

SQL

SELECT – Aggregate Functions

Aggregate Functions

An **aggregate** is a collection of items that are gathered together to form a total quantity.

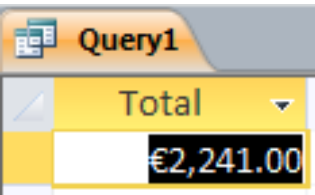
In statistics, an aggregate is data combined from several measurements (categories).

Consider the following table *Sales*:

Sale_Id	Sale_Date	Sale_Value	Stock_No	Qty_Sold
1	01/08/2010	€130.00	002	010
2	02/08/2010	€150.00	001	002
3	08/08/2010	€315.00	003	003
4	15/08/2010	€375.00	001	005
5	16/08/2010	€150.00	001	002
6	21/08/2010	€195.00	002	003
7	01/09/2010	€150.00	001	002
8	02/09/2010	€420.00	003	004
9	06/09/2010	€225.00	001	003
10	09/09/2010	€75.00	001	001
11	12/09/2010	€56.00	001	025

What is the total value of all sales?

The following results is required:



Query1
Total
€2,241.00

How is this achieved?

Must use an **aggregate function**.

The ISO SQL standard defines **five** aggregate functions:

- COUNT(col name)
- SUM(col name)
- AVG(col name)
- MIN(col name)
- MAX(col name)

(more have been added)

Aggregate functions operate on a single column of a table and return a single value.

COUNT / MIN / MAX can be applied to both numeric and non-numeric fields. **SUM()** and **AVG()** can be applied to numeric fields **only** .

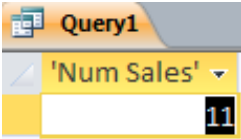
COUNT()

Returns the *number* of values in a specified column.

When counting the number of rows in a table, any column may be used in the COUNT function.

For example:
List the number of sales in the Sales table.

```
SELECT COUNT(Sale_ID) AS 'Num Sales'
FROM Sales;
```



```
SELECT COUNT(Sale_Date) AS 'Num Sales'
FROM Sales;
```

Returns the same result set.

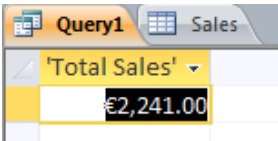
SUM()

Returns the *sum* of the values in a specified column.

Must be applied to a numeric field.

For example:
List the value of all sales.

```
SELECT SUM(Sale_Value) AS 'Total Sales'
FROM Sales;
```



MIN()/MAX()

Returns the *smallest* / largest value in a specified column

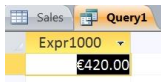
Find the smallest sale in the table Sales

```
SELECT MIN(Sale_Value)
FROM Sales;
```



Find the largest sale in the table Sales:

```
SELECT MAX(Sale_Value)
FROM Staff;
```



A screenshot of a database query result. The table has two columns: 'Sales' and 'Query1'. The 'Sales' column contains 'Expr1000' and the 'Query1' column contains '£420.00'.

MIN() / MAX() can be applied to non-numeric values:

```
SELECT MIN(Supp_Name)
FROM Suppliers;
```



A screenshot of a database query result. The table has two columns: 'Sales' and 'Query1'. The 'Sales' column contains 'Expr1000' and the 'Query1' column contains 'Levi Co'.

```
SELECT MAX(Surname)
FROM Staff;
```



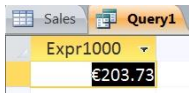
A screenshot of a database query result. The table has two columns: 'Sales' and 'Query1'. The 'Sales' column contains 'Expr1000' and the 'Query1' column contains 'Wrangler'.

AVG()

Returns the *average* of the values in a specified column

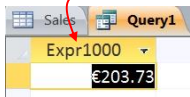
Find the average sale value

```
SELECT AVG(Sale_Value)
FROM Sales;
```



A screenshot of a database query result. The table has two columns: 'Sales' and 'Query1'. The 'Sales' column contains 'Expr1000' and the 'Query1' column contains '£203.73'.

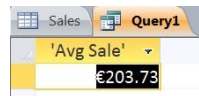
Notice the header on the output column.



A screenshot of a database query result. The table has two columns: 'Sales' and 'Query1'. The 'Sales' column contains 'Expr1000' and the 'Query1' column contains '£203.73'. A red arrow points to the 'Query1' column header.

Using a *column alias* makes the output better

```
SELECT AVG(Sale_Value) AS 'Avg Sale'
FROM Sales;
```



A screenshot of a database query result. The table has two columns: 'Sales' and 'Query1'. The 'Sales' column contains 'Avg Sale' and the 'Query1' column contains '£203.73'.

Aggregate Functions & NULL Values

If a column of the table to which the aggregate function is being applied contains NULL values, then the record is eliminated from the evaluation.

Be careful!

COUNT excludes cells with a NULL value.

Consider the following table Stock:

Stock_No	Description	Cost_Price	Sale_Price	Qty	Supp_Id
1	Levis 501	€50.00	€75.00	100	1
2	Wrangler Regular	€40.00		50	2
3	Levis 901	€80.00	€105.00	35	1
4	Pepe Skinny	€45.00	€70.00	20	3

It has not yet been decided what the Sale_Price for Stock_No 2 is to be (Null value).

There are clearly four (4) stock items in the table Stock

Finding the number of stock items using the Sale_Price column as follows:

```
SELECT COUNT(Sale_Price) AS 'No Stock'
FROM Stock;
```

Returns the following result:

Query1	Stock
'No Stock'	3

Useful Information

- The use of the column alias gives a better presentation of the result (interactive SQL)
- It is possible to apply more than one aggregate function in a singles SELECT statement:

```
SELECT MIN(Sale_Value), MAX(Sale_Value), AVG(Sale_Value)
FROM Stock;
```

- Aggregate functions can be used only in the **SELECT** and the **HAVING** clause of a query.

The following is invalid:

```
SELECT *
FROM Sales
WHERE Sale_Value > AVG(Sale_Value);
```



Consider the following query:

```
SELECT Stock_No, COUNT(Description)
FROM Stock_No; ❌
```

Which Stock_No would be returned in the result?

What is the result of this query?

GROUP BY

Sometimes we require subtotals in reports. To do this, use the **GROUP BY** clause.

The query is then referred to as a **Grouped Query**.

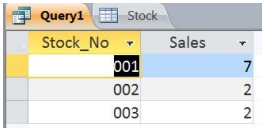
A single summary row is produced for each group.

Columns named in the GROUP BY clause are referred to as the **grouping columns**.

- All columns in the SELECT list **must** appear in the GROUP BY clause **unless** the name is used in a aggregate function.
- There may be column names used in the GROUP BY clause that do not appear in the SELECT list.
- If a WHERE clause is used, it must come before GROUP BY.

Find the number of times each stock has been sold

```
SELECT Stock_No, COUNT(Sale_Value)
FROM Sales
GROUP BY Stock_No;
```



Stock_No	Sales
001	7
002	2
003	2

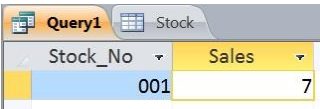
HAVING

The **HAVING** clause is used to restrict the summary groups that appear in the final result table.

This clause can only be used in conjunction with the **GROUP BY** clause.

Find the Stock which has been sold more than five (5) times.

```
SELECT Stock_No, COUNT(*)
FROM Sales
GROUP BY Stock_No HAVING COUNT(*) > 5;
```



Stock_No	Sales
001	7