



30  
años

Licenciada por  
SUNEDU  
para que puedas  
salir adelante

## SESIÓN 11:

Gestionar Sistemas de Información para satisfacer necesidades organizacionales de forma innovadora, respondiendo a estándares de calidad



# Resultados de Aprendizaje

- ✓ Diseña una Base de Datos Relacional para un caso en una empresa y usa el lenguaje SQL como herramienta de manipulación de datos.
- ✓ Aplica los Procedimientos Almacenados, en SQL Server.
- ✓ Crea proyectos en Visual Basic .Net para aplicar el uso de Procedimientos Almacenados.

## TEMARIO:

### TEMA 6: CONSULTAS SQL SERVER.

#### Contenido

-  Consultas SQL Server
-  Funciones de agregado
-  Uso de Group By





Una **función de agregado SQL** acepta un grupo de datos (normalmente una columna de datos) como argumento, y produce un único dato que resume el grupo. **Por ejemplo** la función AVG() acepta una columna de datos numéricos y devuelve la media aritmética (average) de los valores contenidos en la columna.

## SINTAXIS:

**Función** ([ALL|DISTINCT] expression)



- ✓ La palabra **ALL** indica que se tiene que tomar en cuenta todos los valores de la columna. Es el valor por defecto.
- ✓ La palabra **DISTINCT** hace que se consideren todas las repeticiones del mismo valor como uno sólo (considera valores distintos).
- ✓ Una función de agregado puede aparecer en la lista de selección en cualquier lugar en el que puede aparecer un nombre de columna.
- ✓ Puede, **por ejemplo**, formar parte de una expresión pero no se pueden anidar funciones de agregado.



# FUNCIÓN COUNT

Pregrado

Ingeniería de  
Sistemas

## COUNT ([ALL|DISTINCT] expresión | \* )

- ✓ **Devuelve el número total de filas seleccionadas por la consulta.**
- ✓ La función cuenta los valores distintos de NULL que hay en la columna.
- ✓ Expresión puede ser de cualquier tipo excepto text, image o ntext.
- ✓ La palabra **ALL** indica que se tienen que tomar todos los valores de la columna, mientras que **DISTINCT** hace que se consideren todas las repeticiones del mismo valor como uno solo. Estos parámetros son opcionales, por defecto se considera ALL.

**SELECT COUNT**(region) **FROM** oficinas;

**SELECT COUNT**(DISTINCT region) **FROM** oficinas;

**SELECT COUNT**(\*) **FROM** empleados **WHERE** oficina **IS NOT NULL**;

Si utilizamos \* en vez de expresión, devuelve el número de filas del origen que nos quedan después de ejecutar la cláusula **WHERE**.

**COUNT(\*)** no acepta parámetros y no se puede utilizar con **DISTINCT**. **COUNT(\*)** no requiere un parámetro expresión porque, por definición, no utiliza información sobre ninguna columna específica. En el recuento se incluyen las filas que contienen valores **NULL**.





# FUNCIÓN SUM

Pregrado

Ingeniería de  
Sistemas

## SUM ([ALL|DISTINCT] expression)

- ✓ Devuelve la suma de los valores devueltos por la expresión.
- ✓ Sólo puede utilizarse con columnas numéricas.
- ✓ El resultado será del mismo tipo aunque puede tener una precisión mayor.

```
SELECT SUM(importe)  
FROM pedidos;
```





# FUNCIÓN MAX

Pregrado

Ingeniería de  
Sistemas

## MAX ([ALL|DISTINCT] expression)

- ✓ Devuelve el valor máximo de la expresión sin considerar los nulos.
- ✓ **MAX** se puede usar con columnas numéricas, de caracteres y de datetime, pero no con columnas de bit. No se permiten funciones de agregado ni subconsultas.
- ✓ Utilizar DISTINCT no tiene ningún sentido con **MAX** (el valor máximo será el mismo si consideramos las repeticiones o no) y sólo se incluye para la compatibilidad con SQL-92.

```
SELECT SUM(ventas) AS VentasTotales, MAX(objetivo) AS  
MayorObjetivo  
FROM oficinas;
```





# FUNCIÓN MIN

Pregrado

Ingeniería de  
Sistemas

## MIN ([ALL|DISTINCT] expression)

- ✓ Devuelve el valor máximo de la expresión sin considerar los nulos.
- ✓ **MIN** se puede usar con columnas numéricas, de caracteres y de datetime, pero no con columnas de bit. No se permiten funciones de agregado ni subconsultas.
- ✓ Utilizar DISTINCT no tiene ningún sentido con **MIN** (el valor mínimo será el mismo si consideramos las repeticiones o no) y sólo se incluye para la compatibilidad con SQL-92.

```
SELECT SUM(ventas) AS VentasTotales, MIN(objetivo) AS MayorObjetivo  
FROM oficinas;
```



**AVG ([ALL|DISTINCT] expression)**

- ✓ Devuelve el **promedio** de los valores de un grupo, para calcular el promedio se omiten los valores nulos.
- ✓ El grupo de valores lo determina el resultado de la expresión que será un nombre de columna o una expresión basada en una columna o varias del origen de datos.
- ✓ La función se aplica también a campos numéricos, y en este caso el tipo de dato del resultado puede cambiar según las necesidades del sistema para representar el valor del resultado.

```
SELECT AVG(cuota) AS [Cuota media], AVG(ventas) AS [Ventas medias]  
FROM empleados;
```







# OTRAS FUNCIÓN DE AGREGADO

Pregrado

Ingeniería de  
Sistemas

- ✓ **VAR.-** Devuelve la varianza estadística de todos los valores de la expresión especificada.
- ✓ **VARP.-** Devuelve la varianza estadística de la población para todos los valores de la expresión especificada.
- ✓ **STDEV.-** Devuelve la desviación típica estadística de todos los valores de la expresión especificada.
- ✓ **STDEVP.-** Devuelve la desviación estadística estándar para la población de todos los valores de la expresión especificada
- ✓ **COUNT\_BIG.-** Funciona igual que la función COUNT. La única diferencia entre ambas funciones está en los valores devueltos, COUNT\_BIG siempre devuelve un valor de tipo **bigint** y por lo tanto admite más valores de entrada, no está limitado.





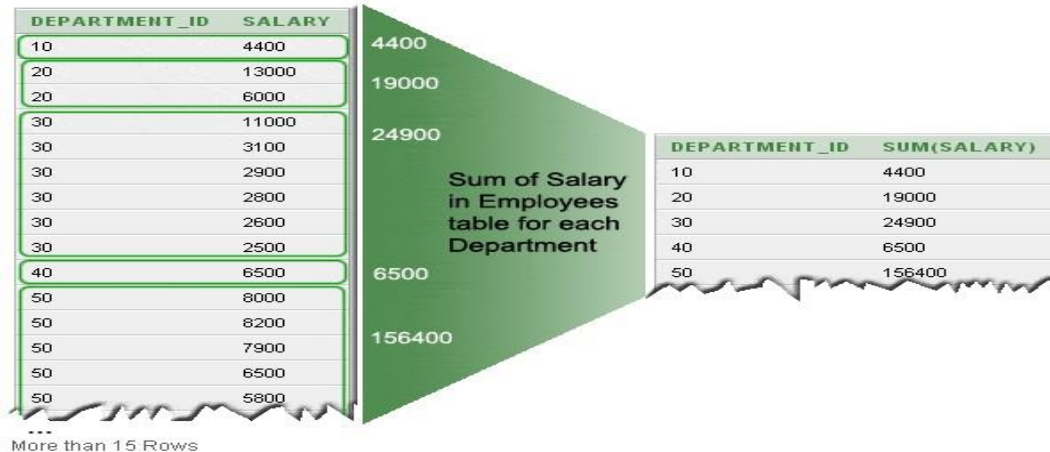
## GROUP BY

Pregrado

Ingeniería de  
Sistemas



La instrucción agrupa filas que tienen los mismos valores en filas de resumen, como "Buscar el total de salarios por cada departamento"



La instrucción se usa a menudo con funciones agregadas (, , , , ) para agrupar el conjunto de resultados por una o más columnas. GROUP BYCOUNT()MAX()MIN()SUM()AVG()



## ESTRUCTURA: CONSULTAS SQL SERVER

```
SELECT working_area, COUNT(*)  
FROM agents  
GROUP BY working_area;
```

agents

AGENT_NAME	WORKING_AREA
Alex	London
Subbarao	Bangalore
Benjamin	Hampshair
Ramasundar	Bangalore
Alford	New York
Ravi Kumar	Bangalore
Santakumar	Chennai
Lucida	San Jose
Anderson	Brisban
Mukesh	Mumbai
McDen	London
Ivan	Torento

the working\_area have  
been grouped and appearing  
once

WORKING_AREA	COUNT(*)
San Jose	1
Torento	1
London	2
Hampshair	1
New York	1
Brisban	1
Bangalore	3
Chennai	1
Mumbai	1

result



## Sintaxis

La sintaxis de la cláusula GROUP BY en SQL es:

```
SELECT expression1, expression2, ... expression_n,  
       aggregate_function (aggregate_expression)  
FROM tables  
[WHERE conditions]  
GROUP BY expression1, expression2, ... expression_n  
[ORDER BY expression [ ASC | DESC ]];
```



## Ejemplo

La siguiente instrucción SQL enumera el número de clientes en cada país

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

```
SELECT COUNT(CustomerID), Country  
FROM Customers  
GROUP BY Country;
```



## Ejemplo: uso de GROUP BY con la función SUM

Veamos cómo usar la cláusula GROUP BY con la función SUM en SQL.

En este ejemplo, tenemos una tabla llamada *empleados* con los siguientes datos:

employee_number	last_name	first_name	sueldo	dept_id
1001	Herrero	Juan	62000	500
1002	Anderson	Jane	57500	500
1003	Everest	Puntilla	71000	501
1004	Horvath	Sota	42000	501

Escriba la siguiente instrucción SQL:

```
SELECT dept_id, SUM(salary) AS total_salaries  
FROM employees  
GROUP BY dept_id;
```

Habrán 2 registros seleccionados. Estos son los resultados que deberías ver:

dept_id	total_salaries
500	119500
501	113000



## Ejemplo: uso de GROUP BY con la función COUNT

Veamos cómo usar la cláusula GROUP BY con la función COUNT en SQL.

En este ejemplo, tenemos una tabla llamada *productos* con los siguientes datos:

product_id	product_name	category_id
1	Pera	50
2	Plátano	50
3	Naranja	50
4	Manzana	50
5	Pan	75
6	Jamón en lonchas	25
7	Kleenex	NULO

Escriba la siguiente instrucción SQL:

```
SELECT category_id, COUNT(*) AS total_products
FROM products
WHERE category_id IS NOT NULL
GROUP BY category_id
ORDER BY category_id;
```

Habrán 3 registros seleccionados. Estos son los resultados que deberías ver:

category_id	total_products
25	1
50	4
75	1



## Ejemplo - Uso de GROUP BY con la función MIN

A continuación, veamos cómo usar la cláusula GROUP BY con la función MIN en SQL.

En este ejemplo, volveremos a utilizar la tabla *de empleados* que rellena los siguientes datos:

employee_number	last_name	first_name	sueldo	dept_id
1001	Herrero	Juan	62000	500
1002	Anderson	Jane	57500	500
1003	Everest	Puntilla	71000	501
1004	Horvath	Sota	42000	501

Escriba la siguiente instrucción SQL:

```
SELECT dept_id, MIN(salary) AS lowest_salary  
FROM employees  
GROUP BY dept_id;
```

Habrán 2 registros seleccionados. Estos son los resultados que deberías ver:

dept_id	lowest_salary
500	57500
501	42000





# HAVING

- Obtener el código y el número de productos que suministra cada proveedor

```
SELECT sn, COUNT(*)  
FROM sp  
GROUP BY sn;
```

- La misma consulta anterior pero solo para aquellos proveedores con más de dos productos. Imprimir en orden por el conteo.

```
SELECT sn, COUNT(*) AS conteo  
FROM sp  
GROUP BY sn  
HAVING COUNT(*) > 2  
ORDER BY 2;
```

Significa que ordene por la segunda columna del SELECT, aquí, conteo.

Nótese el renombrado para la función COUNT.



## Ejemplo - Uso de GROUP BY con la función MAX

Finalmente, veamos cómo usar la cláusula GROUP BY con la función MAX.

Usemos la tabla de *empleados* nuevamente, pero esta vez encontremos el salario más alto para cada *dept\_id*:

employee_number	last_name	first_name	sueldo	dept_id
1001	Herrero	Juan	62000	500
1002	Anderson	Jane	57500	500
1003	Everest	Puntilla	71000	501
1004	Horvath	Sota	42000	501

Escriba la siguiente instrucción SQL:

```
SELECT dept_id, MAX(salary) AS highest_salary
FROM employees
GROUP BY dept_id;
```

Habrán 2 registros seleccionados. Estos son los resultados que deberías ver:

dept_id	highest_salary
500	62000
501	71000




# CASE

Evalúa una lista de condiciones y devuelve una de las varias expresiones de resultado posibles.

## Sintaxis

syntaxsql

 Copiar

```
-- Syntax for SQL Server, Azure SQL Database and Azure Synapse Analytics
```

```
--Simple CASE expression:
```

```
CASE input_expression  
    WHEN when_expression THEN result_expression [ ...n ]  
    [ ELSE else_result_expression ]  
END
```

```
--Searched CASE expression:
```

```
CASE  
    WHEN Boolean_expression THEN result_expression [ ...n ]  
    [ ELSE else_result_expression ]  
END
```



## Ejemplos

### A. Usar una instrucción SELECT con una expresión CASE sencilla

En una instrucción `SELECT`, una expresión `CASE` sencilla solo permite una comprobación de igualdad; no se pueden hacer otras comparaciones. En este ejemplo se utiliza la expresión `CASE` para cambiar la presentación de categorías de línea de productos con el fin de hacerla más comprensible.

SQL

 Copiar


```
USE AdventureWorks2012;
GO
SELECT ProductNumber, Category =
    CASE ProductLine
        WHEN 'R' THEN 'Road'
        WHEN 'M' THEN 'Mountain'
        WHEN 'T' THEN 'Touring'
        WHEN 'S' THEN 'Other sale items'
        ELSE 'Not for sale'
    END,
    Name
FROM Production.Product
ORDER BY ProductNumber;
GO
```



## B. Usar una instrucción SELECT con una expresión CASE de búsqueda

En una instrucción `SELECT`, la expresión `CASE` de búsqueda permite sustituir valores en el conjunto de resultados basándose en los valores de comparación. En el ejemplo siguiente se presenta el precio de venta como un comentario basado en el intervalo de precios de un producto.

SQL

 Copiar

```
USE AdventureWorks2012;
GO
SELECT  ProductNumber, Name, "Price Range" =
        CASE
            WHEN ListPrice = 0 THEN 'Mfg item - not for resale'
            WHEN ListPrice < 50 THEN 'Under $50'
            WHEN ListPrice >= 50 and ListPrice < 250 THEN 'Under $250'
            WHEN ListPrice >= 250 and ListPrice < 1000 THEN 'Under $1000'
            ELSE 'Over $1000'
        END
FROM Production.Product
ORDER BY ProductNumber ;
GO
```



# Insertar datos desde una Consulta

```
INSERT INTO clientes2014 (dni, nombre, localidad, direccion)  
  
SELECT dni, nombre, localidad, direccion  
FROM clientes  
WHERE problemas=0;
```



## ¿QUÉ HEMOS APRENDIDO HOY?



Para que reflexionen y entiendan la importancia de los temas tratados y el mejoramiento de su propio proceso de aprendizaje.



Universidad **César Vallejo**

Licenciada por Sunedu  
para que puedas salir adelante