**COMANDO DE SQL:**

DDL: Data definition Lenguages

\* Crear tablas

\* Elminar tablas

\* Modificar tablas

DML: Data manipulation Lenguege

\* Seleccionar registros de una BBDD

\* Consulta de selección y acción

DCL: Data Control Lenguages

\* Proporcionar seguridad a la información de la BBDD

TCL: Transation Control Lenguages

\*Gestión de los cambios de los datos

**COMANDOS DEL DDL:**

\* CREATE

\* ALTER

\* DROP

\* TRUNCATE

**COMANDOS DML:**

\* SELECT

\* INSERT

\* UPDATE

\* DELETE

**COMANDOS DCL:**

\* GRANT

\* REVOKE

**COMANDOS TCL:**

\* COMMIT

\* ROLLBACK

\* SAVEPOINT

**CLÁUSULAS DE SQL:**

\* FROM: especifica la tabla de la que se quieren obtener los registros.

\* WHERE: especifica las condiciones o criterios de los registros seleccionados.

\* GROUP BY: Para agrupar los registros seleccionados en función de un campo.

\* HAVING: Especifica las condiciones o criterios que deben cumplir los grupos.

\* ORDEN BY: Ordena los registros seleccionados en función de un campo.

\*\*\*En el caso de que una consulta requiera todas las cláusulas entonces su orden sería:

\* comando + from + where + group by + having + orden by

**INSTRUCCIÓN SQL:**

Es la unión de:

COMANDO + CLÁUSULAS + OPERADORES + FUNCIONES

**###### IMPORTAR LAS TABLAS A MYSQL ########**

UNA VEZ ESTAMOS EN ACCESS LE DAMOS A CLIC A LA TABLA QUE QUEREMOS EXPORTAR Y LE DAMOS A EXCEL. (OTRA FORMA ES IR A DATOS EXTERNOS Y CLIC EN EXCEL)

DESPUÉS, ABRIMOS EL ARCHIVO DE EXCEL Y LO GUARDAMOS EN FORMATO .ods (openDocument) PARA PODER IMPORTARLO EN MYSQL

**### CREAR INSTRUCCIÓN SQL EN ACCESS ###**

PARA CREAR UNA CONSULTA SQL EN ACCESS LE DAMOS A CREAR/DISEÑO DE CONSULTA/VER Y SELECCIONAMOS SQL

DESPUÉS SOLAMENTE LE DAMOS EN EJECUTAR Y YA NOS MOSTRARÁ TODA LA INFORMACIÓN DE NUESTRA INSTRUCCIÓN

**### CREAR INSTRUCCIÓN SQL EN MYSQL (PHPMYADMIN) ###**

SELECCIONAR NUESTRA BBBDD Y CLIC EN SQL, ESCRIBIR LA INSTRUCCIÓN Y DARLE A CONTINUAR.

**### OPERADORES DE COMPARACIÓN ###**

<, >, =, >=, <=,

<> distinto que

BETWEEN, entre, utlizado para especificar rangos de valores.

LIKE, como, utilizado con caracteres comodín.

In, en, utlizado para especificar registros en un campo en concreto.

Any: ALGUNO

All: TODOS

**### Operadores lógicos ###**

AND, OR, NOT

**###### COMANTARIO IMPORTANTE AL TENER NOMBRES DE COLUMNAS CON ESPACIOS #####**

Al momento de hacer consultas los nombres de las columnas no deben tener espacios, si es el caso entonces escribir lo siguiente `nombre columna`

\*\*\*\*\* EN ACCESS SE ENCIERRAN CON [ ]

\*\*\*\*\* las comillas `` salen presionando Alt Gr + }

**###### COMENTARIO IMPORTANTE ACERCA DE LOS FORMATOS DE LOS CAMPOS ######**

LOS CAMPOS DEBEN TENER EL FORMATO CORRECTO PORQUE SINO MYSQL PUEDE HACER CONSULTAS ERRONEAS.

POR EJEMPLO: EL CAMPO PRECIO NO DEBERÍA IR EN FORMATO VARCHAR (TEXTO) PORQUE AL MOMENTO DE HACER CONSULTAS HABRÁN FALLOS AL REALIZAR UNA CONSULTA PORQUE NO LO DETECTA COMO VALOR NÚMERO

\*\* CORREGIR EL FORMATO (EN ESTE CASO DEL CURSO) EN EL EXCEL Y LUEGO VOLVER A IMPORTAR LA TABLA

**\*\*\*\* LOS CRITERIOS DE TIPO STRING DEBEN IR EN COMILLAS ####**

**####### ORDER BY ########**

La cláusula ORDER BY permite ordenar varias categorías, por ejemplo, puedo ordenar primero por el tipo de grupo que pertenece un producto y luego ese grupo lo puedo ordenar por su precio

ORDEN BY x,y asc/desc

**####### CONSULTAS DE AGRUPACIÓN O TOTALES #######**

FUNCIONES DE AGREGADO:

\* AVG: Calcula el promedio de un campo

\* COUNT: Cuenta los registros de un campo

\* SUM: Suma los valores de un campo

\* MAX: Devuelve el máximo de un campo

\* MIN: Devuelve el mínimo de un campo

LAS FUNCIONES DE AGREGADO SE UTILIZAN DESPUÉS DEL SELECT Y ANTES DEL CAMPO EN LA QUE VAMOS A APLICAR LA FUNCION POR EJ: SELECT SECCIÓN, SUM(PRECIO) ...

\*\*\*la cláusula WHERE no debe contener funciones de agrupación o agregación, mientras que la cláusula HAVING siempre contiene funciones de agregación.

**------------ EJERCICIO SUM ---------------**

\*\*\*AQUI VAMOS A AGRUPAR POR SECCIÓN Y SUMAREMOS TODOS LOS PRECIOS DE CADA SECCIÓN

SELECT SECCIÓN, SUM(PRECIO) FROM productos GROUP BY SECCIÓN

\*\*\*\*AL AGRUPAR POR SECCIÓN Y COLOCAMOS UN TERCER CAMPO (NOMBRE ARTICULO) EN MYSQL APARECERÁ EL PRIMER ARTICULO DE CADA SECCIÓN

\*\*\*\*EN ACCESS, SIMPLEMENTE NO APARECERÁN LAS CONSULTAS O DIFICILES DE INTERPRETAR

**####### EL ALIAS NOS PERMITE CAMBIARLE EL NOMBRE A UN CAMPO ########**

**------------ EJERCICIO SUM Y AS ------------**

La cláusula AS nos permite cambiarle el nombre a un campo

\*AQUÍ VAMOS A ORDENAR POR LA SUMA DE LOS PRECIOS DE LOS ARTICULOS, PERO COMO NO EXISTE ESA COLUMNA ENTONCES LE COLOCAMOS UN ALIAS PARA PODER UTLIZAR EL RESULTADO DE SUM(PRECIO)

SELECT SECCIÓN, SUM(PRECIO) AS SUMA\_PRODUCTOS FROM productos GROUP BY SECCIÓN ORDER BY SUMA\_PRODUCTOS ASC

**------------ EJERCICIO CON HAVING Y AVG ------------**

AGRUPAR POR SECCIÓN LOS PRODUCTOS DE CERAMICA Y DEPORTES Y CALCULAR EL PROMEDIO DE SUS PRECIOS

AQUÍ COMO ESTAMOS TRABAJANDO CON

SELECT SECCIÓN, AVG(PRECIO) AS PROMEDIO\_PRODUCTOS FROM productos GROUP BY SECCIÓN HAVING SECCIÓN = "CERÁMICA" OR SECCIÓN = "DEPORTES" ORDER BY PROMEDIO\_PRODUCTOS ASC

**------------ EJERCICIO CON COUNT Y AS ------------**

CONTAR LA CANTIDAD DE CLIENTES DE UNA CIUDAD

SELECT POBLACIÓN, COUNT(`CÓDIGO CLIENTE`) AS CANTIDAD\_CLIENTES FROM clientes GROUP BY POBLACIÓN

**------------ EJERCICIO MAX Y MIN ------------**

EL PRECIO MÁS CARO DEL ARTÍCULO DE LA SECCIÓN CONFECCIÓN

SELECT SECCIÓN, MAX(PRECIO) AS PRECIO\_MAX FROM productos GROUP BY SECCIÓN HAVING SECCIÓN = "CONFECCIÓN"

SELECT SECCIÓN, MIN(PRECIO) AS PRECIO\_MIN FROM productos GROUP BY SECCIÓN HAVING SECCIÓN = "CONFECCIÓN"

**######## CONSULTAS DE CÁLCULOS (SE APLICAN EN REGISTROS INDIVIDUALES Y NO EN GRUPOS) O CONSULTAS CON CAMPO CALCULADO**

CADA GESTOR DE BBDD TIENE SUS PROPIAS CONSULTAS DE CÁLCULOS (ALGUNA QUE OTRA COMÚN)

Now(): día y hora actuales

Datediff(): diferencia entre dos fechas, recibe tres argumentos, el primero es el tipo de comparación si por día “D”, mes “M”, etc y los otros dos son las fechas a comparar. EN MYSQL SOLO RECIBE DOS PARAMETROS.

Date\_format(): formatear los resultados (‘%D-%M-%Y’)

Concat(): Concatenar (se suele usar para concatenar textos)

Round(): redondear resultados, le podemos pasar un segundo parámetro con la cantidad de decimales a mostrar

Truncate():

**------- EJERCICIO CON ROUND Y SOLO CÁLCULOS ------**

SELECT `NOMBRE ARTÍCULO`, PRECIO, ROUND(PRECIO\*1.21,2) AS PRECIO\_IVA FROM productos

SELECT `NOMBRE ARTÍCULO`,`PRECIO`, PRECIO-3 AS PRECIO\_DESCUENTO FROM productos

**-------- EJERCICIO CON DATE Y DEMÁS-------**

SELECT `NOMBRE ARTÍCULO`, PRECIO, FECHA, Date\_Format(Now(), "‘%D-%M-%Y’") AS FECHA\_ACT, DateDiff(Now(),FECHA) AS DIAS\_DIFERENCIA FROM productos WHERE SECCIÓN = "DEPORTES"

**########### CONSULTAS MULTITABLA / CONSULTAS DE UNIÓN #############**

UNIÓN EXTERNA:

**LAS TABLAS DEBEN TENER EL MISMO TAMAÑO Y TIPOS DE DATOS COMPATIBLES.**

* UNION: (NO MUESTRA TODOS LOS REGISTROS REPETIDOS)
* UNION ALL: (MUESTRA TODOS LOS REGISTROS REPETIDOS)
* EXCEPT:
* INTERSECT:
* MINUS: A-B MUESTRA LOS REGISTROS QUE ESTÁN PRESENTES EN A Y NO EN B

\*\*\* LOS TRES ÚLTIMOS NO SON SOPORTADOS POR ALGUNAS BBDD EN MYSQL Y ACCESS NO LOS ACEPTAN

UNIÓN INTERNA:

**LOS JOIN PERMITEN HACER CONSULTAS ENTRE TABLAS RELACIONADAS**

**EL ON DEBE IR PARA ESPECIFICAR LAS COLUMNAS EN COMÚN**

* INNER JOIN: REFLEJA LA INFORMACIÓN EN COMÚN ENTRE DOS TABLAS
* LEFT JOIN: NOS DEVUELVE LA INFORMACIÓN COMÚN ENTRE DOS TABLAS Y LA INFORMACIÓN DE LA PRIMERA TABLA
* RIGHT JOIN: INFORMACIÓN COMÚN E INFORMACIÓN DE LA SEGUNDA TABLA

---------- EJERCICIOS UNION Y UNION ALL ------------

SELECT \* FROM productos WHERE SECCIÓN = "DEPORTES" UNION SELECT \* FROM productosnuevos WHERE SECCIÓN="DEPORTES DE RIESGO"

SELECT \* FROM productos WHERE SECCIÓN = "CONFECCIÓN" UNION ALL SELECT \* FROM productosnuevos WHERE SECCIÓN = "DEPORTES"

\*\*SI SOLAMENTE SE QUIEREN TRABAJAR CON X CANTIDAD DE CAMPOS ENTONCES LA OTRA CONSULTA DEBE TRABAJAR CON LA MISMA CANTIDAD, POR EJEMPLO:

SELECT PRECIO,NOMBRE FROM productos UNION SELECT PRECIO,NOMBRE FROM productosNuevos…

**----------- EJERCICIOS JOIN EN ACCESS ------------**

SELECT \* FROM CLIENTES INNER JOIN PEDIDOS ON CLIENTES.`CÓDIGO CLIENTE`= PEDIDOS.`CÓDIGO CLIENTE` WHERE POBLACIÓN = "MADRID" ORDER BY [NÚMERO DE PEDIDO]

\*\* REALIZADA EN ACCESS, COMO AMBAS TABLAS TIENEN EL MISMO NOMBRE DE COLUMNA ENTONCES SE ESPECÍFICA CON SU RESPECTIVA TABLA.

\*\* EN MYSQL HAY QUE RELACIONAR LAS TABLAS PORQUE AL IMPORTARLAS NO QUIERE DECIR QUE YA SE ESTÉN RELACIONANDO

**############ CREAR RELACIÓN ENTRE TABLAS EN MYSQL ############**

* HAY QUE ESPECIFICAR LA COLUMNA POR LA CUAL LAS TABLAS SE VAN A RELACIONAR Y ESPECIFICAR CUALES SON SUS LLAVES PRIMARIAS
* PARA LAS LLAVES PRIMARIAS:
  + LE DAMOS A LA TABLA, LUEGO A ESTRUCTURA, INDICES…
  + TIPO INDICE = PRIMARY Y ESPECIFICAR LA COLUMNA Y LISTO
* PARA LA RELACIÓN:
  + LA TABLA DE LA BBD DEBE ESTAR EN INNODB (SELECCIONAR LA TABLA/DARLE EL OPERACIONES/ IR A OPCIONES DE TABLA Y CAMBIAR EL MOTOR DE ALMACENAMIENTO POR INNODB)
  + TIPO INDICE = INDEX Y ESPECIFICAR LA COLUMNA QUE SE RELACIONA CON LA OTRA TABLA (LA RELACIÓN SE HACE EN LA TABLA NECESARIA)
  + SE ESCOGE ELIMINAR Y ACTUALIZAR EN CASCADA POR EL TIPO DE RELACIÓN.
  + SE ESCRIBE LOS CAMPOS RESPECTIVOS Y GUARDAR

**-------------- EJERCICIO CON NULL MYSQL Y ACCESS ----------**

* SE UTILIZA EL IS NULL SI UN CAMPO EN NULO

SELECT \* FROM CLIENTES LEFT JOIN pedidos ON clientes.`CÓDIGO CLIENTE`= pedidos.`CÓDIGO CLIENTE` WHERE `NÚMERO DE PEDIDO` IS NULL

########## SUBCONSULTAS #####################

SON CONSULTAS DENTRO DE OTRAS (UN SELECT DENTRO DE OTRO SELECT)

SUBCONSULTA ESCALONADA: Devuelve una única columna con un único registro. (se utiliza como consulta o criterio de evaluación):

SUBCONSULTA DE LISTA: Devuelve una lista de registros. AQUÍ SE UTLIZAN MUCHO LOS OPERADORES ANY, IN Y ALL

SUBCONSULTA CORRELACIONADA:

---------- EJERCICIOS SUBCONSULTA ESCALONADA -----------

* SELECT \* FROM productos WHERE PRECIO < (SELECT AVG(PRECIO) FROM productos)
* SELECT \* FROM PRODUCTOS WHERE PRECIO > (SELECT MAX(PRECIO) FROM PRODUCTOS WHERE SECCIÓN = 'CERÁMICA'

---------------- EJERCICIOS SUBCONSULTA DE LISTA ----------------

* SELECT \* FROM productos WHERE PRECIO > ALL (SELECT PRECIO FROM productos WHERE productos.SECCIÓN = "CERÁMICA")

---------- EJERICICIOS IN ------------

UNA CONSULTA QUE DEVUELVE EL NOMBRE Y PRECIO DEL ARTICULO QUE SE HAYAN PEDIDO 20 O MÁS UNIDADES

* SELECT [NOMBRE ARTÍCULO], [PRECIO] FROM PRODUCTOS WHERE [CÓDIGO ARTÍCULO] IN (SELECT [CÓDIGO ARTÍCULO] FROM [PRODUCTOS- PEDIDOS] WHERE [PRODUCTOS- PEDIDOS].UNIDADES >= 20)
* CON INNER JOIN EN VEZ DE IN:
* SELECT [NOMBRE ARTÍCULO], [PRECIO] FROM PRODUCTOS INNER JOIN [PRODUCTOS- PEDIDOS] ON PRODUCTOS.[CÓDIGO ARTÍCULO]=[PRODUCTOS- PEDIDOS].[CÓDIGO ARTÍCULO] WHERE UNIDADES >= 20

CONSULTA DONDE LOS CLIENTES NO HAN HECHO PEDIDOS O QUE LO HAN HECHO, PERO SIN PAGAR CON TARJETA

* SELECT EMPRESA, POBLACIÓN FROM CLIENTES WHERE [CÓDIGO CLIENTE]NOT IN (SELECT [CÓDIGO CLIENTE] FROM PEDIDOS WHERE [FORMA DE PAGO] = "TARJETA")
* SELECT EMPRESA,POBLACIÓN FROM CLIENTES LEFT JOIN PEDIDOS ON PEDIDOS.[CÓDIGO CLIENTE] = CLIENTES.[CÓDIGO CLIENTE] WHERE [FORMA DE PAGO] <> "TARJETA" OR [FORMA DE PAGO] IS NULL

CONSULTA DONDE LOS CLIENTES HAN HECHO PEDIDOS, PERO PAGANDO SIN TARJETA

* SELECT EMPRESA, POBLACIÓN FROM CLIENTES WHERE [CÓDIGO CLIENTE] NOT IN (SELECT [CÓDIGO CLIENTE] FROM PEDIDOS WHERE [FORMA DE PAGO] = "TARJETA") AND [CÓDIGO CLIENTE] IN (SELECT [CÓDIGO CLIENTE] FROM PEDIDOS)
* SELECT EMPRESA,POBLACIÓN FROM CLIENTES LEFT JOIN PEDIDOS ON PEDIDOS.[CÓDIGO CLIENTE] = CLIENTES.[CÓDIGO CLIENTE] WHERE [FORMA DE PAGO] <> "TARJETA"