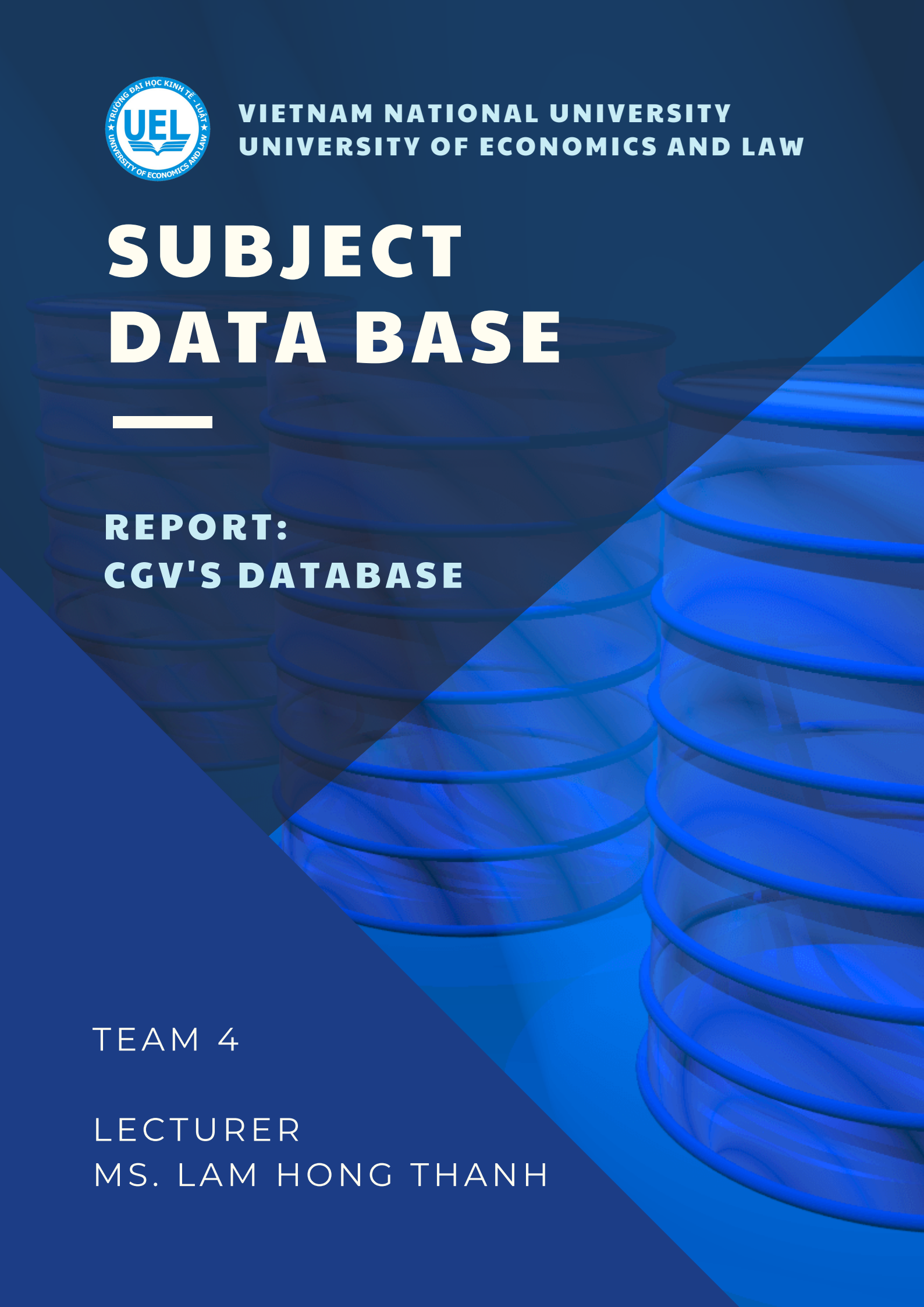


VIETNAM NATIONAL UNIVERSITY
UNIVERSITY OF ECONOMICS AND LAW

SUBJECT
DATA BASE

REPORT:
CGV'S DATABASE

TEAM 4

LECTURER
MS. LAM HONG THANH

FINAL REPORT

CGV'S DATABASE

No.	Team Member	Student code
1	Man Đắc Sang	K204061446
2	Thái Thiên Trúc	K204060310
3	Nguyễn Ngọc Kim Hoàng	K204061429
4	Ngô Song Tuyết Ngân	K204060305
5	Trần Minh Hoài	K204061428

TABLE OF CONTENTS

TABLE OF FIGURES

CHAPTER I: INTRODUCTION

1. Why CGV?

After nearly 3 years of Covid-19, our country has suffered many long-term as well as short-term social distancing periods. This tragic pandemic is also the reason that led to many limitations in human activities such as leisure, manufacturing and entertainment. But in recent days, since Covid has been controlled and most of the restriction rules have been completely removed, it's time for us to enjoy our lives, to entertain. Going to the cinema to watch some movies is one of the most low-cost entertainment activities chosen by a lot of people, especially young people. It was indicated in a statistic about Vietnamese watching behavior with 500,000 candidates that 84% of them have at least once gone to the cinema and more than half of them use cinema apps (52%). Moreover, on the ticket purchase procedure, 36% chose to book and purchase ticket(s) online. From the convenience of cinema applications and the need for cash-free payments, many firms have taken care of developing the application.

In Vietnam, CGV is ranked as the No.1 cinema and takes 66% of the movie theater proportion. CGV Cinemas is also the app with the most downloads and uses. It is the convenient features that this app has provided to users, more and more people trust to use this application.

2. Target

Improve the quality of business:

This database helps managers have a better view of the sales process to find out the advantages and disadvantages of the process and offer strategies for improvement.

Know customer’s behavior:

In addition, this database also helps managers know the decision-making trends of customers by collecting data, thereby planning to build marketing strategies to improve the services and bring a better experience to find more potential partners, filmmakers and customers. And that also leads to an increase the revenue.

3. Expected results

Design and build the general database model of the sales process of CGV cinemas.

CHAPTER II: BUSINESS MODEL INTRODUCTION

1. Objects participating in the model

Staff, bank, customer, CGV Cinemas application.

2. General description of company activities

CGV is a cinema system that attracts a large number of people, especially young people. CGV includes positions of departments working together: accounting department, human resources department, technical department, marketing/pr, customer service room.

The main activity of the CGV system is showing movies, selling movie tickets and accompanying items in compliance with regulations. CGV does business in CGV cinemas or on CGV apps and e-wallets.

In addition, CGV also has attractive incentive programs to attract customers:

Incentive programs such as discount tickets for customers carrying Student-Student cards or recently, discounts for young customers under 22 years old.

Periodic promotions such as the Thu Tu Vui Ve event applied to special screenings every Wednesday.

3. General business process

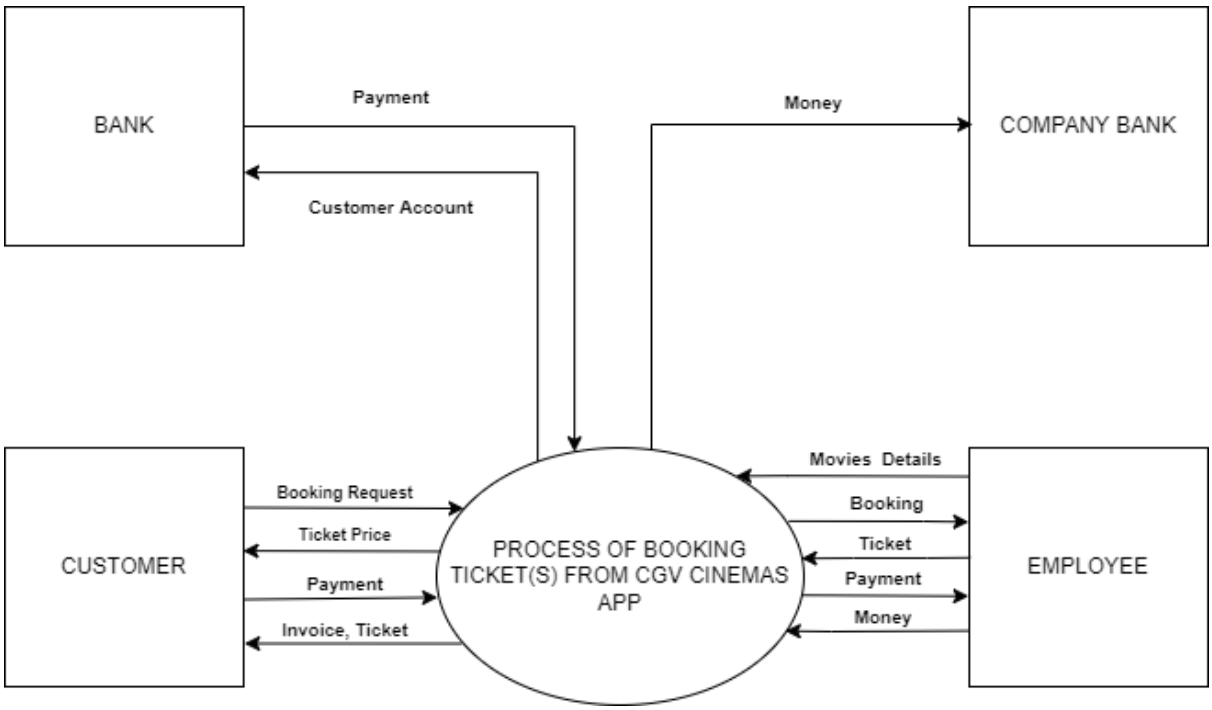


Figure 1. General business process diagram

Step 1:

Customers open the CGV app and pick which film they want to buy ticket(s). Then, they will show the available movie showtimes, theater, type of cinema rooms, seats, combos,...

Step 2:

Sales Departments control order information through Booking Database.

Step 3:

Booking Information will be converted into Invoice Information

4. Description of the main function of business processes

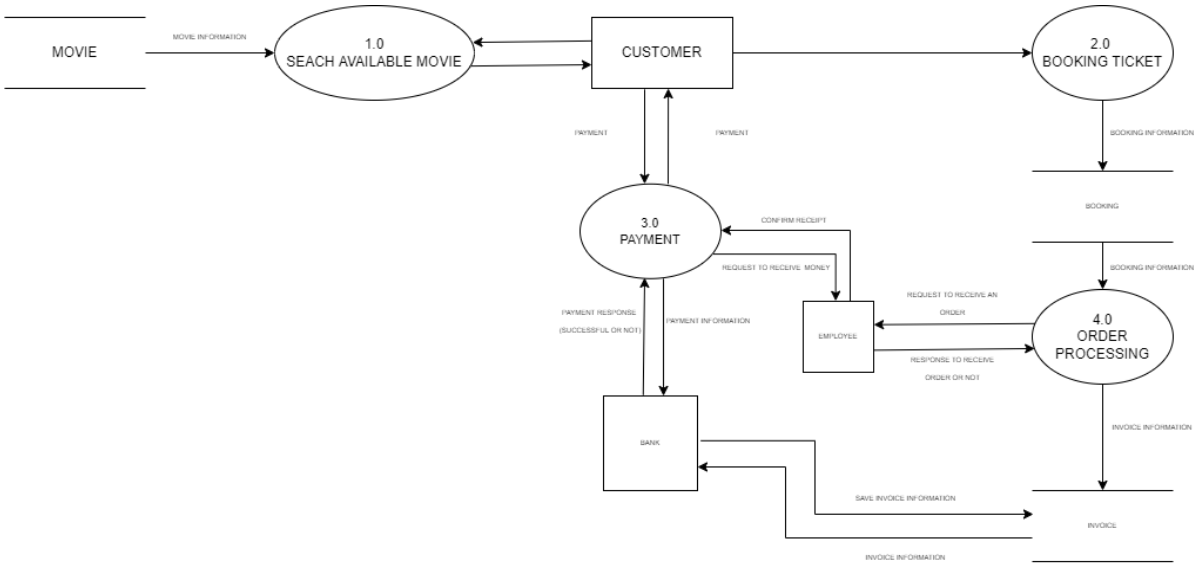
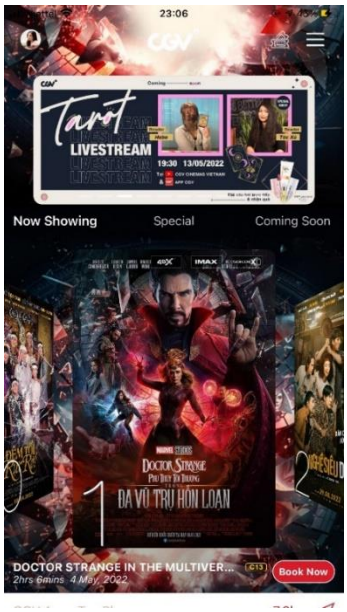


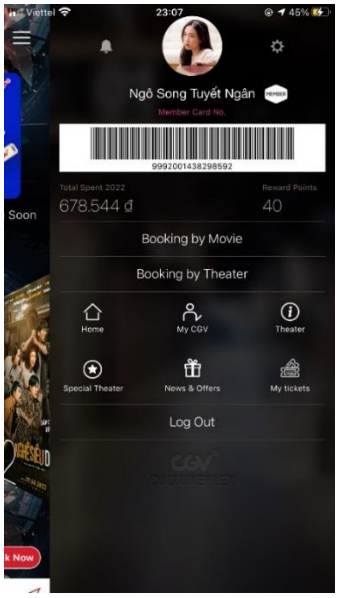
Figure 2. Booking movie process diagram

The process of customers looking for the movie:

Open the CGV app.

Swipe to choose one among available movies on the Home Screen. Or open the Menu Screen and select one of the functions, Booking by Movie and Booking by Theater.





Select the button Book Now to get the movie showtimes at theaters closest to customers' location or the recent theater customers have been, or customers can choose any available theaters.

The movie showtimes database will send movie information (movie name, showtime, cinema type) displayed on the app. Customers will choose the time, then select the seat.

Process customers booking:

Customers submit booking information on the CGV app. When the booking is successfully confirmed, customers' booking information, including Ticket information, Bank payment will be stored in the Booking database for control.

Customer payment process:

When customers place the booking, the system will send payment requests to customers through the app.

Customers can choose to pay through Card, Momo, ZaloPay, ShopeePay, VinIDPay. When the client pays successfully, the money will be transferred to the company's account. At the same time, the system will convert booking information into invoice information and store it in the Invoice database.

Order processing process:

The system receives booking information from the Booking database. When customers come and request to check in to enter the cinema, the employee will check the booking through customers' ticket number and create the invoice information of the order, which is the paper movie ticket.

Invoice information is stored in the Invoice database.

5. The main function of the app

The app is considered a miniature CGV with almost all the basic functions required, such as booking tickets, choosing seat positions, ordering snack combos, displaying time frames and movies being played premiered. In addition, you can also register for a membership card right on this application to accumulate points and experience exclusive member discount codes. Of course, you can pay quickly via bank card payment method. After successful booking, the application will display information for confirmation and code. What you need to do then is go to CGV and give it to the staff to confirm the valid ticket and enjoy the movie.

6. Data flow diagram for CGV

6.1 Context level DFD for CGV

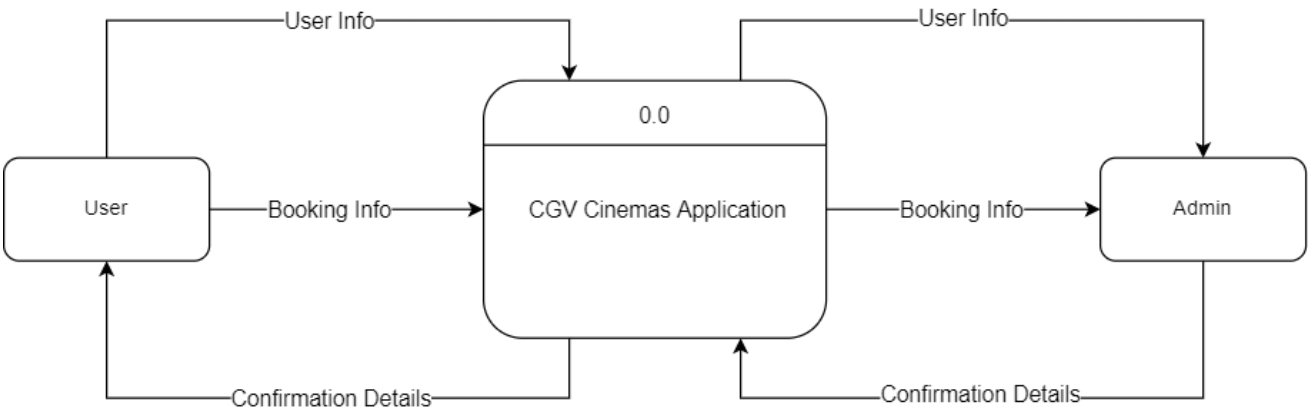


Figure 3. Context level DFD for CGV

The first level DFD of CGV App has shown the general process booking system monitoring.

Input and Output are shown as:

Input:

Booking information is input as the user's booking for the movie ticket(s).

Output:

Confirmation Details since the booking is succeed

6.2 The first level DFD for User

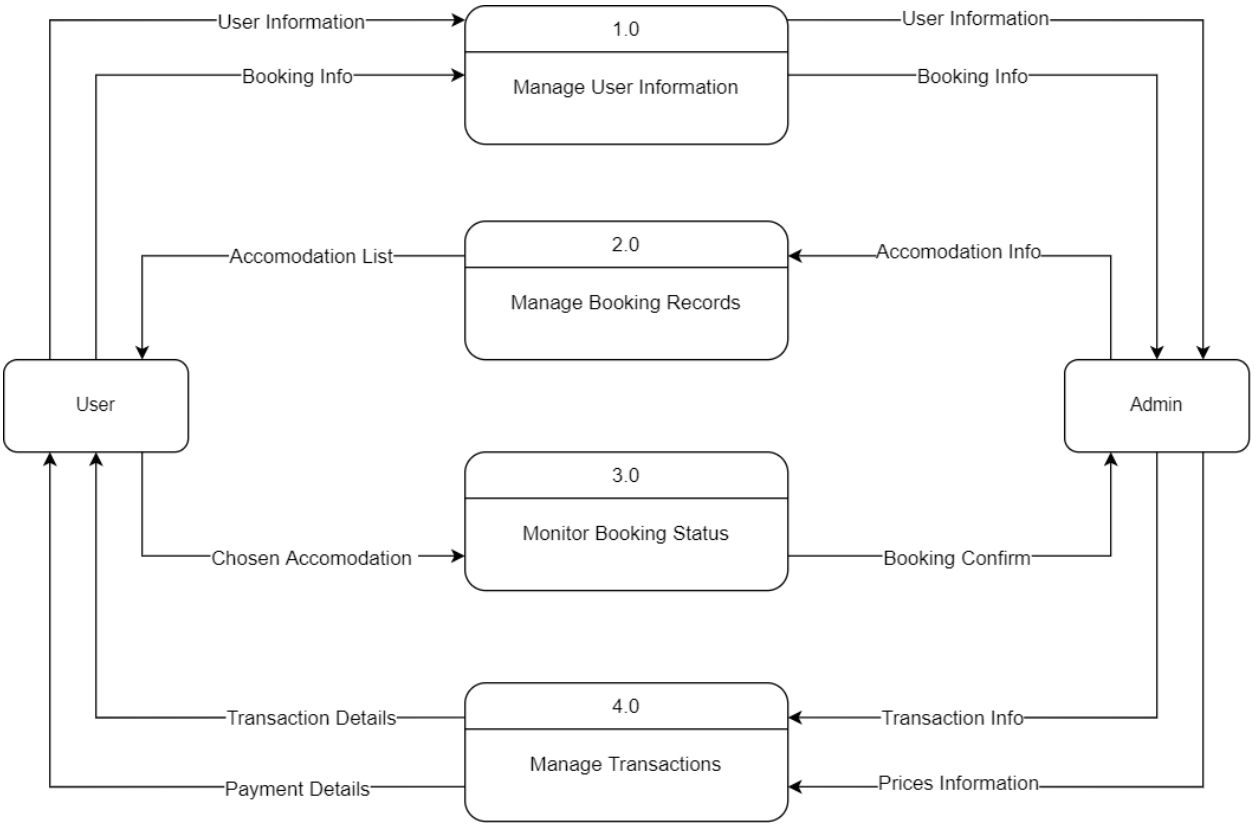


Figure 4. The first level DFD for user

This level provides more essential data to accommodate including:

- Booking Records
- Customer Information
- Revenue
- Transactions Records
- Date of Booking

CGV App DFD Level 1 will inform you about how the input is provided in the system. Next, it will shown the outputs this system gives.

CGV App has 5 main management levels, including:

- Login Management
- System User Management
- Movie Management
- Ticket Management
- Seat Management

CHAPTER III: LOGICAL DATABASE DESIGN

1. Identify and describe entities, define entity attributes

1. **Region**: RegionID, RegionName
Key attribute: RegionID
Not null attributes: RegionID
Single value attributes: RegionID, RegionName

2. **Agency:** AgencyID, AmanagementID, RegionID, Address
Key attribute: AgencyID
Foreign key attribute: AmanagementID,RegionID
Not null attributes: AgencyID, AmanagementID, RegionID, Address
Single value attributes: AgencyID, AmanagementID, RegionID, Address

3. **Team:** TeamName, RoomID, EmployeeID
Composite primary key form foreign keys: RoomID, EmployeeID
Not null attributes: RoomID
Single value attributes: TeamName, RoomID, EmployeeID

4. **Employee:** EmployeeID, EmLastname, EmFirstName, EmPhoneNumber, managementID, AgencyID
Key attribute: EmployeeID
Foreign key attribute: AgencyID
Not null attributes: EmployeeID, AgencyID
Single value attributes: EmployeeID, EmLastname, EmFirstName, EmPhoneNumber, managementID, AgencyID

5. **Customer:** CustomerID, CusLastName, CusFirstName, CusPhoneNumber, Student, MembershipID
Key attribute: CustomerID
Foreign key attribute: MembershipID
Not null attributes: CustomerID
Single value attributes: CusLastName, CusFirstName, CusPhoneNumber, Student, MembershipID

6. **Membership** : MembershipID, MembershipType
Key attribute: MembershipID
Not null attributes:MembershipID
Single value attributes: MembershipID, MembershipType

7. Left: LMembershipID, LEFT
primary key form foreign keys: LMembershipID
Single value attributes: LMembershipID, LEFT
Not null attributes: LMembershipID, LEFT

8. Joined: JMembershipID, JOIN, CouponID
primary key form foreign keys: JMembershipID
Foreign key attribute: CouponID
Single value attributes: JMembershipID, JOIN, CouponID
Not null attributes: JMembershipID, JOIN

9. Coupon: CouponID, couponName, Discount, MaximumUses, Code
Key attribute: CouponID
Not null attributes:CouponID, Discount, MaximumUses, Code
Single value attributes: CouponID, couponName, Discount, MaximumUses, Code

10. Order: OrderID, CustomerID, CouponID, InvoiceID
Key attribute: OrderID
Foreign key attribute: CustomerID, CouponID, InvoiceID
Not null attributes:OrderID, InvoiceID

11. Menu order combo: OrderID, ComboID, Note
Composite primary key form foreign keys: OrderID, ComboID

12. Menu food drink: ComboID, FoodDrinkID, Note
Composite primary key form foreign keys: ComboID, FoodDrinkID

13. Menu order FD: OrderID, FoodDrinkID, Note
Composite primary key form foreign keys: OrderID, FoodDrinkID

14. Ticket Type: TicketTypeID, TickeTypeName, PriceID
Key attribute: TicketTypeID
Foreign key attribute: TicketID, PriceID
Not null attributes: TicketTypeID, TickeTypeName, PriceID

15. Room: RoomID, EmployeeID, RoomName, No.Seat
Key attribute: RoomID
Not null attributes: RoomID, EmployeeID
Single value attributes: RoomName, No.Seat, RoomID
Foreign key attribute: EmployeeID

16. **Seat:** RoomID, SeatID, SeatName
Key attribute: SeatID
Not null attributes: SeatID, Name
Single value attributes: Name, SeatID, SeatName
Foreign key attribute: RoomID

17. **Invoice**: EmpoyeeID, InvoiceID, InvoiceDetail, InvoiceDay, PaymentID
Key attribute: InvoiceID
Not null attributes: EmpoyeeID, InvoiceID, PaymentID
Single value attributes: EmpoyeeID, InvoiceID, InvoiceDetail, InvoiceDay, PaymentID
Foreign key attribute: EmpoyeeID, PaymentID

18. **Payments:** PaymentID, PaymentMethod, Description
Key attribute: PaymentID
Not null attributes: PaymentID, PaymentMethod
Single value attributes: PaymentID, PaymentMethod, Description

19. **Combo:** ComboID, Name, PriceID
Key attribute: ComboID
Not null attributes: ComboID, PriceID
Foreign key attribute: PriceID
Single value attributes: ComboID, Name, PriceID

20. **Food & drink:** FoodDrinkID, Name, Size, PriceID
Key attribute: FoodDrinkID
Not null attributes: FoodDrinkID, Name, PriceID
Foreign key attribute: PriceID
Single value attributes: FoodDrinkID, Name, Size, PriceID

21. **Price:** PriceID, Price, Description
Key attribute: PriceID
Not null attributes: PriceID, Price
Single value attributes: PriceID, Price

22. **Ticket:** TicketID, FilmID, InvoiceID, CouponID, TypeID, SeatID
Key attribute: TicketID
Not null attributes: TicketID, FilmID, InvoiceID, TypeID, SeatID
Foreign key attribute: FilmID, InvoiceID, CouponID, TypeID

23. Film: FilmID, AllowedAge, Type, FilmName
Key attribute: FilmID
Not null attributes: FilmID, FilmName
Single value attributes: FilmID, AllowedAge, Type, FilmName

2. Identify relationships between entities

Name	Relationship	Type	Description
R1	REGION - AGENCY	1-N	Each REGION has many AGENCY, each AGENCY belongs to one REGION only
R2	AGENCY - EMPLOYEE	1-N	Each AGENCY has many EMPLOYEE, each EMPLOYEE belongs to one AGENCY only
R3	EMPLOYEE - ROOM	N-N, 1-1	Each ROOM is responsible by many EMPLOYEES, Each ROOM is managed by one EMPLOYEE only, each Management manage to one ROOM only
R4	ROOM - SEATS	1-N	Each ROOM has many SEATS, each SEATS belongs to one ROOM only
R5	ROOM - TICKET	1-N	Each ROOM has many TICKET, each TICKET belongs to one ROOM only
R6	TICKET - FILM	1-1	Each TICKET has one FILM only, each FILM belongs to many TICKET
R7	TICKET - TICKET TYPE	1-1	Each TICKET has one type, each TICKET TYPE belongs to all tickets
R8	EMPLOYEE - INVOICE	1-1	Each EMPLOYEE has one invoice, each INVOICE belongs to one EMPLOYEE only
R9	TICKET - INVOICE	1-N	Each INVOICE has many TICKET, each TICKET belongs to one INVOICE only
R10	INVOICE - PAYMENTS	1-N	Each INVOICE has one PAYMENT only, each PAYMENTS belongs to many invoice

3. Mapping the relationship & functional dependency diagram

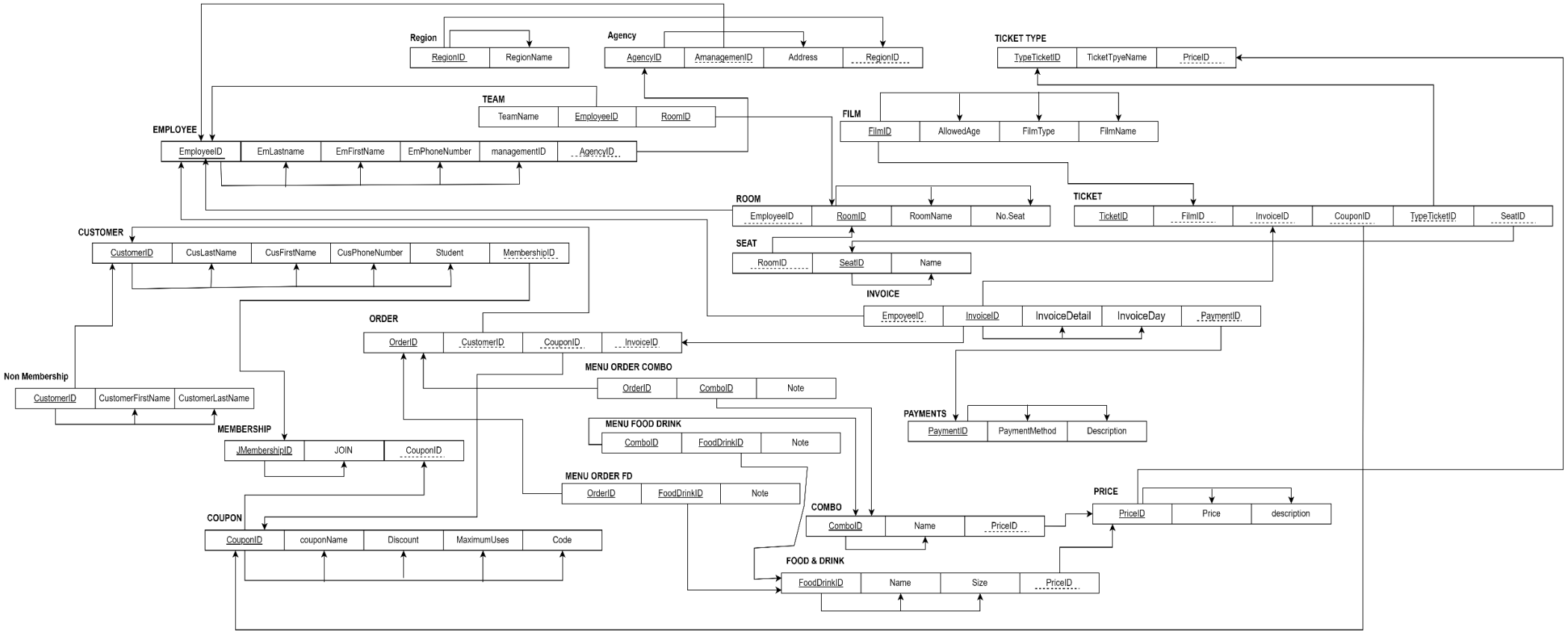


Figure 5. Mapping the relationship & functional dependency diagram

4. Logical ERD

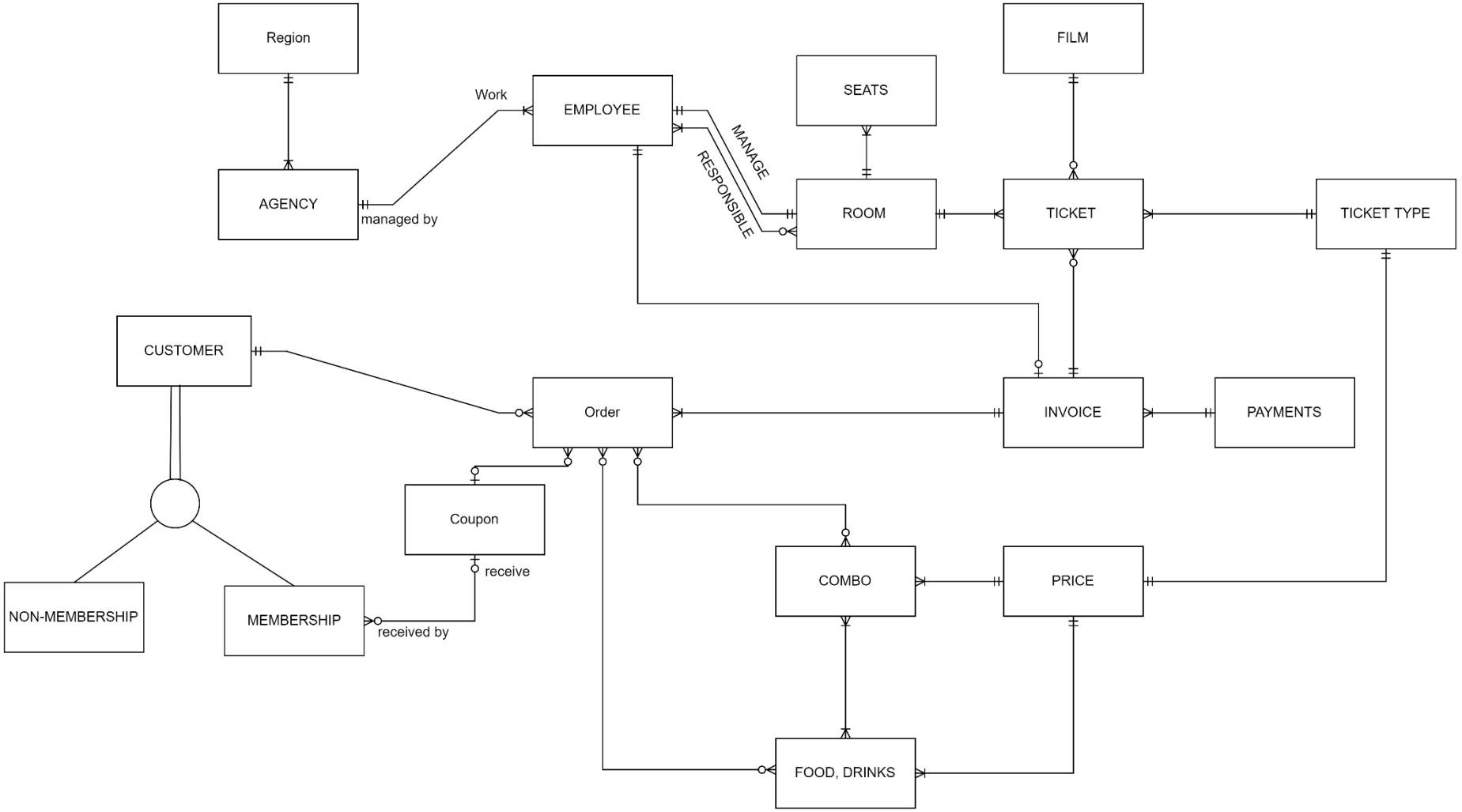


Figure 6. Logical ERD

CHAPTER V: PHYSICAL DATABASE DESIGN

1. Define data type

Table 1: Region

Column name	DataType	Key	Allow null
RegionID	Char(5)	PK	
RegionName	Nvarchar(25)		x

Table 2: Agency

Column name	DataType	Key	Allow null
AgencyID	Varchar(20)	PK	
ManagmentID	Varchar(20)	FK	
RegionID	Char(5)	FK	
Address	Nvarchar(200)		

Table 3: Team

Column name	DataType	Key	Allow null
TeamName	Nvarchar(200)		x
RoomID	Varchar(5)	PFK	
EmployeeID	Varchar(20)	PFK	

Table 4: Employee

Column name	DataType	Key	Allow null
EmployeeID	Varchar(20)	PK	
EmLastname	Nvarchar(50)		x
EmFirstName	Nvarchar(50)		x
EmPhoneNumber	Int		x
ManagementID	Varchar(20)		x

AgencyID	Varchar(20)	FK	

Table 5: Customer

Column name	DataType	Key	Allow null
CustomerID	Varchar(50)	PK	
CusLastName	Nvarchar(50)		x
CusFirstName	Nvarchar(50)		x
CusPhoneNumber	Int		x
Student	Char(1)		x
MembershipID	Varchar(50)	FK	

Table 6: Membership

Column name	DataType	Key	Allow null
MembershipID	Varchar(50)	PK	
MembershipType	char(10)		x

Table 7: LeftMB

Column name	DataType	Key	Allow null
LMembershipID	Varchar(50)	PFK	
LEFT	Int		

Table 8: Joined

Column name	DataType	Key	Allow null
JMembershipID	Varchar(50)	PFK	
JOINED	Int		
CouponID	Varchar(10)	FK	x

Table 9: Coupon

Column name	DataType	Key	Allow null
CouponID	Varchar(10)	PK	
couponName	Nvarchar(200)		x
Discount	Decimal(5,2)		
MaximumUses	INT		
Code	Varchar(10)		

Table 10: Order

Column name	DataType	Key	Allow null
OrderID	Varchar(50)	PK	
CustomerID	Varchar(50)	FK	x
CouponID	Varchar(10)	FK	x
InvoiceID	Varchar(200)	FK	

Table 11: Menu order combo

Column name	DataType	Key	Allow null
OrderID	Varchar(50)	PFK	
ComboID	Varchar(50)	PFK	
Note	Nvarchar(256)		x

Table 12: Menu food drink

Column name	DataType	Key	Allow null
ComboID	Varchar(50)	PFK	
FoodDrinkID	Varchar(50)	PFK	
Note	Nvarchar(256)		x

Table 13: Menu order food drink

Column name	DataType	Key	Allow null
OrderID	Varchar(50)	PFK	
FoodDrinkID	Varchar(50)	PFK	
Note	Nvarchar(256)		x

Table 14: Ticket Type

Column name	DataType	Key	Allow null
TicKetTypeID	Varchar(50)	PK	
PriceID	Varchar(50)	FK	
TicketTpyeName	Nvarchar(50)		

Table 15: Room

Column name	DataType	Key	Allow null
RoomID	Varchar(5)	PK	
EmployeeID	Varchar(20)	FK	
RoomName	Nvarchar(50)		x
NoSeat	Char(1)		x

Table 16: Seat

Column name	DataType	Key	Allow null
RoomID	Varchar(5)	PFK	
SeatID	Varchar(10)	PK	
Name	Nvarchar(50)		x

Table 17: Invoice

Column name	DataType	Key	Allow null
EmpoyeeID	Varchar(20)	FK	
InvoiceID	Varchar(200)	PK	
InvoiceDetail	Nvarchar(50)		x
InvoiceDay	DATE		x
PaymentID	Varchar(50)	FK	

Table 18: Payments

Column name	DataType	Key	Allow null
PaymentID	Varchar(50)	PK	
PaymentMethod	Nvarchar(50)		
Description	Nvarchar(256)		x

Table 19: Combo

Column name	DataType	Key	Allow null
ComboID	Varchar(50)	PK	
Name	Nvarchar(50)		x
PriceID	Varchar(50)	FK	

Table 20: Food & Drink

Column name	DataType	Key	Allow null
FoodDrinkID	Varchar(50)	PK	
Name	Nvarchar(50)		
Size	Nvarchar(5)		x
PriceID	Varchar(50)	FK	

Table 21: Price

Column name	DataType	Key	Allow null
PriceID	Varchar(50)	PK	
Price	Float		
Description	Nvarchar(256)		x

Table 22: Ticket

Column name	DataType	Key	Allow null
TicketID	Varchar(50)	PK	
FilmID	Varchar(50)	PFK	
InvoiceID	Varchar(200)	PFK	
CouponID	Varchar(10)	FK	x
TicKetTypeID	Varchar(50)	PFK	
SeatID	Varchar(10)	PFK	

Table 23: Film

Column name	DataType	Key	Allow null
FilmID	Varchar(50)	PK	
FilmName	Nvarchar(50)		
AllowedAge	int		x
FilmType	Nvarchar(50)		

2. Physical ERD

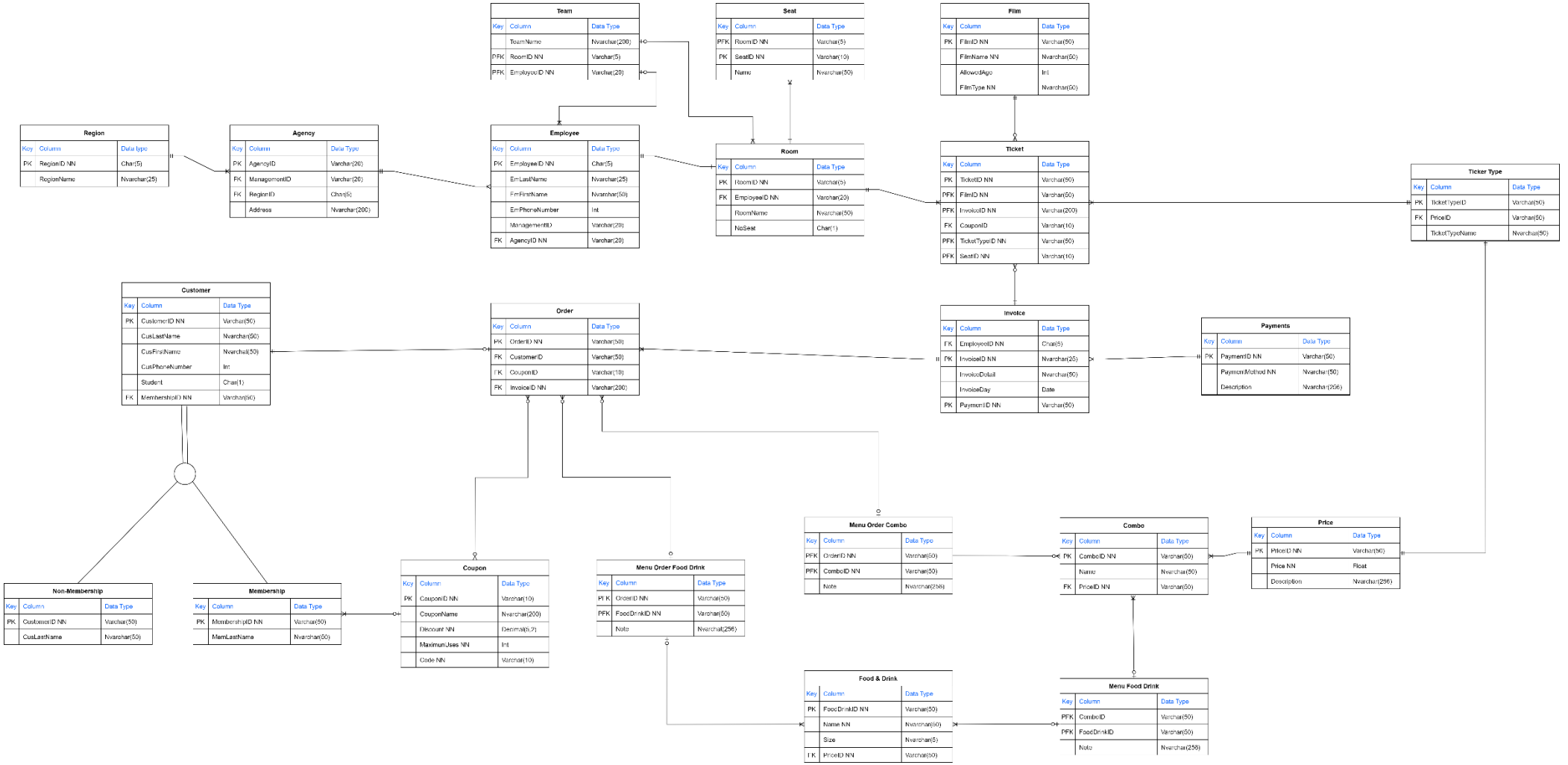


Figure 7. Physical ERD

3. Syntax of database creating

```
create database CGV
--region
create table region (
  region_id char(5) not null,
  region_name nvarchar(25),
  constraint region_pk primary key (region_id)
)
--coupon
create table coupon (
  coupon_id varchar(10) not null,
  coupon_name nvarchar(200),
  discount float not null,
  maximumuses int not null,
  code varchar(10) not null constraint coupon_pk primary key(coupon_id)
)
--18 payments
create table payments (
  payment_id varchar(50) not null,
  payment_method nvarchar(50) not null,
  description nvarchar(256) constraint payment_pk primary key(payment_id)
)
--23 film
create table film (
  film_id varchar(50) not null,
  film_name nvarchar(50) not null,
  allowed_age int,
  film_type nvarchar(50) not null constraint film_pk primary key (film_id)
)
--21 price
create table price (
  price_id varchar(50) not null,
  price float not null,
  description nvarchar(256),
  constraint price_pk primary key (price_id)
)
--2 agency
create table agency (
  agency_id varchar(20),
  region_id char(5),
  address nvarchar(200),
  constraint agency_pk primary key (agency_id),
  constraint agency_fk foreign key (region_id) references region(region_id)
)
--4 employee
create table employee (
  employee_id varchar(20) not null,
  em_lastname nvarchar(50),
```

```
  em_firstname varchar(50),
  em_phonenumber int,
  management_id varchar(20),
  agency_id varchar(20) constraint employees_pk primary key (employee_id),
  constraint employees_fk foreign key (agency_id) references
agency(agency_id)
)
--15 room
create table room (
  room_id varchar(5) not null,
  employee_id varchar(20) not null,
  room_name nvarchar(50),
  no_seat char(1) constraint room_pk primary key (room_id) constraint room_fk
foreign key (employee_id) references employee(employee_id)
)
--3 team
create table team (
  team_name nvarchar(200),
  room_id varchar(5) not null,
  employee_id varchar(20) not null,
  constraint team_pk primary key(room_id, employee_id),
  constraint team_fk1 foreign key (employee_id) references
employee(employee_id),
  constraint team_fk2 foreign key (room_id) references room(room_id)
)
--6 membership
create table membership (
  membership_id varchar(50) not null,
  membership_lastname nvarchar(50),
  constraint membership_pk primary key (membership_id)
)
--5 customer
create table customer (
  customer_id varchar(50) not null,
  cus_lastname nvarchar(50),
  cus_firstname nvarchar(50),
  cus_phonenumber int,
  student char(1),
  membership_id varchar(50) not null,
  constraint customer_pk primary key (customer_id),
  constraint customer_fk foreign key (membership_id) references
membership(membership_id)
)
--7 nonmembership
create table nonmembership (
  customer_id varchar(50) not null foreign key references
customer(customer_id),
  nonmembership_lastname nvarchar(50),
  primary key(customer_id)
```

```
)
--17 invoice
create table invoice (
  employee_id varchar(20) not null,
  invoice_id varchar(200) not null,
  invoice_detail nvarchar(50),
  invoice_day date,
  payment_id varchar(50) not null,
  constraint invoice_pk primary key (invoice_id),
  constraint invoice_fk1 foreign key (employee_id) references
employee(employee_id),
  constraint invoice_fk2 foreign key (payment_id) references
payments(payment_id),
)
--10 order
create table order_ (
  order_id varchar(50) not null,
  customer_id varchar(50),
  coupon_id varchar(10),
  invoice_id varchar(200) not null,
  constraint order_pk primary key (order_id),
  constraint order_fk1 foreign key (customer_id) references
customer(customer_id),
  constraint order_fk2 foreign key (coupon_id) references coupon(coupon_id),
  constraint order_fk3 foreign key (invoice_id) references
invoice(invoice_id)
)
--19 combo
create table combo (
  combo_id varchar(50) not null,
  name nvarchar(50),
  price_id varchar(50) not null,
  constraint combo_pk primary key (combo_id),
  constraint combo_fk foreign key (price_id) references price(price_id)
)
--11 menu order combo
create table moc (
  note nvarchar(256),
  order_id varchar(50) not null,
  combo_id varchar(50) not null,
  constraint moc_pk primary key(order_id, combo_id),
  constraint moc_fk1 foreign key (order_id) references order_(order_id),
  constraint moc_fk2 foreign key (combo_id) references combo(combo_id)
)
--20 food & drink
create table fooddrink (
  fooddrink_id varchar(50) not null,
  name nvarchar(50),
  size nvarchar(50),
```

```
  price_id varchar(50) not null,
  constraint fooddrink_pk primary key (fooddrink_id),
  constraint fooddrink_fk foreign key (price_id) references price(price_id)
)
--12 menu food drink
create table mfd (
  note nvarchar(256),
  fooddrink_id varchar(50) not null,
  combo_id varchar(50) not null,
  constraint mfd_pk primary key(fooddrink_id, combo_id),
  constraint mfd_fk1 foreign key (fooddrink_id) references
fooddrink(fooddrink_id),
  constraint mfd_fk2 foreign key (combo_id) references combo(combo_id)
)
--13 menu order food drink
create table mofd (
  note nvarchar(256),
  fooddrink_id varchar(50) not null,
  order_id varchar(50) not null,
  constraint mofd_pk primary key(fooddrink_id, order_id),
  constraint mofd_fk1 foreign key (fooddrink_id) references
fooddrink(fooddrink_id),
  constraint mofd_fk2 foreign key (order_id) references order_(order_id)
)
--14 ticket type
create table tickettype (
  tickettype_id varchar(50) not null,
  price_id varchar(50) not null,
  tickettype_name nvarchar(50) not null,
  constraint tickettype_pk primary key (tickettype_id),
  constraint tickettype_fk foreign key (price_id) references price(price_id)
)
--16 seat
create table seat (
  room_id varchar(5) not null,
  seat_id varchar(10) not null,
  name nvarchar(50) constraint seat_pk primary key (seat_id),
  constraint seat_fk foreign key (room_id) references room(room_id)
)
--22 ticket
create table ticket (
  ticket_id varchar(50) not null,
  film_id varchar(50) not null,
  invoice_id varchar(200) not null,
  coupon_id varchar(10),
  tickettype_id varchar(50) not null,
  seat_id varchar(10) not null,
  constraint ticket_pk primary key (ticket_id),
  constraint ticket_fk1 foreign key (film_id) references film(film_id),
```

```
  constraint ticket_fk2 foreign key (invoice_id) references
invoice(invoice_id),
  constraint ticket_fk3 foreign key (coupon_id) references coupon(coupon_id),
  constraint ticket_fk4 foreign key (tickettype_id) references
tickettype(tickettype_id)
)

--insert data--------------------------------
--1 region
INSERT INTO Region VALUES ('1', 'Bac');
INSERT INTO Region VALUES ('2', 'Trung');
INSERT INTO Region VALUES ('3', 'Nam');
INSERT INTO Region VALUES ('4', 'Tay');
--9 COUPON
INSERT INTO COUPON
VALUES
('CI001','Bronze1',0.01,'1','CCI1'),
('CI002','Bronze2',0.02,'1','CCI2'),
('CI003','Bronze3',0.03,'2','CCI3'),
('CI004','Silver1',0.04,'1','CCI4'),
('CI005','Silver2',0.05,'1','CCI5'),
('CI006','Silver3',0.06,'1','CCI6'),
('CI007','Gold1',0.07,'2','CCI7'),
('CI008','Gold2',0.08,'1','CCI8'),
('CI009','Gold3',0.09,'1','CCI9'),
('CI010','Diamond',0.1,'1','CCI10');
--18 Payments
insert into PAYMENTS
values
('PI01', 'BANK', 'CASH TRANSFERS'),
('PI02', 'PAY', 'CASH PAYMENT')
--23 film
INSERT INTO FILM
VALUES
('FLI01', 'Doctor Strange 2', 13, 'Science fiction'),
('FLI02', 'Ironman 1',  10, 'Science fiction'),
('FLI03', 'Ironman 2',  13, 'Science fiction'),
('FLI04', 'Ironman 3',  13, 'Science fiction'),
('FLI05', 'Avenger infinity war', 16, 'Science fiction'),
('FLI06', 'Avenger end game', 12, 'Science fiction'),
('FLI07', 'Spider-man far from home', 12, 'Science fiction'),
('FLI08', 'Spider-man no way home', 13, 'Science fiction'),
('FLI09', 'Bố Già', 18, 'Comedy genre'),
('FLI010', 'Chị 13',  18, 'Comedy genre')
--21 Price
INSERT INTO PRICE(PRICE_ID,PRICE)
VALUES
('PC01', 10),
('PC02' ,20),
```

```
('PC03' ,23),
('PC04', 4),
('PC05' ,23),
('PC06' ,45),
('PC07', 26),
('PC08', 1),
('PC09', 42),
('PC10', 11)
--2 Agency
INSERT INTO Agency VALUES ('AC01', '1', 'Cao Bang');
INSERT INTO Agency VALUES ('AC02', '1', 'Lang Son');
INSERT INTO Agency VALUES ('AC03', '1', 'Hai Duong');
INSERT INTO Agency VALUES ('AC04', '3', 'TP HCM');
INSERT INTO Agency VALUES ('AC05', '3', 'Dong Nai');
INSERT INTO Agency VALUES ('AC06', '3', 'Binh Duong');
INSERT INTO Agency VALUES ('AC07', '4', 'Ben Tre');
INSERT INTO Agency VALUES ('AC08', '2', 'Binh Dinh');
INSERT INTO Agency VALUES ('AC09', '3', 'Binh Phuoc');
INSERT INTO Agency VALUES ('AC010', '1', 'Bac Ninh');
--4 Employee
INSERT INTO Employee VALUES ('E001', 'Hoang', 'Nguyen', 101010, 'MI001',
'AC01');
INSERT INTO Employee VALUES ('E002', 'Sang', 'Man', 101012, 'MI002', 'AC02');
INSERT INTO Employee VALUES ('E003', 'Hoai', 'Tran', 1010103, 'MI003',
'AC03');
INSERT INTO Employee VALUES ('E004', 'Thanh', 'Ha', 101014, 'MI004',
'AC04' );
INSERT INTO Employee VALUES ('E005', 'Ngan', 'Ngo',101015, 'MI005' ,'AC05' );
INSERT INTO Employee VALUES ('E006', 'Truc', 'Thai', 101016,  'MI006',
'AC06');
INSERT INTO Employee VALUES ('E007', 'Nguyen', 'Tran', 10107, 'MI007',
'AC07');
INSERT INTO Employee VALUES ('E008', 'A', 'Nguyen', 10108, 'MI008', 'AC08');
INSERT INTO Employee VALUES ('E009', 'B', 'Nguyen', 10109, 'MI009', 'AC09');
INSERT INTO Employee VALUES ('E010', 'C', 'Nguyen', 1010101 ,'MI010',
'AC010');
--15  ROOM
INSERT INTO ROOM(ROOM_ID, EMPLOYEE_ID, ROOM_NAME) VALUES('R1',  'E002',
'Phòng 1');
INSERT INTO ROOM(ROOM_ID, EMPLOYEE_ID, ROOM_NAME)
VALUES('R2',  'E004',   'Phòng 2');
INSERT INTO ROOM(ROOM_ID, EMPLOYEE_ID, ROOM_NAME)
VALUES('R3',  'E005',   'Phòng 3');
INSERT INTO ROOM(ROOM_ID, EMPLOYEE_ID, ROOM_NAME)
VALUES('R4',  'E003',   'Phòng 4');
INSERT INTO ROOM(ROOM_ID, EMPLOYEE_ID, ROOM_NAME)
VALUES('R5',  'E001',   'Phòng 5');
INSERT INTO ROOM(ROOM_ID, EMPLOYEE_ID, ROOM_NAME)
VALUES('R6',  'E007',   'Phòng 6');
```

```
INSERT INTO ROOM(ROOM_ID, EMPLOYEE_ID, ROOM_NAME)
VALUES('R7',  'E006',   'Phòng 7');
INSERT INTO ROOM(ROOM_ID, EMPLOYEE_ID, ROOM_NAME)
VALUES('R8',  'E008',   'Phòng 8');
INSERT INTO ROOM(ROOM_ID, EMPLOYEE_ID, ROOM_NAME) VALUES('R9',  'E010',
'Phòng 9');
INSERT INTO ROOM(ROOM_ID, EMPLOYEE_ID, ROOM_NAME) VALUES('R10', 'E009',
'Phòng 10');
--3 Team
INSERT INTO Team VALUES ('Team 1', 'R1', 'E002');
INSERT INTO Team VALUES ('Team 2', 'R2', 'E004');
INSERT INTO Team VALUES ('Team 3', 'R3', 'E005');
INSERT INTO Team VALUES ('Team 4', 'R4', 'E003');
INSERT INTO Team VALUES ('Team 5', 'R5', 'E001');
INSERT INTO Team VALUES ('Team 6', 'R6', 'E007');
INSERT INTO Team VALUES ('Team 7', 'R7', 'E006');
INSERT INTO Team VALUES ('Team 8', 'R8', 'E008');
INSERT INTO Team VALUES ('Team 9', 'R9', 'E010');
INSERT INTO Team VALUES ('Team 10', 'R10', 'E009');
--6 MEMBERSHIP
INSERT INTO MEMBERSHIP
VALUES
  ('MEM01','C'),
  ('MEM02','A'),
  ('MEM03','U'),
  ('MEM04','P'),
  ('MEM05','Q');
--5.Bảng Customer
INSERT INTO Customer VALUES ('CUS01', 'C', 'Nguyen', 10101, 'X', 'MEM01');
INSERT INTO Customer VALUES ('CUS02', 'S', 'Man', 101231, 'X', 'MEM03');
INSERT INTO Customer VALUES ('CUS03', 'A', 'Tran', 141424, 'X', 'MEM02');
INSERT INTO Customer VALUES ('CUS04', 'E', 'Ha', 213123, 'X','MEM03');
INSERT INTO Customer VALUES ('CUS05','Y', 'Ngo', 144343,' ','MEM05');
INSERT INTO Customer VALUES ('CUS06', 'U', 'Thai', 1231235,'X', 'MEM03');
INSERT INTO Customer VALUES ('CUS07', 'V', 'Tran', 132123,' ','MEM03');
INSERT INTO Customer VALUES ('CUS08', 'P', 'Nguyen', 1312415, 'X', 'MEM04');
INSERT INTO Customer VALUES ('CUS09', 'Q', 'Nguyen', 4567551, '', 'MEM05');
INSERT INTO Customer VALUES ('CUS10', 'M', 'Nguyen', 144441,'','MEM04');
--7 nonmembership
INSERT INTO NONMEMBERSHIP
VALUES
('CUS02', 'S'),
('CUS04', 'E'),
('CUS05', 'Y'),
('CUS07', 'V'),
('CUS10', 'M')
--17 Invoice
insert into Invoice
values
```

```
('E001', 'IV01', 'Doctor Strange 2',   convert(datetime,'18-06-12 10:34:09
PM',5), 'PI01'),
('E002', 'IV02', 'Ironman 1',               convert(datetime,'18-06-12
10:34:09 PM',5),  'PI01'),
('E003', 'IV03', 'Bố Già',                  convert(datetime,'18-06-12
10:34:09 PM',5),  'PI02'),
('E004', 'IV04', 'Ironman 3',               convert(datetime,'18-06-12
10:34:09 PM',5),  'PI02'),
('E005', 'IV05', 'Avenger infinity war',    convert(datetime,'18-06-12
10:34:09 PM',5),  'PI01'),
('E001', 'IV06', 'Avenger end game',        convert(datetime,'18-06-12
10:34:09 PM',5),  'PI01'),
('E002', 'IV07', 'Spider-man far from home', convert(datetime,'18-06-12
10:34:09 PM',5),  'PI02'),
('E003', 'IV08', 'Spider-man no way home',   convert(datetime,'18-06-12
10:34:09 PM',5),  'PI01'),
('E004', 'IV09', 'Ironman2',                convert(datetime,'18-06-12
10:34:09 PM',5),  'PI01'),
('E005', 'IV10', 'Chị 13',                  convert(datetime,'18-06-12
10:34:09 PM',5),  'PI02')
--10 ORDER
INSERT INTO ORDER_ VALUES ('OR01',  'CUS03',  'CI009',  'IV01');
INSERT INTO ORDER_ VALUES ('OR02',  'CUS02',  'CI002',  'IV02');
INSERT INTO ORDER_ VALUES ('OR03',  'CUS01',  'CI008',  'IV03');
INSERT INTO ORDER_ VALUES ('OR04',  'CUS08',  'CI004',  'IV07');
INSERT INTO ORDER_ VALUES ('OR05',  'CUS05',  'CI005',  'IV05');
INSERT INTO ORDER_ VALUES ('OR06' , 'CUS06',  'CI006',  'IV01');
INSERT INTO ORDER_ VALUES('OR07', 'CUS10',  'CI007',  'IV09');
INSERT INTO ORDER_ VALUES ('OR08',  'CUS04',  'CI003',  'IV03');
INSERT INTO ORDER_ VALUES ('OR09',  'CUS09',  'CI001',  'IV08');
INSERT INTO ORDER_ VALUES ('OR10',  'CUS07',  'CI004',  'IV05');
--19 Combo
insert into Combo
values
('1', 'Ve + Nuoc',          'PC01'),
('2', 'Ve + Nuoc + Bap',      'PC03'),
('3', 'Ve + Nuoc + Trai cay', 'PC07'),
('4', 'Ve + Nuoc + My',       'PC09')
--11 MOC
INSERT INTO MOC(ORDER_ID, COMBO_ID)
VALUES('OR01',  1),
  ('OR02',  2),
  ('OR03',  4),
  ('OR04',  2),
  ('OR05',  4),
  ('OR06',  3),
  ('OR07',  1),
  ('OR08',  3),
  ('OR09',  2),
```

```
  ('OR10',  1)
--20 Food & Drink
insert into FOODDRINK
values
('FDI01', '7 UP',       'Min',  'PC04'),
('FDI02', 'Pessi',      'Max',  'PC04'),
('FDI03', 'Cocacola',   'Min',  'PC04'),
('FDI04', 'Bắp',        'Min',  'PC01'),
('FDI05', 'Mỳ',         'Min',  'PC02')
--12 MFD
INSERT INTO MFD(COMBO_ID,FOODDRINK_ID) VALUES(1,  'FDI05');
INSERT INTO MFD(COMBO_ID,FOODDRINK_ID) VALUES (2, 'FDI04');
INSERT INTO MFD(COMBO_ID,FOODDRINK_ID) VALUES(3,  'FDI03');
INSERT INTO MFD(COMBO_ID,FOODDRINK_ID) VALUES(4,  'FDI01');
INSERT INTO MFD(COMBO_ID,FOODDRINK_ID) VALUES(1,  'FDI02');
INSERT INTO MFD(COMBO_ID,FOODDRINK_ID) VALUES(3,  'FDI02');
INSERT INTO MFD(COMBO_ID,FOODDRINK_ID) VALUES(4,  'FDI03');
INSERT INTO MFD(COMBO_ID,FOODDRINK_ID) VALUES(2,  'FDI01');
INSERT INTO MFD(COMBO_ID,FOODDRINK_ID) VALUES(3,  'FDI05');
INSERT INTO MFD(COMBO_ID,FOODDRINK_ID) VALUES(4,  'FDI04');
--13 MOFD
INSERT INTO MOFD(ORDER_ID, FOODDRINK_ID)
VALUES('OR05',  'FDI05'),
('OR07',  'FDI04'),
('OR03',  'FDI03'),
('OR10',  'FDI01'),
('OR01',  'FDI02'),
('OR06',  'FDI02'),
('OR02',  'FDI03'),
('OR08',  'FDI01'),
('OR09',  'FDI05'),
('OR09',  'FDI04')
--14 TICKETTYPE
INSERT INTO TICKETTYPE(TICKETTYPE_ID, PRICE_ID, TICKETTYPE_NAME)
VALUES('TK01',  'PC05', 'Ve don'),
('TK02',  'PC07', 'Combo ve nuoc'),
('TK03',  'PC09', 'Combo ve nuoc do an')
--16 Seat
insert into SEAT values('R1', 'A2', 'SINGLE');
insert into SEAT values('R2', 'A1', 'SINGLE');
insert into SEAT values('R3', 'A3', 'COUPLE');
insert into SEAT values('R4', 'A4', 'COUPLE');
insert into SEAT values('R6', 'A5', 'SINGLE');
insert into SEAT values('R5', 'B1', 'SINGLE');
insert into SEAT values('R7', 'B2', 'COUPLE');
insert into SEAT values('R9', 'B3', 'COUPLE');
insert into SEAT values('R8', 'B4', 'COUPLE');
insert into SEAT values('R10', 'B5', 'COUPLE');
--22 Ticket
```

```
INSERT INTO TICKET VALUES ('T01', 'FLI01', 'IV01', 'CI003', 'TK01', 'A1');
INSERT INTO TICKET VALUES ('T02', 'FLI02', 'IV02', 'CI003','TK03', 'A2');
INSERT INTO TICKET VALUES ('T03', 'FLI03', 'IV03','CI003','TK02', 'A3');
INSERT INTO TICKET VALUES('T04', 'FLI04', 'IV04', 'CI001', 'TK02', 'A4');
INSERT INTO TICKET VALUES('T05', 'FLI05', 'IV05', 'CI002', 'TK01', 'A5');
INSERT INTO TICKET VALUES('T06', 'FLI06', 'IV06', 'CI003', 'TK03', 'B1');
INSERT INTO TICKET VALUES('T07', 'FLI07', 'IV07','CI003 ', 'TK02', 'B2');
INSERT INTO TICKET VALUES('T08', 'FLI08', 'IV08', 'CI003','TK02', 'B3');
INSERT INTO TICKET VALUES('T09', 'FLI09', 'IV09', 'CI003','TK03', 'B4');
INSERT INTO TICKET VALUES('T10', 'FLI010', 'IV10', 'CI003', 'TK02', 'B5');
```

CHAPTER VI: QUERY

```
--SELECT
--1
SELECT *  FROM FILM
--2
SELECT TICKET_ID, TICKETTYPE_ID FROM TICKET
WHERE COUPON_ID = 'CI001'
--3
SELECT REGION_ID, ADDRESS FROM Agency
GROUP BY REGION_ID, ADDRESS
--4
SELECT REGION_ID, ADDRESS FROM Agency
GROUP BY REGION_ID, ADDRESS
HAVING REGION_ID ='3'
--5
SELECT REGION_ID, ADDRESS FROM Agency
ORDER BY REGION_ID
--FUNCTION
--1
SELECT SUM(PRICE) FROM PRICE
WHERE PRICE > 4
--2
SELECT AVG(PRICE) FROM PRICE
WHERE PRICE > 7
--3
SELECT MIN(PRICE) FROM PRICE
WHERE PRICE > 6
--4
SELECT MAX(PRICE) FROM PRICE
WHERE PRICE > 10
--Join
SELECT A.ADDRESS ,'Mien '+ r.REGION_NAME  AS MIEN
FROM AGENCY AS A
JOIN REGION AS R
ON A.REGION_ID = R.REGION_ID
WHERE A.AGENCY_ID = 'AC09'
```

```sql
-- Sub query
SELECT EMPLOYEE_ID, EM_FRISTNAME +' '+ EM_LASTNAME AS NAME  FROM EMPLOYEE
WHERE AGENCY_ID =
(
SELECT A.AGENCY_ID FROM AGENCY AS A
JOIN REGION AS R
ON A.REGION_ID = R.REGION_ID
WHERE A.AGENCY_ID = 'AC09'
)
--SET
--1 EXCEPT
SELECT REGION_ID
FROM AGENCY
EXCEPT
SELECT REGION_ID
FROM REGION
WHERE REGION_NAME = 'Nam'
--2 UNION
SELECT PRICE_ID FROM PRICE
  WHERE PRICE = 4
UNION
SELECT PRICE_ID FROM PRICE
  WHERE PRICE = 10
UNION
SELECT PRICE_ID FROM PRICE
  WHERE PRICE = 45
UNION
SELECT PRICE_ID FROM PRICE
  WHERE PRICE = 1
--3 INTERSECT
SELECT REGION_ID
FROM AGENCY
INTERSECT
SELECT REGION_ID
FROM REGION
WHERE REGION_NAME = 'Nam'

--STORED PROCEDURE
CREATE PROCEDURE PAY
@CUPON FLOAT
AS
SELECT @CUPON * PRICE AS 'TỔNG TIỀN ' FROM PRICE
WHERE PRICE_ID = 'PC07'
GO
EXEC PAY 0.2
```

REFERENCES

Asia Plus Inc. (2022, February 25). *Survey about how Vietnamese enjoy cinema*. Telefilm. Retrieved from https://telefilm.vn/vietnamese-cinema-watching-behavior-118.html