Advantages of derived tables:

1. A derived table is part of a larger, single query, and will be optimized in the context of the rest of the query. This can be an advantage, if the query optimization helps performance (it usually does, with some exceptions). Example: if you populate a temp table, then consume the results in a second query, you are in effect tying the database engine to one execution method (run the first query in its entirety, save the whole result, run the second query) where with a derived table the optimizer might be able to find a faster execution method or access path.
2. A derived table only "exists" in terms of the query execution plan - it's purely a logical construct. There really is no table.

Advantages of temp tables

1. The table "exists" - that is, it's materialized as a table, at least in memory, which contains the result set and can be reused.
2. In some cases, performance can be improved or blocking reduced when you have to perform some elaborate transformation on the data - for example, if you want to fetch a 'snapshot' set of rows out of a base table that is busy, and then do some complicated calculation on that set, there can be less contention if you get the rows out of the base table and unlock it as quickly as possible, then do the work independently. In some cases the overhead of a real temp table is small relative to the advantage in concurrency.

Derived table is a logical construct.

It may be stored in the `tempdb`, built at runtime by reevaluating the underlying statement each time it is accessed, or even optimized out at all.

Temporary table is a physical construct. It is a table in `tempdb` that is created and populated with the values.

Which one is better depends on the query they are used in, the statement that is used to derive a table, and many other factors.

For instance, `CTE` (common table expressions) in `SQL Server` can (and most probably will) be reevaluated each time they are used. This query:

```
WITH    q (uuid) AS
        (
        SELECT  NEWID()
        )
SELECT  *
FROM    q
UNION ALL
SELECT  *
FROM    q
```

will *most probably* yield two different `NEWID()`'s.

In this case, a temporary table should be used since it guarantees that its values persist.

On the other hand, this query:

```
SELECT  *
FROM    (
        SELECT  *, ROW_NUMBER() OVER (ORDER BY id) AS rn
        FROM    master
        ) q
WHERE   rn BETWEEN 80 AND 100
```

is better with a derived table, because using a temporary table will require fetching all values from `master`, while this solution will just scan the first `100` records using the index on `id`.