID
JOIN FILM f ON i.Film_ID = f.Film_ID
JOIN CUSTOMER c ON r.Customer_ID = c.Customer_ID
ORDER BY r.Rental_Date DESC LIMIT 1;
```

First_Name	Last_Name	Film_Title	Rental_Date
Feyza	Yaralı	Spiderman: Homecoming	2024-12-27

```
Open   *project.py   Save
1 import pymysql
2 connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
3 cursor = connection.cursor()
4
5
6 import pymysql
7 connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
8 cursor = connection.cursor()
9
10 print("Connection Established")
11
12
13 cursor.execute("SELECT c.First_Name, c.Last_Name, f.Title AS Film_Title, r.Rental_Date FROM
  RENTAL r JOIN INVENTORY i ON r.Inventory_ID = i.Inventory_ID JOIN FILM f ON i.Film_ID =
  f.Film_ID JOIN CUSTOMER c ON r.Customer_ID = c.Customer_ID ORDER BY r.Rental_Date DESC LIMIT
  1;")
14
15
16 myresult = cursor.fetchall()
17
18 for x in myresult:
19   print(x)
20
21
22 myresult2 = cursor.fetchall()
23
24
25
26 connection.close()
```

```
student@student:~$ python3 project.py
Connection Established
('Feyza', 'Yaralı', 'Spiderman: Homecoming', datetime.date(2024, 12, 27))
```

16. Which movie is available in which store?

With this query, we can see which movie is available in which store. When a customer requests a movie, we can direct them to the store where it's available.

```sql
SELECT f.Title, i.Store_ID
FROM FILM f
JOIN INVENTORY i ON f.Film_ID = i.Film_ID;
```

Title	Store_ID
The Godfather	1
The Dark Knight	1
The Lord of the Rings: The Return of the King	1
The Lord of the Rings: The Return of the King	2
Before Sunrise	2
Catch Me If You Can	1
Catch Me If You Can	2
Me Before You	2
Tenet	1
Tenet	2
IT	2
The Silence of the Lambs	1
Harry Potter and the Order of the Phoenix	2
Spiderman: Homecoming	1
Spiderman: Homecoming	2
Jigsaw X	2

Open *project.py ~/ Save

```
import pymysql
connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
cursor = connection.cursor()


import pymysql
connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
cursor = connection.cursor()

print("Connection Established")


cursor.execute("SELECT f.Title, i.Store_ID FROM FILM f JOIN INVENTORY i ON f.Film_ID =
  i.Film_ID;")


myresult = cursor.fetchall()

for x in myresult:
  print(x)


myresult2 = cursor.fetchall()



connection.close()
```

```
student@student:~$ python3 project.py
Connection Established
('The Godfather', 1)
('The Dark Knight', 1)
('The Lord of the Rings: The Return of the King', 1)
('The Lord of the Rings: The Return of the King', 2)
('Before Sunrise', 2)
('Catch Me If You Can', 1)
('Catch Me If You Can', 2)
('Me Before You', 2)
('Tenet', 1)
('Tenet', 2)
('IT', 2)
('The Silence of the Lambs', 1)
('Harry Potter and the Order of the Phoenix', 2)
('Spiderman: Homecoming', 1)
('Spiderman: Homecoming', 2)
('Jigsaw X', 2)
```

17. How many days did it take for rented films to be returned? (Sorted from the longest to the shortest duration)

Here, we wrote a query that shows how long it took for a customer to return the rented film to the store where it was rented. While doing this, we sorted the number of days it took from largest to smallest.

```
1 SELECT c.First_Name, c.Last_Name, f.Title ,(r.Return_Date-
  r.Rental_Date) AS 'Rental Duration (Day)'
2 FROM INVENTORY i
3 JOIN RENTAL r ON i.Inventory_ID=r.Inventory_ID
4 JOIN CUSTOMER c ON r.Customer_ID=c.Customer_ID
5 JOIN FILM f ON i.Film_ID=f.Film_ID
6 ORDER BY (r.Return_Date-r.Rental_Date) DESC;
```

First_Name	Last_Name	Title	Rental Duration (Day)
Veli	Çetin	Before Sunrise	103
Onur	Özdemir	Jigsaw X	99
Ayşe	Duru	The Lord of the Rings: The Return of the King	98
Büşra	Güzel	The Lord of the Rings: The Return of the King	98
Mehmet	Kaya	Harry Potter and the Order of the Phoenix	97
Derya	Yılmazer	Catch Me If You Can	96
Zeynep	Öztürk	The Lord of the Rings: The Return of the King	94
Ali	Demir	The Lord of the Rings: The Return of the King	93
Yusuf	Taş	Tenet	93
Zafer	Mert	Catch Me If You Can	92
Burcu	Arslan	The Godfather	91
Ali	Demir	Spiderman: Homecoming	89
Aylin	Kara	The Dark Knight	88
Hüseyin	Güler	The Lord of the Rings: The Return of the King	86
Büşra	Güzel	The Dark Knight	82
Fatma	Aslan	The Godfather	82
Aylin	Demirtaş	Spiderman: Homecoming	80
Aylin	Demirtaş	The Silence of the Lambs	79
Murat	Kurt	Before Sunrise	78
Can	Yıldız	Before Sunrise	77
Yusuf	Taş	The Silence of the Lambs	77
Onur	Özdemir	The Lord of the Rings: The Return of the King	76

```
Open   *project.py   Save

1 import pymysql
2 connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
3 cursor = connection.cursor()
4
5
6 import pymysql
7 connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
8 cursor = connection.cursor()
9
10 print("Connection Established")
11
12
13 cursor.execute("SELECT c.First_Name, c.Last_Name, f.Title, r.Rental_Date, (r.Return_Date -
  r.Rental_Date) AS 'Rental Duration (Day)' FROM INVENTORY i JOIN RENTAL r ON i.Inventory_ID =
  r.Inventory_ID JOIN CUSTOMER c ON r.Customer_ID = c.Customer_ID JOIN FILM f ON i.Film_ID =
  f.Film_ID ORDER BY r.Return_Date DESC;")
14
15
16 myresult = cursor.fetchall()
17
18 for x in myresult:
19   print(x)
20
21
22 myresult2 = cursor.fetchall()
23
24
25
26 connection.close()
```

```
student@student:~$ python3 project.py
Connection Established
('Ozan', 'Şahin', 'Spiderman: Homecoming', datetime.date(2024, 12, 27), 3)
('Feyza', 'Yaralı', 'Spiderman: Homecoming', datetime.date(2024, 12, 27), 2)
('Elif', 'Köksal', 'Jigsaw X', datetime.date(2024, 12, 15), 8)
('Elif', 'Köksal', 'Spiderman: Homecoming', datetime.date(2024, 12, 15), 6)
('Zafer', 'Mert', 'Spiderman: Homecoming', datetime.date(2024, 12, 7), 10)
('Sibel', 'Çiçek', 'Harry Potter and the Order of the Phoenix', datetime.date(2024, 12, 7), 4)
('Aylin', 'Demirtaş', 'Spiderman: Homecoming', datetime.date(2024, 11, 27), 80)
('Aylin', 'Demirtaş', 'The Silence of the Lambs', datetime.date(2024, 11, 27), 79)
('Yusuf', 'Taş', 'The Silence of the Lambs', datetime.date(2024, 11, 27), 77)
('Serap', 'Tosun', 'The Silence of the Lambs', datetime.date(2024, 11, 16), 14)
('Tuncay', 'Erdoğan', 'IT', datetime.date(2024, 11, 16), 7)
('Nihal', 'Bölükbaşı', 'The Silence of the Lambs', datetime.date(2024, 11, 4), 12)
('Merve', 'Sönmez', 'Tenet', datetime.date(2024, 10, 30), 75)
('Hüseyin', 'Güler', 'Me Before You', datetime.date(2024, 10, 19), 5)
('Deniz', 'Kucuk', 'Tenet', datetime.date(2024, 10, 5), 5)
('Berk', 'Karaca', 'Tenet', datetime.date(2024, 9, 27), 75)
('Berk', 'Karaca', 'Catch Me If You Can', datetime.date(2024, 9, 15), 10)
('Tolga', 'Aydın', 'Me Before You', datetime.date(2024, 9, 1), 14)
('Onur', 'Özdemir', 'Jigsaw X', datetime.date(2024, 8, 13), 99)
('Berk', 'Karaca', 'Catch Me If You Can', datetime.date(2024, 9, 3), 5)
('Kemal', 'Tuna', 'Catch Me If You Can', datetime.date(2024, 8, 30), 74)
('Tolga', 'Aydın', 'The Lord of the Rings: The Return of the King', datetime.date(2024, 8, 16), 7)
('Selin', 'Yıldırım', 'The Lord of the Rings: The Return of the King', datetime.date(2024, 8, 5), 2)
('Yusuf', 'Taş', 'Tenet', datetime.date(2024, 7, 12), 93)
('Onur', 'Özdemir', 'The Lord of the Rings: The Return of the King', datetime.date(2024, 7, 29), 76)
('Okan', 'Polat', 'Catch Me If You Can', datetime.date(2024, 7, 3), 27)
('Burcu', 'Arslan', 'Me Before You', datetime.date(2024, 7, 14), 5)
('Burcu', 'Arslan', 'The Lord of the Rings: The Return of the King', datetime.date(2024, 7, 7), 9)
('Burcu', 'Arslan', 'The Godfather', datetime.date(2024, 6, 23), 91)
('Derya', 'Yılmazer', 'Me Before You', datetime.date(2024, 6, 26), 2)
('Aylin', 'Kara', 'The Silence of the Lambs', datetime.date(2024, 6, 11), 14)
('Emre', 'BüyükgüngÖr', 'The Godfather', datetime.date(2024, 6, 18), 3)
('Mehmet', 'Kaya', 'Harry Potter and the Order of the Phoenix', datetime.date(2024, 5, 21), 97)
('Aylin', 'Kara', 'The Dark Knight', datetime.date(2024, 5, 30), 88)
('Aylin', 'Kara', 'Me Before You', datetime.date(2024, 6, 9), 8)
('Derya', 'Yılmazer', 'Catch Me If You Can', datetime.date(2024, 5, 14), 96)
```

18. Which city that our customers are registered in has the highest payment in total?

With this query, we can see which city has generated the most payment based on our customers' registered cities.

```sql
SELECT ci.City, SUM(p.Amount) AS Total_Payment FROM PAYMENT p
JOIN CUSTOMER c ON p.Customer_ID = c.Customer_ID
JOIN ADDRESS a ON c.Address_ID = a.Address_ID
JOIN CITY ci ON a.City_ID = ci.City_ID
GROUP BY ci.City
ORDER BY Total_Payment DESC LIMIT 1;
```

City	Total_Payment
Antalya (Merkez)	918.74

```
Open   *project.py   Save
1 import pymysql
2 connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
3 cursor = connection.cursor()
4
5
6 import pymysql
7 connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
8 cursor = connection.cursor()
9
10 print("Connection Established")
11
12
13 cursor.execute("SELECT ci.City, SUM(p.Amount) AS Total_Payment FROM PAYMENT p JOIN CUSTOMER c ON
   p.Customer_ID = c.Customer_ID JOIN ADDRESS a ON c.Address_ID = a.Address_ID JOIN CITY ci ON
   a.City_ID = ci.City_ID GROUP BY ci.City ORDER BY Total_Payment DESC LIMIT 1;")
14
15
16 myresult = cursor.fetchall()
17
18 for x in myresult:
19   print(x)
20
21
22 myresult2 = cursor.fetchall()
23
24
25
26 connection.close()
```

```
student@student:~$ python3 project.py
Connection Established
('Antalya (Merkez)', Decimal('918.74'))
```

19. Who are the actors featured in 'The Godfather'?

This query allows us to see the names of actors who featured in a specific movie. We chose The Godfather as an example.

```
1 SELECT a.Actor_Name, a.Actor_Surname, f.Title FROM ACTOR a
2 JOIN FILM_ACTOR fa ON a.Actor_ID = fa.Actor_ID
3 JOIN FILM f ON fa.Film_ID = f.Film_ID
4 WHERE f.Title = 'The Godfather';
```

Actor_Name	Actor_Surname	Title
Al	Pacino	The Godfather
Marlon	Brando	The Godfather

Open *project.py Save

```
1 import pymysql
2 connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
3 cursor = connection.cursor()
4
5
6 import pymysql
7 connection = pymysql.connect(host="localhost", port=3306, user="root", passwd="_23StudenT47_",
  database="STAT311_PROJECT")
8 cursor = connection.cursor()
9
10 print("Connection Established")
11
12
13 cursor.execute("SELECT a.Actor_Name, a.Actor_Surname, f.Title FROM ACTOR a JOIN FILM_ACTOR fa ON
  a.Actor_ID = fa.Actor_ID JOIN FILM f ON fa.Film_ID = f.Film_ID WHERE f.Title = 'The
  Godfather';")
14
15
16 myresult = cursor.fetchall()
17
18 for x in myresult:
19   print(x)
20
21
22 myresult2 = cursor.fetchall()
23
24
25
26 connection.close()
```

```
student@student:~$ python3 project.py
Connection Established
('Al', 'Pacino', 'The Godfather')
('Marlon', 'Brando', 'The Godfather')
```

7. SUMMARY

In the initial phase of our project, we decided as a group on the topic we wanted to focus on. Based on our shared interest in films and inspired by a sense of nostalgia, we chose to work on a DVD rental database system. As a first step, we identified 13 entities. We decided not to include more entities because we decided that this number sufficient and wanted to avoid redundancy, ensuring the system remained efficient and consistent.

The tables we created were named as follows: actor, address, category, city, customer, film, film_actor (relationship between the film and actor tables), film_category (relationship between the film and category tables), inventory (relationship between the film and store tables), staff, payment, rental, and store. Attributes were added to each table, and relationships between the tables were established. Once this structure was finalized, we generated our dataset and inserted the data.

In the second phase of the project, we conducted create, read, update, and delete (CRUD) operations to check for any inconsistencies in the dataset and confirmed that no such issues existed. Following this, we formulated questions and wrote queries to answer them. Additionally, we connected phpMyAdmin on our virtual machine to Python, allowing us to execute the queries through a Python file in the terminal.

Through this project, we gained valuable experience in teamwork, database creation, data insertion, writing and analyzing queries, and executing these queries in Python. This process allowed us to successfully complete our project.