HW2B

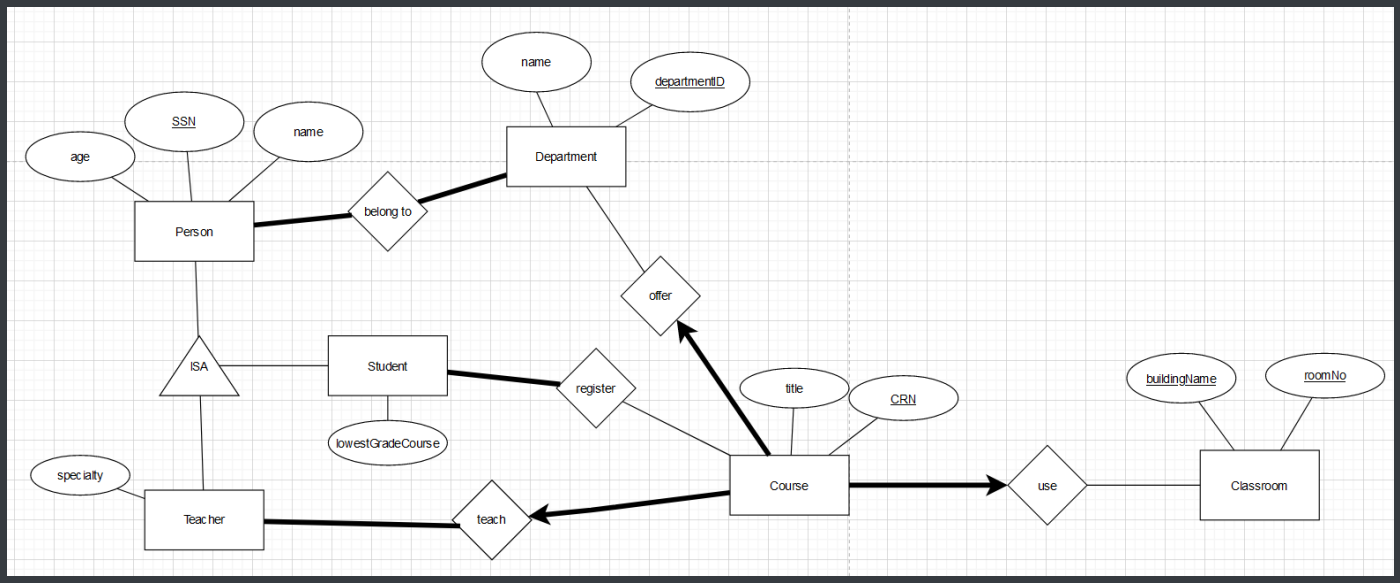
Note:

開頭大寫英文名詞是指 entity（ISA、CRN、SSN 除外，ISA 是 relationship，CRN 和 SSN 是 attribute），開頭小寫英文動詞是指 relationship，開頭小寫名詞是 attribute

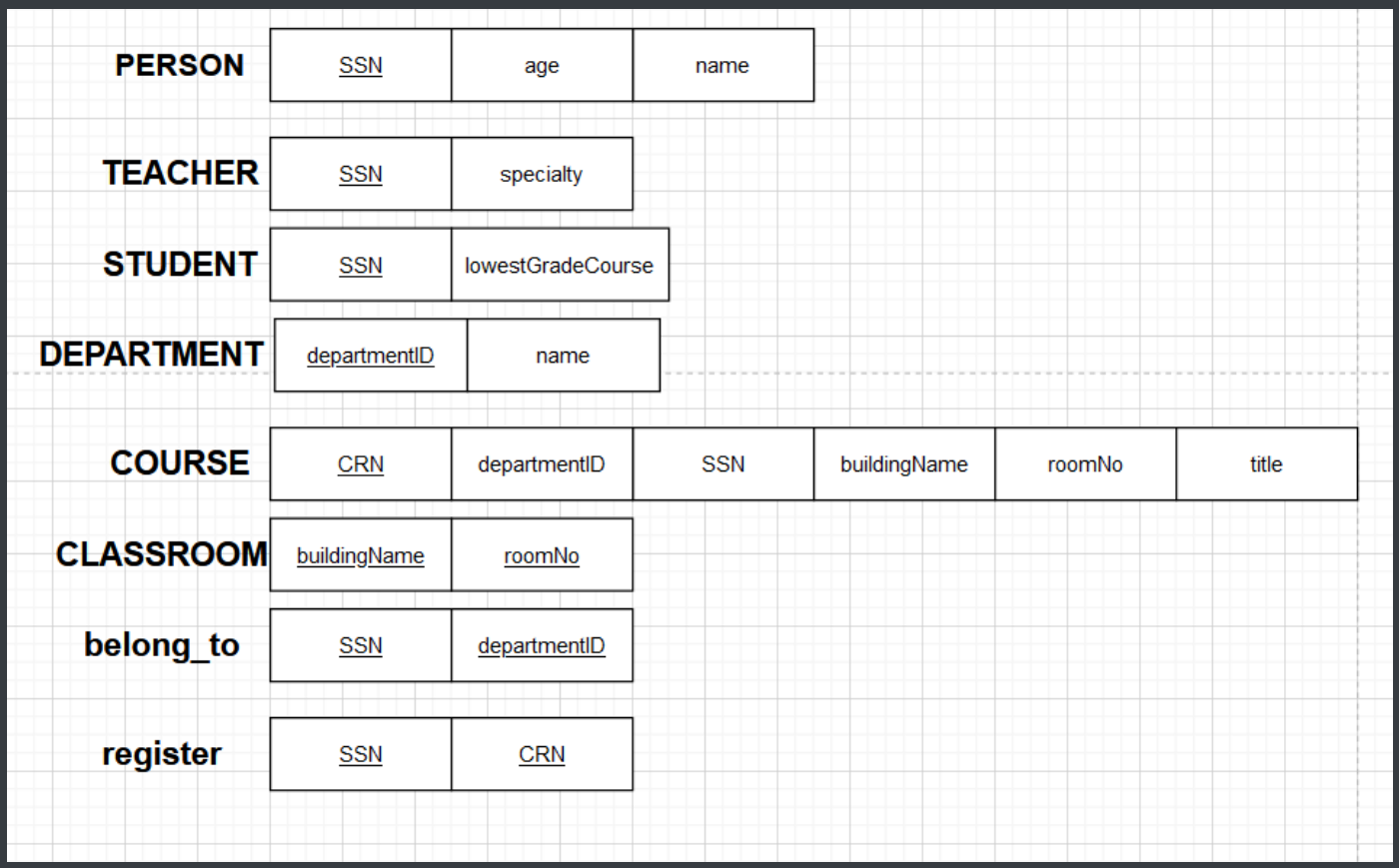
3 .

A .

ER model:



Relational schema:



邏輯設計說明：

Teacher 我有設置額外 attribute: specialty, Student 我有設置額外 attribute: lowestGradeCourse 因為把 Person 抽象了出來。

Person 要有 SSN、age 和 name。

Teacher 要有 specialty，剩餘 attributes 可從 Person 存取。

Student 要有 lowestGradeCourse，剩餘 attributes 可從 Person 存取。

Course 要有 title 和 CRN。

Department 要有 departmentID 和 name。

Classroom 要有 buildingName 和 roomNo。

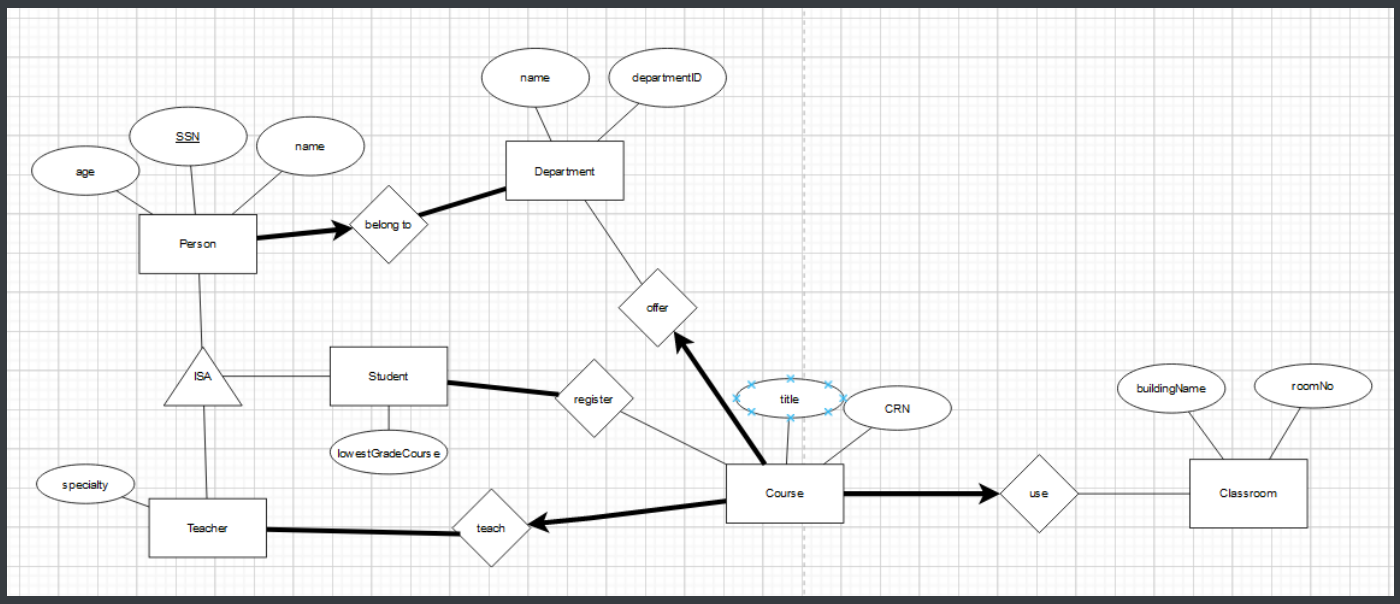
Person 可以泛指 Teacher 和 Student，所以可以看到 Teacher 和 Student 有一個 SSN，該 SSN 和 Person 的 SSN 一致，當 query Teacher 或 Student 的時候需要存取兩張 table 才能完成 query。

Course 和 Teacher 有 many-to-one teach relationship，所以 Course 有一個 SSN foreign key 指向 Teacher 的 SSN。Course 和 department 有 many-to-one offer relationship，所以 Course 有一個 departmentID foreign key 指向 Department 的 departmentID。Course 和 Classroom 有 many-to-one use relationship，所以 Course 有 buildingName foreign key 和 roomNo foreign key 指向 Classroom 的 buildingName 和 roomNo。

由於 Person 可以隸屬多個 Department，Department 也能有多人隸屬於他，因此 Person 和 Department 有一個 many-to-many 的關係，需要用到一個 belong_to table 來記錄 Person 的

SSN 和 Department 的 departmentID。
由於 Student 可以修習多個 Course，Course 也能被多人修習，因此 Student 和 Course 有一個 many-to-many 的關係，需要用到一個 register table 來記錄 Student 的 SSN 和 Course 的 CRN。

B .
ER model:



Relational schema:

PERSON	SSN	age	name	departmentID		
TEACHER	SSN	specialty				
STUDENT	SSN	lowestGradeCourse				
DEPARTMENT	departmentID	name				
COURSE	CRN	departmentID	SSN	buildingName	roomNo	title
CLASSROOM	buildingName	roomNo				
register	SSN	CRN				

邏輯設計說明：
Teacher 我有設置額外 attribute: specialty, Student 我有設置額外 attribute: lowestGradeCourse 因為把 Person 抽象了出來。

Person 要有 SSN、age 和 name。
Teacher 要有 specialty，剩餘 attributes 可從 Person 存取。
Student 要有 lowestGradeCourse，剩餘 attributes 可從 Person 存取。
Course 要有 title 和 CRN。
Department 要有 departmentID 和 name。
Classroom 要有 buildingName 和 roomNo。
Person 可以泛指 Teacher 和 Student，所以可以看到 Teacher 和 Student 有一個 SSN，該 SSN 和 Person 的 SSN 一致，當 query Teacher 或 Student 的時候需要存取兩張 table 才能完成 query。
Course 和 Teacher 有 many-to-one teach relationship，所以 Course 有一個 SSN foreign key 指向 Teacher 的 SSN。Course 和 department 有 many-to-one offer relationship，所以 Course 有一個 departmentID foreign key 指向 Department 的 departmentID。Course 和 Classroom 有 many-to-one use relationship，所以 Course 有 buildingName foreign key 和 roomNo foreign key 指向 Classroom 的 buildingName 和 roomNo。
由於 Person 最多隸屬於一個 Department，因此 Person 有一個 departmentID foreign key 指向 Department 的 departmentID。
由於 Student 可以修習多個 Course，Course 也能被多人修習，因此 Student 和 Course 有一個 many-to-many 的關係，需要用到一個 register table 來記錄 Student 的 SSN 和 Course 的 CRN。

4 .

A.

```sql
CREATE TABLE Person (
    name VARCHAR(50) NOT NULL,
    age INT NOT NULL,
    SSN VARCHAR(50) NOT NULL,
    PRIMARY KEY(SSN)
);

explain Person;
```

Result Grid | Filter Rows: | Export:

Field	Type	Null	Key	Default	Extra
name	varchar(50)	NO		NULL	
age	int	NO		NULL	
SSN	varchar(50)	NO	PRI	NULL	

Person 有三個不可以為 null 的 attributes，分別是 name、age、SSN，SSN 是 Person 的 primary key。

B .

```sql
CREATE TABLE Movie (
    Release_year INT NOT NULL,
    Rating INT NOT NULL,
    Title VARCHAR(50) NOT NULL,
    PRIMARY KEY(Title)
);

explain Movie;
```

Result Grid | Filter Rows: | Export: | Wra

Field	Type	Null	Key	Default	Extra
Release_year	int	NO		NULL	
Rating	int	NO		NULL	
Title	varchar(50)	NO	PRI	NULL	

Movie 有三個不可以為 null 的 attributes，分別是 Release_year、Rating、Title， Title 是 Movie 的 primary key。

C .

```sql
CREATE TABLE Art (
    SSN VARCHAR(50) NOT NULL,
    Title VARCHAR(50) NOT NULL,
    PRIMARY KEY(SSN, Title),
    FOREIGN KEY(SSN) REFERENCES Person(SSN),
    FOREIGN KEY(Title) REFERENCES Movie(Title)
);

explain Art;
```

Result Grid | Filter Rows: | Export: | Wrap Cell C

Field	Type	Null	Key	Default	Extra
SSN	varchar(50)	NO	PRI	NULL	
Title	varchar(50)	NO	PRI	NULL	

Art 有兩個不可以為 null 的 attributes，分別是 SSN 和 Title，SSN 和 Title 是 Art 的 primary key，SSN 是 foreign key 指向 Person 的 SSN，Title 是 foreign key 指向 Movie 的 Title。

D .

```
31 •  INSERT INTO Person (name, age, SSN)
32    VALUES ("John", 58, "1234567");
33
34 •  SELECT * FROM Person;
35
```

Result Grid | Filter Rows: | Edit:

	name	age	SSN
▶	John	58	1234567

新增一個 tuple: ("John", 58, "1234567") 到 Person。

E .

```
36 •  DROP TABLE Art;
37 •  DROP TABLE Movie;
38
```

Output

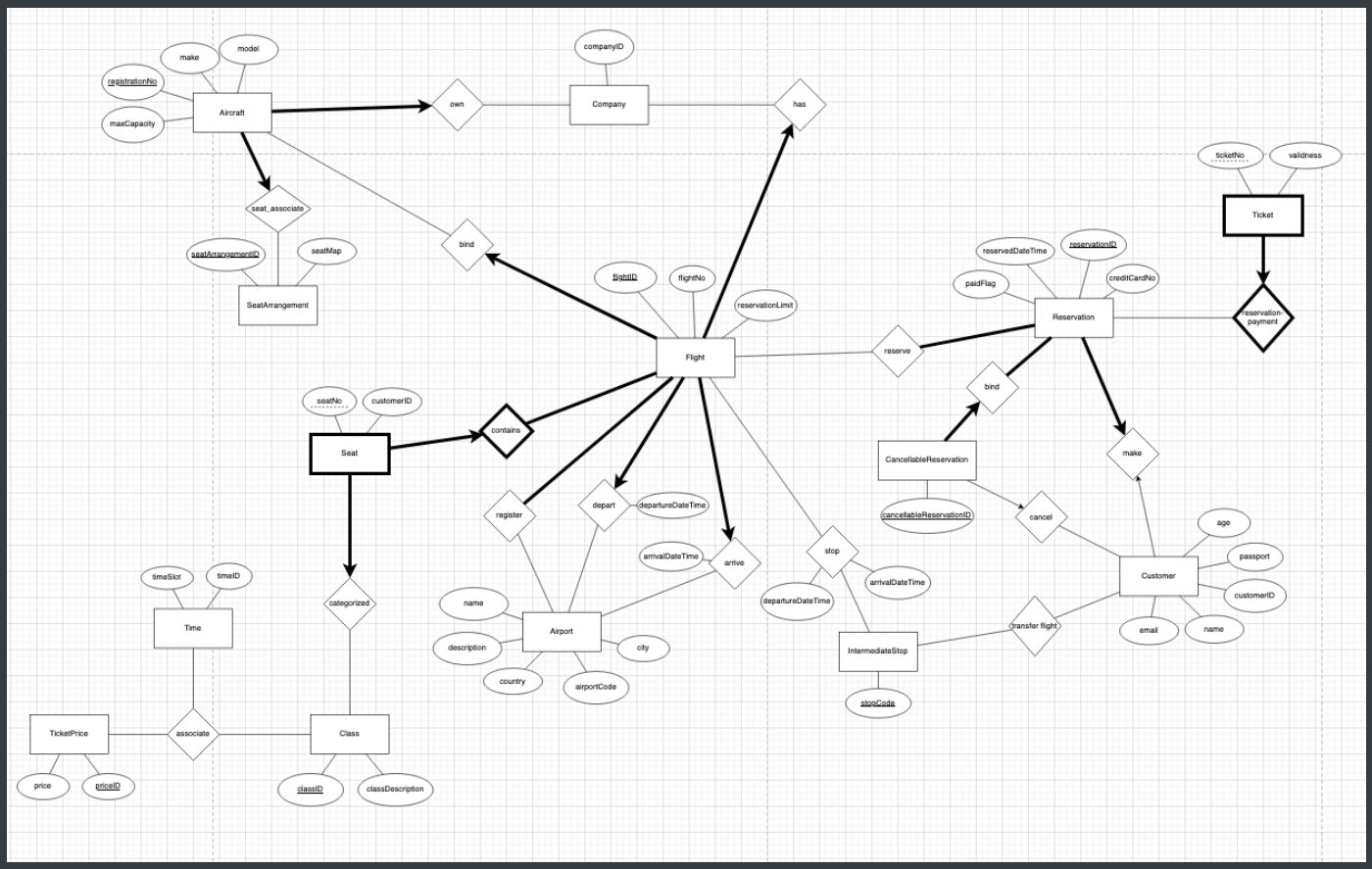
Action Output

#	Time	Action
1	21:21:27	DROP TABLE Art
2	21:21:30	DROP TABLE Movie

由於 Art 有一個 foreign key 指到 Movie 的 Table，因此需要先刪掉 Art table 才能刪掉 Movie table。

5B.

ER diagram:



設計邏輯說明:

```
Each company has an ID and owns many
aircrafts. Each aircraft has a registration number,
make, model, and the maximum capacity of
passengers.
```

根據以上，Company 有 ID，跟 Aircraft 有一個 one-to-many has relationship, Company 是 partial participation 因為公司可能一架 Aircraft 都沒有，而 Aircraft 是 total participation 因為 Aircraft 一定屬於某家公司，這邊排除私人 Aircraft。Aircraft 有 registrationNo, make, model, maxCapacity。

```
Each company has many flights. Flight has a
flight number that can be classified into domestic
or international routes. Each flight departs from
and arrives at registered airports.
```

根據以上，Flight 和 Company 有一個 many-to-one has relationship，Flight 是 total participation 因為航班應該都要屬於某家公司，Company 是 partial participation 因為公司可能基於某些原因沒有任何航班。Flight 有 flightNo，敘述只說明 flightNo 可以用來分辨國內或國外航班，所以這邊增加了一個 flightID 用來當作 flight 的 primary key。Flight 和 Airport 有一個 many-to-one depart、many-to-one arrive、many-to-many register relationship。在 depart relationship 中，Flight 是 total participation 因為 Flight 一定要在某個 airport 起飛，Airport 則是 partial participation 因為 Airport 可能一架飛機都沒有。在 arrive relationship 中，Flight 是 total participation 因為 Flight 最終一定要在某個 airport 降落，Airport 則是 partial participation 因為 Airport 可能不接受航班降落。在 register relationship 中，Flight 是 total participation 因為 Flight 一定要和某些 Airport 註冊後才能起飛或降落，Airport 則是 partial participation 因為 Airport 可能沒有一家航班註冊。

```
An airport also has an airport code, description
name, city and country information. Flight may
stop at some intermediate stops. Customers
could take a transfer flight from intermediate
stops
```

根據以上，Airport 有 airportCode，description、name, city, country。Flight 和 intermediateStop 有一個 many-to-many stop relationship，Flight 和 intermediateStop 都是 partial participation 因為 Flight 可能是直飛航班，intermediaStop 不一定有任何航班使用。Customer 和 intermediateStop 有一個 many-to-many transfer flight relationship，Customer 和 intermediateStop 都是 partial participation 因為 Customer 可能預訂的航班不需要轉機，intermediateStop 不一定提供所有轉機航班轉機服務。

```
Each flight has three seat classes: First Class,
Business Class, and Economy class.
For each flight, the system should store a list of
available seats. The seat arrangements are
associated with aircrafts. Each seat can only be
reserved to one customer. Ticket prices are
associated with classes and time.
```

根據以上，我把 Seat 當作是 Flight 的 weak entity 因為 Seat 在多個航班看來是重複的東西比如座位1號、2號、3號等等。Flight 和 Seat 有一個 one-to-many contains relationship，Flight 和 Seat 都是 total participation 因為 Flight 一定要有座位給乘客搭乘且這是 weak entity 的特性，Seat 一定屬於某些航班。此外，Seat 和 Class 有一個 many-to-one categorized relationship 因為 Seat 分為了三種等級，分別是 First Class, Business Class 和 Economy Class，Seat 是 total participation 因為 Seat 一定被分類到其中一個等級，Class 則是 partial participation 因為有些航班只有三種等級中的兩種比如經濟艙。再來，Class 的價位會隨著時間不同而有所不同，所以 Class、Time、TicketPrice 有一個 many-to-many associate relationship，Class、Time、TicketPrice 都是 partial participation 因為某些 Class 可能沒有提供預訂，Time 可能不涵蓋24小時，TicketPrice 有些價位可能沒有用到。Aircraft 和 SeatArrangement 有 many-to-one associate relationship，Aircraft 是 total participation 因為 Aircraft 一定有對應的座位分佈圖，而 SeatArrangement 是 partial participation 因為不是所有座位分佈圖都會被使用。這邊我有幫 SeatArrangement 設置兩個屬性，分別是 seatArrangementID 和 seatMap，seatMap 就是分佈圖。

```
Customer can make reservation without
reserving seats.
```

根據以上，我幫 Seat 新增一個 customerID attribute，這個 attribute 可為 null，null 表示乘客在預訂航班的時候沒有選擇座位，非 null 則有選擇座位。

```
Each flight allows 20% overbooking.
```

根據以上，幫 Flight 新增一個 reservationLimit attribute。

```
The reservation system should keep the
following information: 1) a unique identifier for
each customer. This identifier is assigned
automatically once customers make a new
reservation in the system; 2) customer
information; 3) flight information; 4) the time that
reservation was made; 5) a credit-card number
associated with the reservation;6) A flag which indicates whether this
particular
reservation is paid or not. When a reservation is
```

```
paid it is transformed automatically to a ticket.
The reservation can be cancelled before flight
departure, however, the ticket is still valid (to
exchange some products).
Each reservation might be associated with many
flights.
```

根據以上，Reservation 有 reservationID(unique identifier)，reservedDateTime, creditCardNo. Reservation 和 Flight 有 many-to-many reserve relationship，Reservation 是 total participation 因為 Reservation 一定和某些航班綁定在一起，而 Flight 是 partial participation 因為不是所有航班都會被 reserve。Reservation 和 Customer 有 one-to-one make relationship，Reservation 是 total participation 因為 Reservation 一定是由一些 Customer 所完成，而 Customer 是 partial participation 因為 Customer 註冊航班預訂系統後可能沒有完成任何 Reservation，但 Customer 記錄還是需要被記錄。Reservation 和 CancellableReservation 有一個 one-to-many bind relationship，Reservation 和 CancallableReservation 都是 total participation 因為 Reservation 一定和某些 CancellableReservation 綁定。 CancellableReservation 和 Customer 有 many-to-one cancel relationship，CancellableReservation 和 Customer 都是 partial participation 因為預訂的航班可能已經起飛了所以不能取消預訂，Customer 不一定會取消預訂。我把 Ticket 當作是 Reservation 的 weak entity 因為 ticketNo 是重複的，所以需要搭配 owner Reservation 才能分辨 Ticket。Reservation 和 Ticket 有 one-to-many reservation-payment relationship，Reservation 是 partial participation 因為有些 Reservation 並未付款所以沒有對應的 Ticket，而 Ticket 是 total participation 因為 ticket 一定由某個 Reservation 付款得到且這是 weak entity 的特性。我有幫 Ticket 設置一個 validness 用來判斷 Ticket 是否有效。

Company:

```
create table Company(
    companyID int not null auto_increment,
    primary key(companyID)
);
explain Company;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
companyID	int	NO	PRI	NULL	auto_increment

Company 只有一個 int 型別不為 null 自動遞增的 companyID 且為 primary key。

SeatArrangement:

```
create table SeatArrangement(
    seatArrangementID int not null auto_increment,
    seatMap blob not null,
    primary key(seatArrangementID)
);
explain SeatArrangement;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
seatArrangementID	int	NO	PRI	NULL	auto_increment
seatMap	blob	NO		NULL	

SeatArrangement 有一個 int 形態不為 null 自動遞增的 seatArrangementID 且為 primary key。此外，還有一個不為 null blob 型別的 seatMap，seatMap 是我假設的 attribute，之所以選擇 blob 型別是其能夠存放圖像。

Aircraft:

```sql
create table Aircraft(
    registrationNo int not null auto_increment,
    companyID int not null,
    seatArrangementID int not null,
    make varchar(50) not null,
    model varchar(50) not null,
    maxCapacity int not null,
    primary key(registrationNo),
    foreign key(companyID) references Company(companyID),
    foreign key(seatArrangementID) references SeatArrangement(seatArrangementID)
);
explain Aircraft;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
registrationNo	int	NO	PRI	NULL	auto_increment
companyID	int	NO	MUL	NULL	
seatArrangementID	int	NO	MUL	NULL	
make	varchar(50)	NO		NULL	
model	varchar(50)	NO		NULL	
maxCapacity	int	NO		NULL	

Aircraft 有一個 int 型別不為 null 自動遞增的 registrationNo 且為 primary key。此外，Aircraft 還有不為 null 型別為 varchar 的 make，model，型別為 int 的 maxCapacity，companyID foreign key 指向 Company 的 companyID 因為 Aircaft 和 Company 是 many-to-one own relationship，seatArrangementID foreign key 指向 SeatArrangement 的 seatArrangementID 因為 Aircaft 和 SeatArrangement 是 many-to-one associate relationship。

IntermediateStop:

```sql
create table IntermediateStop(
    stopCode int not null auto_increment,
    primary key(stopCode)
);
explain IntermediateStop;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
stopCode	int	NO	PRI	NULL	auto_increment

IntermediateStop 有一個 int 型別不為 null 自動遞增的 stopCode 且為 primary key。

Customer:

```
53  create table Customer(
54      customerID int not null auto_increment,
55      age int not null,
56      passport varchar(50) not null,
57      `name` varchar(50) not null,
58      email varchar(50) not null,
59      primary key(customerID)
60  );
61  explain Customer;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
customerID	int	NO	PRI	NULL	auto_increment
age	int	NO		NULL	
passport	varchar(50)	NO		NULL	
name	varchar(50)	NO		NULL	
email	varchar(50)	NO		NULL	

Customer 有一個 int 型別不為 null 自動遞增的 customerID 且為 primary key。此外，Customer 還有不為 null 型別為 int 的 age，型別為 varchar 的 passport，name，email。

Reservation:

```sql
create table Reservation(
    reservationID int not null auto_increment,
    customerID int not null,
    paidFlag boolean,
    reservedDateTime datetime not null,
    creditCardNo varchar(50) not null,
    primary key(reservationID),
    foreign key(customerID) references Customer(customerID)
);
explain Reservation;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
reservationID	int	NO	PRI	NULL	auto_increment
customerID	int	NO	MUL	NULL	
paidFlag	tinyint(1)	YES		NULL	
reservedDateTime	datetime	NO		NULL	
creditCardNo	varchar(50)	NO		NULL	

Reservation 有一個 int 型別不為 null 自動遞增的 reservationID 且為 primary key。此外，Aircraft 還有不為 null 型別為 boolean 的 paidflag，型別為 datetime 的 reservedDateTime，型別為 varchar 的 creditCardNo，型別為 int 的 customerID foreign key 指向 Customer 的 customerID 因為 Reservation 和 Customer 是 many-to-one make relationship。

Ticket:

```sql
create table Ticket(
    reservationID int not null,
    ticketNo varchar(50) not null,
    validness varchar(50) not null,
    primary key(reservationID, ticketNo),
    foreign key(reservationID) references Reservation(reservationID)
);
explain Ticket;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
reservationID	int	NO	PRI	NULL	
ticketNo	varchar(50)	NO	PRI	NULL	
validness	varchar(50)	NO		NULL	

Ticket 有一個 int 型別不為 null 的 reservationID 和 ticketNo，(reservationID, ticketNo) 為 primary key 因為 Ticket 是 weak entity，owner entity 是 Reservation。此外，Ticket 還有不為 null 型別為 varchar 的 validness，用來判斷是否能用作上機或更換產品。由於 owner entity 是 Reservation，所以 reservationID 也算是 foreign key 指向 Reservation 的 reservationID。

CancellableReservation:

```
89  create table CancellableReservation(
90      cancellableReservationID int not null auto_increment,
91      reservationID int not null,
92      customerID int not null,
93      primary key(cancellableReservationID),
94      foreign key(reservationID) references Reservation(reservationID),
95      foreign key(customerID) references Customer(customerID)
96  );
97  explain CancellableReservation;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
cancellableReservationID	int	NO	PRI	NULL	auto_increment
reservationID	int	NO	MUL	NULL	
customerID	int	NO	MUL	NULL	

CancellableReservation 有一個 int 型別不為 null 自動遞增的 cancellableReservationID 且為 primary key。不為 null 型別為 int 的 reservationID foreign key 指向 Reservation 的 reservationID 因為 CancellableReservation 和 Reservation 是 many-to-one bind relationship（bind relationship 是我假設出來的，原因是一個 Reservation 可以包含多趟 Flight，取消 Reservation 也可以取消多個 Flight，所以需要獨立一個 CancellableReservation table 出來，Customer 才可以取消一個 Reservation 裡部分的 Flight），不為 null 型別為 int 的 customerID foreign key 指向 Customer 的 customerID 因為 CancellableReservation 和 Customer 是 many-to-one cancel relationship。

Airport:

```
103 create table Airport(
104     airportCode int not null auto_increment,
105     country varchar(50) not null,
106     city varchar(50) not null,
107     `description` varchar(50) not null,
108     `name` varchar(50) not null,
109     primary key(airportCode)
110 );
111 explain Airport;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
airportCode	int	NO	PRI	NULL	auto_increment
country	varchar(50)	NO		NULL	
city	varchar(50)	NO		NULL	
description	varchar(50)	NO		NULL	
name	varchar(50)	NO		NULL	

Airport 有一個 int 型別不為 null 自動遞增的 AirportCode 且為 primary key。此外，Airport 還有不為 null varchar 型別的country，city，description 和 name。

Class:

```
114 create table Class(
115     classID int not null,
116     classDescription varchar(50) not null,
117     primary key(classID)
118 );
119 explain Class;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
classID	int	NO	PRI	NULL	
classDescription	varchar(50)	NO		NULL	

Class 有一個 int 型別不為 null 的 classID 且為 primary key。此外，Class 還有不為 null varchar 型別的 classDescription。

Time:

```
123 create table `Time`(
124     timeID int not null,
125     timeSlot time not null,
126     primary key(timeID)
127 );
128 explain `Time`;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
timeID	int	NO	PRI	NULL	
timeSlot	time	NO		NULL	

Time 有一個 int 型別不為 null 的 timeID 且為 primary key。此外，Time 還有不為 null time 型別的 timeSlot。

TicketPrice:

```
131 create table TicketPrice(
132     priceID int not null,
133     price int not null,
134     primary key(priceID)
135 );
136 explain TicketPrice;
137
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	Field	Type	Null	Key	Default	Extra
▶	priceID	int	NO	PRI	NULL	
	price	int	NO		NULL	

TicketPrice 有一個 int 型別不為 null 的 priceID 且為 primary key。此外，TicketPrice 還有不為 null int 型別的 price。

Flight:

```sql
create table Flight(
    flightID int not null auto_increment,
    flightNo int not null,
    departureDateTime datetime not null,
    arrivalDateTime datetime not null,
    companyID int not null,
    departureAirportCode int not null,
    arrivalAirportCode int not null,
    registrationNo int not null,
    primary key(flightID),
    foreign key(companyID) references Company(companyID),
    foreign key(departureAirportCode) references Airport(airportCode),
    foreign key(arrivalAirportCode) references Airport(airportCode),
    foreign key(registrationNo) references Aircraft(registrationNo)
);
explain Flight;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
flightID	int	NO	PRI	NULL	auto_increment
flightNo	int	NO		NULL	
departureDateTime	datetime	NO		NULL	
arrivalDateTime	datetime	NO		NULL	
companyID	int	NO	MUL	NULL	
departureAirportCode	int	NO	MUL	NULL	
arrivalAirportCode	int	NO	MUL	NULL	
registrationNo	int	NO	MUL	NULL	

Flight 有一個 int 型別不為 null 自動遞增的 flightID 且為 primary key。此外， Flight 還有不為 null 型別為 int 的 flightNo，不為 null 型別為 datetime 的 departureDateTime, arrivalDateTime。不為 null 型別為 int 的 companyID foreign key 指向 Company 的 companyID 因為 Flight 和 Company 是 many-to-one has relationship，departureAirpostCode foreign key 指向 Airport 的 airportCode 因為 Flight 和 Airport 有 many-to-one depart relationship，arriavalAirpostCode foreign key 指向 Airport 的 airportCode 因為 Flight 和 Airport 有 many-to-one arrive relationship。在 depart 和 arrive 中，Flight 是 total participation 所以把 depart 和 arrive 的 departureDateTime 和 arrivalDateTime 加入到 Flight 的 table 裡。registrationNo foreign key 指向 Aircraft 的 registrationNo 因為 Flight 和 Aircraft 有 many-to-one bind relationship（這個 bind relationship 是假設出來的，因為我看完題目後覺得 Flight 應該要和 Aircraft 有一個

relationship 才對）。

Seat:

```sql
create table Seat(
    flightID int not null,
    seatNo int not null,
    customerID int,
    primary key(flightID, seatNo),
    foreign key(flightID) references Flight(flightID)
);
explain Seat;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
flightID	int	NO	PRI	NULL	
seatNo	int	NO	PRI	NULL	
customerID	int	YES		NULL	

Seat 有一個 int 型別不為 null 的 flightID 和 seatNo，(flightID, seatNo) 為 primary key 因為 Seat 是 weak entity，owner entity 是 Flight。此外，Seat 還有可為 null 型別為 int 的 customerID，用來判斷 Seat 是否已經被預訂(null 為空位，非 null 為已預訂)。由於 owner entity 是 Flight，所以 flightID 也算是 foreign key 指向 Flight 的 flightID。

stop:

```
153 create table `stop`(
154     flightID int not null,
155     stopCode int not null,
156     primary key(flightID, stopCode),
157     foreign key(flightID) references Flight(flightID),
158     foreign key(stopCode) references IntermediateStop(stopCode)
159 );
160 explain `stop`;
161
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
flightID	int	NO	PRI	NULL	
stopCode	int	NO	PRI	NULL	

stop 是 Flight 和 IntermediateStop 的 many-to-many relationship。不為 null 型別為 int 的 flightID 和 stopCode 組合起來 (flightID, stopCode) 是 primary key。flightID 實際上算是 foreign key 指向 Flight 的 flightID，stopCode 實際上算是 foreign key 指向 IntermediateStop 的 stopCode。

reserve:

```
162 create table reserve(
163     flightID int not null,
164     reservationID int not null,
165     primary key(flightID, reservationID),
166     foreign key(flightID) references Flight(flightID),
167     foreign key(reservationID) references Reservation(reservationID)
168 );
169 explain reserve;
170
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
flightID	int	NO	PRI	NULL	
reservationID	int	NO	PRI	NULL	

reserve 是 Flight 和 Reservation 的 many-to-many relationship。不為 null 型別為 int 的 flightID 和 reservationID 組合起來 (flightID, reservationID) 是 primary key。flightID 實際上算

是 foreign key 指向 Flight 的 flightID，reservationID 實際上算是 foreign key 指向 Reservation 的 reservationID。

register:

```
171 create table register(
172     flightID int not null,
173     airportCode int not null,
174     primary key(flightID, airportCode),
175     foreign key(flightID) references Flight(flightID),
176     foreign key(airportCode) references Airport(airportCode)
177 );
178 explain register;
179
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
flightID	int	NO	PRI	NULL	
airportCode	int	NO	PRI	NULL	

register 是 Flight 和 Airport 的 many-to-many relationship。不為 null 型別為 int 的 flightID 和 airportCode 組合起來 (flightID, airportCode) 是 primary key。flightID 實際上算是 foreign key 指向 Flight 的 flightID，airportCode 實際上算是 foreign key 指向 Airport 的 airportCode。

associate:

```sql
create table associate(
    classID int not null,
    timeID int not null,
    priceID int not null,
    primary key(classID, timeID, priceID),
    foreign key(classID) references Class(classID),
    foreign key(timeID) references `Time`(timeID),
    foreign key(priceID) references TicketPrice(priceID)
);
explain associate;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
classID	int	NO	PRI	NULL	
timeID	int	NO	PRI	NULL	
priceID	int	NO	PRI	NULL	

associate 是 Class 、 Time 和 TicketPrice 的 many-to-many ternary relationship。不為 null 型別為 int 的 classID 、 timeID 和 priceID 組合起來 (classID, timeID, priceID) 是 primary key。classID 實際上算是 foreign key 指向 Class 的 classID，timeID 實際上算是 foreign key 指向 Time 的 timeID，priceID 實際上算是 foreign key 指向 TicketPrice 的 priceID。

transfer_flight:

```sql
create table transfer_flight(
    customerID int not null,
    stopCode int not null,
    primary key(customerID, stopCode),
    foreign key(customerID) references Customer(customerID),
    foreign key(stopCode) references IntermediateStop(stopCode)
);
explain transfer_flight;
```

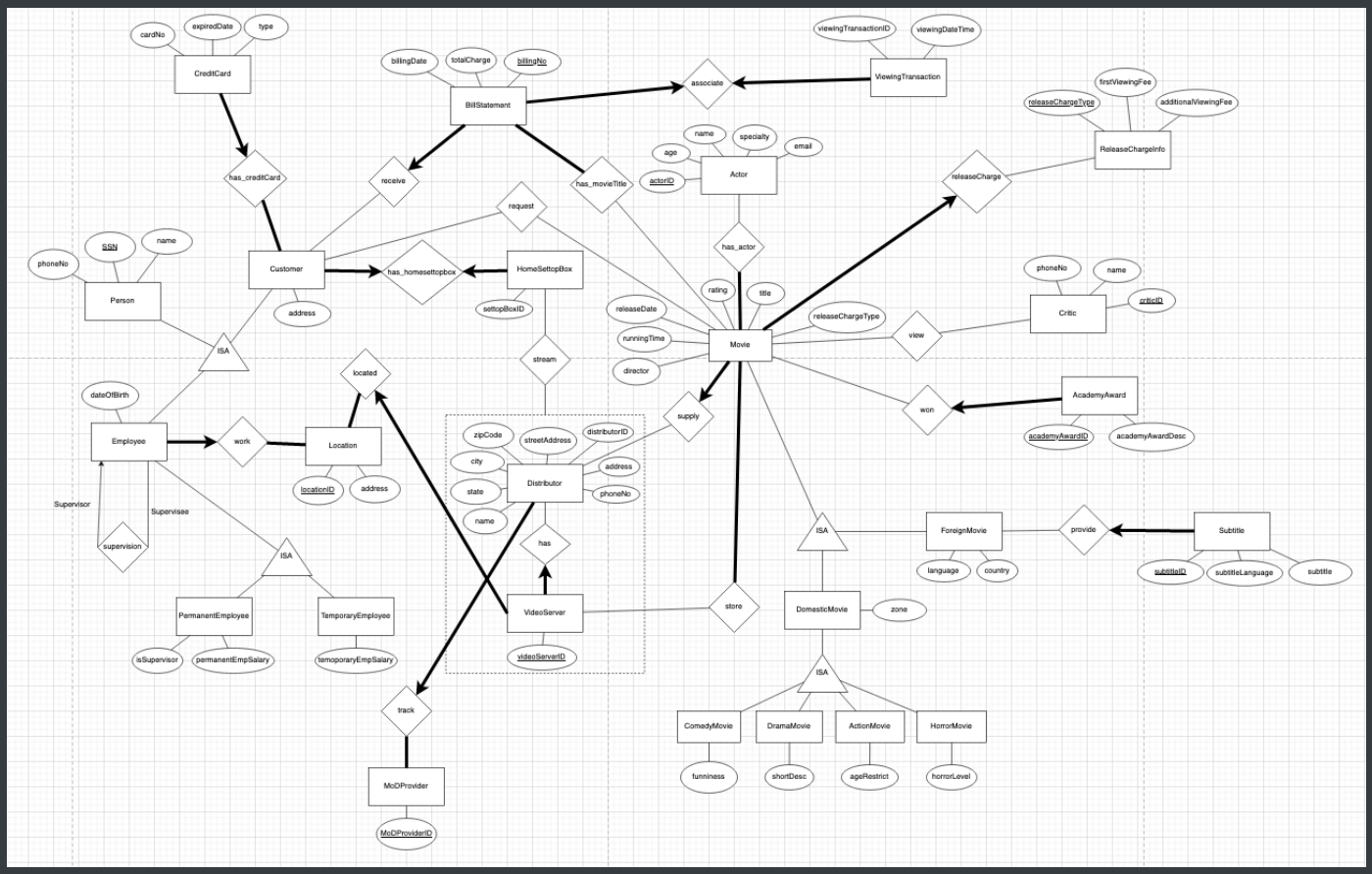
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
customerID	int	NO	PRI	NULL	
stopCode	int	NO	PRI	NULL	

transfer_flight 是 Customer 和 IntermediateStop 的 many-to-many relationship。不為 null 型別為 int 的 customerID 和 stopCode 組合起來 (customerID, stopCode) 是 primary key。customerID 實際上算是 foreign key 指向 Customer 的 customerID，stopCode 實際上算是 foreign key 指向 IntermediateStop 的 stopCode。

6B.

ER diagram:



設計邏輯說明:

```
Video movie information: A movie has a title,
release date, rating, running time, director, and
one or more actors.
Movies are split between domestic and foreign
movies.
Domestic movies are further categorized into
comedy, drama, action, and horror movies.
A comedy movie has the degree of “funniness”
which ranges between 1 and 5. A drama movie
has a short description such as “love story”,
“documentary”紀錄片, “humanity”人性, etc.
```

根據以上，Movie 有 title，releaseDate, rating, runningTime, director. Movie 和 Actor 有 many-to-many has_actor relationship, Movie 是 total participation 因為 Movie 至少有一個 Actor 來演，Actor 是 partial participation 因為有些演員沒有一部戲可以演。Movie 還被分為 DometicMovie 和 ForeignMovie，這邊是用一個 ISA relationship 連接他們。既然用了 ISA

relationship，那麼 ForeignMovie 需要一些 DomesticMovie 沒有的 attribute，在這裡我設置了 language 和 country，同樣的，DomesticMovie 也需要一些獨有的 attribute，這邊我設置了 zone。DomesticMovie 再往下分裂為4種類別，分別是 Comedy, Drama, Action, Horror，這邊一樣可以用 ISA relationship 來連接他們。題目只說明了 Comedy 有 funniness 以及 Drama 有 shortDescription，因此我幫 Action 新增了 ageRestrict，Horror 新增了 horrorLevel。有了不同的 attribute 才適合使用 ISA relationship。

```
A foreign movie uses a language other than
English and may provide English subtitle.
```

根據以上，ForeignMovie 和 Subtitle 有一個 one-to-many provide relationship，ForeignMovie 和 Subtitle 都是 partial participation 因為以英語為主的 ForeignMovie 就不會提供 Subtitle，Subtitle 一定屬於某一個 Movie。

```
Each movie could be viewed by one or more
critics, each of whom evaluates it as between
zero and five stars.
A critic has a name and a phone number. He or
she might review one or more movies.
```

根據以上，Movie 和 Critic 有一個 many-to-many viewed relationship，Movie 和 Critic 都是 partial participation 因為 Movie 可能沒有 Critic，而 Critic 可能沒有對任何 Movie 做影評。Critic 有 name，phoneNo。

```
A movie may have won one or more academy
awards (i.e., “Oscars”). If that is the case, then
the movie has a list of all the categories in which
it won, e.g., “best picture”, “best actor”, “best
actress”, etc.
```

根據以上，Movie 和 AcademyAward 有一個 one-to-many won relationship，Movie 是 partial participation 因為 Movie 可能沒有得過獎，而 AcademyAward 是 total participation 因為 AcademyAward 一定會有獲獎電影。AcademyAward 有 academyAwardDesc 用來描述獎項為何，例如 best picture。

```
Each movie is stored on a number of different
video servers. Each video server has a unique
ID and an address (location).
```

根據以上，Movie 和 videoServer 有一個 many-to-many store relationship，Movie 是 total participation 因為 Movie 一定會上映，排除被下架的可能性，而 VideoServer 是 partial participation 因為 VideoServer 不需要 store 所有 Movie。

```
If a customer requests a movie then that movie
is delivered (streamed) from one of the video
servers to the customer's home.
Each movie can be supplied by only one
distributor.
One distributor might provide several movies.
For each distributor, the MoD provider keeps
track of information about its name, address,
and phone number.
A distributor's address can be accessed as
street address, city, state, and zip code
individually. However, the entire address of a
distributor can also be retrieved as a unit.
```

根據以上，Customer 和 HomeSet-topBox 有一個 one-to-one has_homesettopbox relationship，Customer 和 HomeSet-topBox 都是 total participation 因為既然身為 Customer 就一定會有 HomeSet-topBox，HomeSet-topBox 會出現在 db 就代表被 Customer 買走了。Customer 和 Movie 有一個 many-to-many request relationship，Customer 和 Movie 都是 partial participation 因為 Customer 可能沒有 request 任何 Movie，而 Movie 則不一定需要被 request。Movie 和 Distributor 有一個 many-to-one supply relationship，Movie 是 total participation 因為之前有假設 Movie 一定上映，那麼必然要有 Distributor，而 Distributor 是 partial participation 因為 Distributor 可能沒有 supply 任何 Movie。這邊有把 Distributor 和 VideoServer 看成是一體的（aggregation）因為之前有提到 Movie 由 Distributor supply 又 Movie 儲存在 VideoServer，所以看起來是 Distributor 和 VideoServer 有一個 one-to-many has relationship，Distributor 是 partial participation 因為不是所有 VideoServer 都屬於同一個 Distributor，VideoServer 是 total participation 因為 VideoServer 一定屬於某一個 Distributor。因此該 aggregation 和 HomeSet-topBox 有一個

many-to-many stream relationship，aggregation 和 HomeSet-topBox 都是 partial participation 因為不是所有 aggregation 都會參與 stream，而 HomeSet-topBox stream。Distributor 有 streetAddress, city, state, zipCode，phoneNo 以及 （streetAddress, city, state, zipCode） 為一體的 address。VideoServer 和 Location 有一個 many-to-one located relationship，VideoServer 和 Location 都是 total participation 因為 VideoServer 必然存放在某個 Location， Location 必然存在。VideoServer 不是用 location 而是用 locationID 因為 Employee 的工作地點和一些 VideoServer 一樣，所以我為了 Employee 和 VideoServer table 的 data integrity 因此只要更換 Location table 匹配 locationID 的 address 即可改變 Employee 和 VideoServer 的 location。 Distributor 和 MoDProvider 有一個 many-to-one track relationship，Distributor 和 MoDProvider 都是 total participation 因為 MoDProvider 需要知道 Distributor 的 name、address、phoneNo，Distributor 也需要被 track 因為這樣才能透過 MoDProvider 來 supply Movie。

```
Employee information: An employee has a name,
a social security number, date of birth, salary,
and a phone number.
He or she may have a supervisor.
Employees are either permanent or temporary,
but not both.
A supervisors are permanent employees.
An employee works at the location of one of the
video server.
Customer information: A customer has a social
security number, a name, a phone number, and
an address.
Each customer must have one or more credit
cards. Each credit card includes a type of credit
card (Visa / MasterCard /...), a card number,
and an expiration date.
```

根據以上，Employee 和 Customer 有共同的 attribute: name, SSN, phoneNo，因此我把 Employee 和 Customer 抽像出了 Person，Person 有 name, SSN, phoneNo。Employee 獨有的 attribute 是 dateOfBirth，Customer 獨有的 attribute 是 address。Employee 還可再分成 PermanentEmployee 和 TemporaryEmployee，因此我用了 ISA 連接了 Employee，PermanentEmployee 和 TemporaryEmployee。PermanentEmployee 獨有的 attribute 有 permanentEmpSalary，isSupervisor，isSupervisor 是為了分辨 permanentEmployee 是不是

supervisor。TemporaryEmployee 獨有的 attribute 有 temporaryEmpSalary。Employee 可能會有 Supervisor 所以 Employee 有 one-to-many recursive supervision relationsip。在該 recursive supervision relationship 中，Supervisor 和 Supervisee 是 partial participation，因為如果一人是 Supervisor 那麼他就不是 Supervisee，如果一人是 Supervisee 那麼他就不是 Supervisor。Employee 和 Location 有一個 many-to-one work relationship，Employee 和 Location 都是 total participation 因為身為 Employee 需要在其中一台 VideoServer 的所在地工作，而 Location 也必須能夠 work 因為那邊存放了 VideoServer。 Customer 和 CreditCard 有一個 one-to-many has_creditCard relationship，Customer 和 CreditCard 都是 total participation 因為題目提到 Customer 必須有一張或多張 CreditCard，而 CreditCard 必須有持卡者。CreditCard 有 cardNo, type, expiredDate。

```
Viewing transaction information: Each customer
can view movies from a video server. For each
viewing transaction, the viewing date and time is
recorded.
Billing statement information: Each customer
receives one bill statement after each viewing
transaction. Each statement has a billing date, a
billing number, one or more movie titles, and a
total charge.
A billing number is unique for a particular
customer. It is not unique across different
customers.
```

根據以上，ViewingTransaction 有 viewingDate 和 viewingTime 但我把這兩個 attributes 合併了變成 viewingDateTime。BillStatement 和 Movie 有一個 many-to-many has_movieTitle relationship，BillStatement 是 total participation 因為 BillStatement 至少有一個 Movie title，而 Movie 則是 partial participation 因為不是所有 Movie 都需要被 request 到。ViewingTransaction 和 BillStatement 有一個 one-to-one associate relationship，ViewingTransaction 和 BillStatement 都是 total participation 因為一個 ViewingTransaction 會產生一個 Billstatement。BillStatement 有 billingDate, totalCharge, billingNo。Customer 和 Billstatement 有一個 one-to-many receive relationship，Customer 是 partial participation 因為不是每個 Customer 都有 request Movie，而 BillStatement 是 total participation 因為若有 BillStatement 的話必然屬於某個 Customer。billingNo 根據題意應該是 BillStatement 的 primary key。

```
Release charge information: For each movie, a
release charge type is defined. Arbitrarily recent
movies are marked as “new release”, whereas
others are marked as “ordinary release”.

The store keeps track of different charge
information for different release types.

For newly released movies, the fee is $100 for
the first viewing and $30 for each additional
viewing.

For the other ones, the fee is $50 for the first
viewing and $15 for each additional viewing.
```

根據以上，Movie 應該新增一個 attribute: releaseChargeType，這個 attribute 的值可能是 "new release" 或 "ordinary release"。那可能日後會有更多的 releaseChargeType 出現，所以應該要新增一個 table，我這把叫作 ReleaseChargeInfo。ReleaseChargeInfo 有 releaseChargeType， firstViewingFee, additionalViewingFee。Movie 和 ReleaseChargeInfo 有一個 many-to-one releaseCharge relationship，Movie 是 total participation 因為 request Movie 一定需要有一個價位才能產生繳費單給 Customer，而 ReleaseChargeInfo 則是 partial participation 因為不是所有的 ReleaseChargeInfo 都會被 Movie 所使用到。

Critic:

```
5  create table Critic(
6      criticID int not null auto_increment,
7      name varchar(50) not null,
8      phoneNo varchar(50) not null,
9      primary key(criticID)
10 );
11 explain Critic;
12
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
criticID	int	NO	PRI	NULL	auto_increment
name	varchar(50)	NO		NULL	
phoneNo	varchar(50)	NO		NULL	

Critic 有一個 int 型別不為 null 自動遞增的 criticID 且為 primary key。此外，Critic 還有不為 null varchar 型別的 name 和 phoneNo。

ForeignMovie:

```
15 create table ForeignMovie(
16     title varchar(50) not null,
17     `language` varchar(50) not null,
18     country varchar(50) not null,
19     primary key(title)
20 );
21 explain ForeignMovie;
22
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
title	varchar(50)	NO	PRI	NULL	
language	varchar(50)	NO		NULL	
country	varchar(50)	NO		NULL	

ForeignMovie 有一個 varchar 型別不為 null 的 title 且為 primary key。此外，ForeignMovie 還有不為 null varchar 型別的 language 和 country。

Subtitle:

```sql
create table Subtitle(
    subtitleID int not null auto_increment,
    title varchar(50) not null,
    subtitleLanguage varchar(50) not null,
    subtitle longtext not null,
    primary key(subtitleID),
    foreign key(title) references ForeignMovie(title)
);
explain Subtitle;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
subtitleID	int	NO	PRI	NULL	auto_increment
title	varchar(50)	NO	MUL	NULL	
subtitleLanguage	varchar(50)	NO		NULL	
subtitle	varchar(50)	NO		NULL	

Subtitle 有一個 int 型別不為 null 自動遞增的 subtitleID 且為 primary key。此外，Subtitle 還有不為 null varchar 型別的 title、 subtitleLanguage 和不為 null longtext 型別的 subtitle。title 實際上是一 foreign key 指向 ForeignMovie 的 title 因為 Subtitle 和 ForeignMovie 是 many-to-one provide relationship。

DomesticMovie:

```
37  create table DomesticMovie(
38      title varchar(50) not null,
39      zone varchar(50) not null,
40      primary key(title)
41  );
42  explain DomesticMovie;
43
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
title	varchar(50)	NO	PRI	NULL	
zone	varchar(50)	NO		NULL	

DomesticMovie 有一個 varchar 型別不為 null 的 title 且為 primary key。此外，DomesticMovie 還有不為 null varchar 型別的 zone（zone 是我假設的 attribute 因為這樣才有建立新 table 的意義）。

ComedyMovie:

```
45  create table ComedyMovie(
46      title varchar(50) not null,
47      funniness int not null,
48      primary key(title)
49  );
50  explain ComedyMovie;
51
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
title	varchar(50)	NO	PRI	NULL	
funniness	int	NO		NULL	

由於 ComedyMovie 和 DramaMovie 都有獨有的 attributes(funniness 和 shortDesc) 導致不適合用 attribute 區別 ComedyMovie, DramaMovie, ActionMovie 和 HorrorMovie, 因此另外建一個 table 存放 ComedyMovie，ComedyMovie 有一個 varchar 型別不為 null 的 title 且為 primary key。此外，ComedyMovie 還有不為 null int 型別的 funniness。

DramaMovie:

```
53  create table DramaMovie(
54      title varchar(50) not null,
55      shortDesc varchar(50) not null,
56      primary key(title)
57  );
58  explain DramaMovie;
59
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
title	varchar(50)	NO	PRI	NULL	
shortDesc	varchar(50)	NO		NULL	

由於 ComedyMovie 和 DramaMovie 都有獨有的 attributes(funniness 和 shortDesc) 導致不適合用 attribute 區別 ComedyMovie, DramaMovie, ActionMovie 和 HorrorMovie, 因此另外建一個 table 存放 DramaMovie，DramaMovie 有一個 varchar 型別不為 null 的 title 且為 primary key。此外，DramaMovie 還有不為 null varchar 型別的 shortDesc。

ActionMovie:

```
61  create table ActionMovie(
62      title varchar(50) not null,
63      ageRestrict int not null,
64      primary key(title)
65  );
66  explain ActionMovie;
67
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
title	varchar(50)	NO	PRI	NULL	
ageRestrict	int	NO		NULL	

由於 ComedyMovie 和 DramaMovie 都有獨有的 attributes(funniness 和 shortDesc) 導致不適合用 attribute 區別 ComedyMovie, DramaMovie, ActionMovie 和 HorrorMovie, 因此另外建一個 table 存放 ActionMovie，ActionMovie 有一個 varchar 型別不為 null 的 title 且為 primary key。此外，ActionMovie 還有不為 null int 型別的 ageRestrict（ageRestrict 是我自己假設

的）。

HorrorMovie:

```
69  create table HorrorMovie(
70      title varchar(50) not null,
71      horrorLevel int not null,
72      primary key(title)
73  );
74  explain HorrorMovie;
75
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	Field	Type	Null	Key	Default	Extra
▶	title	varchar(50)	NO	PRI	NULL	
	horrorLevel	int	NO		NULL	

由於 ComedyMovie 和 DramaMovie 都有獨有的 attributes(funniness 和 shortDesc) 導致不適合用 attribute 區別 ComedyMovie, DramaMovie, ActionMovie 和 HorrorMovie, 因此另外建一個 table 存放 HorrorMovie，HorrorMovie 有一個 varchar 型別不為 null 的 title 且為 primary key。此外，HorrorMovie 還有不為 null int 型別的 horrorLevel（horrorLevel 是我自己假設的）。

CreditCard:

```
79  create table CreditCard(
80      cardNo varchar(50) not null,
81      SSN varchar(50) not null,
82      expiredDate date not null,
83      `type` varchar(50) not null,
84      primary key(cardNo),
85      foreign key(SSN) references Customer(SSN)
86  );
87  explain CreditCard;
88
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
cardNo	varchar(50)	NO	PRI	NULL	
SSN	varchar(50)	NO	MUL	NULL	
expiredDate	date	YES		NULL	
type	varchar(50)	YES		NULL	

CreditCard 有一個 varchar 型別不為 null 的 cardNo 且為 primary key。此外，CreditCard 還有不為 null varchar 型別的 SSN、 type 和不為 null date 型別的 expiredDate。SSN 實際上是一 foreign key 指向 Customer 的 SSN 因為 CreditCard 和 Customer 是 many-to-one has_creditCard relationship。

Customer:

```
90  create table Customer(
91      SSN varchar(50) not null,
92      address varchar(50) not null,
93      primary key(SSN)
94  );
95  explain Customer;
96
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
SSN	varchar(50)	NO	PRI	NULL	
address	varchar(50)	NO		NULL	

Customer 有一個 varchar 型別不為 null 的 SSN 且為 primary key。此外，Customer 還有不為 null varchar 型別的 address。

Location:

```
98  create table Location(
99      locationID int not null auto_increment,
100     address varchar(50) not null,
101     primary key(locationID)
102 );
103 explain Location;
104
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
locationID	int	NO	PRI	NULL	auto_increment
address	varchar(50)	NO		NULL	

Location 有一個 int 型別不為 null 自動遞增的 locationID 且為 primary key。此外，Location 還有不為 null varchar 型別的 address。

Employee:

```sql
create table Employee(
    SSN varchar(50) not null,
    locationID int not null,
    dateOfBirth date not null,
    supervisor_SSN varchar(50) not null,
    primary key(SSN),
    foreign key(locationID) references Location(locationID)
);
explain Employee;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
SSN	varchar(50)	NO	PRI	NULL	
locationID	int	NO	MUL	NULL	
dateOfBirth	date	NO		NULL	
supervisor_SSN	varchar(50)	NO		NULL	

Employee 有一個 varchar 型別不為 null 的 SSN 且為 primary key。此外，Employee 還有不為 null int 型別的 locationID、 不為 null date 型別的 dateOfBirth 以及不為 null varchar 型別的 supervisor_SSN。locationID 實際上是一 foreign key 指向 Location 的 locationID 因為 Employee 和 Location 是 one-to-many work relationship。

PermanentEmployee:

```sql
create table PermanentEmployee(
    SSN varchar(50) not null,
    isSupervisor Boolean not null,
    permanentEmpSalary int not null,
    primary key(SSN)
);
explain PermanentEmployee;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
SSN	varchar(50)	NO	PRI	NULL	
isSupervisor	tinyint(1)	NO		NULL	
permanentEmpSalary	int	NO		NULL	

PermanentEmployee 有一個 varchar 型別不為 null 的 SSN 且為 primary key。此外，PermanentEmployee 還有不為 null int 型別的 permanentEmpSalary 和不為 null Boolean 型別的 isSupervisor 用來區別 PermanentEmployee 是不是 supervisor。

TemporaryEmployee:

```sql
create table TemporaryEmployee(
    SSN varchar(50) not null,
    temporaryEmpSalary int,
    primary key(SSN)
```

TemporaryEmployee 有一個 varchar 型別不為 null 的 SSN 且為 primary key。此外，TemporaryEmployee 還有不為 null int 型別的 temporaryEmpSalary。

BillStatement:

```
139 create table BillStatement(
140     billingNo int not null auto_increment,
141     SSN varchar(50) not null,
142     billingDate date not null,
143     totalCharge int not null,
144     primary key(billingNo),
145     foreign key(SSN) references Customer(SSN)
146 );
147 explain BillStatement;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
billingNo	int	NO	PRI	NULL	auto_increment
SSN	varchar(50)	NO	MUL	NULL	
billingDate	date	NO		NULL	
totalCharge	int	NO		NULL	

BillStatement 有一個 int 型別不為 null 的 billingNo 且為 primary key。此外，BillStatement 還有不為 null varchar 型別的 SSN, 不為 null date 型別的 billingDate, 不為 null int 型別的 totalCharge。SSN 實際上是一 foreign key 指向 Customer 的 SSN 因為 BillStatement 和 Customer 是 many-to-one receive relationship。

Actor:

```
149 create table Actor(
```

Actor 有一個 int 型別不為 null 自動遞增的 actorID 且為 primary key。此外，Actor 還有不為 null varchar 型別的 name、specialty、email 和不為 null int 型別的 age。

MoDProvider:

```
159 create table MoDProvider(
160     MoDProviderID int not null auto_increment,
161     primary key(MoDProviderID)
162 );
163 explain MoDProvider;
164
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
MoDProviderID	int	NO	PRI	NULL	auto_increment

MoDProvider 有一個 int 型別不為 null 自動遞增的 MoDProviderID 且為 primary key。

Distributor:

```
173 • create table Distributor(
```

Distributor 有一個 int 型別不為 null 自動遞增的 distributorID 且為 primary key。此外，Distributor 還有不為 null varchar 型別的 name、state、city、zipCode、streetAddress、address、phoneNo, 不為 null int 型別的 MoDProviderID。address 實際上是 state、city、zipCode、streetAddress 合併在一起的單一 attribute 因為題目提到 "the entire address of a distributor can also be retrieved as a unit" 。MoDProviderID 實際上是一 foreign key 指向 MoDProvider 的 MoDProviderID 因為 Distributor 和 MoDProvider 是 many-to-one track relationship。

ReleaseChargeInfo:

```
190 • create table ReleaseChargeInfo(
```

ReleaseChargeInfo 有一個 varchar 型別不為 null 的 releaseChargeType 且為 primary key。此外，ReleaseChargeInfo 還有不為 null int 型別的 firstViewingFee 和 additionalViewingFee。

Movie:

Movie 有一個 varchar 型別不為 null 的 title 且為 primary key。此外，Movie 還有不為 null varchar 型別的 releaseChargeType、director ，不為 null int 型別的 distributorID、rating、runningTime 和不為 null date 型別的 releaseDate。releaseChargeType 實際上是一 foreign key 指向 releaseChargeInfo 的 releaseChargeType 因為 Movie 和 releaseChargeInfo 是 many-to-one releaseCharge relationship。distributorID 實際上是一 foreign key 指向 Distributor 的 distributorID 因為 Movie 和 Distributor 是 many-to-one supply relationship。

AcademyAward:

AcademyAward 有一個 int 型別不為 null 自動遞增的 academyAwardID 且為 primary key。此外，AcademyAward 還有不為 null varchar 型別的 title 和 academyAwardDesc。title 實際上是一 foreign key 指向 Movie 的 title 因為 AcademyAward 和 Movie 是 many-to-one won relationship。

ViewingTransaction:

ViewingTransaction 有一個 int 型別不為 null 自動遞增的 viewingTransactionID 且為 primary key。此外，ViewingTransaction 還有不為 null datetime 型別的 viweingDateTime，不為 null int 型別的 billingNo。billingNo 實際上是一 foreign key 指向 BillStatement 的 billingNo 因為 ViewingTransaction 和 BillStatement 是 one-to-one associate relationship(這個 associate relationship 是假設出來的 relationship 因為題目沒有明確說出他們的 relationship 但他們存在某種 relationship 因為 "Each customer receives one bill statement after each viewing transaction.")。

VideoServer:

VideoServer 有一個 int 型別不為 null 自動遞增的 videoServerID 且為 primary key。此外，VideoServer 還有不為 null int 型別的 locationID 和 distributorID。locationID 實際上是一 foreign key 指向 Location 的 locationID 因為 VideoServer 和 Location 是 many-to-one located relationship。distributorID 實際上是一 foreign key 指向 Distributor 的 distributorID 因為 VideoServer 和 Distributor 是 many-to-one has relationship。

HomeSettopBox:

HomeSettopBox 有一個 int 型別不為 null 自動遞增的 settopBoxID 且為 primary key。此外，HomeSettopBox 還有不為 null varchar 型別的 SSN。SSN 實際上是一 foreign key 指向 Customer 的 SSN 因為 HomeSettopBox 和 Customer 是 one-to-one has_homesettopbox relationship。

view:

view 有一個 varchar 型別不為 null 的 title 和 int 型別不為 null 的 criticID，（title, criticID）為 primary key。title 實際上是一 foreign key 指向 Movie 的 title， criticID 實際上是一 foreign key 指向 Critic 的 criticID 因為 Movie 和 Critic 是 many-to-many view relationship。

stream:

stream 有一個 int 型別不為 null 的 settopBoxID 和 videoServerID，（settopBoxID, videoServerID）為 primary key。settopBoxID 實際上是一 foreign key 指向 HomeSettopBox 的 settopBoxID， videoServerID 實際上是一 foreign key 指向 VideoServer 的 videoServerID

因為 HomeSettopBox 和 VideoServer 是 many-to-many stream relationship。stream table 沒有包含 distributorID 因為 Distributor 和 VideoServer 是 one-to-many has relationship，只要知道其中一個 videoServerID 就能夠找到 Distributor 即使 Distributor 和 VideoServer 是 aggregation。

has_actor:

has_actor 有一個 varchar 型別不為 null 的 title 和 int 型別不為 null 的 actorID，（actorID, title）為 primary key。title 實際上是一 foreign key 指向 Movie 的 title， actorID 實際上是一 foreign key 指向 Actor 的 actorID 因為 Movie 和 Actor 是 many-to-many has_actor relationship。

request:

request 有一個 varchar 型別不為 null 的 title 和 SSN，（SSN, title）為 primary key。title 實際上是一 foreign key 指向 Movie 的 title， SSN 實際上是一 foreign key 指向 Customer 的 SSN 因為 Movie 和 Customer 是 many-to-many request relationship。

store:

store 有一個 varchar 型別不為 null 的 title 和 int 型別不為 null 的 videoServerID，（videoServerID, title）為 primary key。title 實際上是一 foreign key 指向 Movie 的 title，videoServerID 實際上是一 foreign key 指向 VideoServer 的 videoServerID 因為 Movie 和 VideoServer 是 many-to-many store relationship。

has_movieTitle:

has_movie 有一個 varchar 型別不為 null 的 title 和 int 型別不為 null 的 billingNo，（billingNo, title）為 primary key。title 實際上是一 foreign key 指向 Movie 的 title，billingNo 實際上是一 foreign key 指向 BillStatement 的 billingNo 因為 Movie 和 BillStatment 是 many-to-many has_movieTitle relationship。