**GUIA SQL**

**CONSULTAS**

**General:**

FECHAS : “01-02-1992”

TEXTO: “JONATHAN”

NUMEROS: 12

Se aconseja que los títulos de las columnas no tengan espacios o acentos. Podemos solucionarlo de la siguiente forma [CÓDIGO CLIENTE] o modificar el título.

Si en una consulta de multitablas hay dos títulos con el mismo nombre pero diferente tabla usamos el punto:

CLIENTES.CÓDIGOCLIENTE=PEDIDOS.CÓDIGOCLIENTE

|  |
| --- |
| **MOVERSE DETRO DE LA BASE DE DATOS** |
| **SHOW DATABASE**  **SHOW TABLES**  **USE DB\_EMRESA01** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Clausulas:** | | | | | |
| SELECT | FROM | WHERE | GROUP BY | HAVING | ORDER BY |

|  |  |
| --- | --- |
| **Operadores de comparación** | **Operadores lógicos** |
| <  >  =  >=  <=  <>  BETWEEN  LIKE  IN | AND  OR  NOT |

**Ejemplos de consultas:**

SELECT \* FROM CLIENTES

SELECT DIRECCIÓN, NOMBRE, EDAD FROM CLIENTES

## Nos mostrara en ese mismo orden

SELECT \* FROM CLIENTES WHERE PRECIO>1000

SELECT \* FROM CLIENTES WHERE CIUDAD =”B%”

## todos los registros que comiencen con B

SELECT \* FROM CLIENTES WHERE CIUDAD =”BUENOS AIRES” OR CIUDAD=”JUJUY”

## Queremos mostrar resultados de dos ciudades por eso usamos OR y no AND

SELECT \* FROM CLIENTES WHERE CIUDAD =”BUENOS AIRES” AND SEXO=”MASCULINO”

## AND primero filtra por buenos aires **luego** filtra por sexo.

**Ejemplo consulta usando between:**

SELECT\* FROM PRODUCTOS WHERE PRECIO BETWEEN 500 AND 1000

*SELECT\* FROM PRODUCTOS WHERE PRECIO<500 AND PRECIO>1000*

*SELECT \* FROM PRODUCTOS WHERE [PAÍS DE ORIGEN] IS NULL*

*## Nos muestra los registros que están el blanco.*

**Usar ORDER BY:**

SELECT \* FROM PRODUCTOS ORDER BY PRECIO

SELECT \* FROM PRODUCTOS ORDER BY PRECIO DESC

## DESC hace que se ordene de forma descendente Z-A o 9-1

SELECT \* FROM PRODUCTOS ORDER BY SECCIÓN, PAÍSDEORIGEN, PRECIO DESC

## Ordena en la misma orden comenzando por la columna de sección.

SELECT \* FROM PRODUCTOS WHERE SECCIÓN="FERRETERÍA" OR SECCIÓN ="JUGUETERÍA" ORDER BY PRECIO

## Ordenar luego de usar un WHERE.

**GROUP BY y HAVING**

|  |  |
| --- | --- |
| **AVG** | Promedio |
| **COUNT** | Cantidad (no debe tener espacios en blanco) |
| **SUM** | Suma de los valores |
| **MAX** | El máximo |
| **MIN** | El mínimo |

SELECT COUNT(ID) AS CANTIDAD FROM CLIENTES

## AS Le pone un alias a la columna que va ser creada

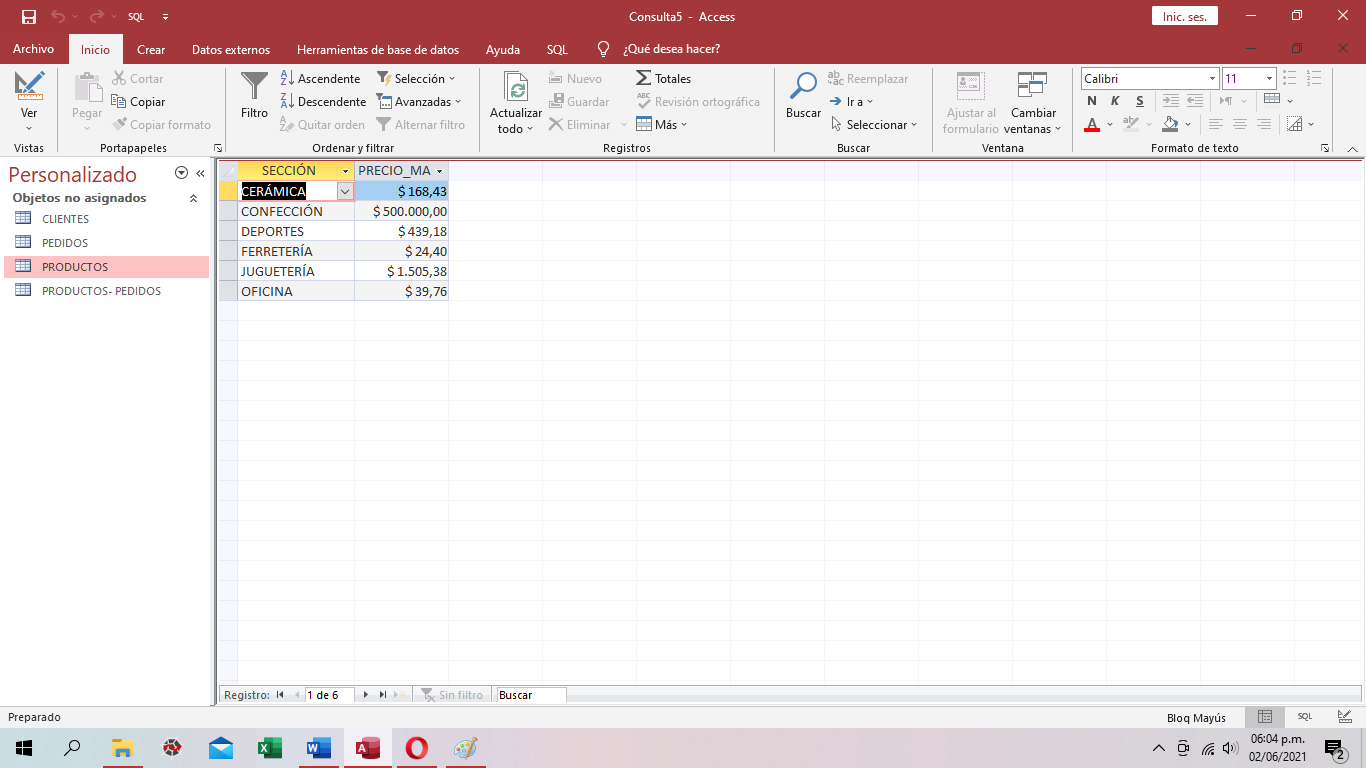
## En este caso cuenta la cantidad que registros de una columna, se aplica a las demás funciones.

SELECT POBLACIÓN FROM CLIENTES GROUP BY POBLACIÓN

## Agrupa, nos muestra solo un registro de cada tipo de dato de una columna

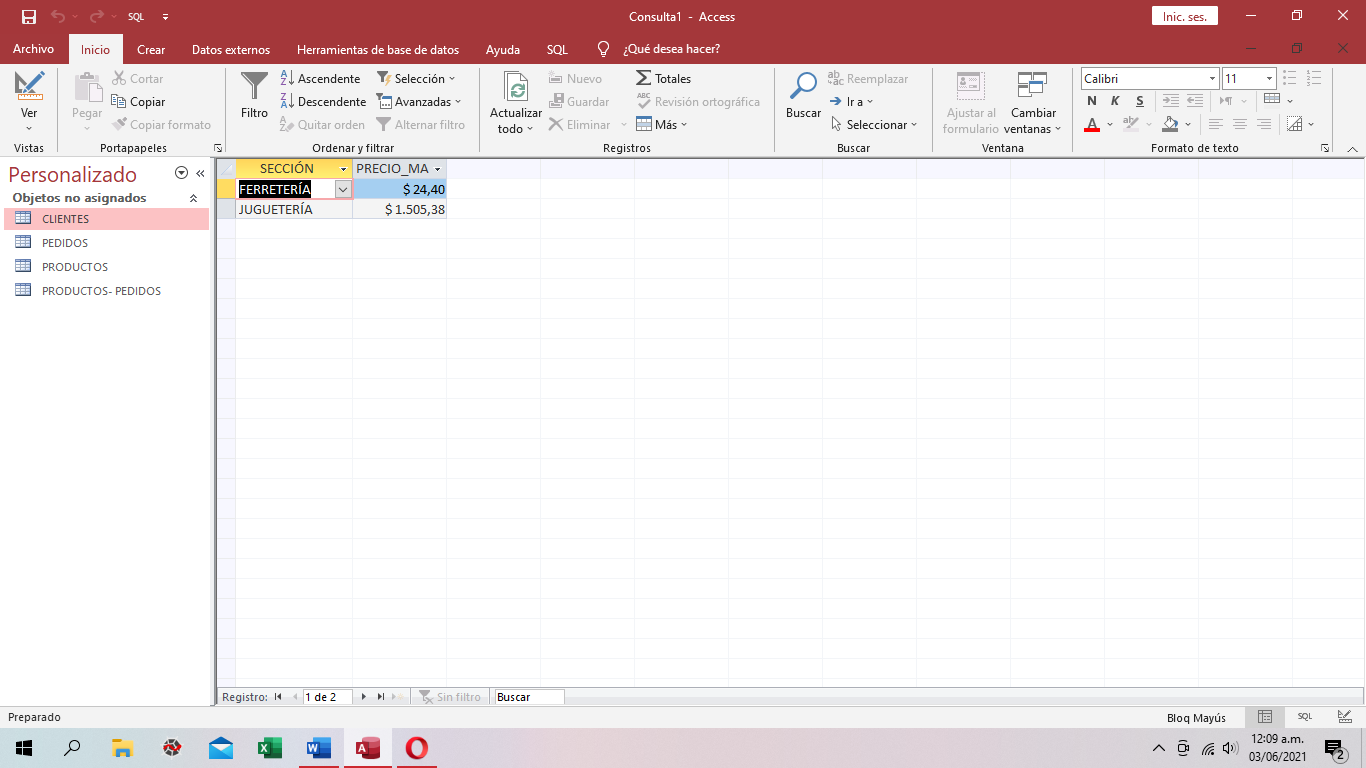
SELECT SECCIÓN, MAX (PRECIO) AS PRECIO\_MAXIMO FROM PRODUCTOS GROUP BY SECCIÓN

## Combinamos una funcion con GROUP BY



**#Usamos HAVING para reemplazar al WHERE para seguir la regla del orden.**

SELECT SECCIÓN, MAX (PRECIO) AS PRECIO\_MAXIMO FROM PRODUCTOS GROUP BY SECCIÓN HAVING SECCIÓN="FERRETERÍA" OR SECCIÓN ="JUGUETERÍA"



**## NOTA: También podemos usar ORDER BY pero teniendo en cuenta que si lo hacemos con una columna creada en la consulta por el AS debemos de hacerlo con el alias definido anteriormente.**

**CONSULTAS CON CALCULOS MATEMATICOS**

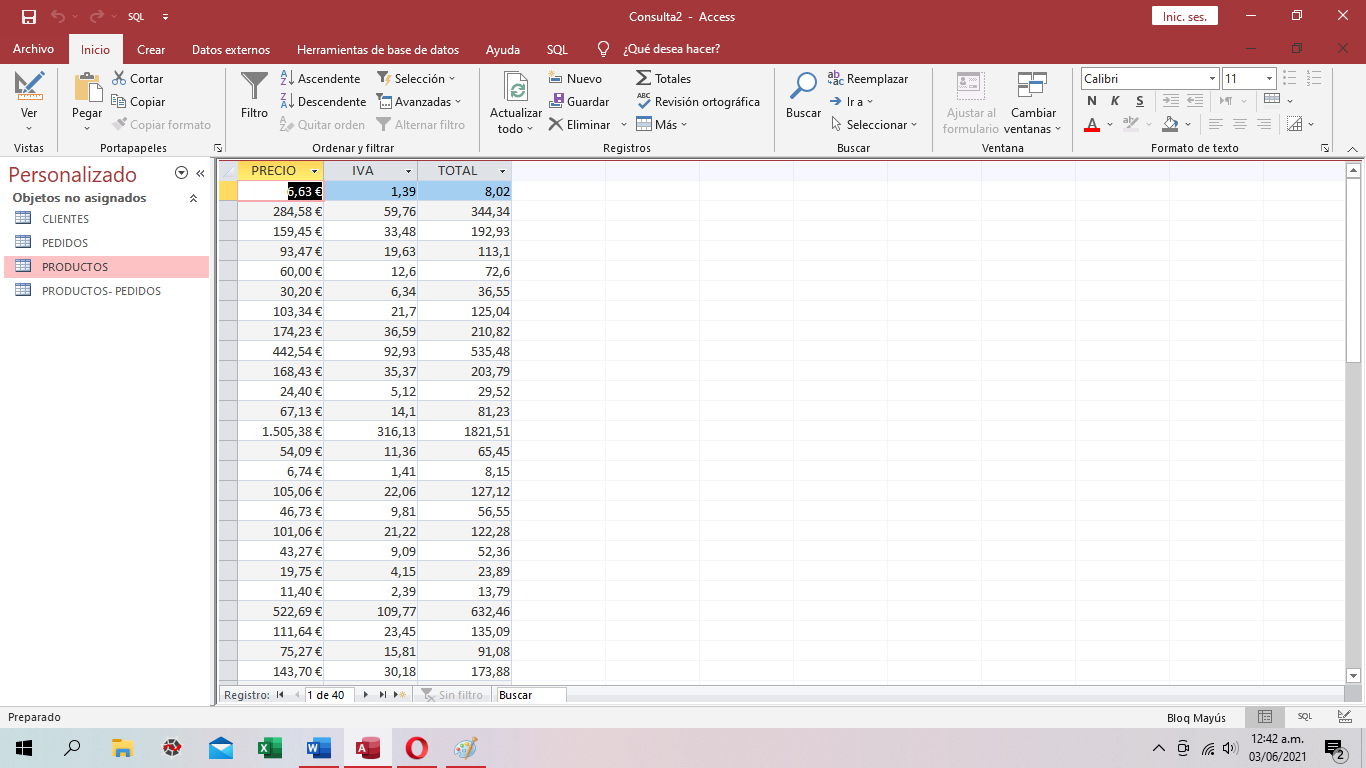
|  |  |
| --- | --- |
| **ROUND** | **ROUND(***columna, numero de decimales***) ## redondear** |
| **TRUNCATE** |  |

SELECT PRECIO, PRECIO\*21/100 AS IVA FROM PRODUCTOS

SELECT PRECIO, ROUND(PRECIO\*21/100, 2) AS IVA FROM PRODUCTOS

## Redondeamos con ROUND a dos decimales

SELECT PRECIO, ROUND(PRECIO\*21/100, 2) AS IVA, ROUND(PRECIO+PRECIO\*21/100,2) AS TOTAL FROM PRODUCTOS



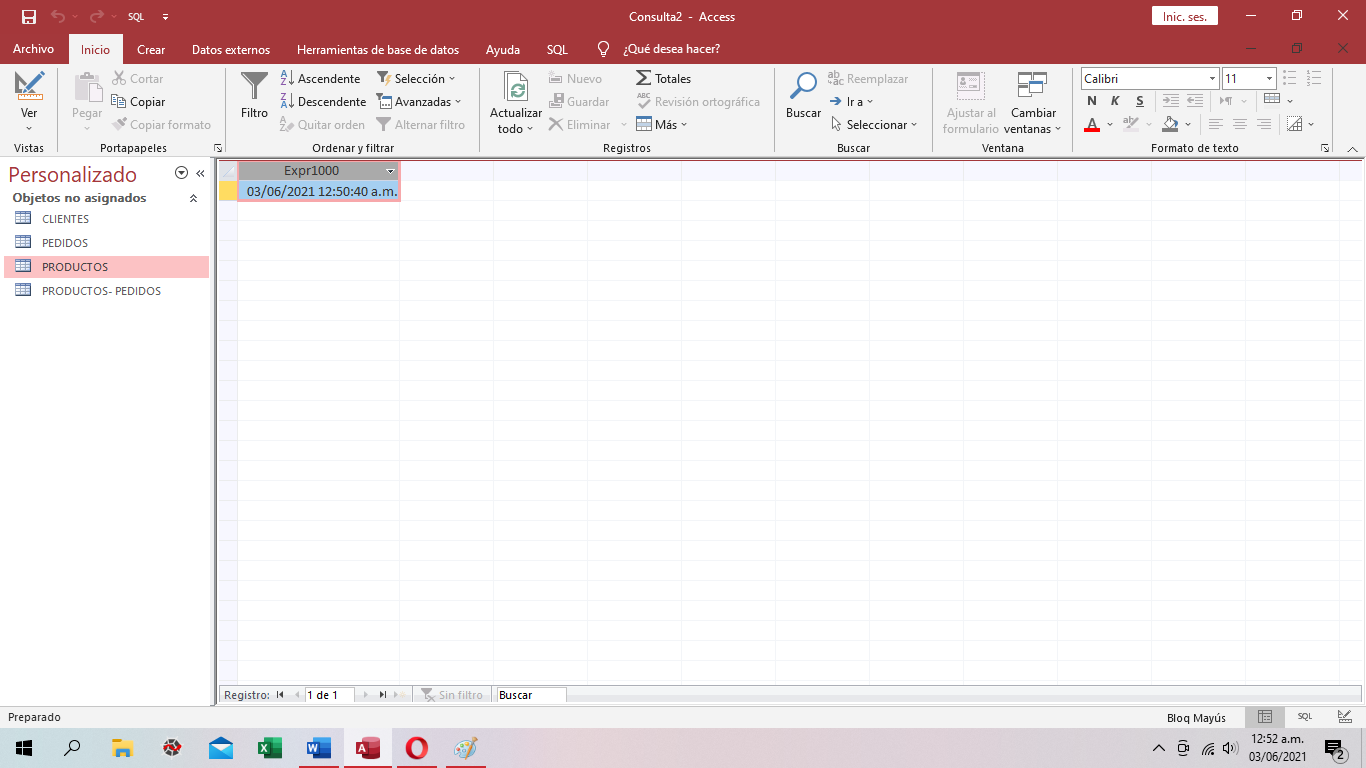
**CONSULTAS CON FECHAS**

|  |  |
| --- | --- |
| **NOW()** | Devuelve la hora y fecha actual |
| **DATEDIFF()** |  |
| **DATE\_FORMAT()** |  |
| **CONCAT** |  |

SELECT NOW()

SELECT FORMAT(NOW(), "DD/MM/YYYY HH:MM:SS")

## Podemos modificar como deseemos el formato agregando o quitando valores.



PARA ACCES ESPECIFICAMENTE

SELECT DATEDIFF("YYYY",NOW(), FECHA) FROM PRODUCTOS

## Nos devuelve la diferencia entre dos fechas “YYYY”, “M”,”D”,”H”,”M”

## Influye el orden de las fechas en caso de un resultado negativo.

SELECT DATEDIFF("YYYY",NOW(), "02/05/1993")

SELECT DATEDIFF("D","01/01/2021","02/05/2021")

**UNIÓN DE CONSULTAS**

|  |  |
| --- | --- |
| **Consultas externas** | **Consultas internas** |
| UNION | INNER JOIN |
| UNION ALL | LEFT JOIN |
| EXCEPT | RIGHT JOIN |
| INTERSECT |  |
| MINUS |  |

## Deben tener el mismo número de campos(columnas) y los datos del mismo deben se conptatibles(numerico, fecha, texto)

## Los nombres pueden variar de las columnas



**SELECT \* FROM PRODNUEVO** WHERE SECCIÓN="ALTA COSTURA" **UNION SELECT \* FROM PRODUCTOS** WHERE SECCIÓN="JUGUETERÍA"

## Unimos dos consultas de dos tablas diferentes un UNION

## Si se puede unir dos totalmente diferentes. La parte en negrita es importante que sean similares mas no la del where

## Pueden haber mas de un UNION en una misma consulta.

## No permite datos OLE (imágenes)

SELECT \* FROM PRODNUEVO UNION ALL SELECT \* FROM PRODUCTOS

## De esta forma si en caso que haya registros repetidos las muestra. Con UNION NO LAS MUETRA

**CONSULTA CON JOINS**



**INNER JOIN**

## Dos tablas se combinan (uno a la izquierda otro a la derecha). INNER JOIN muestra los datos que ambas tablas comparten los que no se comparten no los muestra.

**SELECT \* FROM CLIENTES INNER JOIN PEDIDOS ON** CLIENTES.CÓDIGOCLIENTE=PEDIDOS.[CÓDIGO CLIENTE]

## Luego podemos usar WHERE ORDER BY GROUP BY…G

SELECT \* FROM CLIENTES LEFT JOIN PEDIDOS ON CLIENTES.CÓDIGOCLIENTE=PEDIDOS.[CÓDIGO CLIENTE]

SELECT \* FROM CLIENTES RIGHT JOIN PEDIDOS ON CLIENTES.CÓDIGOCLIENTE=PEDIDOS.[CÓDIGO CLIENTE]

## A diferencia de INNER JOIN si muestra los registros no tienen su correspondiente en CLIENTES Y viceversa.

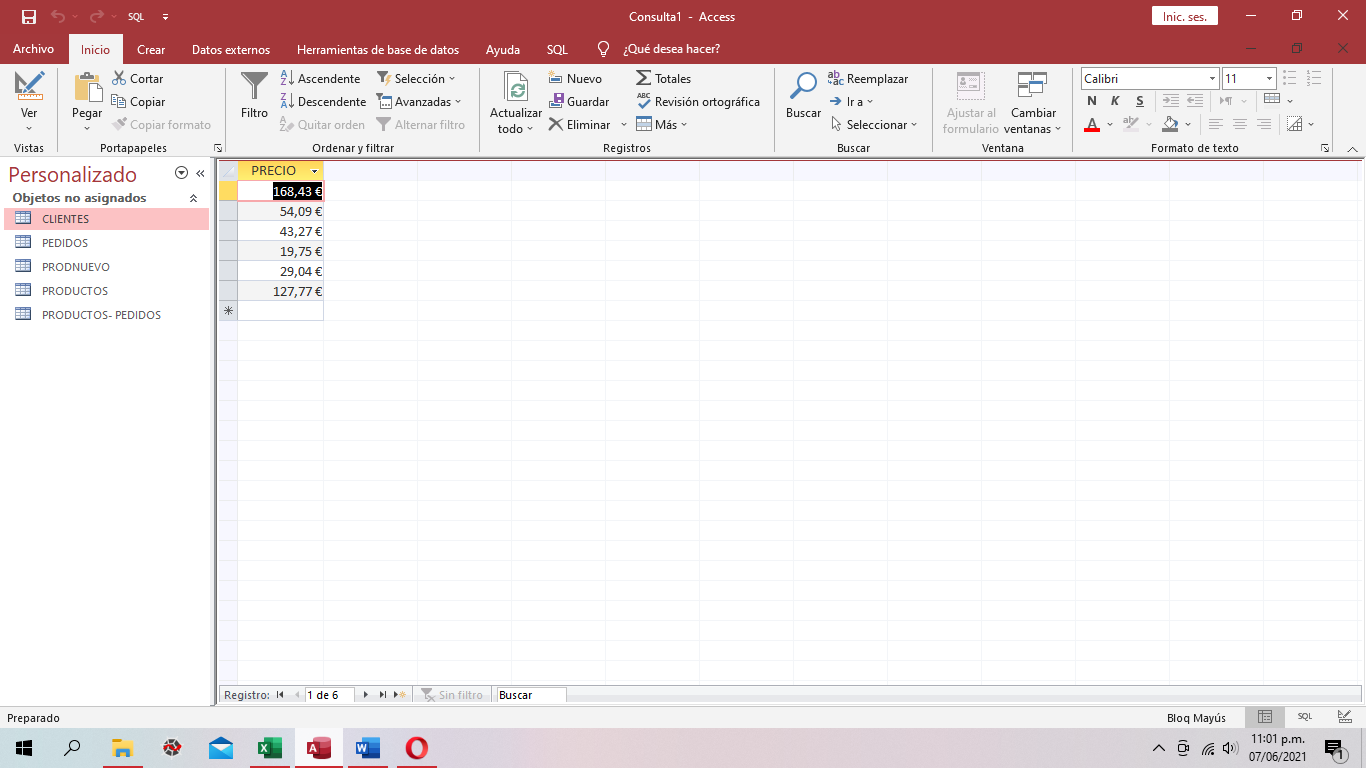
**SUB CONSULTAS**

SELECT\* FROM PRODUCTOS WHERE PRECIO<(SELECT AVG(PRECIO) FROM PRODUCTOS)

## Tenemos una consulta hijo de un solo resultado que usamos como dato de comparación en el WHERE.

|  |
| --- |
| IN / NOT IN |
| ANY |
| ALL |

SELECT PRECIO FROM PRODUCTOS WHERE SECCIÓN="CERÁMICA"



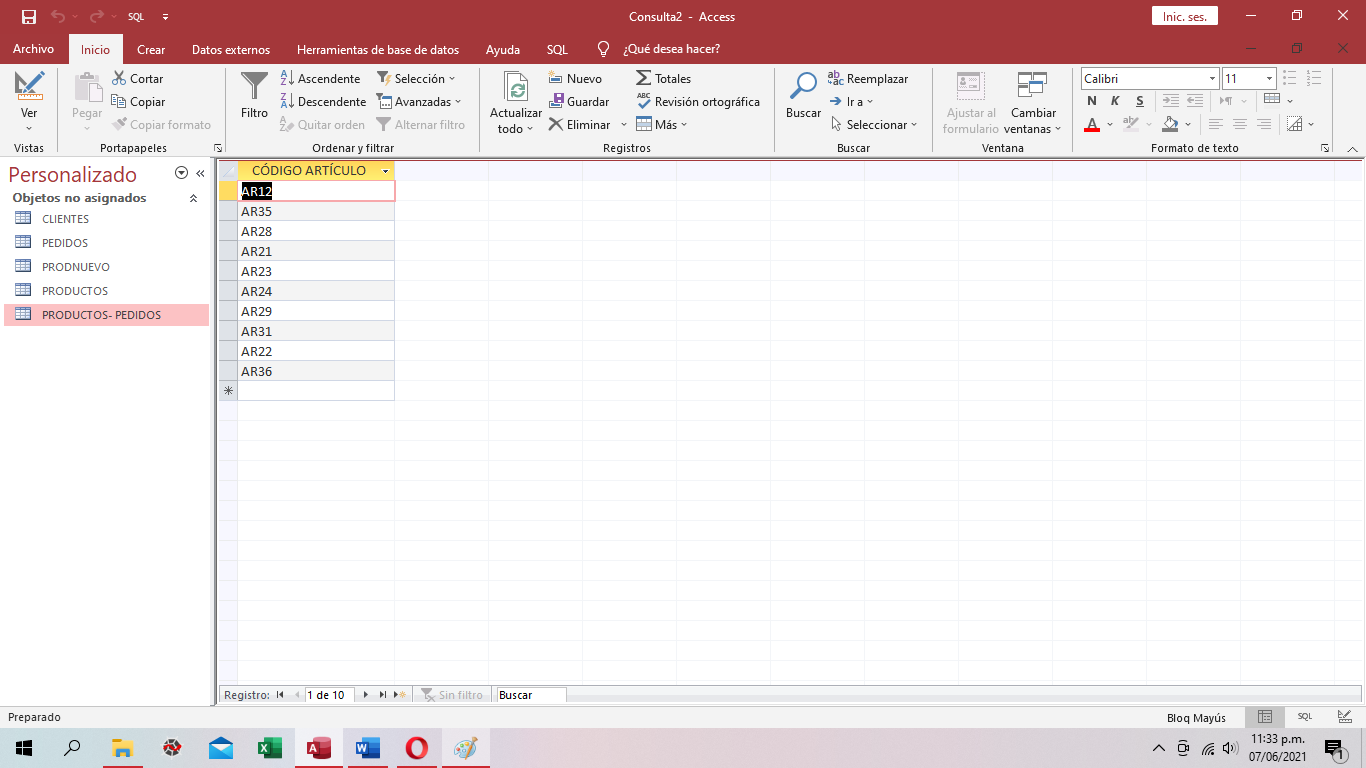
SELECT \* FROM PRODUCTOS WHERE PRECIO> ALL(SELECT PRECIO FROM PRODUCTOS WHERE SECCIÓN="CERÁMICA")

## ALL toma como punto de comparación al mayor de la lista

SELECT \* FROM PRODUCTOS WHERE PRECIO> ANY(SELECT PRECIO FROM PRODUCTOS WHERE SECCIÓN="CERÁMICA")

SELECT [CÓDIGO ARTÍCULO] FROM [PRODUCTOS- PEDIDOS] WHERE UNIDADES>20

##Obtenemos una lista.



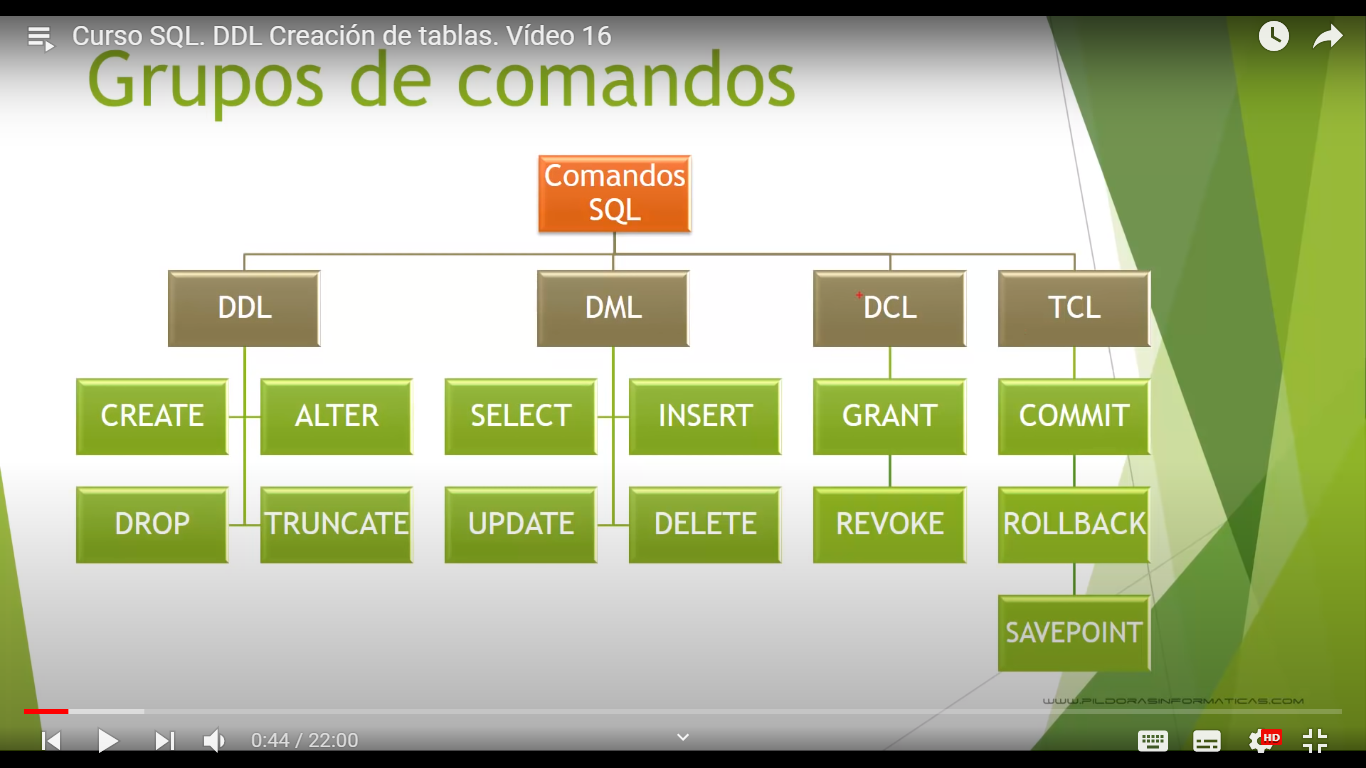
## Relaciona la columna con la lista. Toma solo las que existen en la lista.

SELECT \* FROM PRODUCTOS WHERE CÓDIGOARTÍCULO IN (SELECT [CÓDIGO ARTÍCULO] FROM [PRODUCTOS- PEDIDOS] WHERE UNIDADES>20)

SELECT \* FROM PRODUCTOS INNER JOIN [PRODUCTOS- PEDIDOS] ON PRODUCTOS.CÓDIGOARTÍCULO=[PRODUCTOS- PEDIDOS].[CÓDIGO ARTÍCULO] WHERE UNIDADES>20

## Otra forma de llegar al resultado.

**CONSULTA DE ACCIÓN**



|  |
| --- |
| CREATE |
| UPTADE |
| DELETE |
| INSERT INTO |
| SELECT INTO |

|  |  |  |
| --- | --- | --- |
| UPDATE PRODUCTOS SET PRECIO=PRECIO +10 WHERE SECCIÓN="DEPORTES"  UPDATE PRODUCTOS SET SECCIÓN="DEPORTE" WHERE SECCIÓN="DEPORTIVO"  UPDATE PRODUCTOS SET IMPORTADO=TRUE WHERE CÓDIGOARTÍCULO="AR01" | | |
| CREATE TABLE IF NOT EXISTS NUEVA\_TABLA SELECT \*FROM PRODUCTOS  CREATE TABLE NUEVA\_TABL SELECT \* FROM PRODUCTOS WHERE SECCIÓN=”ARGENTINA”  ##Vamos a crear una tabla nueva con los datos del filtro del SELECT.  **CREATE TABLE IF NOT EXISTS** NUEVA  (  ID INT AUTO\_INCREMENT, PRIMARY KEY(ID),  NOMBRE VARCHAR(20),  APELLIDO VARCHAR(20),  EDAD TINYINT,  NACIMIENTO DATE,  TELÉFONO int(10),  CARNET BOOL,  MOMENTO\_INSERTADO DATETIME  ) | | |
| ALTER TABLE TABLA\_1 RENAME NUEVO\_NOMBRE  RENAME TABLE TB\_1 TO TABLA1, TB2 TO TABLA2  ALTER TABLE NUEVA ADD COLUMN (NACIONALIDAD VARCHAR(20))  ALTER TABLE NUEVA ALTER COLUMN NACIONALIDAD INT(20)  ALTER TABLE NUEVA DROP COLUMN NACIONALIDAD  ALTER TABLE NUEVA ADD PRIMARY KEY(ID)  ALTER TABLE NUEVA ADD PRIMARY KEY(NOMBRE, APELLIDO)  ALTER TABLE NUEVA DROP PRIMARY KEY  ALTER TABLE NUEVA CHANGE ID ID INT AUTO\_INCREMENT  ## Poner luego de crear con el script.  **CREATE INDEX** NOMB\_INDICE **ON** TABLA1 (APELLIDO)  DROP INDEX NOMB\_INDICE ON TABLA1  **CREATE UNIQUE INDEX** NOM\_INDICE *ON* TABLA1(NOMBRE)  **CREATE UNIQUE INDEX** NOM\_INDICE *ON* TABLA1(NOMBRE, APELLIDO) | PRIMARY KEY | Valor único |
| No NULL |
| INDEX | Permite duplicados |
| Si admite NULL |
| UNIQUE INDEX | Sin duplicados |
| Si admite NULL |
|  |  |
|  |
|  | |
| DELETE FROM PRODUCTOS WHERE SECCIÓN=”CERÁMICA”  DELETE CLIENTES FROM CLIENTES RIGHT JOIN PEDIDOS ON CIENTES.CÓDIGOCLIENTE LIKE PEDIDOS.CÓDIGOPEDIDOS  #Ejemplo de eliminación usando joins | | |
| DROP TABLE CLIENTES  DROP TABLE IF EXISTS CLIENTES  ## Borra la table si existe, no detiene el proceso del script en caso de no existir dicha tabla  DROP TALBE IF EXIST CLIENTES, PRODUCTOS, PEDIDOS | | |
| INSERT INTO CLIENTES SELECT \* FROM CLIENTES\_MADRID  INSERT INTO CLIENTES VALUES (‘JONY’, ‘ALANOCA’, 1125607123)  INSERT INTO CLIENTES (NOMBRE, EDAD, TELEFONO) VALUES (‘JONY’, ‘ALANOCA’, 1125607123)  ## Anexar los registros en una sola tabla  INSERT INTO CLIENTES(CÓDIGOCLIENTE, EMPRESA, POBLACIÓN, TELÉFONO) SELECT CÓDIGOCLIENTE, EMPRESA, POBLACIÓN, TELÉFONO FROM CLIENTES\_MADRID  ## Anexamos solo las columnas que queremos. Para esto debemos especificar en cada lado las columnas. | | |
|  | | |
|  | | |

|  |  |
| --- | --- |
| **DISTINCT** | Similar al GROUP BY, esta ocupa menos memoria. Toma solo un registro (no agrupa) |
| **DISTINCTROW** | Agrupa a los registros que son idénticos en todas las colunas va después del SELECT y toma un registro de los duplicados. |
| SELECT DISTINCT POBLACIÓN FROM CLIENTES  SELECT DISTINCT POBLACIÓN, EMPRESA FROM CLIENTES  ## DISCTINCT solo toma los índices que le indicamos para hacer la comparación con los demás registros y los que son similares | |

SELECT DISCTINCTROW POBLACIÓN FROM CLIENTES

## Aunque solo tomamos población DISCTINCT compara todas las columnas del registro y los compara con los demás en la tabla. En caso de haber uno toma solo un registro y lo muestra.

INSERT INTO CLIENTES SELECT \* FROM CLIENTES\_MADRID

## Anexar los registros en una sola tabla

INSERT INTO CLIENTES(CÓDIGOCLIENTE, EMPRESA, POBLACIÓN, TELÉFONO) SELECT CÓDIGOCLIENTE, EMPRESA, POBLACIÓN, TELÉFONO FROM CLIENTES\_MADRID

## Anexamos solo las columnas que queremos. Para esto debemos especificar en cada lado las columnas.

**TRIGGERS**

Se encarga de realizar una acción luego de relazar alguna de las siguientes:

|  |
| --- |
| INSERT |
| UPDATE |
| DELETE |

Previamente creamos una tabla donde vamos a guardar los registros. En este caso Reg\_clientes

|  |
| --- |
| FOR EACH ROW |
|  |

CREATE TRIGGER CLIENTES\_AI AFTER INSERT ON CLIENTES FOR EACH ROW

INSERT INTO REG\_CLIETES(DIRECCIÓN, EMPRESA, TELÉFONO) VALUES (NEW.DIRECCIÓN, NEW.EMPRESA, NEW.TELÉFONO)

## CLIENTES\_AI (AFTER INSERT) es el nombre del TRIGGER

##NEW. Se refiere al valor nuevo de DIRECCIÓN… de la tabla anterior (CLIENTES)

CREATE TRIGGER CLIENTES\_AD AFTER DELETE ON CLIENTES FOR EACH ROW

INSERT INTO REG\_CLIETES(DIRECCIÓN, EMPRESA, TELÉFONO) VALUES (OLD.DIRECCIÓN, OLD.EMPRESA, OLD.TELÉFONO)

CREATE TRIGGER TABLA\_PRINCIP\_BU BEFORE UPDATE ON REG FOR EACH ROW

 INSERT INTO TABLA\_RESPALDO (

 ANTERIOR\_NOMBRE, ANTERIOR\_APELLIDO, ANTERIOR\_EDAD, ANTERIOR\_TELÉFONO,

 NUEVO\_NOMBRE, NUEVO\_APELLIDO, NUEVO\_EDAD, NUEVO\_TELÉFONO, USUARIO, F\_MODIF

)

 VALUES (

OLD.NOMBRE, OLD.APELLIDO, OLD.EDAD, OLD.TELÉFONO,

 NEW.NOMBRE, NEW.APELLIDO, NEW.EDAD, NEW.TELÉFONO,

**CURRENT\_USER (), NOW ()**

)

## Respaldo a TABLA\_PRINCIPAL

##Con OLD guardamos los datos antes de ser midifcados y con NEW los datos luego de ser midiicados.

## CURRENT\_USER() nos devuelve el usuario

DROP TRIGGER IF EXIST RESPALDO\_PEDIDOS

**PROCEDIMIENTOS (FUNCIONES)**

Son funciones dentro del sql, da mas seguridad (el cliente no ve el código de sql sino solo el proceso), ahora procesos

CREATE PROCEDURE FUNC\_01()

SELECT \* FROM PRODUCTOS WHERE SECCIÓN LIKE ‘DEPORTES’

CALL FUNC\_01()

CREATE PROCEDURE FUNC2(N\_PRECIO INT)

SELECT \* FROM PRODUCTOS WHERE PRECIO > N\_PRECIO

CALL FUNC2(100)

## Al llamar a la función le pasamos el valor que va a reemplazar en el PROCEDURE y nos devuelve la instrucción dentro del mismo.

DELIMITER $$

CREATE PROCEDURE FONCI03()

BEGIN

SELECT \* FROM CLIENTES WHERE POBLACIÓN LIKE ‘MADRID’;

DROP TABLE CLIENTES\_RESPALDO;

END; $$

DELIMITER ;

## Estructura de una procedure para una función de varias líneas de código sql

## El punto y coma es importante tras cada línea sql

**Variables, condicionales**

DELIMITER $$

CREATE PROCEDURE FONCI03(MI\_EDAD INT)

BEGIN

DECLARE VAR1 INT DEFAULT 2021;

DECLARE VAR2 INT;

SET VAR2= VAR1-MI\_EDAD;

SELECT VAR2;

End; $$

DELIMITER ;

DELIMITER $$

CREATE TRIGGER VERIF\_PRECIO\_BU BEFORE UPDATE ON PRODUCTOS FOR EACH ROW

BEGIN

IF (NEW.PRECIO<0) THEN

SET NEW.PRECIO=OLD.PRECIO;

ELSEIF (NEW.PRECIO>1000) THEN

SET NEW.PRECIO=OLD.PRECIO;

END IF;

END; $$

DELIMITER ;

**VISTAS**

Mas privacidad de la base de datos

Optimización de la bdd

Se crea una tabla virtual en caso de realizar la misma consulta una y otra vez.

Se modifica tanto en la tabla VIEW como en la principal los valores actualizados.

Luego del AS la consulta puede ser tan compleja como queremos.

* CREATE VIEW ART\_DEPORTES AS SELECT \* FROM PRODUCTOS WHERE SECCIÓN LIKE 'DEPORTES'
* SELECT \* FROM ART\_DEPORTES
* DROP VIEW ART\_DEPORTES
* ALTER VIEW ART\_DEPORTES AS …….(NUEVA CONSULTA )