IE School of Human Sciences & Technology

SQL Based Data Architectures

Data Modeling Assignment
Cinema

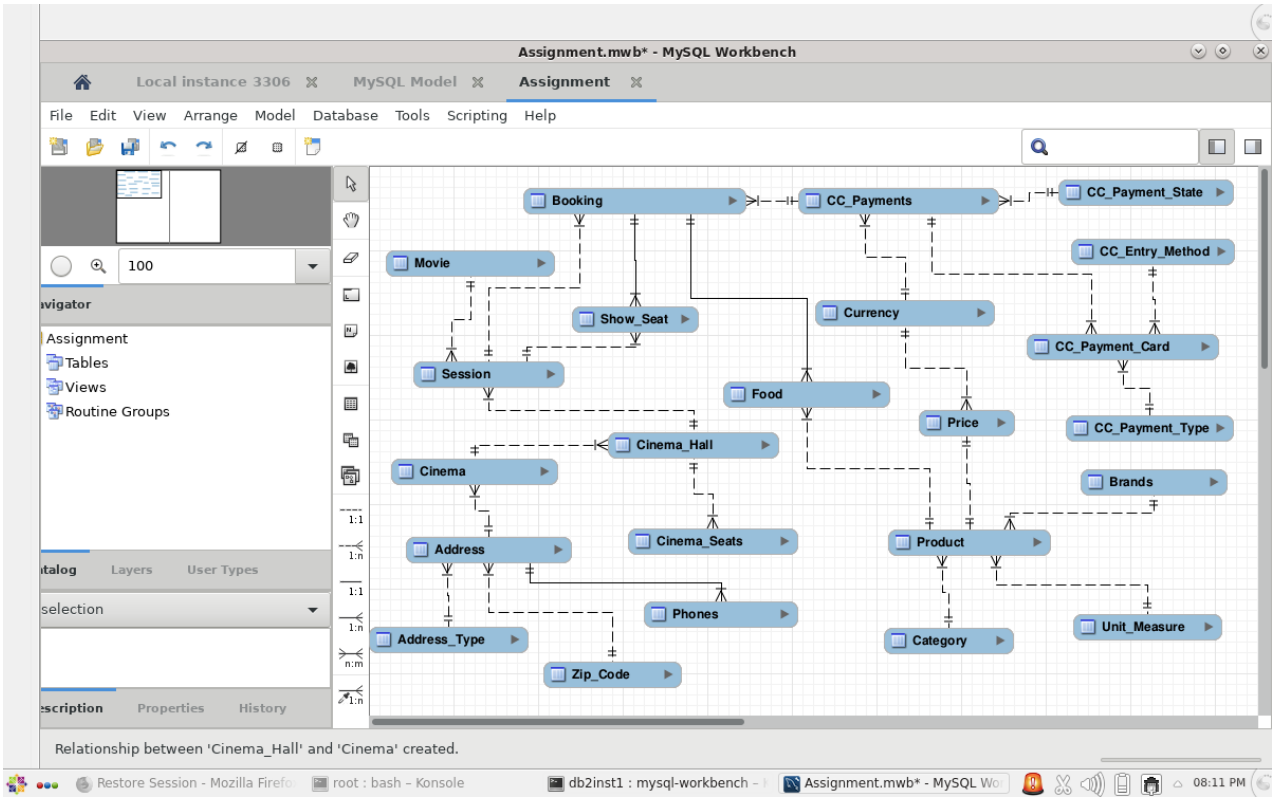
Author - Krishna Agrawal

Course

Master's in Business Analytics and Big Data

DATA WAREHOUSE (Cinema)

ERD Table - Cinema



Description of The Model

The Entity Relationship Diagram (ERD) is a model of movie ticket booking system entity, whose main purpose is to represent data objects and the relationships between structured data groups of Movie ticket booking system functionality. The ERD shows all the attributes of database tables and the relationship between, Booking, Movie, Address, Food, Seats, Payments, Shows and Currency. The ERD was designed with one-to-many and one-to-one relationships between the entities, where each one of the entities contain a primary key and some contain foreign keys to develop the relationship between them. The movie, seats, booking and show entities were normalized to reduce duplicacy of records and reduce the unnecessary relationships that create data inconsistencies.

Description of the Movie Ticket Booking System entities and attributes:

Booking: This table has several important attributes, where Booking_ID was assigned as the primary key. This table display information about the number of seats purchased by each booking, the date when the booking took place, the show_ID, the total price users paid for its ticket and food, the total price with a 10% tax included, the currency used to do the booking and the payment_ID that is specific to each booking and payment and cannot be replicable. The Booking_ID table contains two foreign keys, show_ID and payment_ID, that reference two other tables.

Movie: This table shows the movie_ID which is a unique value for each one of the movies displayed and was assigned as a primary key. It shows the title the cinema is playing now, the description of each movie, the duration of them, the language in which they are played, the release date of the movie, the country where it was created and the genre of the movie.

Session: This table has some important attributes and information about different sessions in which the movies are displayed. It has as a primary key the Show_ID for each one of the sessions, the date and the cinema hall ID where they are played, the start and end of the session, and the movie_ID for each one of the movies. This entity has as foreign key the Cinema_Hall_ID and Movie_ID, referencing two other entities on the ERD.

Show Seat: this entity shows the information about the seats booked for each one of the booking_ID. This entity has two primary keys, the booking_ID and show_seat_ID. The table shows information about the price of each seat, and to which show_ID they were assigned. The table has show_ID as foreign key.

Cinema: The cinema table shows the Cinema_ID which is a unique value for each one of the cinemas and it was assigned as the primary key. The table displays information about the name for each one of the cinemas and the total of cinema halls the cinemas have. Finally, the table has address_ID as the foreign key and it displays information about the address of each cinema.

Cinema Hall: The table shows the cinema hall ID which is a unique value for each one of the different halls at each cinema and assigned as the primary key. The table displays the name for each hall, the total of seats that each hall has, and finally the cinema_ID was assigned as the foreign key to check which cinema hall belongs to each one of the cinemas.

Cinema Seats: the cinema seats table shows the cinema seats ID, which is a unique value for each seat at each cinema hall as it is the identifier for each one of the seats, and it was assigned as the primary key. Furthermore, it shows the seat number for each different seat, the type of seats available at the cinema, and finally it was assigned as a foreign key to the cinema hall ID.

Address: The table shows the address ID as a primary key and it shows the ID for each one of the addresses of the cinemas. Furthermore, it displays the type_ID for each address to identify if it was an avenue, street, or boulevard. Also, it shows the name of the street, the number of the building, the zip code and the country where the cinemas are located. It was assigned as foreign key to the zip code, type_ID, and country.

Address Type: the table shows the description for each one of the Type_ID, and the type_ID was displayed as the primary key of the table.

Zip Code: The table has as primary key the zip code of each address and the country where each cinema is located. It also shows the city and state where the cinemas are situated.

Phones: the phones table displays the address ID and numseq as primary key. The table shows the phone numbers of the cinemas, where it is displaying the external number and the internal number of each one of the phones and the description of the main purpose of each phone number.

Currency: This table shows the Currency_ID which is a unique value for each one of the currency codes and was assigned as a primary key. It further shows the currency code as per ISO 4217, as well as a description of the currency (to what exact currency the ID refers to).

CC_Payments: This table shows the CC_Payment_ID which is a unique value for each one of the payments made and was assigned as a primary key. It further shows the Pay_Tran as a unique value, the specific amount of the payment, and the time at which the payment was done. This entity has as foreign key the Currency_ID and CC_Payment_State, referencing two other entities on the ERD.

CC_Entry_Method: This table shows the CC_Method which is a unique value for each one of the ways the payment was made and was assigned as a primary key. It further shows a more complete description of the way the payment was made, assigning the numbers of the primary key to a better understanding of what they mean.

CC_Payment_Card: This table shows the CC_Payment_ID which is a unique value for each one of the credit card payments made and was assigned as a primary key. It further shows whether the card is encrypted or not, the number of each card that was used for a payment, the name of the bank for each card used and the expiration date of the card used. Finally, this entity has as foreign key the Payment_Type and CC_Entry_Method, referencing two other entities on the ERD.

CC_Payment_Type: This table shows the CC_Type which is a unique value for each one of the types of payments made and was assigned as a primary key. It further shows a more complete description of the exact type of payment made, and which card type was used.

CC_Payment_State: This table shows the CC_State which is a unique value for each one of the state of the payments made, as a number and was assigned as a primary key. It further shows a more complete description of the state of each payment made, whether it was approved or not.

Food: This table shows the combination of the Booking_ID and NumSeq attributes, which were assigned as a composite primary key, showing which bookings bought which food. It further shows the quantity of each food that was bought for each booking. Finally, this entity has as foreign key the Product_ID referencing another entity on the ERD.

Price: This table shows the Product_ID which is a unique value for each one of the products that were bought (in order to show the price for each product), as a number and was assigned as a primary key. It further shows the cost of the product that was bought (being the price the cinema paid to have these products), but also its price, which is the actual price provided for customers buying this product. Taxes is another

attribute of this table which shows the total price paid by a customer for each product, being the price multiplied by a 10% tax on each product. Finally, this entity has as foreign key the Currency_ID, referencing another entity on the ERD.

Product: This table shows the Product_ID which is a unique value for each one of the products that were bought, as a number and was assigned as a primary key. It further shows the name of the product that was bought and shows a description to what this product is more concretely. Finally, this entity has as foreign key the Category_ID, Brand_ID and Size_ID, referencing three other entities on the ERD.

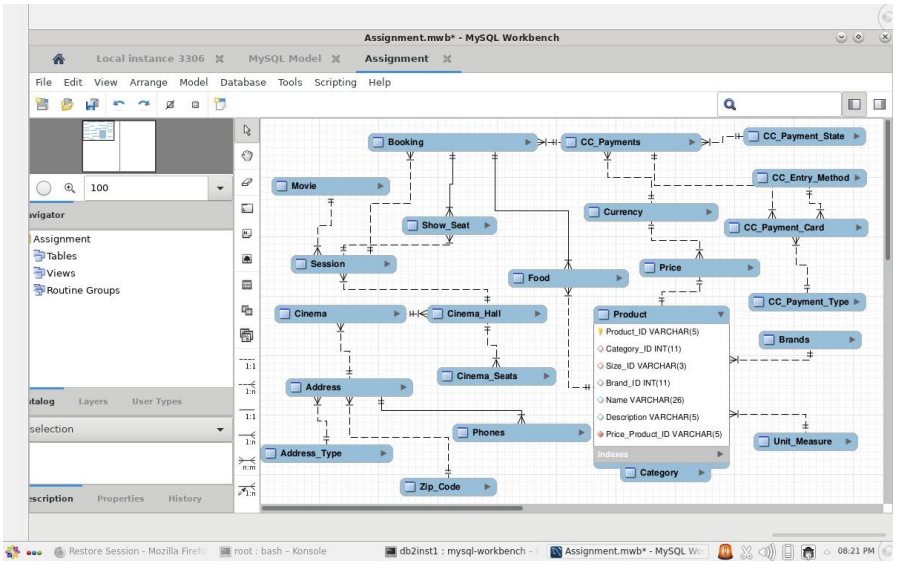
Brands: This table shows the Brand_ID which is a unique value that shows the brand of each product that was bought, as brand for each product bought, and a more complete description of what exactly this name refers to, in terms of the brand of each product.

Category: This table shows the Category_ID which is a unique value that shows the category of each product that was bought, as a number and was assigned as a primary key. It further shows the name of the category, and a more complete description of what exactly this name refers to, in terms of the category of each product bought.

Unit_Measure: This table shows the Size_ID which is a unique value that shows the size of each product that was bought and was assigned as a primary key. It further shows the description of this exact size, in more complete details.

DDL and DML SQL Table Creation with Primary Key and Insertion of data

Product Table



CREATE TABLE Product (

Product_ID VARCHAR(5) NOT NULL,

Category_ID INT,

Size_ID VARCHAR(3),

Brand_ID INT,

Name VARCHAR(26),

Description VARCHAR(5),

PRIMARY KEY(Product_ID)

);

INSERT INTO Product VALUES

('PRK1',600,'SM',500, 'Sweet Popcorn','snack'),

('PRK2',600,'SM',500,'Salt Popcorn','snack'),

('PRK3',600,'SM',500,'Mix Popcorn','snack'),

('PRK5',601,'UN',504,'Hershey','sweet'),

('PRK6',602,'ML',502, 'Fizzy Drinks','drink'),

('PRK8',600,'UN',507,'Hot Dogs','food'),

('PRK9',600,'UN',505,'Nachos','food'),

('PRK10',601,'UN',506,'Gummy Bear','sweet'),

('PRK11',600,'MD',500,'Sweet Popcorn','snack'),

('PRK12',600,'MD',500,'Salt Popcorn','snack'),

('PRK13',600,'MD',500,'Mix Popcorn','snack'),

('PRK14',602,'CL',502,'Fizzy Drinks','drink'),

('PRK15',602,'CL',502,'Fizzy Drinks','drink'),

('PRK16',600,'LG',500,'Sweet Popcorn','snack'),

('PRK17',600,'LG',500,'Salt Popcorn','snack'),

('PRK18',600,'LG',500,'Mix Popcorn','snack'),

('PRK19',604,'LG',508,' Drinks, popcorn, hot dog','combo'),

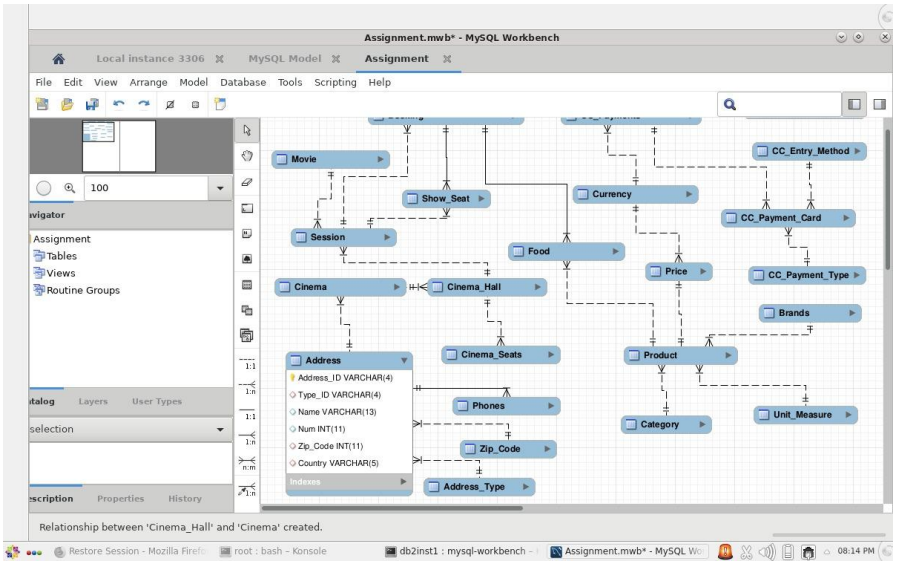
('PRK20',604,'LG',508,'Drinks, popcorn, nachos','combo'),

('PRK21',604,'LG',508,' Drinks, popcorn, hot dogs','combo'),

('PRK22',604,'MD',508,'Drinks, popcorn, sweet','combo'),

('PRK23',604,'MD',508,'Drinks, popcorn, popcorn','combo');

Address Table



CREATE TABLE Address (

Address_ID VARCHAR(4) NOT NULL,

Type_ID VARCHAR(4),

Name VARCHAR(13),

Num INT,

Zip_Code INT,

Country VARCHAR(5),

PRIMARY KEY(Address_ID)

);

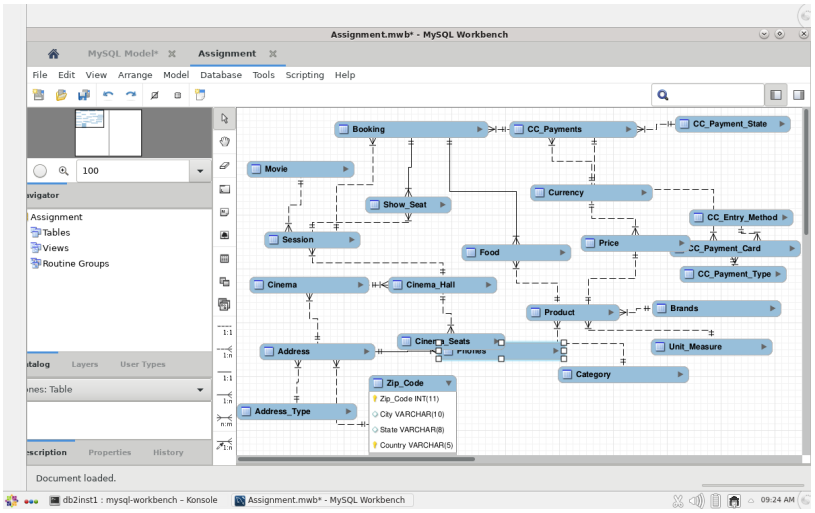
INSERT INTO Address VALUES

('B001','St','Maria Molina',28,8001,'Spain'),

('MA001','Ave','Castelló',92,28006,'Spain'),

('V001','Blvd','Diego de Leon',58,46001,'Spain');

Zip_Code Table



CREATE TABLE Zip_Code (

Zip_Code INT NOT NULL,

City VARCHAR(10),

State VARCHAR(8),

Country VARCHAR(5) NOT NULL,

PRIMARY KEY (Zip_Code, Country)

);

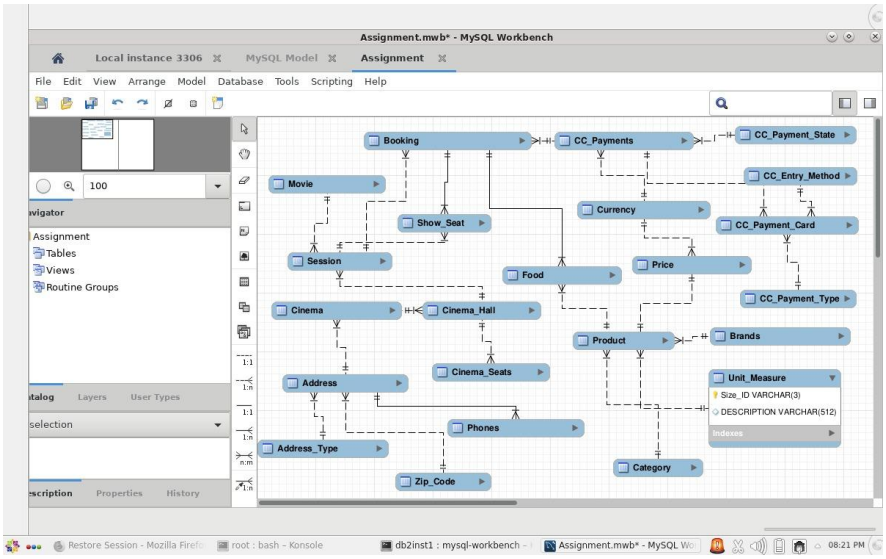
INSERT INTO Zip_Code VALUES

(8001,' Barcelona','Cataluña','Spain'),

(28006,' Madrid','Madrid','Spain'),

(46001,' Valencia','Valencia','Spain');

Unit_Measure Table



CREATE TABLE Unit_Measure (

Size_ID VARCHAR(3) NOT NULL,

DESCRIPTION VARCHAR(512),

PRIMARY KEY(Size_ID)

);

INSERT INTO Unit_Measure VALUES

('SM','SMALL'),

('MD','MEDIUM'),

('LG','LARGE'),

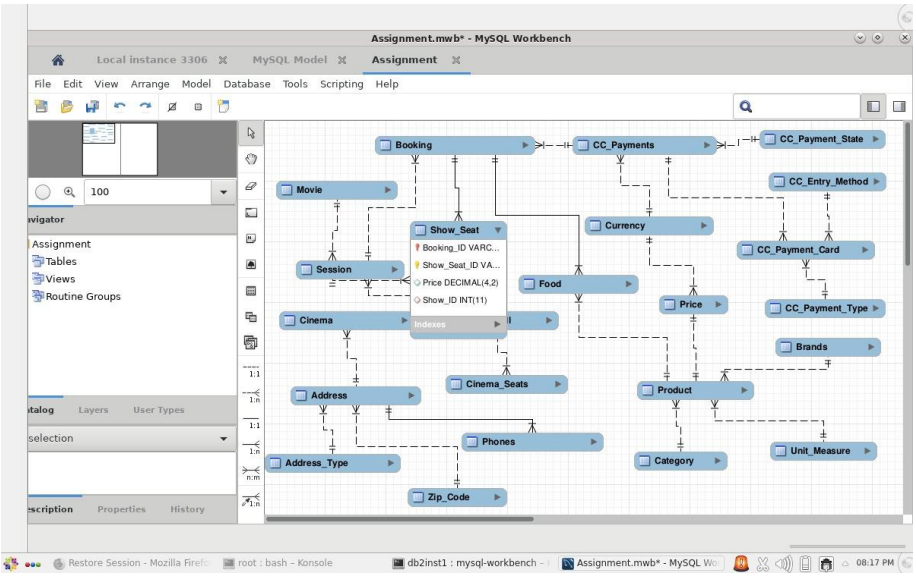
('CL','CENTILITRES'),

('ML','MILLILITRES'),

('LT','LITRES'),

('UN', 'UNIQUE');

Show_Seat Table



CREATE TABLE Show_Seat (

Booking_ID VARCHAR(6) NOT NULL,

Show_Seat_ID VARCHAR(7) NOT NULL,

Price NUMERIC(4, 2),

Show_ID INT,

PRIMARY KEY (Booking_ID, Show_Seat_ID)

);

INSERT INTO Show_Seat VALUES

('ID1000','SSID001',10.99,44197),

('ID1001','SSID002',10.99,44198),

('ID1002','SSID003',10.99,44199),

('ID1003','SSID004',10.99,44200),

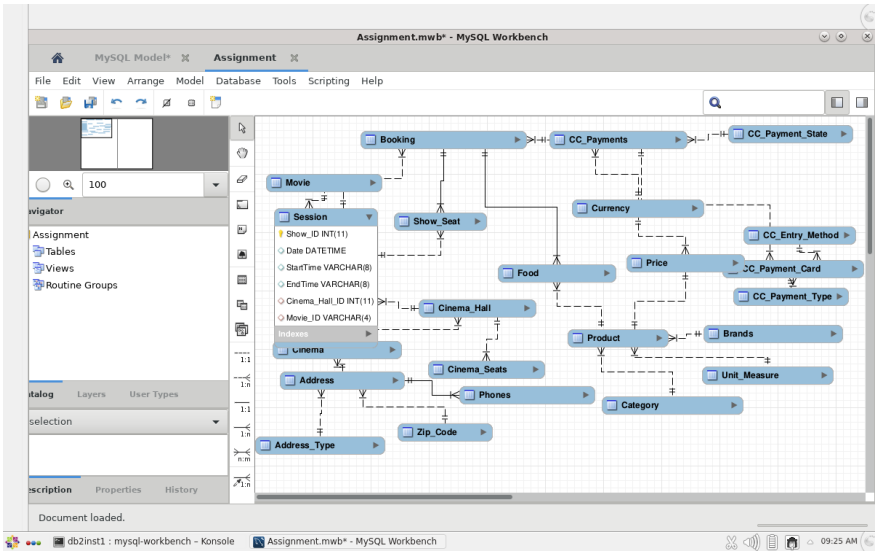
('ID1004','SSID005',10.99,44201),

('ID1005','SSID006',10.99,44202),

('ID1006','SSID007',10.99,44203),

('ID1007','SSID008',10.99,44204),

Session Table



CREATE TABLE Session (

Show_ID INT NOT NULL,

Date DATETIME,

StartTime VARCHAR(8),

EndTime VARCHAR(8),

Cinema_Hall_ID INT,

Movie_ID VARCHAR(4),

PRIMARY KEY (Show_ID)

);

INSERT INTO Session VALUES

(44197,'2021-12-08 00:00:00','16:00:00','18:40:00',1,'M001'),

(44198,'2021-12-08 00:00:00','16:00:00','17:00:00',2,'M002'),

(44199,'2021-12-08 00:00:00','16:00:00','17:30:00',3,'M003'),

(44200,'2021-12-08 00:00:00','16:00:00','18:40:00',4,'M001'),

(44201,'2021-12-08 00:00:00','16:00:00','17:00:00',5,'M002'),

(44202,'2021-12-08 00:00:00','16:00:00','17:30:00',6,'M003'),

(44203,'2021-12-08 00:00:00','16:00:00','18:40:00',7,'M001'),

(44204,'2021-12-08 00:00:00','16:00:00','17:00:00',8,'M002'),

(44205,'2021-12-08 00:00:00','16:00:00','17:30:00',9,'M003'),

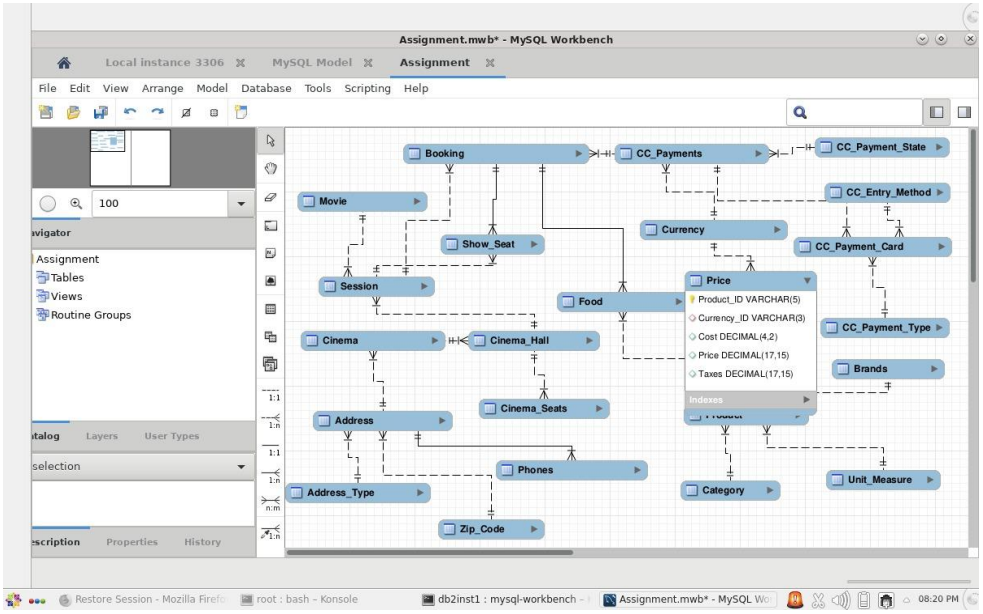
(44206,'2021-12-08 00:00:00','13:00:00','15:40:00',1,'M004'),

(44207,'2021-12-08 00:00:00','13:00:00','14:00:00',2,'M005'),

(44208,'2021-12-08 00:00:00','13:00:00','14:30:00',3,'M006'),

(44209,'2021-12-08 00:00:00','13:00:00','15:40:00',4,'M004'),

Price Table



CREATE TABLE Price (

Product_ID VARCHAR(5) NOT NULL,

Currency_ID VARCHAR(3),

Cost NUMERIC(4, 2),

Price NUMERIC(17, 15),

Taxes NUMERIC(17, 15),

PRIMARY KEY(Product_ID)

);

INSERT INTO Price VALUES

('PRK1','EUR',2.99,5.99,6.589),

('PRK2','EUR',2.99,5.99,6.589),

('PRK3','EUR',2.99,5.99,6.589),

('PRK4','EUR',1.59,4.59,5.049),

('PRK5','EUR',1.59,4.59,5.049),

('PRK6','EUR',0.99,3.99,4.389),

('PRK7','EUR',0.99,3.99,4.389),

('PRK8','EUR',4.99,7.99,8.789000000000001),

('PRK9','EUR',3.5,6.5,7.15),

('PRK10','EUR',1.59,4.59,5.049),

('PRK11','EUR',3.5,6.5,7.15),

('PRK12','EUR',3.5,6.5,7.15),

('PRK13','EUR',3.5,6.5,7.15),

('PRK14','EUR',1.5,4.5,4.95),

('PRK15','EUR',1.99,4.99,5.489000000000001),

('PRK16','EUR',3.99,6.99,7.689000000000001),

('PRK17','EUR',3.99,6.99,7.689000000000001),

('PRK18','EUR',3.99,6.99,7.689000000000001),

('PRK19','EUR',15.99,18.990000000000002,20.889000000000003),

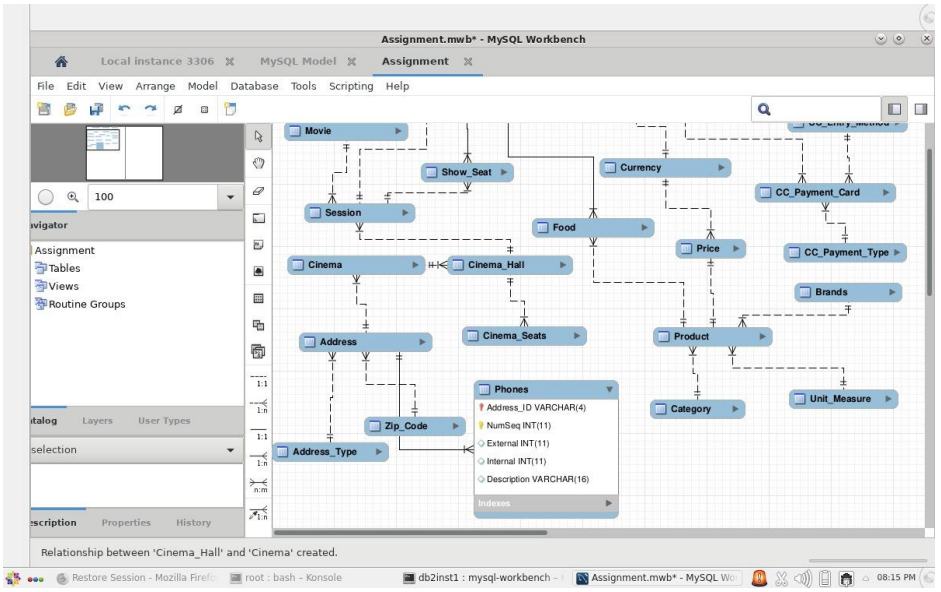
('PRK20','EUR',13.99,16.990000000000002,18.689000000000004),

('PRK21','EUR',13.99,16.990000000000002,18.689000000000004),

('PRK22','EUR',12.5,15.5,17.05),

('PRK23','EUR',11.99,14.99,16.489);

Phones Table




```
CREATE TABLE Phones (
    Address_ID VARCHAR(4) NOT NULL,
    NumSeq INT NOT NULL,
    External INT,
    Internal INT,
    Description VARCHAR(16),
PRIMARY KEY(Address_ID, NumSeq)
);
INSERT INTO Phones VALUES
    ('B001',4000,853,9054,'Customer Service'),
    ('MA001',4001,816,9929,'Customer Service'),
    ('V001',4002,864,8116,'Customer Service'),
    ('B001',4003,718,7911,'Customer Service'),
    ('MA001',4004,738,7229,'Customer Service'),
    ('V001',4005,806,7087,'Customer Service'),
    ('B001',4006,742,8177,'Customer Service'),
```

('MA001',4007,712,7584,'Customer Service'),

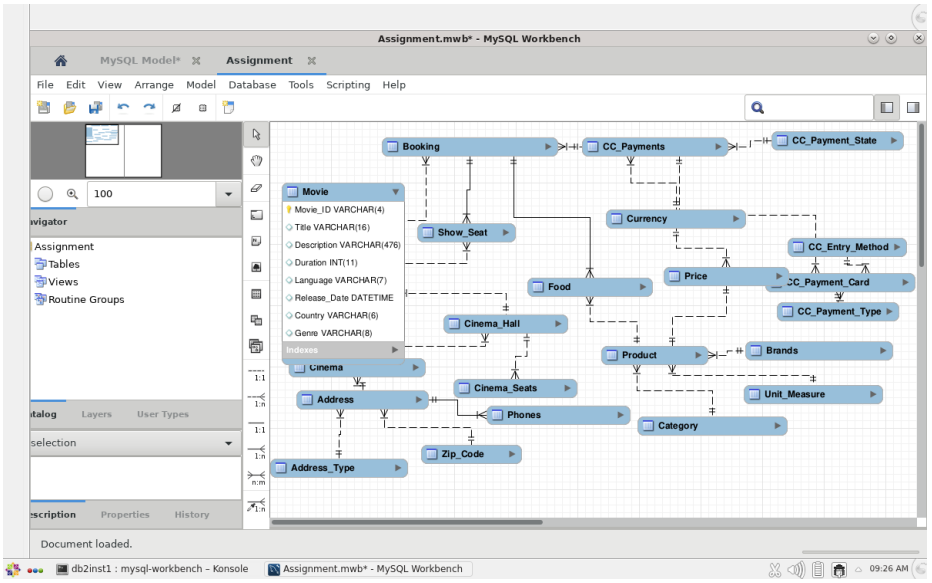
('V001',4008,685,7960,'Customer Service'),

('B001',4009,726,7439,'Customer Service'),

('MA001',4010,694,9331,'Customer Service'),

('V001',4011,600,7741,'Customer Service');

Movie Table



CREATE TABLE Movie (

Movie_ID VARCHAR(4) NOT NULL,

Title VARCHAR(16),

Description VARCHAR(476),

Duration INT,

Language VARCHAR(7),

Release_Date DATETIME,

Country VARCHAR(6),

Genre VARCHAR(8),

PRIMARY KEY(Movie_ID)

);

INSERT INTO Movie VALUES

('M001','Advengers','When Thor''s evil brother, Loki (Tom Hiddleston), gains access to the unlimited power of the energy cube called the Tesseract, Nick Fury (Samuel L. Jackson), director of S.H.I.E.L.D., initiates a superhero recruitment effort to defeat the unprecedented threat to Earth. Joining Fury''s "dream team" are Iron Man (Robert Downey Jr.), Captain America (Chris Evans), the Hulk (Mark Ruffalo), Thor (Chris Hemsworth), the Black Widow (Scarlett Johansson) and Hawkeye (Jeremy Renner).',160,'English','2021-01-01 00:00:00','USA','Comedy'),

('M002','Fast and Furious','Dom Toretto is living the quiet life off the grid with Letty and his son, but they know that danger always lurks just over the peaceful horizon. This time, that threat forces Dom to confront the sins of his past to save those he loves most. His crew soon comes together to stop a world-shattering plot by the most skilled assassin and high-performance driver they''ve ever encountered -- Dom''s forsaken brother.',60,'English','2021-03-02 00:00:00','France','Romantic'),

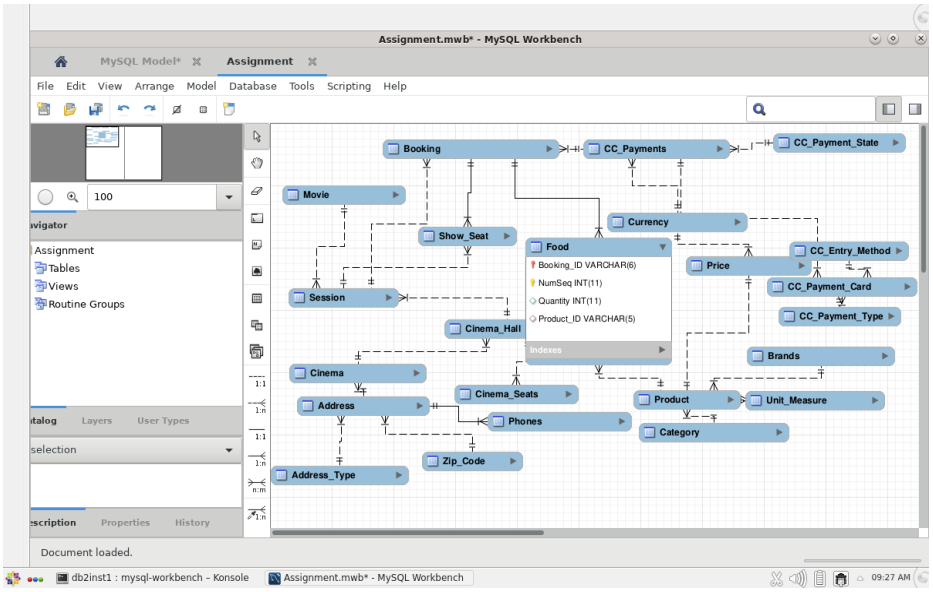
('M003','Spiderman','"Spider-Man" centers on student Peter Parker (Tobey Maguire) who, after being bitten by a genetically-altered spider, gains superhuman strength and the spider-like ability to cling to any surface. He vows to use his abilities to fight crime, coming to understand the words of his beloved Uncle Ben: "With great power comes great responsibility."',90,'English','2021-04-03 00:00:00','Spain','Action'),

('M004','Advengers','When Thor''s evil brother, Loki (Tom Hiddleston), gains access to the unlimited power of the energy cube called the Tesseract, Nick Fury (Samuel L. Jackson), director of S.H.I.E.L.D., initiates a superhero recruitment effort to defeat the unprecedented threat to Earth. Joining Fury''s "dream team" are Iron Man (Robert Downey Jr.), Captain America (Chris Evans), the Hulk (Mark Ruffalo), Thor (Chris Hemsworth), the Black Widow (Scarlett Johansson) and Hawkeye (Jeremy Renner).',160,'Spanish','2021-01-01 00:00:00','USA','Comedy'),

('M005','Fast and Furious','Dom Toretto is living the quiet life off the grid with Letty and his son, but they know that danger always lurks just over the peaceful horizon. This time, that threat forces Dom to confront the sins of his past to save those he loves most. His crew soon comes together to stop a world-shattering plot by the most skilled assassin and high-performance driver they''ve ever encountered -- Dom''s forsaken brother.',60,'Spanish','2021-03-02 00:00:00','France','Romantic'),

('M006','Spiderman','"Spider-Man" centers on student Peter Parker (Tobey Maguire) who, after being bitten by a genetically-altered spider, gains superhuman strength and the spider-like ability to cling to any surface. He vows to use his abilities to fight crime, coming to understand the words of his beloved Uncle Ben: "With great power comes great responsibility."',90,'Spanish','2021-04-03 00:00:00','Spain','Action');

Food Table



CREATE TABLE Food (

Booking_ID VARCHAR(6) NOT NULL,

NumSeq INT NOT NULL,

Quantity INT,

Product_ID VARCHAR(5),

PRIMARY KEY (Booking_ID, Numseq)

);

INSERT INTO Food VALUES

('ID1000',2000,1,'PRK1'),

('ID1001',2001,1,'PRK2'),

('ID1002',2002,3,'PRK3'),

('ID1003',2003,2,'PRK4'),

('ID1004',2004,2,'PRK5'),

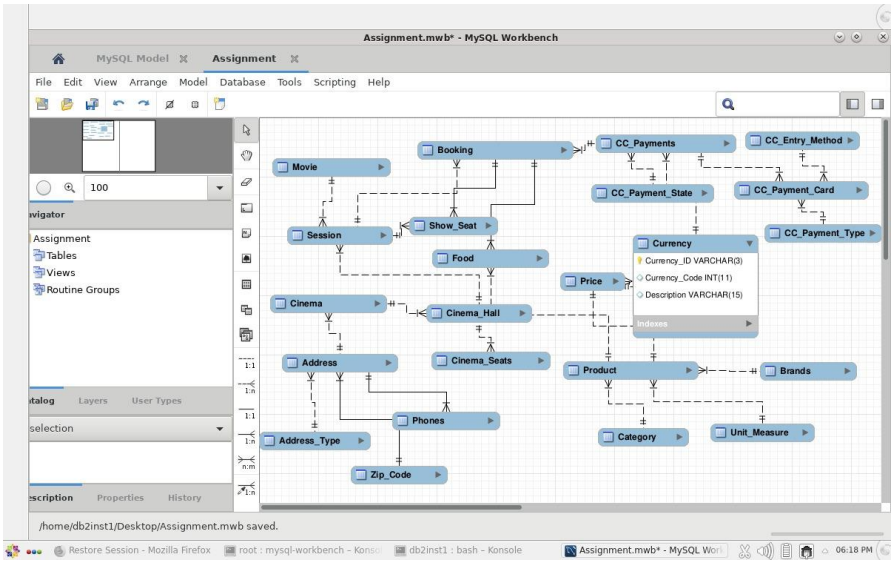
('ID1005',2005,3,'PRK6'),

('ID1006',2006,1,'PRK7'),

('ID1007',2007,2,'PRK8'),

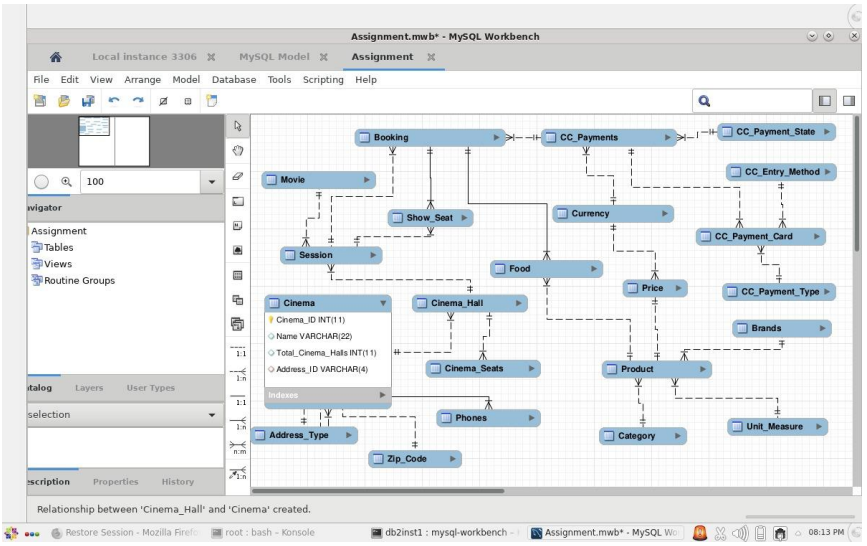
('ID1008',2008,1,'PRK9'),

Currency Table




```
CREATE TABLE Currency (
    Currency_ID VARCHAR(3) NOT NULL,
    Currency_Code INT,
    Description VARCHAR(15),
PRIMARY KEY (Currency_ID)
);
INSERT INTO Currency VALUES
    ('EUR',978,'Euro'),
    ('GBP',826,'British Pounds'),
    ('USD',840,'US Dollars');
```

Cinema Table



CREATE TABLE Cinema (

Cinema_ID INT NOT NULL,

Name VARCHAR(22),

Total_Cinema_Halls INT,

Address_ID VARCHAR(4),

PRIMARY KEY (Cinema_ID)

);

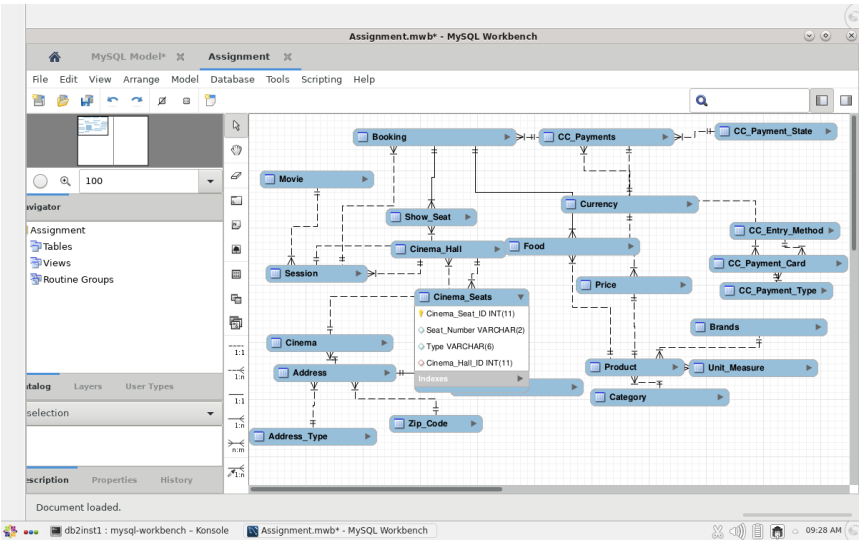
INSERT INTO Cinema VALUES

(001,'Cinema MMPRK Barcelona',3,'B001'),

(002,'Cinema MMPRK Madrid',3,'MA001'),

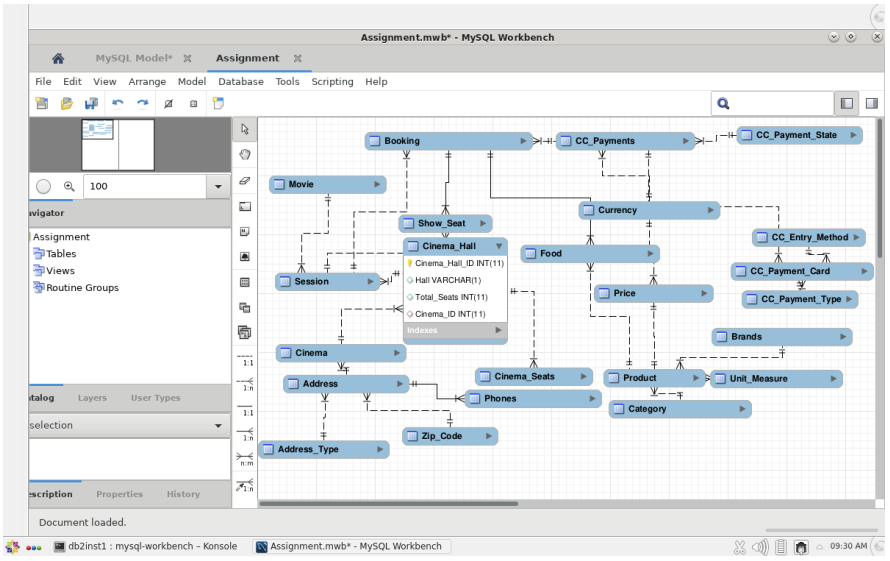
(003,'Cinema MMPRK Valencia',3,'V001');

Cinema_Seats Table




```
CREATE TABLE Cinema_Seats (
    Cinema_Seat_ID INT NOT NULL,
    Seat_Number VARCHAR(2),
    Type VARCHAR(6),
    Cinema_Hall_ID INT,
 PRIMARY KEY (Cinema_Seat_ID)
);
INSERT INTO Cinema_Seats VALUES
    (100,'A1','Normal',1),
    (101,'B1','Normal',2),
    (102,'C1','Normal',3),
    (103,'D1','Normal',4),
    (104,'E1','Normal',5),
    (105,'F1','Normal',6),
     (106,'G1','VIP',7),
      (107,'H1','VIP',8),
       (108,'I1','VIP',9),
```

Cinema_Hall Table



CREATE TABLE Cinema_Hall (

Cinema_Hall_ID INT NOT NULL,

Hall VARCHAR(1),

Total_Seats INT,

Cinema_ID INT,

PRIMARY KEY (Cinema_Hall_ID)

);

INSERT INTO Cinema_Hall VALUES

(1,'A',10,001),

(2,'B',10,002),

(3,'C',10,003),

(4,'A',10,001),

(5,'B',10,002),

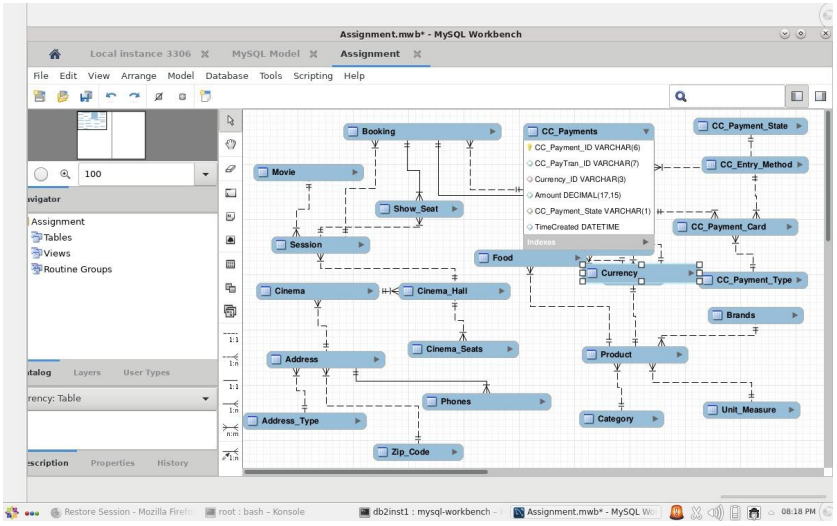
(6,'C',10,003),

(7,'A',10,001),

(8,'B',10,002),

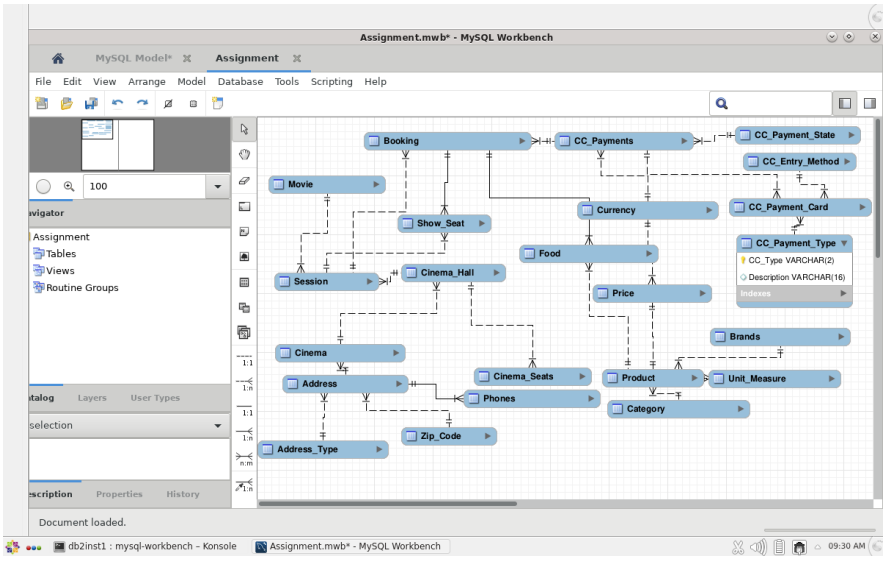
(9,'C',10,003);

CC_Payments Table



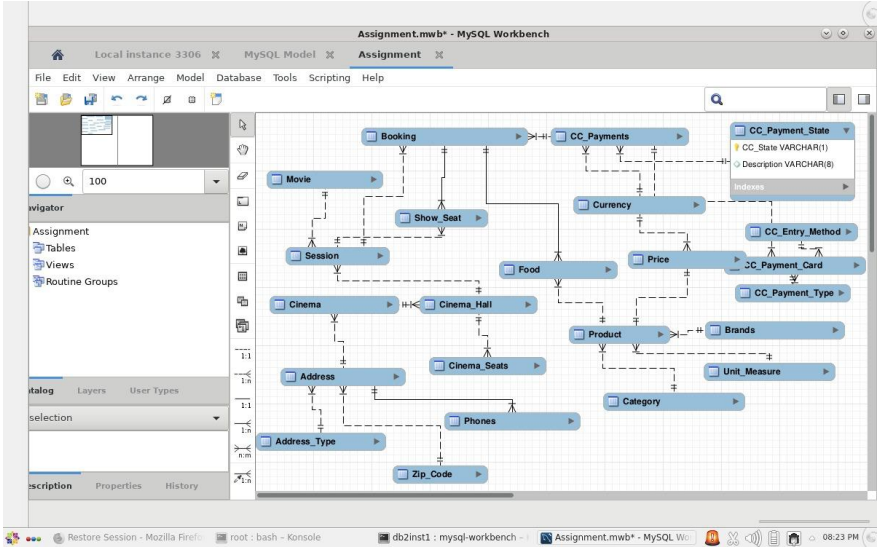

```
CREATE TABLE CC_Payments (
    CC_Payment_ID VARCHAR(6) NOT NULL,
    CC_PayTran_ID VARCHAR(7),
    Currency_ID VARCHAR(3),
    Amount NUMERIC(17, 15),
    CC_Payment_State VARCHAR(1),
    TimeCreated DATETIME,
 PRIMARY KEY(CC_Payment_ID)
);
INSERT INTO CC_Payments VALUES
    ('PY1000','CCPT001','EUR',12.100000000000001,'A','2021-12-08 00:00:00'),
    ('PY1001','CCPT002','EUR',31.900000000000002,'A','2021-12-08 00:00:00'),
    ('PY1002','CCPT003','EUR',22,'A','2021-12-08 00:00:00'),
    ('PY1003','CCPT004','EUR',36.300000000000004,'A','2021-12-08 00:00:00'),
    ('PY1004','CCPT005','EUR',36.300000000000004,'A','2021-12-08 00:00:00'),
    ('PY1005','CCPT006','EUR',14.3,'A','2021-12-08 00:00:00'),
    ('PY1006','CCPT007','EUR',26.400000000000002,'A','2021-12-08 00:00:00'),
    ('PY1007','CCPT008','EUR',36.300000000000004,'A','2021-12-08 00:00:00'),
```

CC_Payment_Type Table



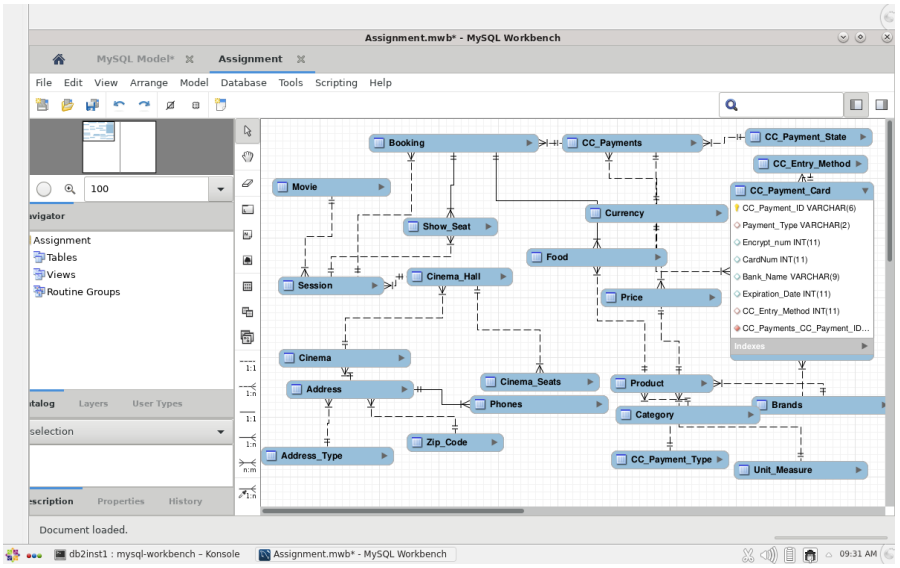

```
CREATE TABLE CC_Payment_Type (
    CC_Type VARCHAR(2) NOT NULL,
    Description VARCHAR(16),
 PRIMARY KEY (CC_Type)
);
INSERT INTO CC_Payment_Type VALUES
    ('MC','Master Card'),
    ('VS','Visa'),
    ('AM','American Express'),
    ('DS','Discover'),
    ('DC','Dinners Club'),
    ('BK','Other Bank Card');
```

CC_Payment_State Table




```
CREATE TABLE CC_Payment_State (
    CC_State VARCHAR(1) NOT NULL,
    Description VARCHAR(8),
PRIMARY KEY(CC_State)
);
INSERT INTO CC_Payment_State VALUES
    ('A','Approved'),
    ('F','Fail'),
    ('E','Expired'),
     ('C','Cancel');
```

CC_Payment_Card Table



CREATE TABLE CC_Payment_Card (

CC_Payment_ID VARCHAR(6) NOT NULL,

Payment_Type VARCHAR(2),

Encrypt_num INT,

CardNum INT,

Bank_Name VARCHAR(9),

Expiration_Date INT,

CC_Entry_Method INT,

PRIMARY KEY (CC_Payment_ID)

);

INSERT INTO CC_Payment_Card VALUES

('PY1000','MC',1,67419,'Santander',2025,1),

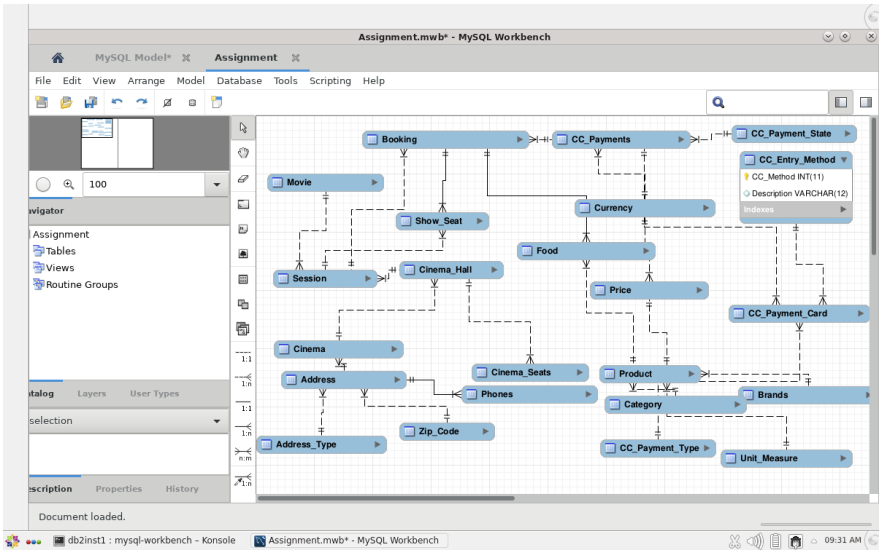
('PY1001','VS',0,46416,'Caixa',2023,1),

('PY1002','AM',1,47282,'BBVA',2027,2),

('PY1003','DS',0,74543,'Sabadell',2025,1),

('PY1004','DC',0,53367,'Abanca',2026,1),

CC_Entry_Method Table



CREATE TABLE CC_Entry_Method (

CC_Method INT NOT NULL,

Description VARCHAR(12),

PRIMARY KEY(CC_Method)

);

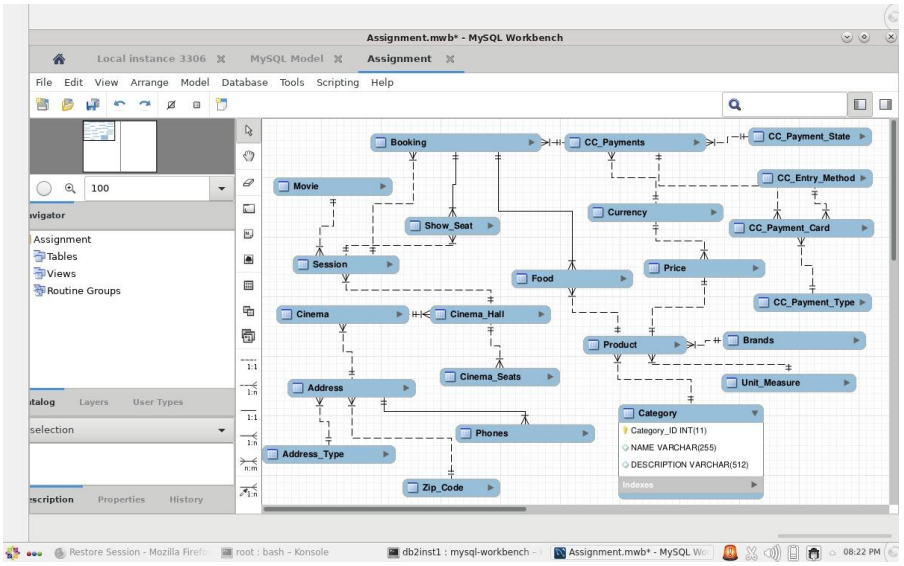
INSERT INTO CC_Entry_Method VALUES

(0,'Card Swiping'),

(1,'Card Dipping'),

(2,'Contactless');

Category Table



CREATE TABLE Category (

Category_ID INT NOT NULL,

NAME VARCHAR(255),

DESCRIPTION VARCHAR(512),

PRIMARY KEY (Category_ID)

);

INSERT INTO Category VALUES

(600,'FOOD','FOOD ITEMS'),

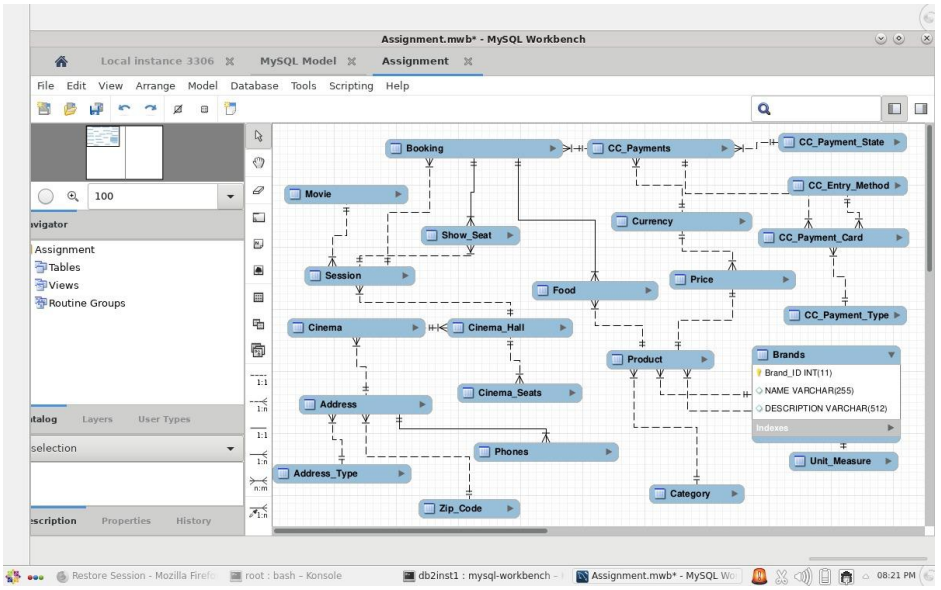
(601,'SWEET','SWEETS'),

(602,'SOFT DRINK','SOFT FIZZY DRINKS'),

(603,'WATER','FILTER WATER'),

(604,'COMBO','COMBO ITEMS');

Brands Table



CREATE TABLE Brands (

Brand_ID INT NOT NULL,

NAME VARCHAR(255),

DESCRIPTION VARCHAR(512),

PRIMARY KEY(Brand_ID)

);

INSERT INTO Brands VALUES

(500,'ACT II','ACTIVE POPCORN'),

(501,'M & M','M & M SWEET'),

(502,'COKE','COKE DRINKS'),

(503,'WATER','WATER DRINK'),

(504,'HERSHEY','HERSHEY SWEET'),

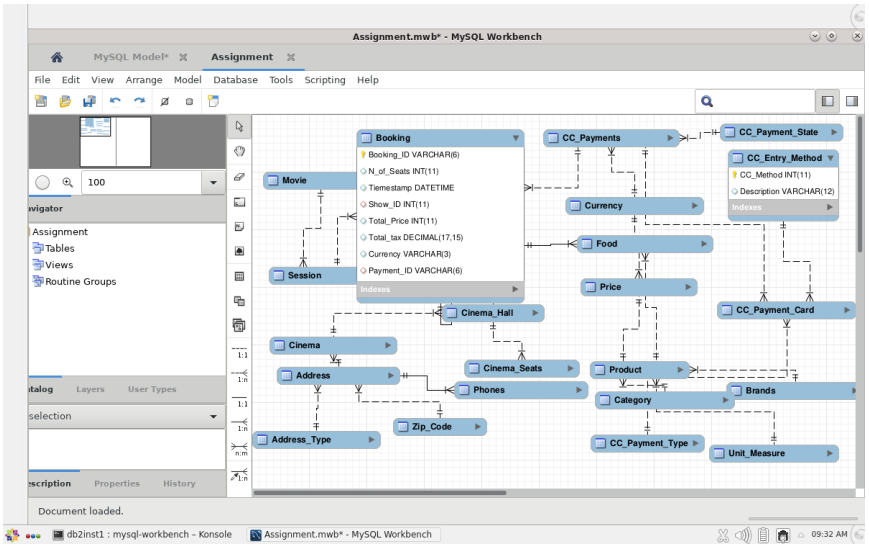
(505,'DORITOS','NACHOS DORITOS'),

(506,'HARIBO','HARIBO GUMMY BEAR'),

(507, 'HOTDOGS', 'HOTDOGS'),

(508, 'COMBO', 'COMBO');

Booking Table



CREATE TABLE Booking (

Booking_ID VARCHAR(6) NOT NULL,

N_of_Seats INT,

Tiemestamp DATETIME,

Show_ID INT,

Total_Price INT,

Total_tax NUMERIC(17, 15),

Currency VARCHAR(3),

Payment_ID VARCHAR(6),

PRIMARY KEY (Booking_ID)

);

INSERT INTO Booking VALUES

('ID1000',1,'2021-12-08 00:00:00',44197,34,37.400000000000006,'EUR','PY1000'),

('ID1001',1,'2021-12-08 00:00:00',44198,30,33,'EUR','PY1001'),

('ID1002',1,'2021-12-08 00:00:00',44199,13,14.3,'EUR','PY1002'),

('ID1003',1,'2021-12-08 00:00:00',44200,21,23.1,'EUR','PY1003'),

('ID1004',1,'2021-12-08 00:00:00',44201,12,13.200000000000001,'EUR','PY1004'),

('ID1005',1,'2021-12-08 00:00:00',44202,29,31.900000000000002,'EUR','PY1005'),

('ID1006',1,'2021-12-08 00:00:00',44203,12,13.200000000000001,'EUR','PY1006'),

('ID1007',1,'2021-12-08 00:00:00',44204,30,33,'EUR','PY1007'),

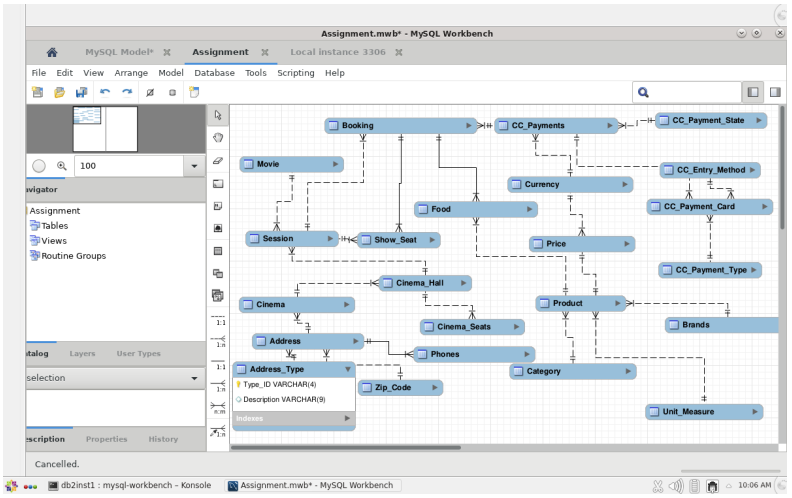
('ID1008',1,'2021-12-08 00:00:00',44205,33,36.300000000000004,'EUR','PY1008'),

('ID1009',1,'2021-12-08 00:00:00',44206,31,34.1,'EUR','PY1009'),

('ID1010',1,'2021-12-08 00:00:00',44207,25,27.500000000000004,'EUR','PY1010'),

('ID1011',1,'2021-12-08 00:00:00',44208,26,28.6,'EUR','PY1011'),

Address_Type Table



CREATE TABLE Address_Type (

Type_ID VARCHAR(4) NOT NULL,

Description VARCHAR(9),

PRIMARY KEY(Type_ID)

);

INSERT INTO Address_Type VALUES

('St','Street'),

('Ave','Avenue'),

('Blvd','Boulevard');

DDL SQL Foreign Keys creation

ALTER Queries

ALTER TABLE Cinema

ADD FOREIGN KEY (Address_ID)

REFERENCES Address (Address_ID);

ALTER TABLE Cinema_Hall

ADD FOREIGN KEY (Cinema_ID)

REFERENCES Cinema (Cinema_ID);

ALTER TABLE Cinema_Seats

ADD FOREIGN KEY (Cinema_Hall_ID)

REFERENCES Cinema_Hall (Cinema_Hall_ID);

ALTER TABLE Session

ADD FOREIGN KEY (Movie_ID)

REFERENCES Movie (Movie_ID);

ALTER TABLE Session

ADD FOREIGN KEY (Cinema_Hall_ID)

REFERENCES Cinema_Hall (Cinema_Hall_ID);

ALTER TABLE Address

ADD FOREIGN KEY (Type_ID)

REFERENCES Address_Type (Type_ID);

ALTER TABLE Address

ADD FOREIGN KEY (Zip_Code, Country)

REFERENCES Zip_Code (Zip_Code, Country);

ALTER TABLE Product

ADD FOREIGN KEY (Category_ID)

REFERENCES Category (Category_ID);

ALTER TABLE Product

ADD FOREIGN KEY (Size_ID)

REFERENCES Unit_Measure (Size_ID);

ALTER TABLE Product

ADD FOREIGN KEY (Brand_ID)

REFERENCES Brands (Brand_ID);

ALTER TABLE Price

ADD FOREIGN KEY (Currency_ID)

REFERENCES Currency (Currency_ID);

ALTER TABLE Show_Seat

ADD FOREIGN KEY (Show_ID)

REFERENCES Session (Show_ID);

ALTER TABLE Booking

ADD FOREIGN KEY (Show_ID)

REFERENCES Session (Show_ID);

ALTER TABLE Booking

ADD FOREIGN KEY (Payment_ID)

REFERENCES CC_Payments (CC_Payment_ID);

ALTER TABLE CC_Payments

ADD FOREIGN KEY (Currency_ID)

REFERENCES Currency(Currency_ID);

ALTER TABLE CC_Payments

ADD FOREIGN KEY (CC_Payment_State)

REFERENCES CC_Payment_State (CC_State);

ALTER TABLE CC_Payment_Card

ADD FOREIGN KEY (Payment_Type)

REFERENCES CC_Payment_Type (CC_Type);

ALTER TABLE CC_Payment_Card

ADD FOREIGN KEY (CC_Entry_Method)

REFERENCES CC_Entry_Method (CC_Method);

ALTER TABLE Food

ADD FOREIGN KEY (Booking_ID)

REFERENCES Booking (Booking_ID);

ALTER TABLE Food

ADD FOREIGN KEY (Product_ID)

REFERENCES Product (Product_ID);

ALTER TABLE Phones

ADD FOREIGN KEY (Address_ID)

REFERENCES Address (Address_ID);

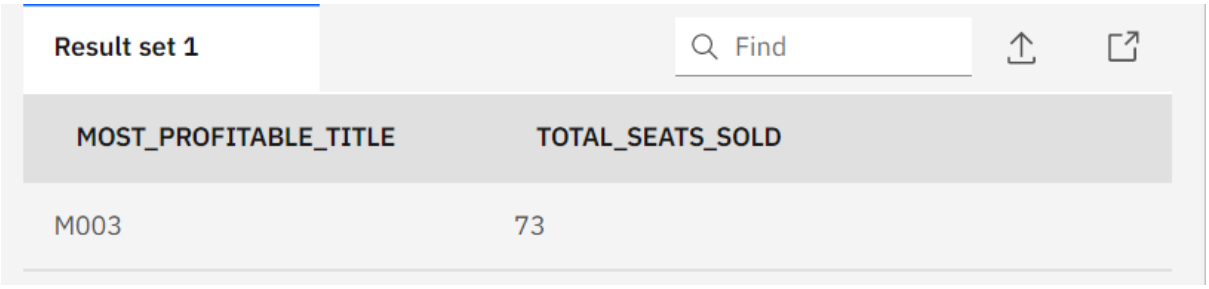
SQL Queries and their outputs

1) Which is the most profitable title?

Query:

```
SELECT M.Movie_ID AS MOST_PROFITABLE_TITLE, SUM(N_of_Seats) AS
TOTAL_SEATS_SOLD
FROM Movie AS M, Session AS S, Booking AS B
WHERE M.Movie_ID = S.Movie_ID AND S.Show_ID = B.Show_ID
GROUP BY M.Movie_ID
HAVING SUM(N_of_Seats) = (SELECT MAX(SUM_SEATS) FROM (SELECT
SUM(N_of_Seats) AS SUM_SEATS, M.Movie_ID AS MOVIES
FROM Movie AS M, Session AS S, Booking AS B
WHERE M.Movie_ID = S.Movie_ID AND S.Show_ID = B.Show_ID
GROUP BY M.Movie_ID));
```

Output:



2) Percentage of people that buy popcorn & soft drinks?

Ans:

```
SELECT SUM(N_of_Seats) AS TOTAL_NO_OF_PEOPLE,
COUNT(F.Product_ID) AS PEOPLE_BOUGHT_FOOD_SPFTDRINKS,
COUNT(F.Product_ID) * 100.0 / SUM(N_of_Seats) AS PERCENTAGE_PEOPLE
FROM Booking AS B
LEFT JOIN Food AS F
ON B.Booking_ID = F.Booking_ID
LEFT JOIN Product AS P
ON F.Product_ID= P.Product_ID
WHERE UPPER(P.Name) LIKE '%Fizzy Drinks%'
OR UPPER(P.Name) LIKE '%Popcorn%';
```

Output:

TOTAL_NO_OF_PEOPLE	PEOPLE_BOUGHT_FOOD_SPFTDRINKS	PERCENTAGE_PEOPLE
315	272	86.34920634920634920

3)At which session is there the most number of people.

Ans:

```
SELECT S.Show_ID AS SESSIONS_WITH_MOST_PEOPLE, SUM(N_of_Seats) AS
TOTAL_SEATS_SOLD
FROM Session AS S, Booking AS B
WHERE S.Show_ID = B.Show_ID
GROUP BY S.Show_ID
HAVING SUM(N_of_Seats) =
(SELECT MAX(SUM_SEATS)
FROM (SELECT SUM(N_of_Seats) AS SUM_SEATS,
S.Show_ID FROM Session AS S, Booking AS B
WHERE S.Show_ID = B.Show_ID
GROUP BY S.Show_ID));
```

Output:

SESSIONS_WITH_MOST_PEOPLE	TOTAL_SEATS_SOLD
44197	10

4)Show the revenue per session.

Ans:

```
SELECT S.Show_ID AS Session, SUM(Total_Price) AS REVENUE_PER_SESSION
FROM Session AS S, Booking AS B
WHERE S.Show_ID = B.Show_ID
GROUP BY S.Show_ID;
```

Output:

SESSION ↑↓	REVENUE_PER_SESSION
44197	234
44198	174
44199	175
44200	106
44201	90
44202	132
44203	106
44204	120
44205	123
44206	117
44207	122
44208	100

5) Own Query - What is the most preferred payment type while booking a ticket.

Query:

```
SELECT T.CC_Type AS MOST_COMMON_PAYMENT_TYPE, COUNT(*) AS
NUM_OF_BOOKING from Booking B, CC_Payments P, CC_Payment_Card C,
CC_Payment_Type T
WHERE B.Payment_ID = P.CC_Payment_ID
AND P.CC_Payment_ID = C.CC_Payment_ID
AND C.Payment_Type = T.CC_Type
GROUP BY T.CC_Type
HAVING COUNT(*) = (SELECT MAX(NUM_OF_BOOKING) FROM
(SELECT T.CC_Type, COUNT(*) AS NUM_OF_BOOKING
FROM Booking B, CC_Payments P, CC_Payment_Card C, CC_Payment_Type T
WHERE B.Payment_ID = P.CC_Payment_ID
AND P.CC_Payment_ID = C.CC_Payment_ID
AND C.Payment_Type = T.CC_Type
GROUP BY T.CC_Type
));
```

Output:

MOST_COMMON_PAYMENT_TYPE	NUM_OF_BOOKING
MC	68