Presenter, company name and addressTitle and subtitleVersion number and dateDecorative sidebar

Table of Contents

[I)](#_gjdgxs) **INTRODUCE THE PROBLEM** 2

[1) Describe the problem 2](#_30j0zll)

[II)](#_1fob9te) **Entity – relationship – er** 4

[1) Definition entity – attribute 4](#_3znysh7)

[2) Set-up entity – relationship 5](#_2et92p0)

[LINK: 6](#_tyjcwt)

[III)](#_3dy6vkm) **Data dictionary** 6

[1) Database and table 7](#_1t3h5sf)

[**A.** Create DATABASE](#_4d34og8) **Assignment\_DBI202** 7

[**B.** Create table](#_2s8eyo1)  **Location** 7

[C. Create table](#_17dp8vu) **Employee** 8

[**D.** Create table](#_3rdcrjn) **Category Computer** 9

[**E.** Create TABLE](#_26in1rg) **Computer** 9

[F. Create table](#_lnxbz9) **Product** 11

[**G.** Create table](#_35nkun2) **Voucher** 12

[**H.** Create table](#_1ksv4uv) **Work** 13

[I. Create table](#_44sinio)  **Customer** 14

[**J.** Create table](#_2jxsxqh) **Bill** 15

[K. Create table](#_z337ya) **Account** 17

[L. Create table](#_3j2qqm3) **Order detail** 17

[IV. **Entity relationship diagram (erd)** 20](#_1y810tw)

[A. Location 20](#_4i7ojhp)

[B. EMPLOYEE 21](#_2xcytpi)

[C. CategoryComputer 21](#_1ci93xb)

[D. COMPUTER 22](#_3whwml4)

[E. Product 22](#_2bn6wsx)

[F. Voucher 24](#_qsh70q)

[J. Account 28](#_3as4poj)

[K. Order detail 28](#_1pxezwc)

[L. Sign In 29](#_49x2ik5)

[M. Full diagram 30](#_147n2zr)

[V. SQL COMMAND 32](#_3o7alnk)

[A. query using order by 32](#_23ckvvd)

[B. query using inner join 32](#_ihv636)

[C. query using aggregate functions 34](#_32hioqz)

[D. query using the group by and having clauses 34](#_1hmsyys)

[E. query that uses a sub-query as a relation 35](#_41mghml)

[F. query that uses partial matching in the where clause 36](#_2grqrue)

[G. query that uses a self – join 36](#_vx1227)

[H. store procedure 37](#_3fwokq0)

[I. trigger 38](#_1v1yuxt)

# INTRODUCE THE PROBLEM

## Describe the problem

With the rapid development of information technology and the growing popularity of online gaming, gaming net rooms (internet cafes specifically focused on gaming) have become a common sight. To manage these facilities effectively, a robust database management system is necessary. Based on theoretical foundations and surveys of gaming net rooms, we have conducted a Database Analysis to manage a chain of gaming net rooms efficiently**.**

Based on the theoretical foundation learned in class and through surveys of internet rooms, our group has conducted a Database Analysis on managing a chain of internet rooms.

* One location can have many employees working. Each employee can work at multiple locations at different times. Location information includes M\_ID, L\_ID, Location.
* Information about employees includes: E\_ID, name, email, phone, sex, E\_Address. The employee's personality is "M"(Boy) of "F"(Girl). Phone numbers cannot be larger than 11 digits. E\_ID must follow the form CSD1|CSD2|CSD3\_xxx, x as a number. Email must end with @gmail.com. The employee may be the location manager.
* An employee working at a location will have Salary, Work\_Hours (number of hours worked in a day), From, To (which are the employee's start working day and departure day).
* Each location consists of multiple computers. Each computer is managed only within one location with information including: C\_ID, status.
* The status is “1” (This computer is empty, customers can register to play on this computer) or ”0” (This computer is registered, customers cannot register on this computer This) or “2” (The computer is broken, this status helps the administrator know for maintenance and repair).
* Each computer belongs to a specific configuration. A configuration can include multiple computers. CategoryComputer is logic of configuration, includes CategoryID, Name, Price (the amount to be paid for using the machine for one hour), Place (in a Location, it will be divided into different Places for different computers).
* Vouchers will be given to customers during the event. Each voucher can be applied to multiple bills, but each customer can only apply one voucher. Vouchers will include Voucher\_ID, Begin\_date, End\_date, Discount, V\_Description, Is\_Member (only applicable to people who are members of the club), Minimum\_price (minimum price to apply voucher).
* Each employee can create multiple bills. Each bill is created by an employee at a certain time through Create\_Time and Create\_Date. On one computer, you can place multiple orders, but each order can only be placed on a single computer.
* Bill will include programs such as Bill\_ID, Primary\_Price, Final\_Price, Used\_Point, Earned\_Point, Voucher\_ID (the voucher is applicable to the bill when the customer makes payment), C\_ID, Place (they are the address of the machine where the order is placed), Payment\_Time (Payment\_Time this is the time when customers pay for the bill), CustomerID. The bill will be paid by the customer when receiving the goods.
* One bill can have many Products, and a Product can appear on multiple bills. Products informations include Product\_ID, P\_Name, Price.
* Customer information includes: customerID, Sex, Name, Points.
* When customers register for a gaming account, they will provide information such as Username, Password, and TimeToUse (the time that users ordered).A customer can only have 1 account, the account will be deleted if this customer no longer uses it. Additionally, an account can order products online from cyber.
* One account can only Sign\_In to one computers at specific times: From, To (sign-in time and log-out time).

Request:

* Every day, employees need to calculate the total amount.
* Employees need to check the date customers registered their accounts and products on the shelves
* Managers need to check whether the computers are operating stably or not
* Every month, employees need to calculate the total amount collected.
* Managers can check customer information
* Staff can check empty computers for customers to register.
* Staff can check if the voucher is still valid or not to pay in the bill
* Check out the most used computer types
* Check guests' playing hours and notify guests
* The manager will know which type of machine is used the most and which types of products are popular.

2) MANAGEMENT OBJECTIVES

* Manage schedules and working hours
* Manage Storage and manage customer details, including names, contact information, and preferences.
* Manage the most used computer types
* Manage your most purchased products
* Manage whether the machine is available or empty

Important output

* Total amount earned each time
* Count the most used computers.
* Count the number of best-selling products.

# entity – relationship – er

## definition entity – attribute

Base on the problem description and management objectives, we can present several entities and attributes of the entity as follow:

- Location: **L\_ID,** M\_ID, L\_Address.

- CategoryComputer: **CategoryID,** Name, Price, Place

- Computer: **C\_ID,**  CategoryID, Status, L\_ID.

- Product: **Product\_ID,** P\_Name, Price

- Voucher: **Voucher\_ID,** Discount, Begin\_Date, V\_Description, End\_Date, Mininum\_Price, Is\_Member

- Employee: **E\_ID,** Name**,** Sex, E\_Address, Email, Phone.

- Work: **L\_ID**, **E\_ID, From,** To, Salary, Work\_Hours

- Bill: **Bill\_ID,** Used\_Point, Primary\_Price, Final\_Price, Earned\_Point, E\_ID, Voucher\_ID, C\_ID, Create\_Date, Create\_Time, Payment\_Time, CustomerID.

- Customer: **CustomerID,** Sex, Name, Point.

- Account: **UserName, Password, CustomerID**, TimeToUse

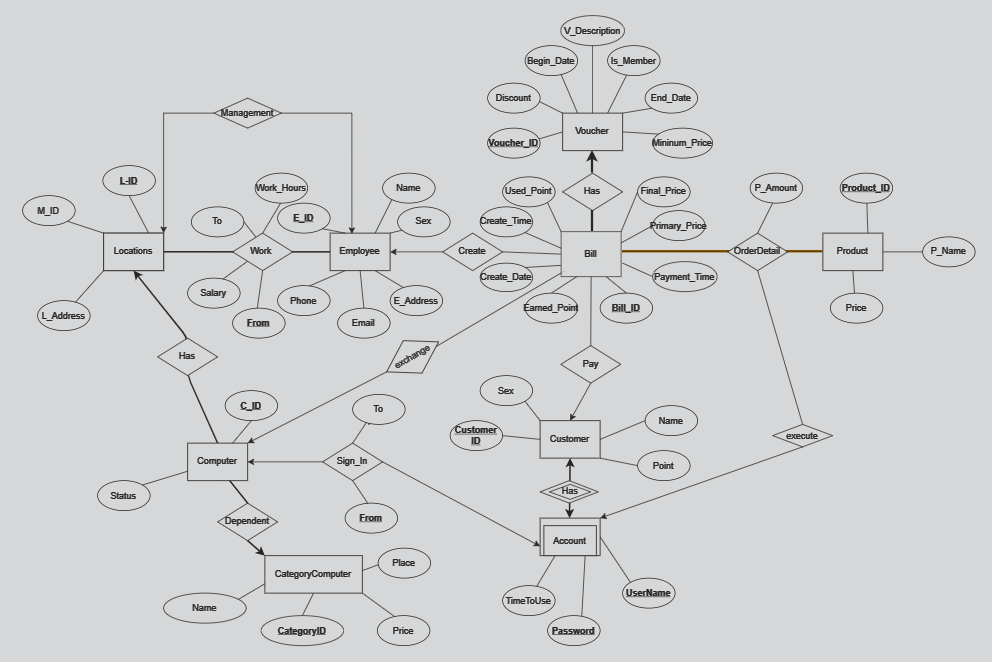
- OrderDetail: **Bil\_ID, Product\_ID, UserName, Password, CustomerID,** P\_Amount

- Sign\_In: **C\_ID,** **UserName, Password,** **From**, **CustomerID**, To

## set-up entity – relationship

\* Some symbols used in the model

|  |  |
| --- | --- |
| * Attribute description / description * Entity * Weak entity * Relationship * Connectivity (force) = 1 * Connectivity = N |  |



## LINK: [ERD](https://drive.google.com/file/d/1naSJkkZ0pnvw4S4alVbXxsI1Cuqy6M4u/view?usp=sharing)

# data dictionary

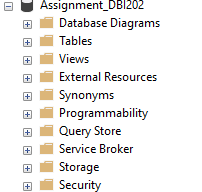
Just for example on some tables (other table are similar, you have to define all the tables in your database). Note: to run the query you have to define the table 1 first then go to the side tables much

## database and table

### CREATE DATABASE **Assignment\_DBI202**

--create database

CREATE DATABASE Assignment\_DBI202



### Create table **Employee**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| E\_ID | Varchar(7) |  | CS1|CS2|CS3\_xxx | PK, Not null, unique |
| Name | Nvarchar(45) |  |  | Not null |
| Address | Nvarchar(45) |  |  | Not null |
| Email | Nvarchar(255) |  | @gmail.com | Not null, unique |
| Phone | Varchar(11) |  | 0 to 9 | Unique, not null |
| Sex | Char(1) | M | ‘F’ or ‘M’ | Not null |

***Code:***

--create table employee

create table employee(

E\_ID Varchar(7) primary key check(E\_ID like 'CS1\_[0-9][0-9][0-9]' or E\_ID like 'CS2\_[0-9][0-9][0-9]' or E\_ID like 'CS3\_[0-9][0-9][0-9]') not null,

[Name] Nvarchar(50) not null,

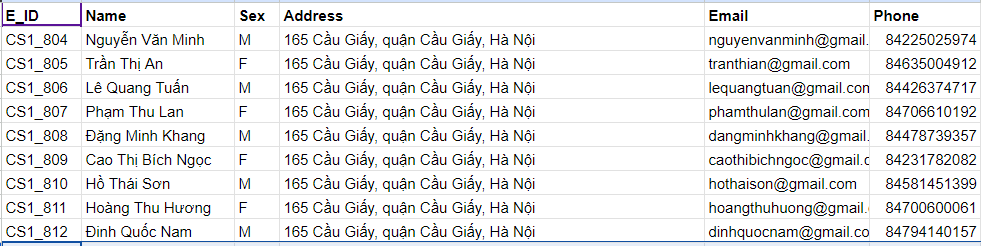
Phone varchar(11) unique check(Phone like '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]' or Phone like '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]') not null,

Email nvarchar(255) check(Email like '%@gmail.com') not null,

[Address] nvarchar(255) not null,

Sex Char(1) check(Sex like 'F' or Sex like 'M') not null)

***Example:***

******

### Create table **Locations**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| L\_ID | Varchar(5) |  | CS1|CS2|CS3 | PK, not null |
| M\_ID | Varchar(7) |  |  | PK, FK reference Employee(E\_ID), Not null |
| L\_Address | Nvarchar(255) |  |  | Not null |

***Code:***

--create table **Location**

create table locations(

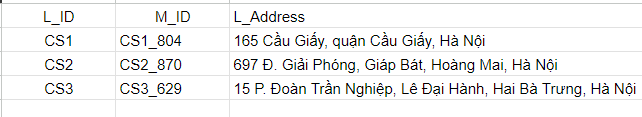
L\_ID varchar(5) check(L\_ID like 'CS1' or L\_ID like 'CS2' or L\_ID like 'CS3') not null,

M\_ID varchar(8) foreign key references employee(E\_ID) not null,

L\_Address nvarchar(255) not null,

primary key (L\_ID))

***Example:***

******

### Create table **Category Computer**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| CategoryID | Varchar(20) |  |  | PK, not null |
| Name | Varchar(50) |  |  | Not null |
| Place | Char(1) |  |  |  |
| Price | Int |  |  | Not null |

***Code:***

--create table **CategoryComputer**

create table CategoryComputer(

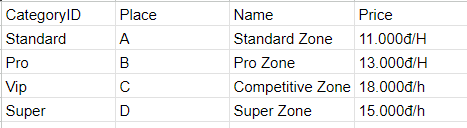
CategoryID varchar(20) not null primary key,

Place Varchar(1) check(Place like 'A' or Place like 'B' or Place like 'C' or Place like 'D') not null,

price int not null,

[Name] Varchar(50) not null)

***Example:***

******

### Create TABLE **Computer**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| C\_ID | Char(6) |  |  | PK, not null |
| L\_ID | Varchar(5) |  |  | FK reference Location(L\_ID), not null |
| M\_ID | Varchar(7) |  |  | FK reference Location(M\_ID), not null |
| Status | Char(1) | 1 | ‘1’ or ‘0’ or’2’ | Not null |
| CategoryID | Char(10) |  |  | PK, not null |

***Code:***

--create table computer

create table computer(

C\_ID char(6) check(C\_ID like 'A\_[0-9][0-9][0-9][0-9]' or C\_ID like 'B\_[0-9][0-9][0-9][0-9]' or C\_ID like 'C\_[0-9][0-9][0-9][0-9]' or C\_ID like 'D\_[0-9][0-9][0-9][0-9]') not null,

[status] char(1) check ([status] like '1' or [status] like '0' or [status] like '2') not null,

L\_ID varchar(5) not null,

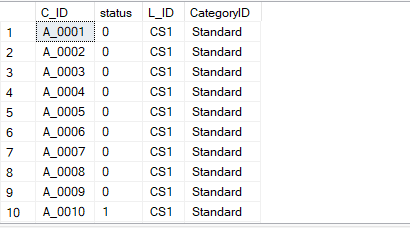
CategoryID varchar(20) not null,

primary key(C\_ID),

foreign key(L\_ID) references locations(L\_ID),

foreign key(CategoryID) references CategoryComputer(CategoryID))

***Example:***

******

### create table Product

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| Product\_ID | Char(5) |  |  | PK, not null, unique |
| P\_Name | Nvarchar(50) |  |  | Not null |
| Price | Int |  |  | Not null |

***Code:***

--create table Product

create table Product(

Product\_ID char(5) primary key not null,

P\_Name nvarchar(50) not null,

Price int not null)

***Example:***

******

### create table **Voucher**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| Voucher\_ID | Nvarchar(255) |  |  | PK, Not null |
| Discount | float |  |  | Not null |
| Begin\_Date | Date |  |  | Not null |
| End\_Date | Date |  |  | Not null |
| V\_description | Nvarchar(255) |  |  |  |
| Mininum\_Price | float |  |  | Not null |
| Is\_Member | Char(1) | 0 | ‘1’ or ‘0’ | Not null |

***Code:***

--create table Voucher

create table Voucher(

Voucher\_ID nvarchar(255) primary key not null,

Discount float not null,

Begin\_Date date not null,

End\_Date date not null,

V\_description nvarchar(255) not null,

Mininum\_Price float not null,

[Is\_Member] char(1) check([Is\_Member] like '1' or [Is\_Member] like '0') not null)

***Example: ***

### create table **Work**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| L\_ID | Nvarchar(45) |  |  | PK, FK reference Location(LocationID), not null |
| E\_ID | Varchar(11) |  |  | PK, FK reference Employee(E\_ID), Not null |
| From | Datetime |  | <= Getdate() | PK, not null |
| To | Datetime |  |  |  |
| Work\_Hours | Int |  |  | Not null |
| Salary | Int |  |  | Not null |

***Code:***

--create table Work

create table work(

L\_ID varchar(5) not null,

E\_ID Varchar(7) foreign key references employee(E\_ID) not null,

[From] datetime check([From] <= getdate()) not null,

[To] datetime not null,

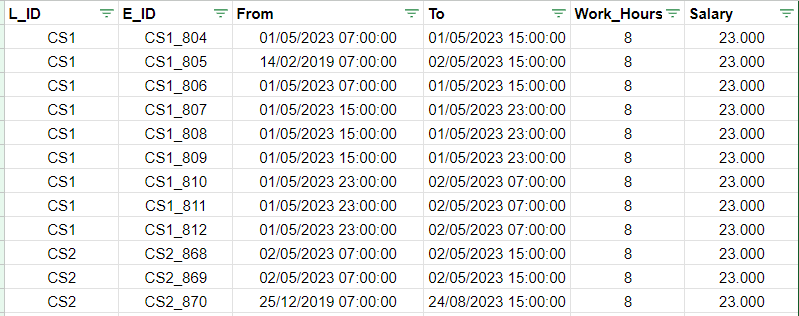
Work\_Hour int not null,

Salary int not null,

primary key (E\_ID, L\_ID, [From]),

foreign key(L\_ID, M\_ID) references locations(L\_ID, M\_ID))

***Example:***

******

### create table **Customer**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| CustomerID | **Varchar(20)** |  |  | **PK, not null** |
| Name | Nvarchar(50) |  |  | Not null |
| Point | Int |  |  |  |
| Sex | Char(1) | M | ‘F’ or ‘M’ | Not null |

***Code:***

--create table Customer

create table Customer(

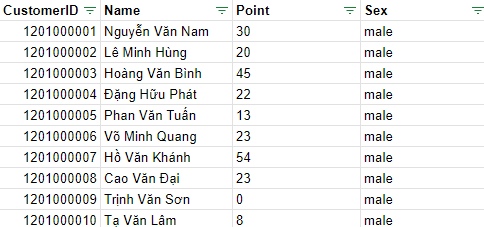
CustomerID varchar(20) primary key not null,

[Name] nvarchar(50) not null,

Point int not null,

Sex Char(1) check(Sex like 'F' or Sex like 'M') not null)

***Example:***

******

### Create table **Bill**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| Bill\_ID | **Varchar(50)** |  |  | **PK, not null, unique** |
| E\_ID | Varchar(11) |  |  | FK reference Employee(E\_ID), Not null |
| Voucher\_ID | Nvarchar(255) |  |  | FK reference Voucher(Voucher\_ID), Not null |
| C\_ID | Char(6) |  |  | FK reference Computer(C\_ID), not null |
| CustomerID | **Varchar(20)** |  |  | **FK** reference Customer(CustomerID)**, not null** |
| Used\_Point | int |  |  |  |
| Primary\_Price | int |  |  | Not null |
| Final\_Price | int |  |  |  |
| Earned\_Point | Varchar(20) |  |  |  |
| Create\_Date | Date |  |  | Not null |
| Create\_Time | Time |  |  | Not null |
| Payment\_Time | DateTime |  |  |  |

***Code:***

--create table Bill

create table Bill(

Bill\_ID varchar(50) primary key not null,

E\_ID varchar(7) foreign key references employee(E\_ID) not null,

Voucher\_ID nvarchar(255) foreign key references Voucher(Voucher\_ID) not null,

C\_ID char(6) not null,

Used\_Point int,

Primary\_Price int not null,

Final\_Price int,

Earned\_Point varchar(20),

Create\_Date date not null,

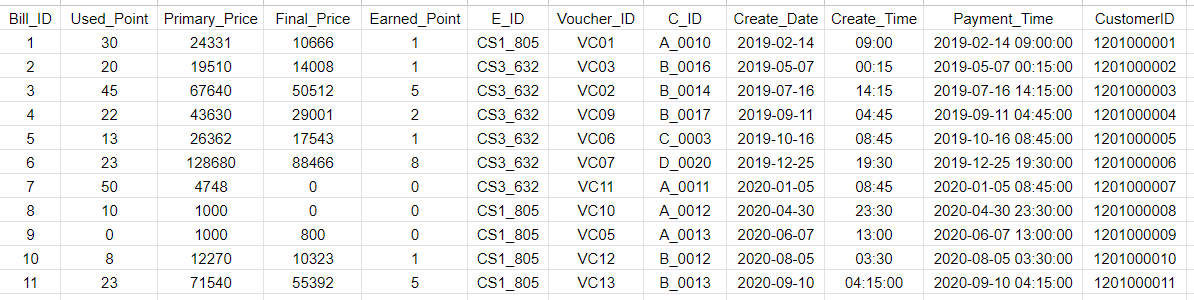
Create\_Time time not null,

Payment\_Time datetime not null,

CustomerID varchar(20) foreign key references Customer(CustomerID),

foreign key(C\_ID) references computer(C\_ID))

***Example:***

******

### Create table Account

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| UserName | **Varchar(50)** |  |  | **PK, not null** |
| Password | Varchar(50) |  |  | PK, not null |
| CustomerID | Varchar(20) |  |  | PK, FK reference Customer(CustomerID), not null |
| TimeToUse | Time |  |  | Not null |

***Code:***

--create table Account

create table Account(

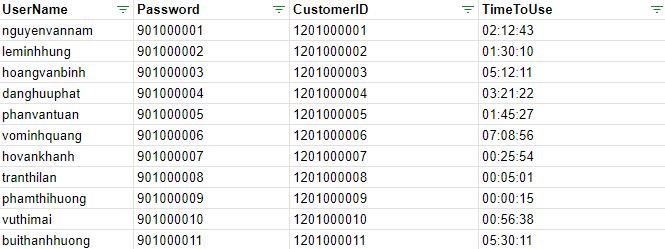
UserName varchar(50) not null,

[Password] varchar(50) not null,

CustomerID varchar(20) foreign key references Customer(CustomerID),

TimeToUse Timenot null,

primary key(UserName, [Password], CustomerID))

***Example: ***

### Create table **Order detail**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| UserName | **Varchar(20)** |  |  | **PK, FK reference Account(UserName), not null** |
| CustomerID | Varchar(20) |  |  | **PK, FK** references Account(CustomerID), not null |
| Password | Varchar(20) |  |  | PK, FK reference Account(Password), not null |
| Bill\_ID | Int |  |  | PK, FK reference Bill(Bill\_ID) |
| Product\_ID | Char(5) |  |  | PK, FK reference Product(Product\_ID) not null |
| P\_Amount | int |  |  | Not null |

***Code:***

--create table OrderDetail

create table OrderDetail(

UserName varchar(50) not null,

[Password] varchar(50) not null,

CustomerID varchar(20) not null,

Bill\_ID varchar(50) foreign key references Bill(Bill\_ID) not null,

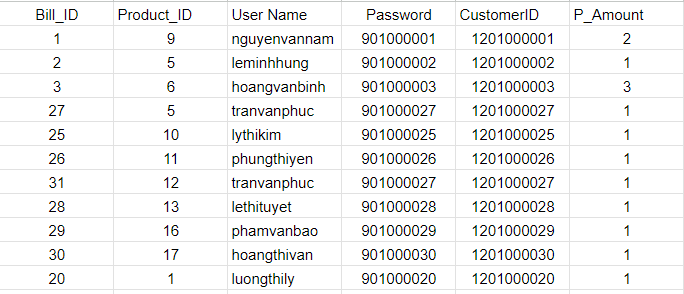
Product\_ID char(5) foreign key references Product(Product\_ID) not null,

P\_Amount int not null,

primary key(UserName, [Password], Bill\_ID, Product\_ID, CustomerID ),

foreign key(UserName, [Password], CustomerID ) references Account(UserName,[Password], CustomerID))

***Example:***

******

1. Create table **Sign In**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| C\_ID | **Char(6)** |  |  | **PK, FK reference Computer(C\_ID), not null** |
| CustomerID | Varchar(20) |  |  | **PK, FK** references Account(CustomerID), not null |
| UserName | Varchar(20) |  |  | PK, FK reference Account(UserName), not null |
| Password | Varchar(20) |  |  | PK, FK reference Account(Password), not null |
| From | DateTime |  | <=GetDate() | Not null |
| To | DateTime |  |  |  |

***Code:***

--create table SignIn

create table SignIN(

C\_ID char(6) not null,

UserName varchar(50) not null,

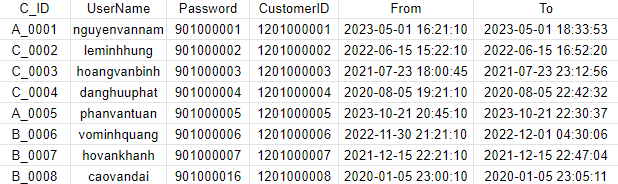
[Password] varchar(50) not null,

CustomerID varchar(20) not null,

[From] datetime check([From] <= getdate()) not null,

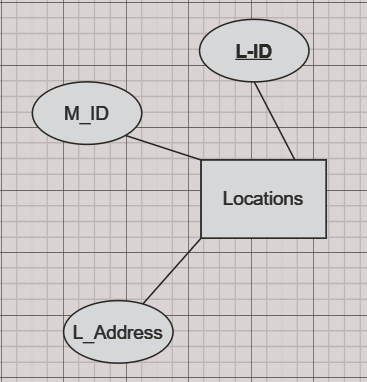
[To] datetime not null,  
 primary key(C\_ID, UserName, [Password], CustomerID, [From]),  
 foreign key(C\_ID) references computer(C\_ID)  
 foreign key(UserName, [Password], CustomerID ) references Account(UserName,[Password], CustomerID))

***Example:***



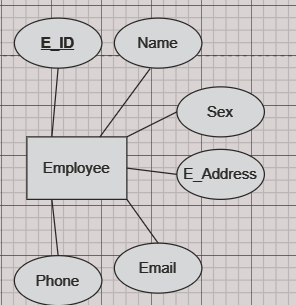
### IV. entity relationship diagram (erd)

## Location



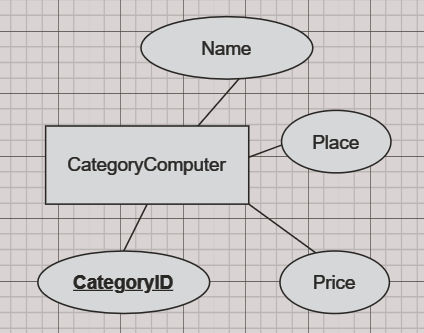
|  |  |
| --- | --- |
| This is the Location entity, the root of whole diagram. Location entity has 3attributes.  The attribute L\_ID also are the primary key of this entity. Each facility will have an address a id of manager who is an employee by. (foreign key references Employee(E\_ID)) |  |

## EMPLOYEE



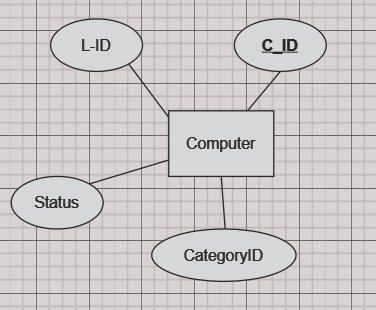
|  |  |
| --- | --- |
| This is the Employee entity. This has 6 attributes.  Each student have E\_ID is Primary key. Contaxt of Employee has Phone number as Phone, Email. Student’s Information has Name, Sex. |  |
|  |  |

## CategoryComputer

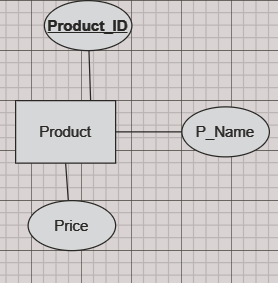


|  |  |
| --- | --- |
| This is category computer. This is 4 attributes.  Each computer configuration have CategoryID is Primary key. Each computer configuration has Name, Price as Cost per hour of computer usage and Place (The area where that type of computer is available at that facility).. |  |
|  |  |

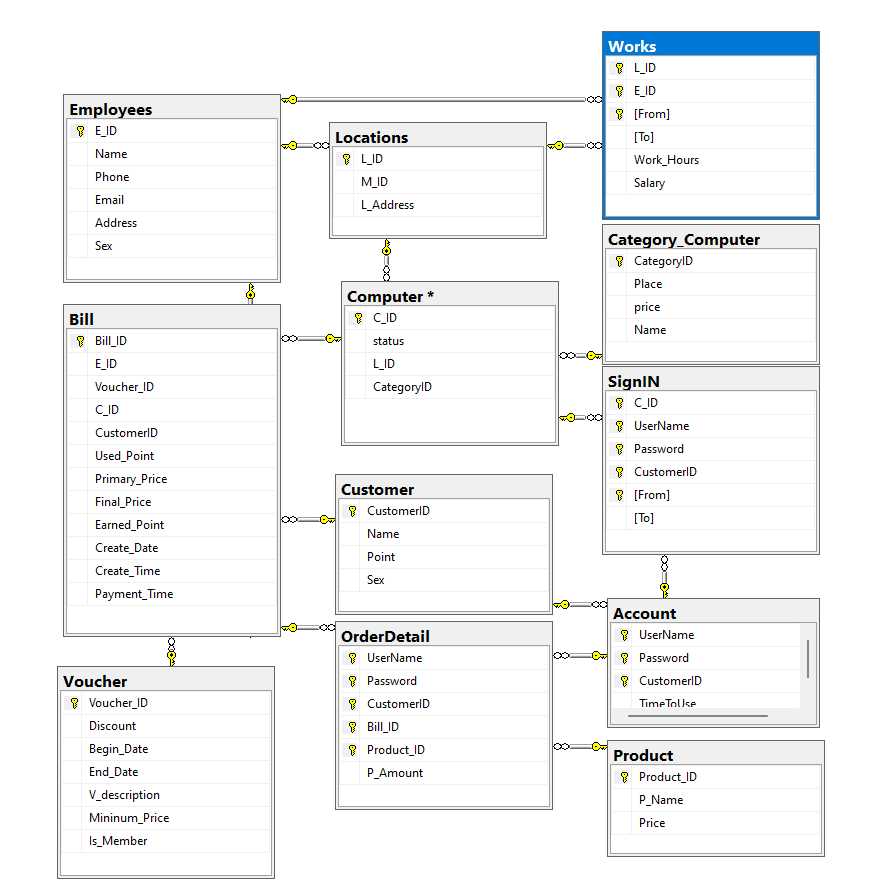
## COMPUTER



|  |  |
| --- | --- |
| CComputer entity have 4 attributes. To search for a computer, you need to enter the C\_ID, which are primary key.  To manage this Computer need information is Status as (0 or 1 or 2). Each computer will belong to a Location, L\_ID are foreign key by Location . The Last, each Computer has a configuration by CategoryComputer so this have CategoryID is foriegn key. |  |
| Product |  |
|  |  |



|  |  |
| --- | --- |
| This is the Product entity. It has 3 attributes. Each product has a Product\_ID as the primary key. The product information includes P\_Name  as the product name and Price as the product price. |  |
| Voucher   This is the Voucher entity. This has 7 attributes.  Each voucher has a Voucher\_ID as the primary key. The Voucher entity includes details such as Discount, Begin\_Date, End\_Date, and V\_Description to describe the voucher. Additionally, the voucher specifies a Mininum\_Price to be eligible for the discount. Finally, the Is\_Member attribute indicates whether the voucher is applicable only to members.     1. Work     This is the Work entity. This has 6 properties.  E\_ID, L\_ID, From are the primary key  Work is associated with the Salary and Work\_Hours, To attributes, indicating that each job will include information about the employee code, salary, location code, and hours worked.  Relationships are specified with To and From labels, which show the direction of the relationship between entities.  E\_ID foreign key references Employee(E\_ID), L\_ID references Location(L\_ID).     1. Bill     This is the Bill entity. This has 12 properties.  Bill\_ID is the primary key  Bill is associated with the Voucher\_ID, Create\_Time, Final\_Price, Primary\_Price, Bill\_ID, C\_ID, Create\_Date, Used\_Point, and Earned\_Point, Customer\_ID, Payment\_TIme attributes, indicating that each invoice will include information about the coupon code, creation time, final price, original price, invoice code, location, creation date, used points and accumulated points.  E\_ID is foreign key reference Employee who create bill at specific time, Customer\_ID is foreign key reference Customer who paid the bill. C\_ID is foreign key reference Computer where the order took place.     1. Customer     This is the Customer entity. This has 4 properties.  CustomerID is the primary key  Customer is associated with the Customer ID, Sex, Name, and Point attributes, indicating that each customer will include information about customer code, gender, name, and reward points.   Account   This is the OrderDetail entity. This has 4 properties.  CustomerID is the primary key  Accounts are associated with the CustomerID, Username, Password, and TimeToUse attributes, indicating that each account will include information about the customer, username, password, and time to use.   **Order detail**   This is the OrderDetail entity. This has 6 properties.  Bill\_ID, Product\_ID, UserName, Password, CustomerID are primary key  OrderDetail binds to the P\_Amount, Product\_ID, UserName, Password, Bill\_ID and CustomerID attributes, indicating that each order will include product, quantity, user, invoice and customer information.  UserName, Password, CustomerID are foreign key references Account, Bill\_ID foreign key reference Bill, Product\_ID foreign key reference Product.   **Sign In**   This is the Sign in entity. This has 6 attributes.  Each Sign In have C\_ID, CustomerID, UserName, Password, From are Primary key. Account has time log-in computer different From, To(Time log out).  UserName, Password, CustomerID are foreign key references Account, C\_ID foreign key reference Computer.   full diagram |  |
|  |  |



# V. SQL COMMAND

I using Microsoft SQL Server 2019, this server build intelligent, mission-critical applications using a scalable, hybrid database platform that has everything built in—from in-memory performance and advanced security to in-database analytics.

## query using order by

Code:

SELECT top 10 \*

FROM Account

ORDER BY UserName ASC;

Result:



We use the ORDER BY clause to sort the list of accounts by their points in ascending order. The SELECT Top 10 \* FROM Account command returns the first 10 records in the Account table and then sorts the records by the UserName according to the alphabet.

## query using inner join

Code:

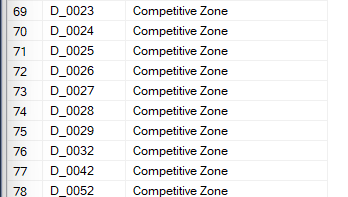
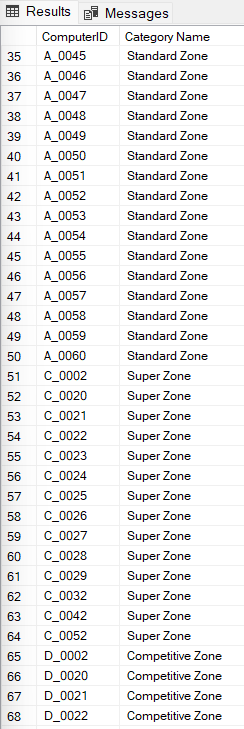
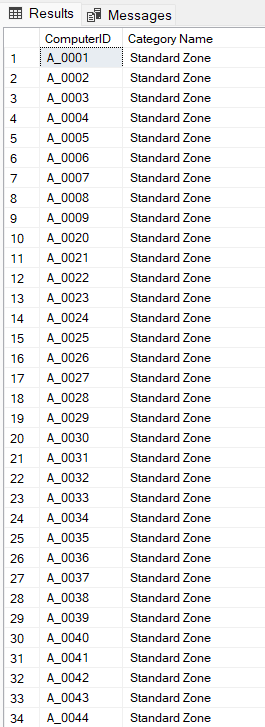
SELECT c.C\_ID AS ComputerID, cc.Name AS 'Category Name'

FROM Computer c

INNER JOIN Category\_Computer cc ON c.CategoryID = cc.CategoryID

WHERE c.Status = '0';

Result:



we use the INNER JOIN command to find computers that are available. We join the Computer table with the Category\_Computer table on the CategoryID. We then select the columns CategoryID and Place from the Computer table, and the Category Name from the Category\_Computer table, where the status of the computer is '0'. This helps us see which available computers belong to which category.

## query using aggregate functions

QUERY USING AGGREGATE FUNCTIONS

Code:

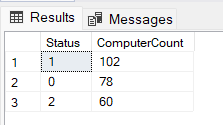
SELECT Status, COUNT(\*) AS ComputerCount

FROM Computer

GROUP BY Status

ORDER BY ComputerCount DESC;

Result:



We count the number of computers by their Status to determine how many are available and how many are in use. Using the COUNT() function with the Status parameter, we group the results by Status and sort the counts in descending order. This query helps in managing computer resources by providing insights into the number of computers that are currently available versus those that are in use.

## D. query using the group by and having clauses

Code:

SELECT CustomerID, SUM(P\_Amount) AS Total\_Products

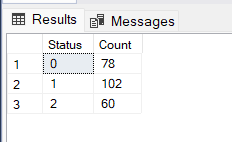
FROM OrderDetail

GROUP BY CustomerID

HAVING SUM(P\_Amount) > 2

ORDER BY CustomerID ASC;

Result:



We group the order details by their customer ID and calculate the total number of products purchased by each customer. The GROUP BY CustomerID clause is used to group the records by the CustomerID column, and SUM(P\_Amount) is used to calculate the total number of products purchased in each group. The HAVING SUM(P\_Amount) > 2 clause ensures that only groups with a total number of products greater than two are included in the results. Finally, the results are sorted by CustomerID in ascending order.

## query that uses a sub-query as a relation

Code:

SELECT

Product\_ID,

Sum(r.Amount) AS Total\_Amount

FROM (

SELECT

od.Product\_ID,

od.P\_Amount AS Amount

FROM

OrderDetail od

JOIN

Bill b ON od.Bill\_ID = b.Bill\_ID

WHERE

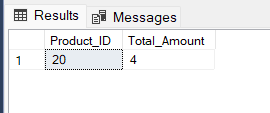
b.Create\_Date = '2022-07-23'

) r

Group by r.Product\_ID

ORDER BY Total\_Amount DESC;

Result:



We use the JOIN command to include product descriptions in our result. We join the Product table with the aggregated data from the OrderDetail and Bill tables on Product\_ID. We select Product\_ID and the amount of products sold (P\_Amount) on a specific date, and sort the results by Total\_Amount in descending order. This allows us to see which products were sold and their descriptions, along with the total quantity sold.

## query that uses partial matching in the where clause

Code:

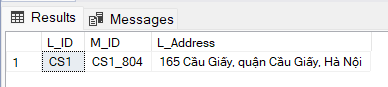
SELECT l.\*

FROM Locations l join Employees e on e.E\_ID = l.M\_ID

WHERE l.L\_Address LIKE N'%Hà Nội' AND e.Name like N'%Minh'

Order by l.L\_Address

Result:



We use the LIKE clause to find Locations whose location ends with "Hà Nội" and whose manager name end with "Anh". The SELECT l.\* FROM Locations l command selects all columns from the Locations table. The WHERE l.L\_Address LIKE N'%Hà Nội' AND e.Name like N'%Minh' condition filters the Locations to those whose location ends with "Hà Nội" and whose name be end with "Minh". The results are then ordered by the L\_Address.

## query that uses a self – join

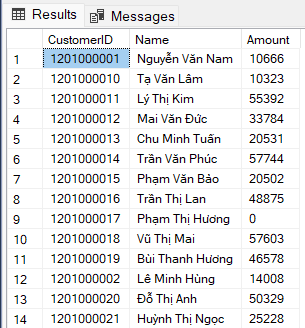
Code:

SELECT c.CustomerID, c.Name, b.Final\_Price AS Amount

FROM dbo.Customer c, dbo.Bill b

WHERE b.CustomerID = c.CustomerID;

Result:



We find customers with their respective bills. We join the Customer table with the Bill table using the WHERE clause on CustomerID. We then select the columns CustomerID, Name from the Customer table, and Amount from the Bill table. This helps us see how much each customer has to pay on their bill.

## store procedure

Code:

CREATE PROCEDURE GetCustomersByBillDate

@BillDate DATE,

@Total int OUTPUT

AS

BEGIN

SELECT @Total = Sum(b.Final\_Price)

FROM Bill b

WHERE b.Create\_Date = @BillDate;

END;

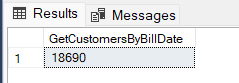
-- check

Declare @x int

exec GetCustomersByBillDate '2022-07-27', @x output

select @x GetCustomersByBillDate

Result:



We create a stored procedure to get detailed information about total price on a specific date. When executed, it returns the CustomerID, Name, Email, Bill\_ID, and Final\_Price for each customer with bills created on the specified Create\_Date.

## trigger

Code:

CREATE TRIGGER UpdateComputerStatusOnSignIn

ON SignIn

AFTER INSERT

AS

BEGIN

DECLARE @C\_ID CHAR(6);

SELECT @C\_ID = i.C\_ID

FROM Inserted i;

UPDATE Computer

SET Status = 1

WHERE C\_ID = @C\_ID

END;

-- check

insert into SignIN

values ('B\_0001', 'huynhthingoc', '901000013', ‘1201000013’, '2024-05-01 16:21:10', '2024-05-01 18:33:53')

We create a trigger named UpdateComputerStatusOnSignIn to update the status of a computer whenever a new sign-in record is inserted into the SignIn table. This trigger executes **after an INSERT** operation on the SignIn table.

THE END