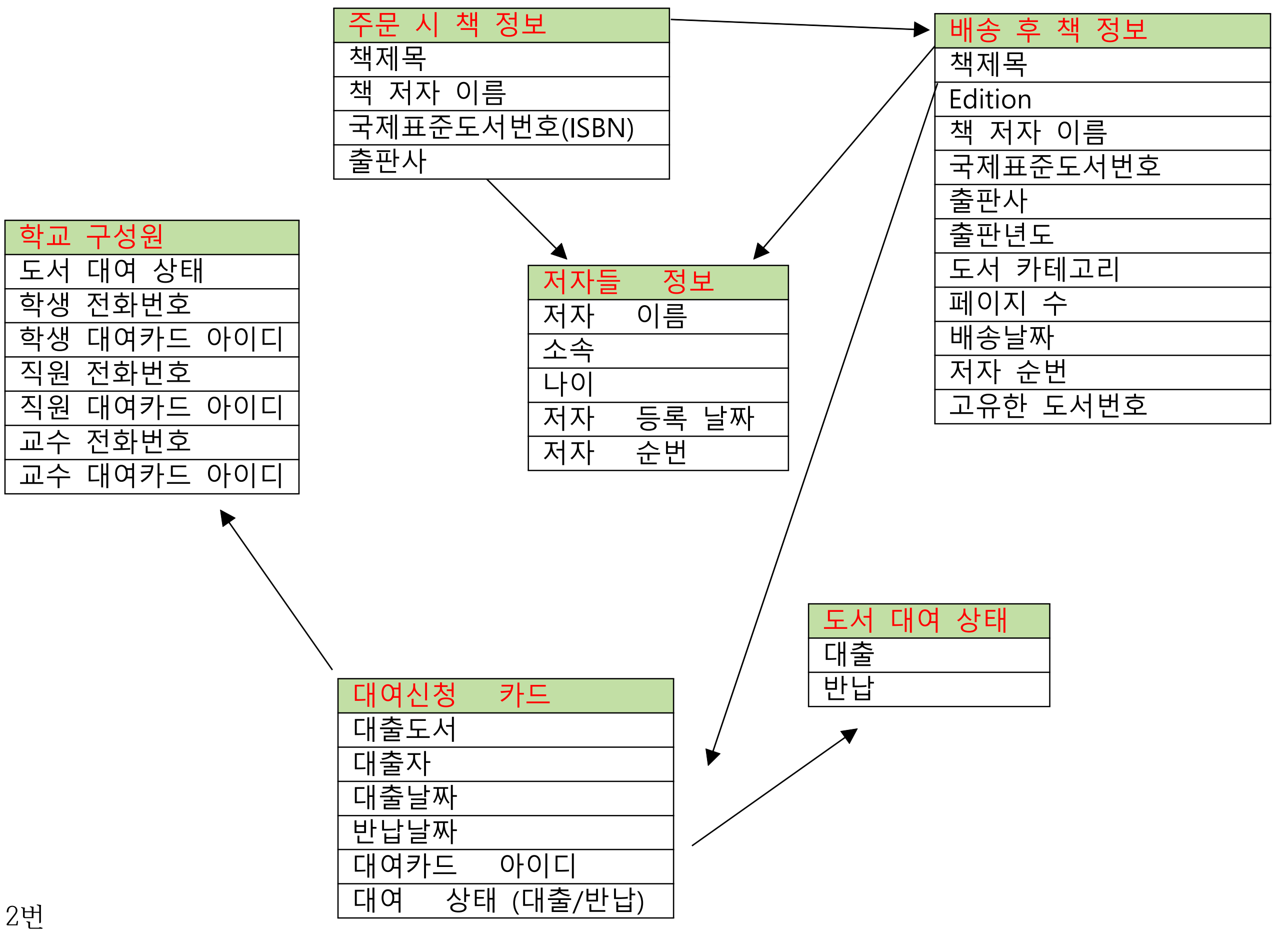
1번



2번

ACID의 개념

Atomicity: 원자성

->데이터 갱신이 전체 성공 또는 실패로 보증하는 구조

->성공시 DB에 반영, 실패시 원래상태로 Roll back

ex) 도서 대출을 할 때 대여 성공을 했다면 대출상태에 반영을 하고

실패 시에는 원래상태로 Roll back을 해야 한다. (실패 시 수행해진 것들을 삭제)

Consistency: 일관성

->트랜잭션이 성공적으로 수행된 이후에도 데이터베이스 의 제약 상태가 유지되어야 함

ex) 대여카드 아이디는 중복되면 안 되고 여러 권을 대여했을 때 일부만 되거나 하나만 되는 데이터 불일치 현상

Isolation: 고립성 또는 격리성

->Database 처리시, 복수의 사용자가 동시에 처리해도 처리가 모순되지 않고 실행을 보증함

->이를 위해 "동시성 제어"를 제공함 (TRANSACTION이 일어나는 동안 해당 Table을 Block 함)

ex) 여러 사람들이 동시에 도서를 대여했을 때 처리가 모순되면 안 됨.

또한 대출상태로 되어 있는 도서는 대출이 되지 않게 떠야함

Durability: 지속성

->Transaction이 종료되면 해당 시점의 데이터 상태가 저장되는 것을 보증함

->이상적인 시스템의 종료, 장애발생 시에도 기존 데이터가 없어지거나 지워지지 않음

->데이터 처리의 기록을 로그로 남겨 저장 보장

ex) 대출 시스템을 종료해도 데이터가 사라지지 않고 다음 날 다시 실행했을 때 각 도서의 대여상태가 유지되어야 한다.

3번
1. 현재 명사형으로 테이블 이름이 적혀있을 시 전체조직이 이해가능한 서술적 이름으로 작성하는 것이 좋다.
(명사 -> 동사화)
2. 이름은 최대한 짧지만 이해할 수 있게 작성해야한다.

<Query>
#1
```
SELECT DAYOFWEEK(orderDate) as DAYOFWEEK, sum(orderID)
From orders
Group by DAYOFWEEK(orderDate);
```

#2

```
SELECT products.CategoryID, products.ProductName,
sum(`order details`.Unitprice*(`order details`.Quantity)*(1-`order details`.Discount)) as `Total Sales`
From products
INNER JOIN `order details`
ON `order details`.ProductID = products.ProductID
Group by ProductName
Order by CategoryID, productName;
```

#3
```
SELECT customers.ContactName, month(orders.OrderDate) as Month,
sum(`order details`.UnitPrice*`order details`.Quantity*(1-`order details`.Discount)) as `Purchase Price`
FRom orders
INNER JOIN customers
ON customers.CustomerID = orders.CustomerID
INNER JOIN `order details`
ON `order details`.OrderID = orders.OrderID
Group by ContactName
Order by contactname DESC, `Purchase Price` DESC  limit 17;
```

#4
```
SELECT products.ProductName, sum(products.UnitsOnOrder) as `Total amount orders`
From products
Group by productName
Order by productName;
```

#5 (한국식 나이로 계산했습니다.)
```
Select concat(employees.FirstName, " ", employees.LastName) as `full name`,
(year(employees.hiredate)-year(employees.birthdate)+1) as `age at the time of employment`
From employees
Where employees.Title = 'Sales Representative'
Order by `age at the time of employment`;
```

```
#6
Select suppliers.CompanyName, round(avg(products.UnitPrice),2) as `Average Unit Price`
From products
INNER JOIN suppliers
ON products.SupplierID = suppliers.SupplierID
Group by companyName
Having avg(products.UnitPrice) <=15
Order by `Average Unit Price`;

#7
use northwind;
Select right(employees.HomePhone,8) as `EmployeeHomePhone`, orders.CustomerID,
substring(customers.Phone, 1, 5) as `CustomerAreaCode`,
substring(customers.Phone, 6, 20) as `customerHomePhone`, right(orders.OrderID,4) as `OrderID`
From employees
INNER JOIN customers
ON employees.City = customers.City
INNER JOIN orders
ON orders.CustomerID = customers.CustomerID
Where (orders.CustomerID = customers.CustomerID) AND (orders.OrderID between 11000 AND 11030)
Order by OrderID;



#8
SELECT products.ProductName, categories.CategoryName, suppliers.CompanyName, suppliers.Phone,
(products.UnitsInStock - products.UnitsOnOrder) as Required
From products
INNER JOIN categories
ON products.CategoryID = categories.CategoryID
INNER JOIN suppliers
ON products.SupplierID = suppliers.SupplierID
where (products.UnitsInStock - products.UnitsOnOrder) <0
Order by ProductName;

#9
use northwind;
SELECT shippers.CompanyName, count(shippers.ShipperID) as NumberOfOrder
From shippers
INNER JOIN orders
ON orders.ShipVia = shippers.ShipperID
Where Datediff(orders.ShippedDate, orders.OrderDate) <7
Group by CompanyName;
```

#10

```
SELECT left(orders.ShippedDate, 7) as ShippedMonth, count(orders.ShippedDate) as TotalOrder, round(avg(`order details`.UnitPrice*`order details`.Quantity), 2) as AvgAmount

From `order details`

INNER JOIN orders

ON orders.OrderID = `order details`.OrderID

Group by left(orders.ShippedDate, 7);
```

#11

```
SELECT suppliers.CompanyName, suppliers.City, concat(categories.CategoryName, " / ",products.ProductName) as Orderproducts, products.ReorderLevel, products.UnitsOnOrder*products.UnitPrice as TotalPrice

From products

INNER JOIN suppliers

ON products.SupplierID = suppliers.SupplierID

INNER JOIN categories

ON products.CategoryID = categories.CategoryID

Where UnitsOnOrder !=0

Group by CompanyName

Order by ReorderLevel DESC;
```

#12

```
SELECT products.ProductName, categories.Description, products.UnitPrice, suppliers.HomePage

From products

INNER JOIN categories

ON categories.CategoryID = products.CategoryID

INNER JOIN suppliers

ON suppliers.SupplierID = products.SupplierID

Where products.UnitsInStock !=0 and suppliers.HomePage is NOT NULL

Group by ProductName

Order by UnitPrice;
```

```
#13

SELECT customers.CompanyName, products.ProductName, round(`order details`.UnitPrice, 2) as UnitPrice, round(orders.Freight, 2) as Freight, orders.RequiredDate

From orders

INNER JOIN customers

ON customers.CustomerID = orders.CustomerID

INNER JOIN `order details`

ON `order details`.orderID = orders.OrderID

INNER JOIN products

ON `order details`.ProductID = products.ProductID

Where orders.Freight > 500

Order by customers.CompanyName ASC, products.ProductName ASC, orders.RequiredDate DESC;


#14

SELECT customers.ContactName, round(avg(orders.Freight), 4) as `Avg of freight`

From orders

INNER JOIN customers

ON customers.CustomerID = orders.CustomerID

Where orders.ShipCountry = 'USA'

Group by contactName

Having avg(orders.Freight) >=50

Order by avg(orders.Freight);


#15

SELECT concat(employees.FirstName, " ", employees.LastName) as EmployeeName,

count(distinct(orders.ShipCity)) as TotalShipCountry

From orders

INNER JOIN employees

ON orders.EmployeeID = employees.EmployeeID

Where concat(employees.FirstName, " ", employees.LastName) like 'A%'

Group by concat(employees.FirstName, " ", employees.LastName);
```