**Using CTE:**

**with** uo\_cte(id,CustomerID,ShoppingListstatus,ShoppingListdate)  
**as**  
(  
SELECT  top 100 Id,CustomerID,ShoppingListStatus,ShoppingListDate FROM UserShoppingList where ShoppingListDate<’2013-02-10′  
)  
SELECT ShoppingListstatus,COUNT(id) from uo\_cte  
group by ShoppingListstatus

**Using SubQuery:**

**SELECT** subq1.ShoppingListStatus,COUNT(subq1.Id) from  
(  
SELECT  top 100 Id,CustomerID,ShoppingListStatus,ShoppingListDate FROM UserShoppingList where ShoppingListDate<’2013-02-10′  
)**subq1**  
GROUP by subq1.ShoppingListStatus

<http://bradsruminations.blogspot.in/2011/05/cte-coolest-t-sql-enhancement.html>

**Difference between CTE and View**

               A CTE is a temporary/logical View, it is not store physically. It is a named query, the result for which is only available to the very next query after the CTE is defined. CTE is defined using WITH clause.

                  A View is a physical object that is present in the database. View is as good as a Table but it doesn't store data physically as compared to a table, only the data schema is stored in View. View,when referred, pulls data by executing the query that is associated with it.

**Stackoverflow:**

                 The biggest difference between a CTE and View, is that, View or derived table cannot call itself, whereas CTE can call itself and hence support recursion.

A table contains data, a view is just a SELECT statement which has been saved in the database (more or less, depending on your database).

The advantage of a view is that it can join data from several tables thus creating a new view of it. Say you have a database with salaries and you need to do some complex statistical queries on it.

Instead of sending the complex query to the database all the time, you can save the query as a view and then SELECT \* FROM vie

**What is the difference between a Table, View and Synonym in SQL?**

A Table is a repository of data, where in the table itself is a physical entity. The table resides physically in the database.

A View is not a part of the database's physical representation. It is precompiled, so that data retrieval behaves faster, and also provide a secure accessibility mechanism.

**See code example below:**

Create table T\_EMPLOYEE

(Emp\_Id integer primary key,

Name varchar2(50)

Skillset varchar2(200),

Salary number(12,2),

DOB datetime);

Say there is a scenario where Salary is not to be shown to a group of users, a View may be created to display allowable information:

Create view EMP\_SOME\_DETAILS

as

(Select Emp\_Id, Name, Skillset, DOB

From T\_EMPLOYEE);

**The advantages of using a view are as follows:**

- It may access data from a table, multiple tables, view, multiple views, or a combination of these

- A view connects to the data of its base table(s).

- Provides a secure mechanism of data accessibility

**Synonym** is alternate name assigned to a table, view, sequence or program unit.

- It may be used to shadow the original name and owner of the actual entity

- Extends the reach of tables, by allowing public access to the synonym

***1) What is a view useful for?***

**IOPO** In One Place Only  
  
•Whether you consider the data itself or the queries that reference the joined tables, utilizing a view avoids unnecessary redundancy.   
  
•Views also provide an abstracting layer preventing direct access to the tables (and the resulting handcuffing referencing physical dependencies). In fact, I think it's *good practice*1 to offer only abstracted access to your underlying data (using views & table-valued functions), including views such as   
  
CREATE VIEW AS   
      SELECT \* FROM tblData  
  
1I hafta admit there's a good deal of "Do as I say; not as I do" in that advice ;)

*2) Are there any situations in which it is tempting to use a view when you shouldn't use one?*

Performance in view joins used to be a concern (e.g. SQL 2000). I'm no expert, but I haven't worried about it in a while. (Nor can I think of where I'm presently using view joins.)  
  
Another situation where a view might be overkill is when the view is only referenced from one calling location and a derived table could be used instead. Just like an anonymous type is preferable to a class in .NET if the anonymous type is only used/referenced once.  
  
    • See the derived table description in   http://msdn.microsoft.com/en-us/library/ms177634.aspx

*3) Why would you use a view in lieu of something like a table-valued function or vice versa?*

(Aside from performance reasons) A table-valued function is functionally equivalent to a parameterized view. In fact, a common simple table-valued function use case is simply to add a WHERE clause filter to an already existing view in a single object.

*4) Are there any circumstances that a view might be useful that aren't apparent at first glance?*

**A view provides several benefits.**

**1. Views can hide complexity**

If you have a query that requires joining several tables, or has complex logic or calculations, you can code all that logic into a view, then select from the view just like you would a table.

**2. Views can be used as a security mechanism**

A view can select certain columns and/or rows from a table, and permissions set on the view instead of the underlying tables. This allows surfacing only the data that a user needs to see.

**3. Views can simplify supporting legacy code**

If you need to refactor a table that would break a lot of code, you can replace the table with a view of the same name. The view provides the exact same schema as the original table, while the actual schema has changed. This keeps the legacy code that references the table from breaking, allowing you to change the legacy code at your leisure.

These are just some of the many examples of how views can be useful.