NIBM WORLD WIDE

SOFTWARE ENGINEERING

DATABASE MANAGEMENT SYSTEMS

SQL VIEWS AND SUB QUERIES USING DATA MANIPULATION

National Institute of Business Management

Lesson 11 – SQL Views and Sub Queries using Data Manipulation Language (DML)

Views

- In SQL, a view is a virtual table based on the result-set of an SQL statement.
- A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depends on the written SQL query to create a view.
- Views, which are a type of virtual tables allow users to do the following –
    - Structure data in a way that users or classes of users find natural or intuitive.
    - **Restrict access to the data in such a way that a user can see and (sometimes) modify exactly what they need and no more.**
    - Summarize data from various tables which can be used to generate reports.

Syntax:

```
CREATE VIEW view_name AS_SELECT column1, column2, ...

FROM table_name_WHERE condition;
```

Examples:

Create a view to display only Ename, Address and Position of the Employee table.

```
Use Company;

CREATE VIEW Employee_View AS

SELECT Ename, Address, Position

FROM Employee;
```

Run the

```
        Select * from Employee_View
```

to see the columns and data

Example 2:

Create a view to display Eno, Ename, DeptNo and Dname of the Employee table and Department table.

```
CREATE VIEW Employee_Dept_View AS

SELECT Employee.Eno, Employee.Ename, Department.DeptNo, Department.Dname

FROM Employee, Department

Where Employee.Dno = Department.DeptNo;
```

Run the

```
Select * from Employee_Dept_View
```

to see the columns and data

Example 3:

Create a view to display Eno, Ename and Department Name of all the Managers from the Employee and Department table.

```
CREATE VIEW EMP_Manager AS
SELECT Employee.Eno, Employee.Ename, Department.Dname
FROM Employee, Department
Where Employee.Dno = Department.DeptNo AND Position = 'Manager';
```

Run the

```
Select * from EMP_Manager
```

to see the columns and data

Example 4:

Create a View called EMP_Insert by selecting all the columns. And then Insert the data into the View. You can see once you insert the data into the View the data will available in the Employee table as well.

```
Use Company;
CREATE VIEW EMP_Insert AS
SELECT *
FROM Employee;
```

Then insert the data to the View

```
Insert into EMP_Insert values ('E12', 'Ishan', 'Galle', 'Cashier',20000, 'D3');
```

Run this query to see E12 new data in the Employee table

```
Select * from Employee
```

Subquery

- A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause.
- A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.
- Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc.
- The comparison operator can also be a multiple-row operator, such as IN, ANY, or ALL.
- **The Subquery or Inner query executes first before its parent query so that the results of an inner (subquery) query can be passed to the outer (Parent) query.**

Subqueries are most frequently used with the SELECT statement. The basic syntax is as follows.

```
SELECT column_names
FROM table list (can be many tables)
WHERE column_name OPERATOR
(SELECT column_names
FROM table list
WHERE condition)
```

Outer Query (Parent Query)

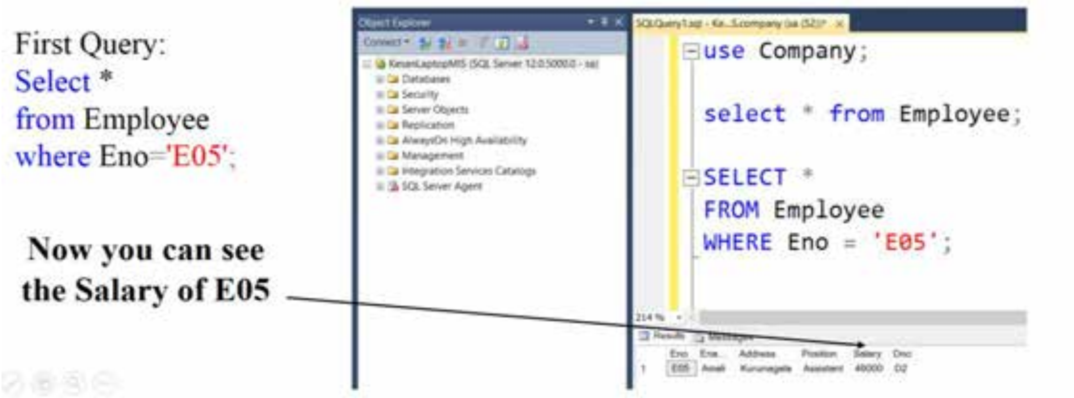
Inner Query (Subquery)

- The subquery (inner query) executes once before the main query (outer query) executes.
- The main query (outer query) use the subquery result.

Subqueries with the SELECT Statement

Suppose we want to write a query to identify all Employees (and their department) who get better salary than that of the Employee who's Eno is 'E05', but we do not know the salary of 'E05'.

To solve the problem, we require two queries. One query returns the salary (stored in Salary field) of 'E05' and a second query identifies the Employees who get better salary than the result of the first query.



First Query:

```
Select *
from Employee
where Eno='E05';
```

Now you can see the Salary of E05

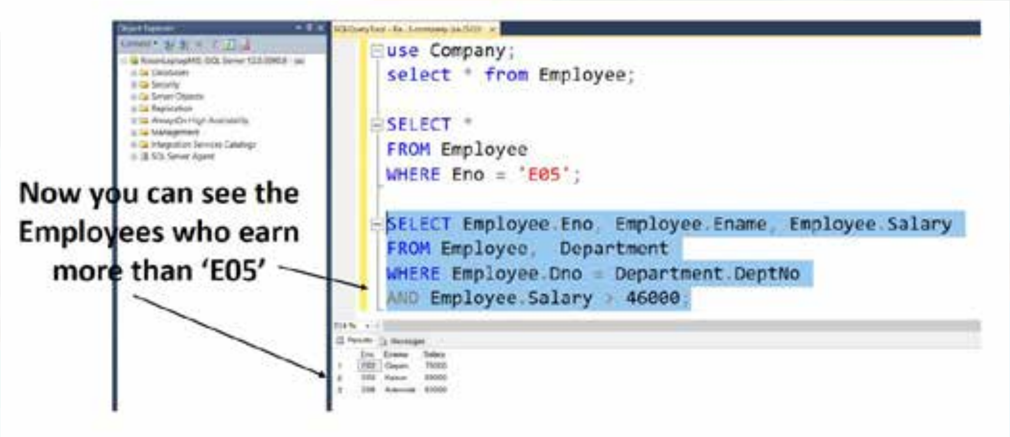
```
use Company;

select * from Employee;

SELECT *
FROM Employee
WHERE Eno = 'E05';
```

	Eno	Ena...	Address	Position	Salary	Dno
1	E05	Amali	Kurunegala	Assistant	46000	D2

Figure 11.0.1 Without Subqueries

Second Query:

```
SELECT Employee.Eno, Employee.Ename, Employee.Salary
FROM Employee,  Department
WHERE Employee.Dno = Department.DeptNo AND Employee.Salary > 46000;
```



Now you can see the Employees who earn more than 'E05'

```
use Company;
select * from Employee;

SELECT *
FROM Employee
WHERE Eno = 'E05';

SELECT Employee.Eno, Employee.Ename, Employee.Salary
FROM Employee,  Department
WHERE Employee.Dno = Department.DeptNo
AND Employee.Salary > 46000;
```

	Eno	Ename	Salary
1	E02	Gayan	75000
2	E04	Karun	59000
3	E06	Ashmonie	63000

Figure 11.0.2 Without Subqueries

Above two queries identified Employees who get the better salary than the Employee who's Eno is 'E05'.

You can combine the above two queries by placing one query inside the other. The subquery (also called the 'inner query') is the query inside the parentheses. See the following code and query result:

```
SELECT Employee.Eno, Employee.Ename, Employee.Salary
FROM Employee, Department
WHERE Employee.Dno = Department.DeptNo AND Employee.Salary >
(SELECT Salary
FROM Employee
WHERE Eno = 'E05');
```

Outer Query (Parent Query)

Inner Query (Subquery)

Example 02:

Find the Employees who has the salary greater than Amali (E05) or Aravinda (E06).

First you have to find Amali's and Aravinda's Salary separately.

```
Select Salary from Employee where Eno='E05';

Select Salary from Employee where Eno='E06';
```

Then you have to find employees whose salary greater than 46000 or 65000

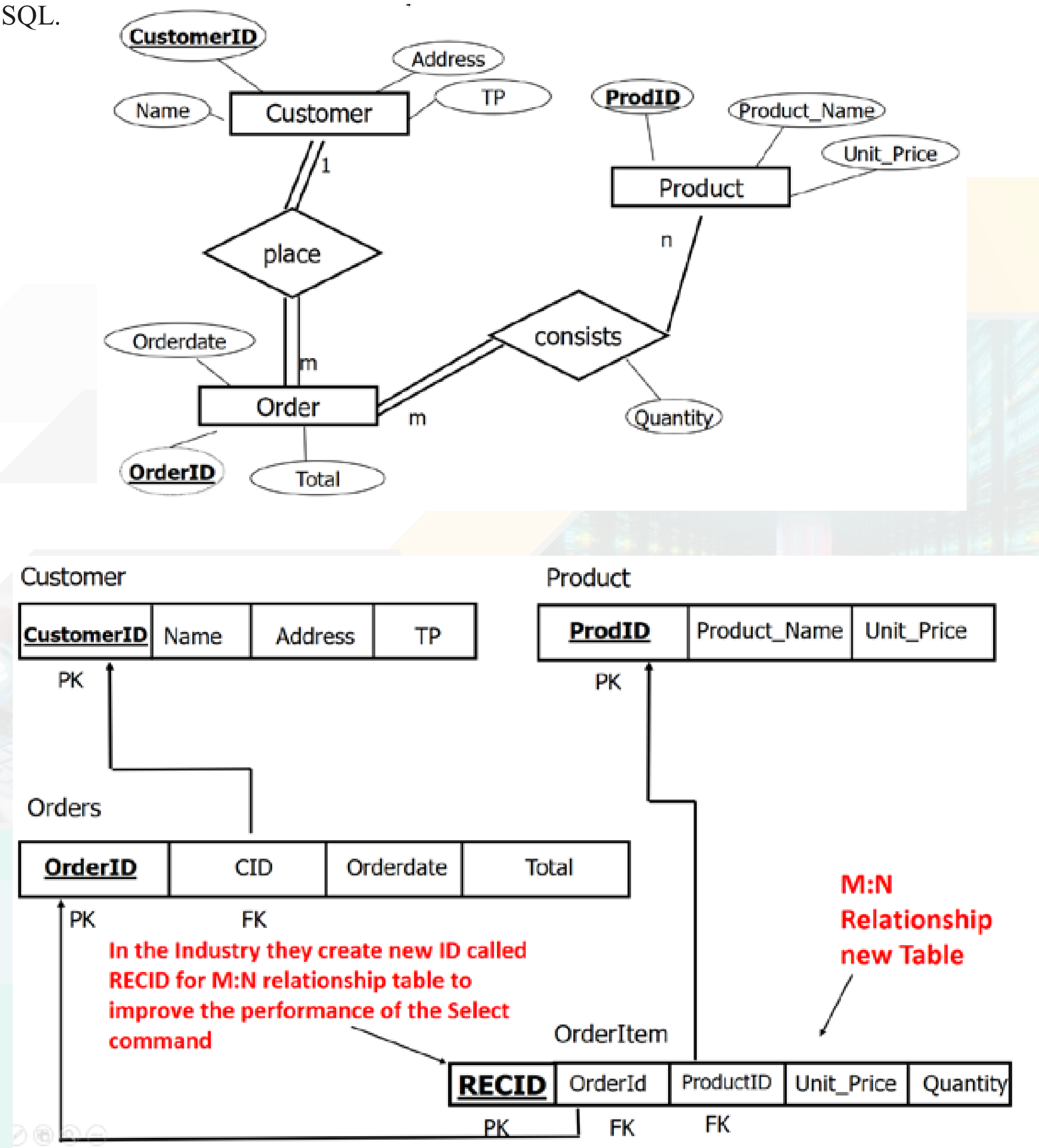
```
Select * from Employee
where Salary > 46000 or Salary > 65000;
```

Instead of running three queries you can run the following sub query

```
Select * from Employee
where Salary >
(Select Salary from Employee
where Eno='E05')
or Salary > (Select Salary from Employee
where Eno='E06');
```

Exercise:

Take the following ER Diagram. Create the Schema Mapping and create the tables in the SQL.



Customer

CustomerID	Name	Address	TP

Product

ProdID	Product_Name	Unit_Price

Orders

OrderID	CID	Orderdate	Total

In the Industry they create new ID called RECID for M:N relationship table to improve the performance of the Select command

M:N Relationship new Table

OrderItem

RECID	OrderId	ProductID	Unit_Price	Quantity

Figure 11.0.3 ER and Schema for Sub Query Exercise

```
use Company;
Create table Customer (CustomerID varchar(10) primary key, Name varchar(20), Address
varchar(20), TP int)

insert into Customer values ('C1', 'Gayan', 'Pliyandala', 0772569014);
insert into Customer values ('C2', 'Waruna','Galle', 0772569016);
insert into Customer values ('C3', 'Tom', 'New york', 0715694781);
insert into Customer values ('C4', 'David', 'Washington', 0715894562);
insert into Customer values ('C5', 'Dutch', 'Washington', 0772584785);

Create table Product (ProdID varchar(10) primary key, Product_Name varchar(20),
Unit_Price int);
insert into Product values ('P1', 'Apple', 45);
insert into Product values ('P2', 'Orange', 55);
insert into Product values ('P3', 'Grapes', 10);
insert into Product values ('P4', 'Pine Apple', 100);
insert into Product values ('P5', 'Mango', 150);
insert into Product values ('P6', 'Berry', 450);
insert into Product values ('P7', 'Sweet Orange', 78);
insert into Product values ('P8', 'Wood Apple', 50);

Create table Orders (OrderID varchar(10) primary key, CID varchar(10), Orderdate date,
Total int, Foreign Key (CID) references Customer (CustomerID));

insert into Orders values ('O1', 'C1', '2019-06-12', 9500);
insert into Orders values ('O2', 'C1', '2019-06-15', 2500);
insert into Orders values ('O3', 'C2', '2019-06-12', 19500);
insert into Orders values ('O4', 'C3', '2019-06-11', 7500);
insert into Orders values ('O5', 'C1', '2019-07-12', 2650);
insert into Orders values ('O6', 'C4', '2019-07-15', 7800);
insert into Orders values ('O7', 'C2', '2019-07-22', 4578);
insert into Orders values ('O8', 'C2', '2019-02-22', 14000);
insert into Orders values ('O9', 'C3', '2019-08-22', 2250);
insert into Orders values ('O10', 'C2', '2019-07-22', 9500);

Create table OrderItem (RECID int IDENTITY primary key, OID varchar(10), PID
varchar(10), Unit_Price int, Quantity int, Foreign Key (OID) references Orders (OrderID),
Foreign Key (PID) references Product (ProdID));

insert into OrderItem values ('O1', 'P1', 45, 15);
insert into OrderItem values ('O1', 'P2', 55, 25);
insert into OrderItem values ('O1', 'P7', 78, 18);
insert into OrderItem values ('O1', 'P8', 50, 12);
insert into OrderItem values ('O2', 'P6', 450, 10);
insert into OrderItem values ('O2', 'P3', 10, 150);
insert into OrderItem values ('O3', 'P7', 78, 20);
insert into OrderItem values ('O4', 'P2', 55, 12);
insert into OrderItem values ('O4', 'P3', 10, 22);
insert into OrderItem values ('O4', 'P8', 50, 40);
```

Subqueries with the SELECT and IN Statement

Syntax:

```
SELECT column-names
FROM table-name1
WHERE value IN
(SELECT column-name
FROM table-name2
WHERE condition);
```

Subqueries can also assign column values for each record:

Syntax:

```
SELECT column1 = (SELECT column-name FROM table-name WHERE condition),
column-names
FROM table-name
WHERE condition
```

Example 1: List products names with order quantities greater than 20.

```
SELECT Product_Name
FROM Product
WHERE ProdID IN
(SELECT PID
FROM OrderItem
WHERE Quantity > 20);
```

Outer Query (Parent Query)

Here the result of the Inner Query takes for the Outer

Inner Query (Subquery)

Example 2: List all customers with their total number of orders.

```
SELECT Name,
OrderCount = (SELECT COUNT(OrderID) FROM Orders
WHERE Orders.CID = Customer.CustomerID)
FROM Customer;
```

Subqueries with the SELECT, WHERE ANY, ALL Clause

- ANY and ALL keywords are used with a WHERE or HAVING clause.
- ANY and ALL operate on subqueries that return multiple values.
- ANY returns true if any of the subquery values meet the condition.
- ALL returns true if all of the subquery values meet the condition.

General ANY syntax:

```
SELECT column-names
FROM table-name
WHERE column-name operator ANY
(SELECT column-name
FROM table-name
WHERE condition)
```

General ALL syntax:

```
SELECT column-names
FROM table-name
WHERE column-name operator ALL
(SELECT column-name
FROM table-name
WHERE condition)
```

Example 1: Which products were sold by the unit (quantity = 10)

```
SELECT Product_Name
FROM Product
WHERE ProdID = ANY
       (SELECT PID
         FROM OrderItem
        WHERE Quantity = 10)
```

Example 2: Which products were sold more than 30 Quantity.

```
SELECT Product_Name
FROM Product
WHERE ProdID = ANY
      (SELECT PID FROM OrderItem
      WHERE Quantity > 30);
```

Example 3: List customers who placed orders that are larger than the average of each customer order.

```
SELECT DISTINCT Name
FROM Customer, Orders
WHERE Customer.CustomerID = Orders.CID
   AND Total > ALL
       (SELECT AVG (Total)
         FROM Orders
        GROUP BY CID);
```

Example 4: Find the OrderID whose maximum Quantity among all product of that OrderID is greater than average quantity of all OrderID.

```
SELECT OID
FROM OrderItem
```

```
GROUP BY OID
HAVING max (Quantity) > ALL (SELECT avg (Quantity)
                       FROM OrderItem
                       GROUP BY OID);
```

Subqueries with the Insert Statement

Create a table called EMP_Copy with all the same columns which are available in the Employee table.

```
Use Company;

Create table EMP_Copy
(Eno varchar(10), Ename varchar(10),
Address varchar(20), Position varchar(10),
Salary int, Dno varchar(10),
primary key (Eno),
foreign key (Dno) references Department (DeptNo));
```

In Company database we already created the Department table

```
INSERT INTO EMP_Copy
SELECT * FROM Employee;
```

Select all the DATA from the Employee table. So all the data of the Employee table will copy to EMP_Copy table

```
INSERT INTO EMP_Copy_Sal
SELECT * FROM Employee
Where Salary > 45000 AND Position = 'Manager';
```

Select the DATA which has Salary > 45000 and Position is Manager from the Employee table. So those specific data of the Employee table will copy to EMP_Copy table