Lista de comandos SQL para Sql Server y MySQL

A continuación veremos los principales comandos SQL utilizados durante la administración de las Bases de Datos y conocerás para qué sirven:
CREATE DATABASE es utilizado para crear una base de datos vacía.
DROP DATABASE es utilizado para eliminar íntegramente una base de datos existente.
CREATE TABLE es utilizado para crear una tabla nueva en una base de Datos existente.
ALTER TABLE es utilizado para modificar una tabla que hemos creado previamente.
DROP TABLE es utilizado para eliminar por completo una tabla de nuestra base de Datos.

Comandos sql para manipular Datos

SELECT es utilizado para recuperar datos de una o más tablas, en otras palabras, nos permite hacer consultas de los registros de las tablas.
INSERT es utilizado añadir o insertar registros a una tabla creada previamente, su complemento es Insert into.
UPDATE es utilizado para modificar los datos de un conjunto de registros existentes en una tabla
DELETE es utilizado para eliminar o borrar todo o una parte de los datos de la tabla indicada por el argumento especificado después de la palabra clave **FROM**.
TRUNCATE es utilizado para borrar todos los registros de una tabla, pero no la tabla, es decir que quedaría una tabla vacía.

Clausulas básicas

Las cláusulas son **condiciones** y las utilizamos para especificar los datos que deseamos seleccionar o manipular.
FROM se utiliza para especificar la tabla de la cual se van a consultar los registros
GROUP BY se utiliza para separar en grupos específicos los registros consultados.
ORDER BY se utiliza para ordenar los registros seleccionados tomando en cuenta los parámetros que le indiquemos.
WHERE Se utilizar para determinar los registros seleccionados con la cláusula FROM
HAVING es parecida a **WHERE**, ya que determina qué registros se seleccionan. Cuando los registros se han agrupado utilizando GROUP BY, la cláusula HAVING determina cuáles de ellos se van a mostrar.

Comandos para la cláusula Select

AVG Se utiliza para determinar el promedio de los registros de un campo determinado, en una tabla especifica.
COUNT Se utiliza para devolver el número de registros que se muestran en una consulta realizada.
SUM Se utiliza para devolver la suma de todos los registros de un campo especifico, dentro de una tabla.

MAX Se utiliza para devolver el registro mayor o cantidad mayor de un campo especifico, dentro de una tabla.
MIN Se utiliza para devolver el registro menor o cantidad menor de un campo especifico, dentro de una tabla.
Todos los comandos sql vistos anteriormente son los mas usados, sin embargo es importante tener claro que deben ser complementados con ciertas palabras reservadas, por lo tanto es importante conocer la sintaxis para cada una de las acciones que deseamos realizar en nuestra Base de Datos. Te invito a que continúes aprendiendo en las diferentes categorías de este sitio web.

Ejemplos

```
CREATE DATABASE mydb;
```

```
USE mydb;

CREATE TABLE mitabla ( id INT PRIMARY KEY, nombre VARCHAR(20) );

INSERT INTO mitabla VALUES ( 1, 'Will' );

INSERT INTO mitabla VALUES ( 2, 'Marry' );

INSERT INTO mitabla VALUES ( 3, 'Dean' );

SELECT id, nombre FROM mitabla WHERE id = 1;

UPDATE mitabla SET nombre = 'Willy' WHERE id = 1;

SELECT id, nombre FROM mitabla;

DELETE FROM mitabla WHERE id = 1;

SELECT id, nombre FROM mitabla;

DROP DATABASE mydb;

SELECT count(1) from mitabla; da el número de registros en la tabla
```

Usar SELECT con encabezados de columna y cálculos

En los siguientes ejemplos se devuelven todas las filas de la tabla `Product`. En el primer ejemplo se devuelven las ventas totales y los descuentos de cada producto. En el segundo ejemplo se calculan los beneficios totales de cada producto.

```
USE AdventureWorks2012;
GO
SELECT p.Name AS ProductName,
NonDiscountSales = (OrderQty * UnitPrice),
Discounts = ((OrderQty * UnitPrice) * UnitPriceDiscount)
FROM Production.Product AS p
INNER JOIN Sales.SalesOrderDetail AS sod
ON p.ProductID = sod.ProductID
ORDER BY ProductName DESC;
GO
```

Ésta es la consulta que calcula el beneficio de cada producto de cada pedido de venta.

```
USE AdventureWorks2012;
GO
SELECT 'Total income is', ((OrderQty * UnitPrice) * (1.0 - UnitPriceDiscount)), ' for
',
p.Name AS ProductName
FROM Production.Product AS p
INNER JOIN Sales.SalesOrderDetail AS sod
ON p.ProductID = sod.ProductID
ORDER BY ProductName ASC;
GO
```

C. Usar DISTINCT con SELECT

En el siguiente ejemplo se utiliza `DISTINCT` para evitar la recuperación de títulos duplicados.

```
USE AdventureWorks2012;
GO
SELECT DISTINCT JobTitle
FROM HumanResources.Employee
ORDER BY JobTitle;
GO
```

D. Crear tablas con SELECT INTO

En el primer ejemplo se crea una tabla temporal denominada `#Bicycles` en `tempdb`.

```
USE tempdb;
GO
IF OBJECT_ID (N'#Bicycles',N'U') IS NOT NULL
DROP TABLE #Bicycles;
GO
```

```
SELECT *
INTO #Bicycles
FROM AdventureWorks2012.Production.Product
WHERE ProductNumber LIKE 'BK%';
GO
```

En el segundo ejemplo se crea la tabla permanente `NewProducts`.

```
USE AdventureWorks2012;
GO
IF OBJECT_ID('dbo.NewProducts', 'U') IS NOT NULL
    DROP TABLE dbo.NewProducts;
GO
ALTER DATABASE AdventureWorks2012 SET RECOVERY BULK_LOGGED;
GO

SELECT * INTO dbo.NewProducts
FROM Production.Product
WHERE ListPrice > $25
AND ListPrice < $100;
GO
ALTER DATABASE AdventureWorks2012 SET RECOVERY FULL;
GO
```

E. Usar subconsultas correlacionadas

Una subconsulta correlacionada es una consulta que depende de la consulta externa para obtener sus valores. Se ejecuta varias veces, una vez por cada fila que la consulta externa pueda seleccionar.
En el primer ejemplo se muestran consultas que son semánticamente equivalentes para demostrar la diferencia entre el uso de la palabra clave `EXISTS` y la palabra clave `IN`. Ambos son ejemplos de subconsultas válidas que recuperan una instancia de cada nombre de producto cuyo modelo es un jersey de manga larga con logotipo y cuyos números de `ProductModelID` coinciden en las tablas `Product` y `ProductModel`.

```
USE AdventureWorks2012;
GO
SELECT DISTINCT Name
FROM Production.Product AS p
WHERE EXISTS
    (SELECT *
     FROM Production.ProductModel AS pm
     WHERE p.ProductModelID = pm.ProductModelID
           AND pm.Name LIKE 'Long-Sleeve Logo Jersey%');
```

```
GO

-- OR

USE AdventureWorks2012;
GO
SELECT DISTINCT Name
FROM Production.Product
WHERE ProductModelID IN
     (SELECT ProductModelID
      FROM Production.ProductModel AS pm
      WHERE p.ProductModelID = pm.ProductModelID
         AND Name LIKE 'Long-Sleeve Logo Jersey%');
GO
```

En el ejemplo siguiente se usa `IN` y se recupera una instancia del nombre y apellido de cada empleado cuya bonificación en la tabla `SalesPerson` sea de `5000.00` y cuyos números de identificación coincidan en las tablas `Employee` y `SalesPerson`.

```
USE AdventureWorks2012;
GO
SELECT DISTINCT p.LastName, p.FirstName
FROM Person.Person AS p
JOIN HumanResources.Employee AS e
     ON e.BusinessEntityID = p.BusinessEntityID WHERE 5000.00 IN
     (SELECT Bonus
      FROM Sales.SalesPerson AS sp
      WHERE e.BusinessEntityID = sp.BusinessEntityID);
GO
```

La subconsulta anterior de esta instrucción no se puede evaluar independientemente de la consulta externa. Necesita el valor `Employee.EmployeeID`, aunque este valor cambia a medida que el Motor de base de datos de SQL Server examina diferentes filas de `Employee`.
Una subconsulta correlativa se puede usar también en la cláusula `HAVING` de una consulta externa. En este ejemplo se buscan los modelos cuyo precio máximo es superior al doble de la media del modelo.

```
USE AdventureWorks2012;
GO
SELECT p1.ProductModelID
FROM Production.Product AS p1
GROUP BY p1.ProductModelID
HAVING MAX(p1.ListPrice) >=
     (SELECT AVG(p2.ListPrice) * 2
      FROM Production.Product AS p2
      WHERE p1.ProductModelID = p2.ProductModelID);
```

```
GO
```

En este ejemplo se utilizan dos subconsultas correlativas para buscar los nombres de los empleados que han vendido un producto específico.

```
USE AdventureWorks2012;
GO
SELECT DISTINCT pp.LastName, pp.FirstName
FROM Person.Person pp JOIN HumanResources.Employee e
ON e.BusinessEntityID = pp.BusinessEntityID WHERE pp.BusinessEntityID IN
(SELECT SalesPersonID
FROM Sales.SalesOrderHeader
WHERE SalesOrderID IN
(SELECT SalesOrderID
FROM Sales.SalesOrderDetail
WHERE ProductID IN
(SELECT ProductID
FROM Production.Product p
WHERE ProductNumber = 'BK-M68B-42')));
GO
```

F. Usar GROUP BY

En este ejemplo se busca el total de cada pedido de venta de la base de datos.

```
USE AdventureWorks2012;
GO
SELECT SalesOrderID, SUM(LineTotal) AS SubTotal
FROM Sales.SalesOrderDetail
GROUP BY SalesOrderID
ORDER BY SalesOrderID;
GO
```

Debido a la cláusula `GROUP BY`, solo se devuelve una fila que contiene la suma de todas las ventas por cada pedido de venta.

G. Usar GROUP BY con varios grupos

En este ejemplo se busca el precio medio y la suma de las ventas anuales hasta la fecha, agrupados por Id. de producto e Id. de oferta especial.

```
USE AdventureWorks2012;
GO
SELECT ProductID, SpecialOfferID, AVG(UnitPrice) AS [Average Price],
```

```
    SUM(LineTotal) AS SubTotal
FROM Sales.SalesOrderDetail
GROUP BY ProductID, SpecialOfferID
ORDER BY ProductID;
GO
```

H. Usar GROUP BY y WHERE

En el siguiente ejemplo se colocan los resultados en grupos después de recuperar únicamente las filas con precios superiores a `$1000`.

```
USE AdventureWorks2012;
GO
SELECT ProductModelID, AVG(ListPrice) AS [Average List Price]
FROM Production.Product
WHERE ListPrice > $1000
GROUP BY ProductModelID
ORDER BY ProductModelID;
GO
```

I. Usar GROUP BY con una expresión

En este ejemplo se agrupa por una expresión. Puede agrupar por una expresión si ésta no incluye funciones de agregado.

```
USE AdventureWorks2012;
GO
SELECT AVG(OrderQty) AS [Average Quantity],
NonDiscountSales = (OrderQty * UnitPrice)
FROM Sales.SalesOrderDetail
GROUP BY (OrderQty * UnitPrice)
ORDER BY (OrderQty * UnitPrice) DESC;
GO
```

J. Usar GROUP BY con ORDER BY

En este ejemplo se busca el precio medio de cada tipo de producto y se ordenan los resultados por precio medio.

```
USE AdventureWorks2012;
GO
```

```
SELECT ProductID, AVG(UnitPrice) AS [Average Price]
FROM Sales.SalesOrderDetail
WHERE OrderQty > 10
GROUP BY ProductID
ORDER BY AVG(UnitPrice);
GO
```

K. Usar la cláusula HAVING

En el primer ejemplo se muestra una cláusula `HAVING` con una función de agregado. Agrupa las filas de la tabla `SalesOrderDetail` por Id. de producto y elimina aquellos productos cuyas cantidades de pedido medias son cinco o menos. En el segundo ejemplo se muestra una cláusula `HAVING` sin funciones de agregado.

```
USE AdventureWorks2012;
GO
SELECT ProductID
FROM Sales.SalesOrderDetail
GROUP BY ProductID
HAVING AVG(OrderQty) > 5
ORDER BY ProductID;
GO
```

En esta consulta se utiliza la cláusula `LIKE` en la cláusula `HAVING`.

```
USE AdventureWorks2012 ;
GO
SELECT SalesOrderID, CarrierTrackingNumber
FROM Sales.SalesOrderDetail
GROUP BY SalesOrderID, CarrierTrackingNumber
HAVING CarrierTrackingNumber LIKE '4BD%'
ORDER BY SalesOrderID ;
GO
```

L. Usar HAVING y GROUP BY

En el siguiente ejemplo se muestra el uso de las cláusulas `GROUP BY`, `HAVING`, `WHERE` y `ORDER BY` en una instrucción `SELECT`. Genera grupos y valores de resumen pero lo hace tras eliminar los productos cuyos precios superan los 25 $ y cuyas cantidades de pedido medias son inferiores a 5. También organiza los resultados por `ProductID`.

```
USE AdventureWorks2012;
GO
```

```
SELECT ProductID
FROM Sales.SalesOrderDetail
WHERE UnitPrice < 25.00
GROUP BY ProductID
HAVING AVG(OrderQty) > 5
ORDER BY ProductID;
GO
```

M. Usar HAVING con SUM y AVG

En el siguiente ejemplo se agrupa la tabla `SalesOrderDetail` por Id. de producto y solo se incluyen aquellos grupos de productos cuyos pedidos suman más de `$1000000.00` y cuyas cantidades de pedido medias son inferiores a `3`.

```
USE AdventureWorks2012;
GO
SELECT ProductID, AVG(OrderQty) AS AverageQuantity, SUM(LineTotal) AS Total
FROM Sales.SalesOrderDetail
GROUP BY ProductID
HAVING SUM(LineTotal) > $1000000.00
AND AVG(OrderQty) < 3;
GO
```

Para ver los productos cuyas ventas totales son superiores a `$2000000.00`, utilice esta consulta:

```
USE AdventureWorks2012;
GO
SELECT ProductID, Total = SUM(LineTotal)
FROM Sales.SalesOrderDetail
GROUP BY ProductID
HAVING SUM(LineTotal) > $2000000.00;
GO
```

Si desea asegurarse de que hay al menos mil quinientos elementos para los cálculos de cada producto, use `HAVING COUNT(*) > 1500` para eliminar los productos que devuelven totales inferiores a `1500` elementos vendidos. La consulta tiene este aspecto:

```
USE AdventureWorks2012;
GO
SELECT ProductID, SUM(LineTotal) AS Total
FROM Sales.SalesOrderDetail
GROUP BY ProductID
HAVING COUNT(*) > 1500;
GO
```

Hora Usar la sugerencia del optimizador INDEX

En el ejemplo siguiente se muestran dos formas de usar la sugerencia del optimizador `INDEX`. En el primer ejemplo se muestra cómo obligar al optimizador a que use un índice no clúster para recuperar filas de una tabla, mientras que en el segundo ejemplo se obliga a realizar un recorrido de tabla mediante un índice igual a 0.

```
USE AdventureWorks2012;
GO
SELECT pp.FirstName, pp.LastName, e.NationalIDNumber
FROM HumanResources.Employee AS e WITH (INDEX(AK_Employee_NationalIDNumber))
JOIN Person.Person AS pp on e.BusinessEntityID = pp.BusinessEntityID
WHERE LastName = 'Johnson';
GO

-- Force a table scan by using INDEX = 0.
USE AdventureWorks2012;
GO
SELECT pp.LastName, pp.FirstName, e.JobTitle
FROM HumanResources.Employee AS e WITH (INDEX = 0) JOIN Person.Person AS pp
ON e.BusinessEntityID = pp.BusinessEntityID
WHERE LastName = 'Johnson';
GO
```

Carácter comodín	Descripción	Ejemplo
%	Cualquier cadena de cero o más caracteres.	WHERE title LIKE '%computer%' busca todos los títulos de libros que contengan la palabra 'computer' en el título.
_ (carácter de subrayado)	Cualquier carácter individual.	WHERE au_fname LIKE '_ean' busca todos los nombres de cuatro letras que terminen en ean (Dean, Sean, etc.)
[ ]	Cualquier carácter individual del intervalo ([a-f]) o del conjunto ([abcdef]) que se ha especificado.	WHERE au_lname LIKE '[C-P]arsen' busca apellidos de autores que terminen en arsen y empiecen por cualquier carácter individual entre C y P, como Carsen, Larsen, Karsen, etc. En las búsquedas de intervalos, los caracteres incluidos en el intervalo pueden variar, dependiendo de las reglas de ordenación de la intercalación.
[^]	Cualquier carácter individual que no se encuentre en el intervalo ([^a-f]) o el conjunto ([^abcdef]) que se ha especificado.	WHERE au_lname LIKE 'de[^l]%' busca todos los apellidos de autores que empiecen por de y en los que la siguiente letra no sea l.

escape_character

Es un carácter que se coloca delante de un carácter comodín para indicar que el comodín se interpreta como un comodín, sino como un carácter normal. *escape_character* es una expresión de caracteres que no tiene valor predeterminado y se debe evaluar como un único carácter.