Introduction to Database Systems

Kisi kisi:
Menjelaskan benefit data warehouse dari perusahaan (diekspektasikan untuk menjawab lebih spesifik dari case : contoh spesifik seperti ada detailnya (medecine seperti apa, insurance seperti apa) (10%)
DML (Multiple Table dengan konsep JOIN, INNER DAN OUTER) (15%)
Inner, irisan
Outer, gabungan (LEFT, RIGHT, FULL)
Antara pake ON, USING, NATURAL JOIN
Union, Intersect, Except (Yang keluar except)
Security and Administrations (DCL) (20%)
Grant, memberikan akses (privileges) ke seseorang
Revoke, copot akses (ambil privilege)
DBMS Architecture (20%)
Teleprocessing
File server
Client Server
Subqueries (35%)
Query pake aggregate
Menggunakan IN, JOIN
Menggunakan EXISTS, NOT EXISTS (T, F)
CREATE VIEW, ALTER, DROP

Menjelaskan benefit data warehouse dari perusahaan (diekspektasikan untuk menjawab lebih spesifik dari case : contoh spesifik seperti ada detailnya (medecine seperti apa, insurance seperti apa)

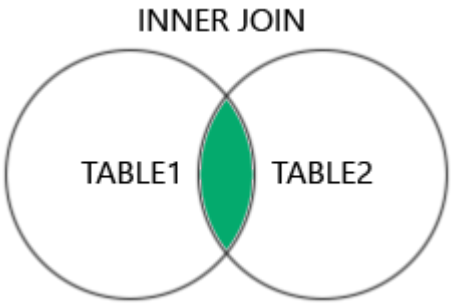
make shit up, pokoknya remember that the benefits are:

- 1 Source as all
- Competitive Advantage in a way
- Increase Productivity

- Cross-Department Integration

DML

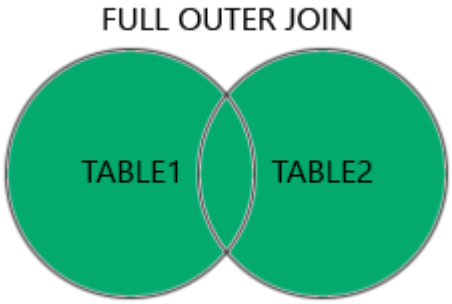
Inner Join




```
select ProductID, ProductName, CategoryName
from Products
INNER join Categories on Products.CategoryID = Categories.CategoryID
```

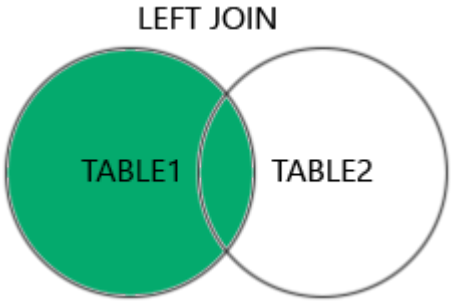
Outer Joins:

- Full Outer Join



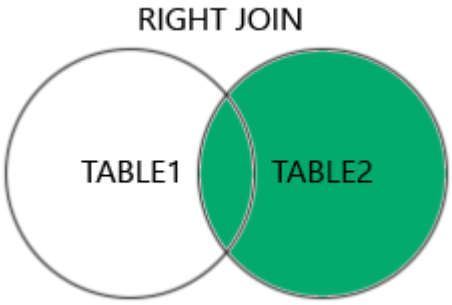

```
select Customers.CustomerName, Orders.OrderID
from Customers
full outer join Orders on Customers.CustomerID=Orders.CustomerID
order by Customers.CustomerName
```

- Left Outer Join




```
select Customers.CustomerName, Orders.OrderID
from Customers
left outer join Orders on Customers.CustomerID = Orders.CustomerID
order by Customers.CustomerName
```

- Right Outer Join




```
select Orders.OrderID, Employees.LastName, Employees.FirstName
from Orders
right outer join Employees on Orders.EmployeeID = Employees.EmployeeID
order by Orders.OrderID
```

Union, Intersect, Except

The `UNION` operator is used to combine the result-set of two or more `SELECT` statements.

- Every `SELECT` statement within `UNION` must have the same number of columns
- The columns must also have similar data types
- The columns in every `SELECT` statement must also be in the same order

```
select City
from Customers
union
select City
```

```
from Suppliers
order by City
```

The `INTERSECT` operator is used to combine the similar result-set of two `SELECT` statements.

```
select City
from Customers
intersect
select City
from Suppliers
order by City
```

The `EXISTS` operator is used to check whether the result of a correlated nested query is empty or not.

```
select SupplierName
from Suppliers
where exists(
select 1
from products
where products.SupplierID = Suppliers,SupplierID and price<20)
```

DCL

Grant

- Grants permission to a user or group of users or role.
  V Just Granting some privilege types

```
grant pvt on Suppliers to user1
```

V with the ability to change one's role

```
grant pvt on Suppliers to user1 with grant option
```

Revoke

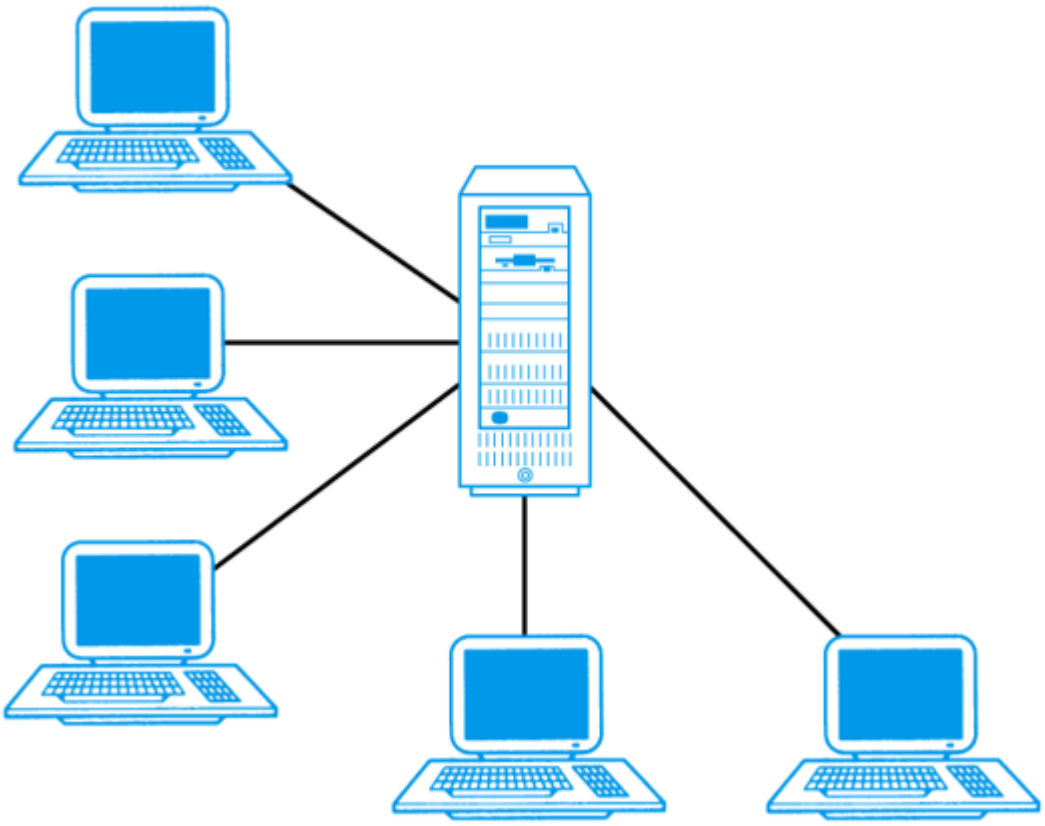
- Removes previous granted privileges from a user account, taking away their access to certain database objects or action.

```
revoke pvt on Suppliers from user1
```

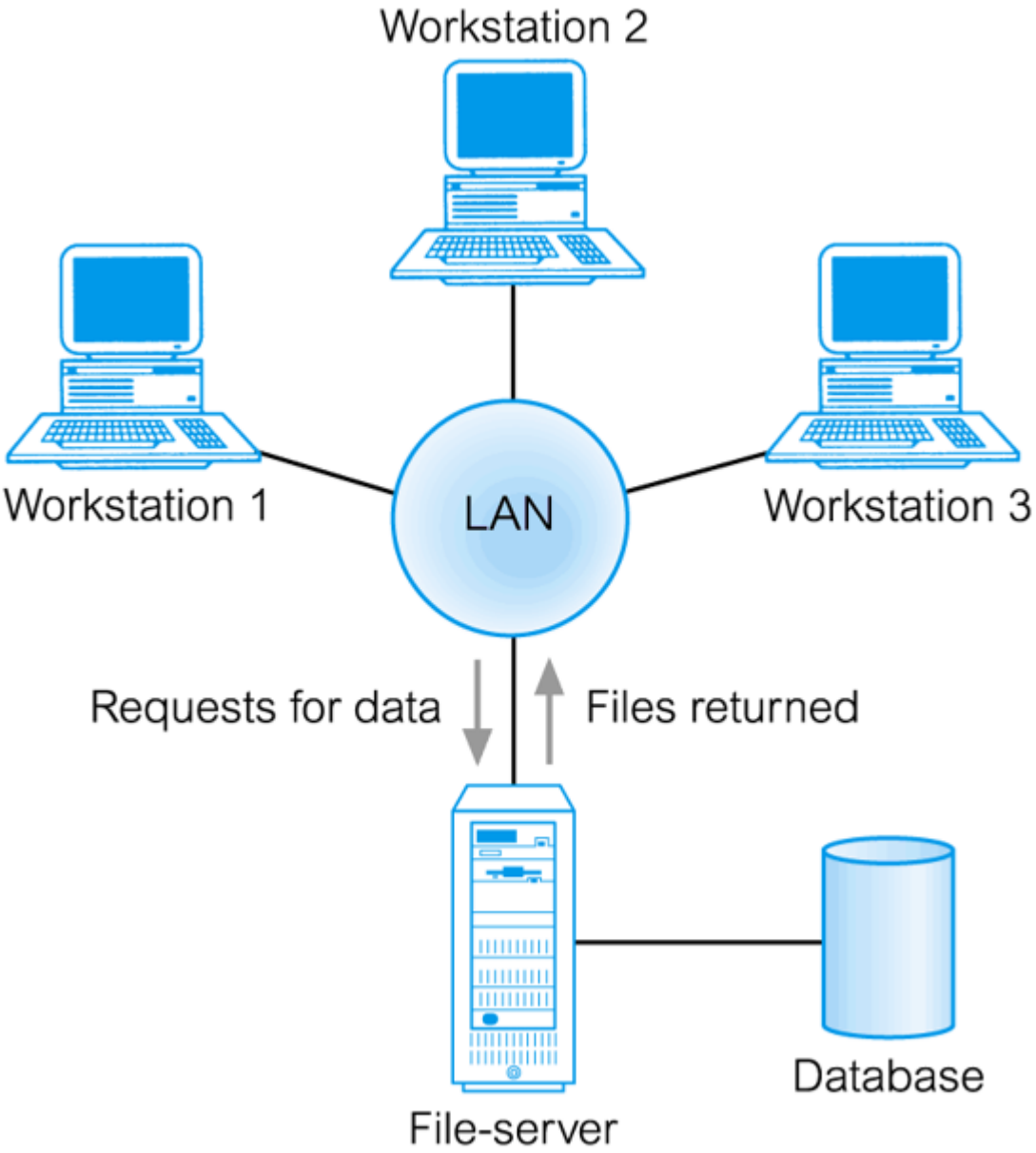
All PVTs:

Privilege Types
Create
Insert
Select
Update

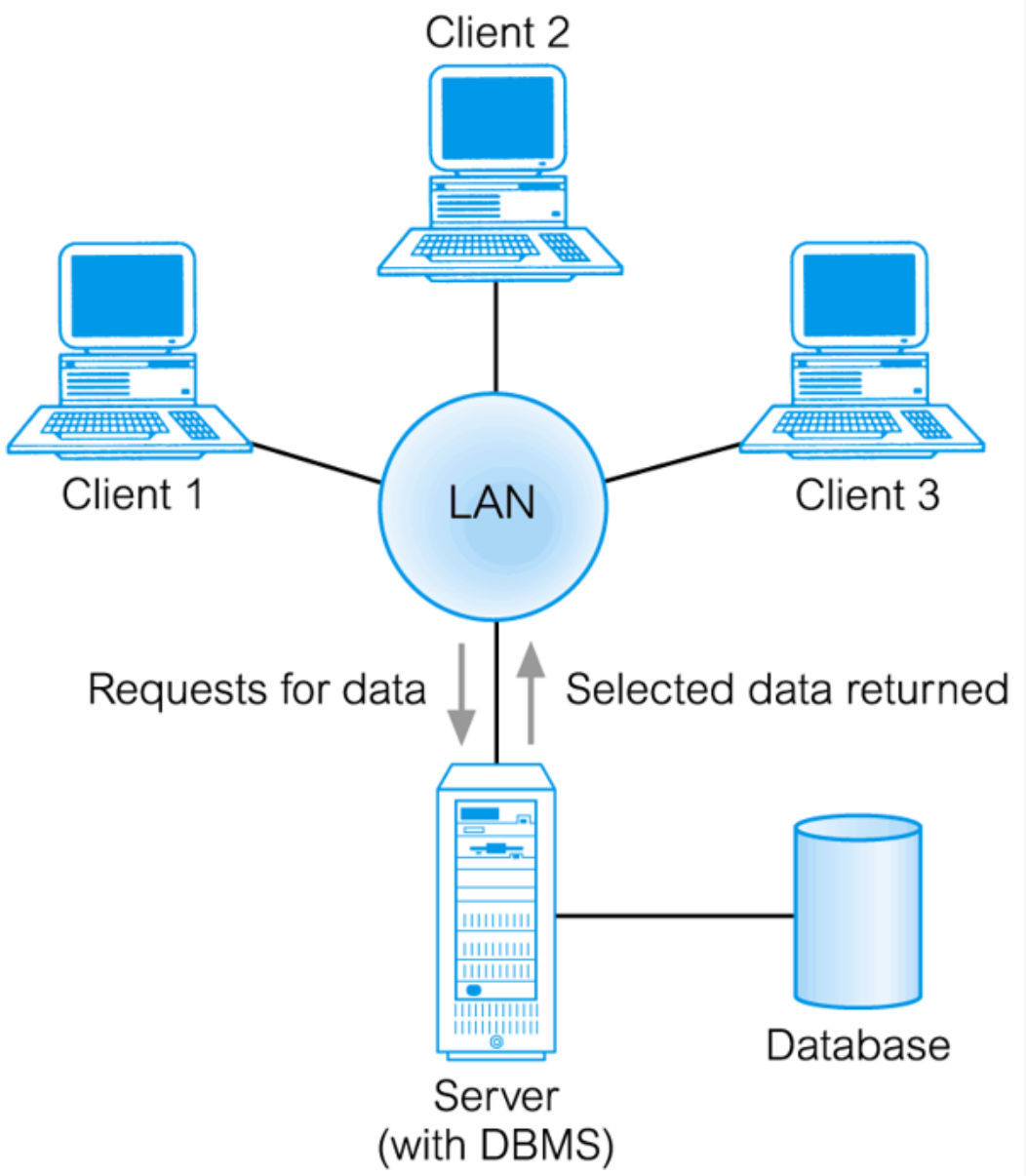
Database Architecture

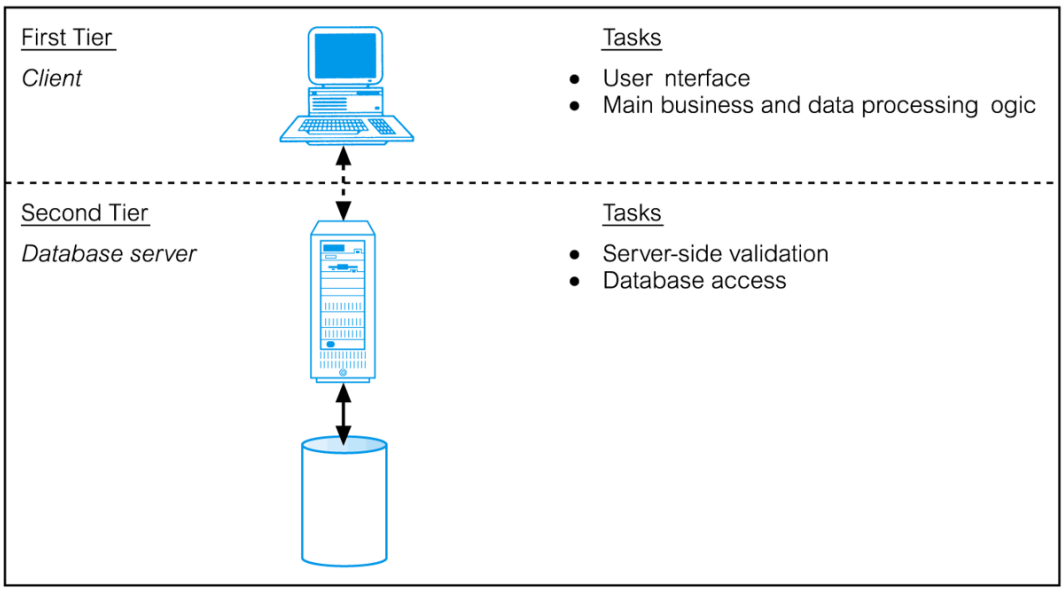
- Teleprocessing
  - Traditional
  - Single mainframe with a number of terminals attached.
  - Single computer processing unit.
  - 
  - Disadvantages:
    - Slow, since the processes is being performed within the same physical computer.

        - If the mainframe is broken, gws.
        - Downsizing trend.
- File server
    - Connection across several workstations in the network (LAN)
    - Database resides on a file-server
    - DBMS and applications work on each workstation



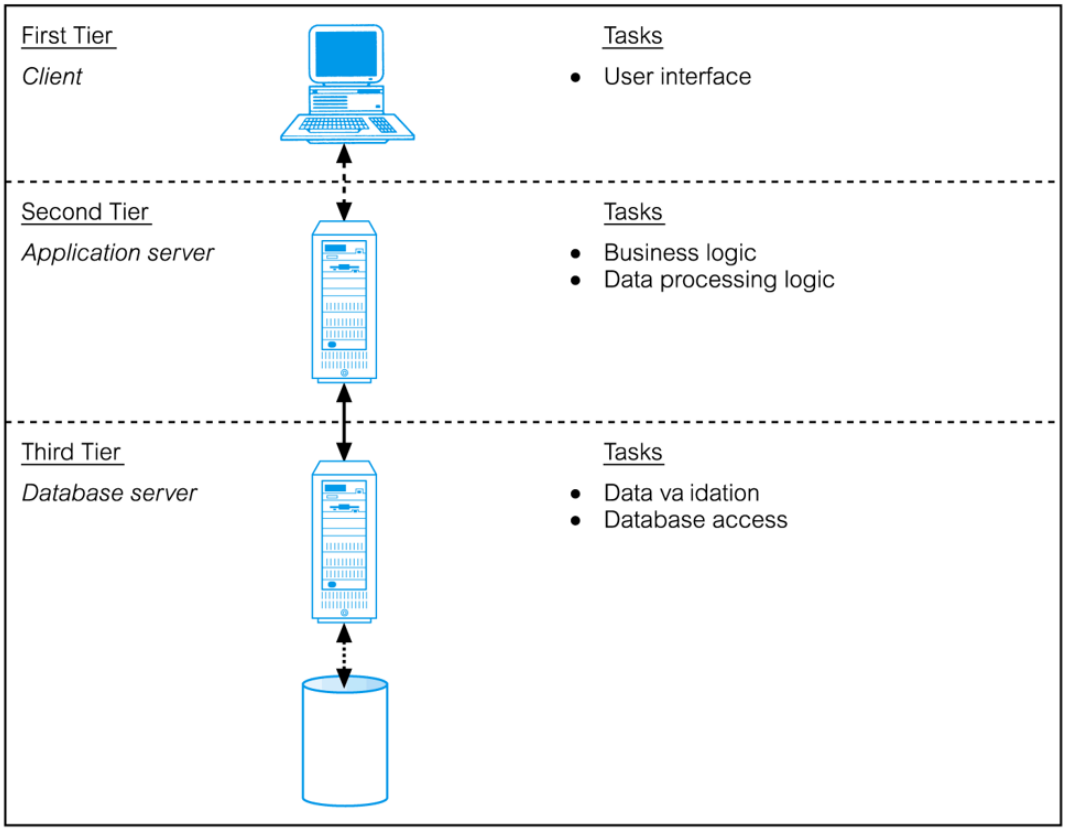
    - Disadvantages:
        - Network traffic jams.
        - Redundancy is inevitable, as each workstation needs a dbms.
        - Concurrent (Recovery and integrity would be harder than ever).
- Client Server
    - Traditional Two-Tier Client (1, apps) - Server(2, db & dbms)
        - Advantages:

- Wider access to existing databases.
- Increased performance.
- Possible reduction in hardware costs.
- Reduction in communication costs.
- Increased consistency.
- 



CLIENT	SERVER
Manages the user interface	Accepts and processes database requests from clients
Accepts and checks syntax of user input	Checks authorization
Processes application logic	Ensures integrity constraints not violated
Generates database requests and transmits to server	Performs query/update processing and transmits response to client
Passes response back to user	Maintains system catalog Provides concurrent database access Provides recovery control

- Three-Tier Client(1,ui) - Server((2,apps),(3,db & dbms))
  - Advantages:
    - 'Thin' client.
    - Application maintenance centralized.
    - Easier to modify or replace one tier without affecting others.
    - Separating business logic from database functions, making it easier to implement load balancing.
    - Maps quite naturally to web environments.



- Subqueries
  - Aggregates
    An aggregate function is a function that performs a calculation on a set of values, and returns a single value.
    - Subquery Examples:

```
-- Min Max
select FilmName,FilmRunTimeMinutes
from tblFilm
where FilmRunTimeMinutes =
(select min / max(FilmRunTimeMinutes)
from tblFilm)
```

```
-- Average
select FilmName, FilmRunTimeMinutes
from tblFilm
where FilmRunTimeMinutes > (select avg(FilmRunTimeMinutes)
from tblFilm)
```

Aggregate Basics
Min
Max

Aggregate Basics
Count
Sum
Avg

```
        - IN
```

```
select CustomerName, CustomerAddress
from MsCustomer mc
where CustomerID in(
        select CustomerID
        from TransactionHeader th
        group by CustomerID
        having count(TransactionID) > (
                select avg(TransactionCount) from (
                        select CustomerID, count(TransactionID) as
TransactionCount
                        from TransactionHeader th
                        group by CustomerID
                )AvgTransaction))
```

```
        - Join
```

```
select CustomerName, CustomerEmail, [Fish Type Variant]
from MsCustomer mc
join TransactionHeader th on mc.CustomerID = th.CustomerID, (
    select th.TransactionId, count(distinct mf.FishTypeID) as [Fish Type
Variant]
    from TransactionHeader th
    join TransactionDetail td on th.TransactionID = td.TransactionID
    join MsFish mf on td.FishID = mf.FishID
    group by th.TransactionId) a, (
    select MIN([Fish Type Variant]) as [Min Fish Type Variant]
    from (
       select th.TransactionId, count(distinct mf.FishTypeID) as [Fish Type
Variant]
       from TransactionHeader th
       join TransactionDetail td on th.TransactionID = td.TransactionID
       join MsFish mf on td.FishID = mf.FishID
       group by th.TransactionId) a) b
```

```
where a.TransactionID = th.TransactionID
and a. [Fish Type Variant] = b.[Min Fish Type Variant]
```

```
        - Exists
```

```
select CustomerName, count(TransactionID) as 'NumberOfTransaction'
from MsCustomer mc
join TransactionHeader th on mc.CustomerID = th.CustomerID
where CustomerGender = 'Female' and exists(
        select 1
        from MsStaff ms
        where th.StaffID = ms.StaffID and StaffGender = 'Male')
group by CustomerName
```

```
        - Not Exists
```

```
select CustomerName, count(TransactionID) as 'NumberOfTransaction'
from MsCustomer mc
join TransactionHeader th on mc.CustomerID = th.CustomerID
where CustomerGender = 'Female' and not exists(
        select 1
        from MsStaff ms
        where th.StaffID = ms.StaffID and StaffGender = 'Male')
group by CustomerName
```

Views

View is a virtual table based on the result-set of an SQL statement.

- Create View Example

```
create view [Alsut Customers] as
select CustomerName, ContactName
from Customer
where City = 'Alam Sutera';
```

- Create or Replace View Example

```
create or replace view [Alsut Customers] as
select CustomerName, ContactName, City
from Customers
where City = 'Alam Sutera';
```

- Dropping a View Example

```
Drop View [Alsut Customers]
```