

# Funciones de agregación

**Las funciones de agregación poseen de manera implícita la función de agrupación**

Las funciones de agregación aparecen en la lista SELECT, que puede incluir una cláusula GROUP BY. Si no hay cláusula GROUP BY en la sentencia SELECT, y la lista SELECT incluye al menos una función de agregación, no se pueden incluir columnas simples en la lista SELECT (salvo como argumentos de una función de agregación) Por lo tanto, el ejemplo siguiente es erróneo.

**--Forma incorrecta--**

```
SELECT emp_lname, MIN(emp_no)
FROM employee
```



**--Forma correcta--**

```
SELECT emp_lname, MIN(emp_no)
FROM employee
GROUP BY emp_lname
```



## SUM

**USE AdventureWorks2019**

Sumar las cantidades pedidas de los distintos pedidos y ordenarlos de mayor a menor

```
SELECT * FROM Sales.SalesOrderDetail
```


```
SELECT SalesOrderID, SUM(OrderQty) AS OrderQty
FROM Sales.SalesOrderDetail
GROUP BY SalesOrderID
ORDER BY OrderQty DESC
```



El uso de la opción DISTINCT elimina todos los valores duplicados en la columna antes de que se aplique la función SUM. Del mismo modo, todos los valores NULL se eliminan siempre antes de aplicar la función SUM.

```
SELECT DISTINCT SUM(campo)
FROM tabla
```

Podemos utilizar el SUM para realizar sumas de operaciones




```
SELECT SalesOrderID, SUM((UnitPrice - UnitPriceDiscount) * OrderQty) 'Ganancia'
FROM Sales.SalesOrderDetail
GROUP BY SalesOrderID
```

---

## AVG


```
USE AdventureWorks2019
```

Promedio del precio unitario de los distintos pedidos



```
SELECT SalesOrderID, AVG(UnitPrice) AS Promedio
FROM Sales.SalesOrderDetail
GROUP BY SalesOrderID
ORDER BY Promedio
```

Promedio del 10% del precio unitario de los distintos pedidos



```
SELECT SalesOrderID, AVG(UnitPrice * 0.10) AS Promedio
FROM Sales.SalesOrderDetail
GROUP BY SalesOrderID
ORDER BY Promedio
```

---

## COUNT

```
USE AdventureWorks2019
```

```
SELECT * FROM Sales.SalesOrderDetail
```

### Contar todos los registros de la tabla

```
SELECT COUNT(*) FROM Sales.SalesOrderDetail
```

```
SELECT COUNT(SalesOrderID) FROM Sales.SalesOrderDetail
```

### Contar todos los pedidos distintos


```
SELECT COUNT(DISTINCT SalesOrderID) FROM Sales.SalesOrderDetail
```

Cuando se utiliza la palabra clave **DISTINCT**, se eliminan todos los valores duplicados antes de aplicar **COUNT**. Esta forma de **COUNT** no cuenta los valores NULL de la columna.


### Contar el número de pedidos donde su pedido se ordenó una cantidad mayor a 10

```
SELECT SalesOrderId, OrderQty
FROM Sales.SalesOrderDetail
WHERE OrderQty > 10
AND
SalesOrderID = 51131
```

Para una orden específica



```
SELECT SalesOrderID, COUNT(*) AS Cantidad_Pedidos
FROM Sales.SalesOrderDetail
WHERE OrderQty > 10
GROUP BY SalesOrderID
ORDER BY Cantidad_Pedidos DESC
```



```
SELECT SalesOrderID, COUNT(SalesOrderID) AS Cantidad_Pedidos
FROM Sales.SalesOrderDetail
WHERE OrderQty > 10
GROUP BY SalesOrderID
ORDER BY Cantidad_Pedidos DESC
```


---

### MAX

USE AdventureWorks2019

```
SELECT * FROM Sales.SalesOrderDetail
```

### Obtener la mayor cantidad de productos pedidos por Orden de venta



```
SELECT SalesOrderID, MAX(OrderQty) AS Maxima_Cantidad
FROM Sales.SalesOrderDetail
GROUP BY SalesOrderID
ORDER BY Maxima_Cantidad DESC
```

La opción **DISTINCT** no puede utilizarse con las funciones de agregación **MIN** y **MAX**. Todos los valores NULL de la columna que son el argumento de la función agregada MIN o MAX se eliminan siempre antes de aplicar MIN o MAX.

### Obtener el nombre y apellido del empleado y el ID empleado que tenga el número ID menor

```
USE sample
SELECT emp_no, emp_fname, emp_lname
FROM employee
WHERE emp_no =
    (SELECT MIN(emp_no) FROM employee)
```

Obtener el número de empleado que fue ingresado en la fecha más reciente en la tabla works\_on

```
USE sample
SELECT emp_no,project_no,job,enter_date
FROM works_on
WHERE enter_date =
      (SELECT MAX(enter_date) FROM works_on)
```

Podemos utilizar un MIN y un MAX en el mismo SELECT. También podemos convertir formatos de fechas que vienen con la hora, a formato solo de fechas y en el orden que necesitamos

```
USE AdventureWorks2019
SELECT OrderDate
FROM Sales.SalesOrderHeader
```

| Results |                         | Messages |
|---------|-------------------------|----------|
|         | OrderDate               |          |
| 1       | 2011-05-31 00:00:00.000 |          |
| 2       | 2011-05-31 00:00:00.000 |          |
| 3       | 2011-05-31 00:00:00.000 |          |
| 4       | 2011-05-31 00:00:00.000 |          |
| 5       | 2011-05-31 00:00:00.000 |          |
| 6       | 2011-05-31 00:00:00.000 |          |

```
USE AdventureWorks2019
SELECT TerritoryID, CONVERT(VARCHAR(10), MIN(OrderDate), 105),
      CONVERT(VARCHAR(10), MAX(OrderDate), 23),CustomerID
FROM Sales.SalesOrderHeader
GROUP BY TerritoryID, CustomerID
```



Results

Messages

|   | TerritoryID | (No column name) | (No column name) | CustomerID |
|---|-------------|------------------|------------------|------------|
| 1 | 9           | 18-07-2012       | 2014-03-23       | 21256      |
| 2 | 9           | 07-03-2013       | 2014-02-20       | 24173      |
| 3 | 1           | 09-09-2013       | 2013-09-09       | 22878      |
| 4 | 9           | 31-12-2011       | 2013-12-21       | 11461      |
| 5 | 9           | 03-02-2012       | 2013-11-22       | 19951      |
| 6 | 4           | 11-08-2013       | 2013-08-11       | 13490      |
| 7 | 4           | 17-05-2014       | 2014-05-17       | 19109      |
| 8 | 8           | 31-05-2014       | 2014-05-31       | 26980      |

De la tabla Sales.CreditCard, se necesita mostrar la última fecha de modificación (ModifiedDate) por cada CardType

```
SELECT CardType, CONVERT(VARCHAR(10), MAX(ModifiedDate), 105)
FROM Sales.CreditCard
GROUP BY CardType
```

