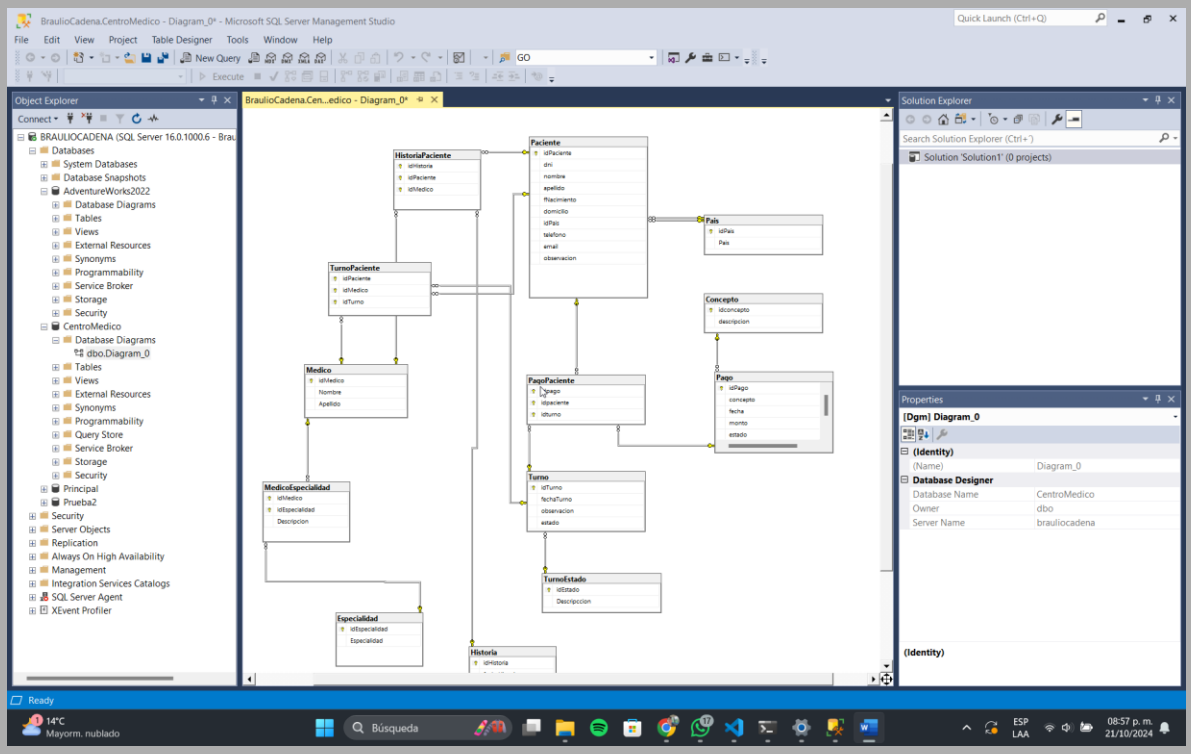
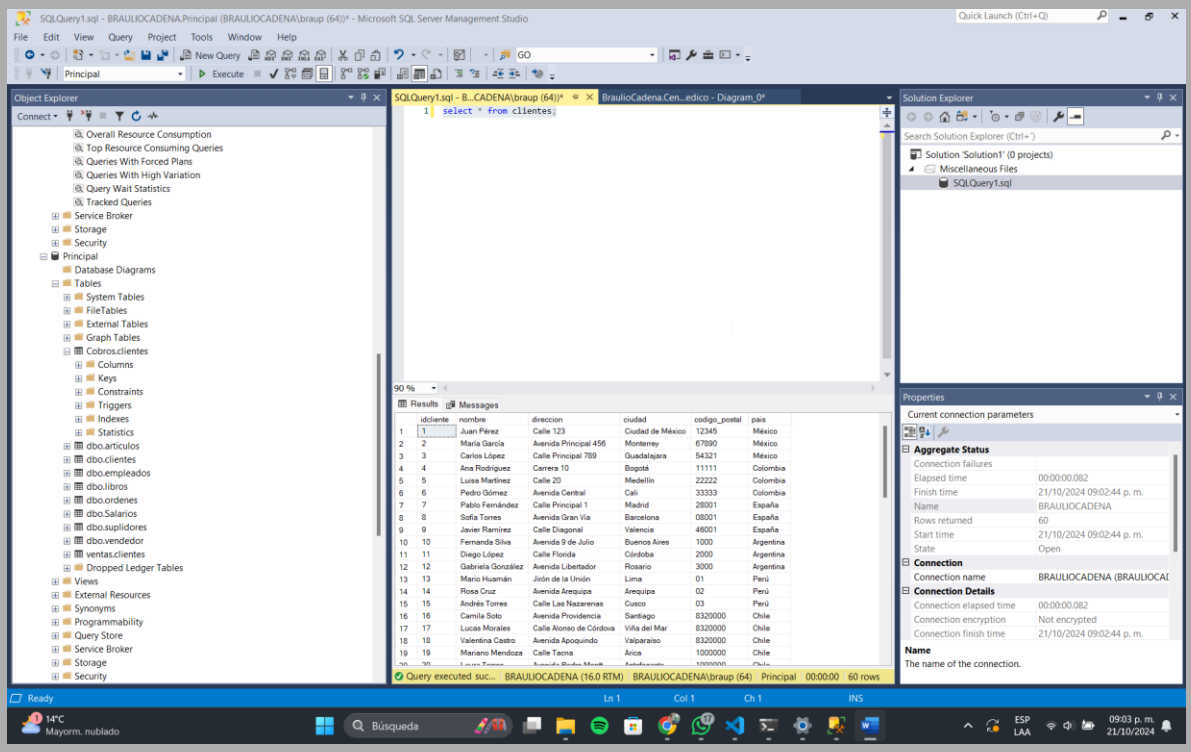
Apuntes propios sql en (SQL SERVER)





Cadena Espinoza Braulio

```
alter database Prueba modify name = Principal;
char --- Almacenar tipo de  datos de ancho fijo
varchar -- almacena tipo de datos alfanumericos de ancho de variable
text-- tipos de datos de base de texto
nchar ---- tipos de datos de ancho fijo
nvarchar--- almacena tipos de datos alfanumericos de ancho de variable
bit --- valores de 1-0
int-- almacena valores entre negativo -9000000 y 9211515155151
bigint ---- almacena valores de e entre nuevo billones y mas
decimal--- almacena valores de -10^38 y 10^38-1
numeric -- almacena valores entre -10^30 + 1 10^38-1
money ---valores de 9billones a -9 se usa para base de datos financiera
float-- almacena valores de -179E + 308 a 1.79E + 308
```

Drop PARA BORRAR TABLA

```
exec sp_rename  'Empleados',''Usuarios'; RENAME PARA RENOMBRAR

select * from empleados where nombre='Braulio'; MUESTRA  DATOS DE BRAULIO (TODOS LOS
BRAULIOS)

select nombre from empleados where edad  ='29'; MUESTRA SOLO LOS QUE TIENEN 29 AÑOS

truncate table empleados;           BORRA TODO SI O SI LA TABLA DE EMPLEADOS

delete from empleados where idempleado = '7'; SOLO BORRA EL NUMERO 7 (ID) DE LA
TABLA EMPLEADOS

delete from empleados;   BORRA TODO A DIFERENCIA DE TRUNCATE ESTE COMANDO ACEPTA
WHERE LO QUE SIGNIFICA QUE TRABAJA CON FILTROS (QUE QUEREMOS ELIMINAR) Y TRUNCATE SE
LIMPIA TODO.-

alter table empleados add Sexo char(1);

alter table empleados drop column fe_Contratacion; /BORRA COLUMNA DE FE_CONTRATACIÓN

select* from Empleados where idempleado =3; SOLO MUESTRA EL ID NUMERO 3 (FILA

-- DOBLE GUION ES PARA METER O USAR COMENTARIOS

/*
COMENTARIO LARGO
ES SLASH
Y ASTERISCO
PERO HAY QUE CERRARLO  CON
ASTERISCO SLASH */

update Empleados set activo ='si';    //  actualizar campo de activo a (todos a si)

update empleados set activo = 'no' where idempleado in (1,2,3,4); // actualiza campo
de activo a no en las filas 1,2,3,4

select * from Empleados where edad in (25,31,21);

seleccionamos la tabla empleados y de ahí sacábamos quien tiene 25,31,21;
```

```
insert into Salarios(Nombre,Apellido,salario)
select nombre, apellido, salario from empleados; // Insertamos en la tabla salarios
donde esta filas nombre apellido y salario, seleccionamos filas nombre apelido y
salario que vamos a agregar de la tabla empleados

insert into Salarios(Nombre,Apellido,salario)
select nombre, apellido, salario from empleados
where salario > 2500;   ///// aquí con el where hace la condición en este caso de
solo importara los que tengan un salario de mas de 2500 …

select top 5 * from empleados; // solo me imprimirá los 5 primeros filas con sus
campos sin nada de exepciones

select top 50 percent * from empleados; // con este condigo se puede mostrara el
porcentaje asignado si son 10 filas y pides un 50%  va imprimir solo las primeras 5
filas por que son el 50%

select top 3 * from empleados where activo= 'no'; // solo va imprimer los 3 primeras
filas que contenga la condición que en activo digan no

select * from clientes where nombre is null;

seleccionamos de la tabla clientes  y va a mostrar los campos que están vacíos de la
columna nombre

select * from clientes where nombre is not null;

seleccionamos de la tabla cliente y condicionamos  a que parezcan solo los que no
tienen campos vacíos en el columna nombre

select * from clientes where direccion is  null;

seleccionas de la tabla clientes y condicionamos que aparezcan solo los campos
vacíos de la columna dirección

update clientes set direccion = 'no tiene'
where direccion is null;  // actualizar de la tabla clientes en la columna dirección
a no tiene en donde este con campo vacio (null)

delete from clientes  where nombre is null; aquí solo eliminamos de la tabla
clientes en la coluna nombre todos los campos que estén vacíos

create table clientes (
idcliente int not null,
nombre varchar(10) not null,
direccion varchar (100) not null,
telefono numeric(10) null,
email varchar (50) null
);

/// Ojo aquí solo estamos creando columnas que en donde diga not null el usuario no
podrá dejar vaio el campo y donde diga nul si podrá dejar vaio el campo
```

Cadena Espinoza Braulio

```sql
--Constrains

create table personas(
idepersonas int primary key,
nombre varchar (10) not null,
edad int  not null);

insert into personas values (1, 'jose','30');
insert into personas values (1, 'tomas',30);
insert into personas values (null, 'jose','30');

// Aquí creamos una tabla personas y utilizamos un primery key que significa llave primaria , a parte  tenemos el not null donde no podremos dejar campos vacíos, cuando insertamos valores la llave primaria por defecto NO PUEDE QUEDAR VACIA NI PUEDE TENER VALORES REPETIDOS, por lo que los últimos dos insert into están mal

create table personas(
idepersonas int,
nombre varchar (10) not null,
edad int  not null,
primary key (idepersonas));  // LA PRIMARI KEY NO ES NECESARIO QUE ESTE ES PRIMER CAMPO COMO ESTE EJEMPLO ESTA AL FINAL.



create table personas(
idepersonas int,
nombre varchar (10) not null,
edad int  not null,
constraint PK_enlace_persona primary key (idepersonas)
);


///  LA FORMA CORRECTA ES NOMBRAR EL PRIMARY KEY COMO LEERA EL SISTEMA, PARA DE ESTA FORMA LO CONFIGURAS O RENOMBRAS, EN EL EJEMPLO SE LLAMA ENLACE PERSONA, (CREAR UNA REGLA QUE SE LLAMA ENLACE PERSONA DE TIPO PRIMARY KEY QUE LE APLICARA A IDEPERSONA)



create table personas(
idepersonas int not null ,
nombre varchar (10) not null,
edad int  not null);

alter table personas
add constraint PK_enlace primary key (idepersonas);

//// MUCHO OJO AQUÍ, CREAMOS UNA TABLA PERSONAS CON CAMPOS SIN PRIMARY KEY Y DESPUES LE ASIGNAMOS UNA PRIMERA KEY CON ADD CONSTRAINT  Y SU NOMBRE PARA SISTEMA PK_ENLACE PERSONA EN IDEPERSONAS, SOLO SE PODRA PASAR A PRIMARY KEY SI ES NOT NULL SI NO LE NEGARA ACCESO A PRIMARY KEY


alter table personas
drop constraint PK_enlace;
```

Cadena Espinoza Braulio

```
PARA BORRAR LA LLAVE PRIMARIA SOLO SE ALTERA LA TABLA Y SE COLOCA DROP CONSTRAINT EL
NOMBRE QUE SE LE PUSO A LA LLAVE PRIMARIA

--CONSTRAINTS RESTRICIONES O LLAVES

create table Persona(
indpersonas  int not null,
clave varchar(10),
nombre varchar (10),
edad int,
constraint uq_pidepersona unique (indpersonas),
constraint uq_clave unique(clave)
);
-- PODEMOS CONFIGURAR UNO O VARIOS CONSTRAINT DE VARIOS CAMPOS EN NUESTRA TABLA CON
UNIQUE
   --- EL UNIQUE NO PERMITE VALORES REPETIDOS PERO SI VACIOS
alter table persona
add constraint uq_indpersona unique (indpersonas);
--ALTERAMOS LA TABLA PARA AGREGAR UN NUEVO CONSTRAINT DONDE NO HABIA QUE VA LLAMAR
UQ_INDEPERSONA  Y ESTARA EN EL CAMPO INDPERSONAS
insert into persona values(1,'clave1','alberto',30);
insert into persona values (2,'clave2','ana',30);

drop table persona;

-- COUNSTRAINT RESTRICCION DE TIPO CHECK
--ES USADO PARA LIMITAR  VALORES, SOLO SE PERMITIRAN VALORES PERMITIDOS POR NOSOTROS

create table persona(
id int not null,
nombre varchar(10),
edad int,
check (edad>=18) --Colocamos una restriccion que todo lo que va en el campo solo
sera numeros mayor o igual a 18
);
select *from persona;
insert into persona values (2,'juan',16);


-- COUNSTRAINT RESTRICCION DEFAULT
----Cuando ingresemos default en un campo automaticamente nos va a colocar un no
tiene
--en nuestro campo  en este caso ciudad
create table persona(
idpersona int not null,
nombre varchar (10),
edad int not null,
ciudad varchar (50) default 'No tiene'
);

insert into persona values (1,'pedro',30,default);


select * from persona;
```

```sql
-- COUNSTRAINT RESTRICCION DEFAULT
----Cuando ingresemos default en un campo automaticamente nos va a colocar un no
tiene
--en nuestro campo  en este caso ciudad
create table persona(
idpersona int not null,
nombre varchar (10),
edad int not null,
ciudad varchar (50)
);
-- En caso que no tengamos esta regla, la podemos crear con add constraint le
ponemos el nombre
-- despues como va ir por defecto que sera no tiene para el campo ciudad.
--(CUANDO ESCRIBAN DEFAULT SE PONDRA NO TIENE)

alter table persona
add constraint Df_ciudad
default 'no tiene' for ciudad;

--PARA BORRARLA SOLO SE ALTERA LA TABLA SE AGREGA EL DROP CONSTRAINT Y EL NOMBRE DE
LA RESTRICCION DEL SISTEMA
alter table persona
drop constraint Df_ciudad;



insert into persona values (1,'pedro',30,default);

drop table persona;
select * from persona;


create table libros(
codigo int identity,
titulo varchar(60) not null,
autor varchar(60) not null
);
--



En el siguiente codigo cuando nosotros ingresemos datos sin ingresar numeros



--IDENTITY  se encargara de enumerar estos datos y haci el lleva su secuencia
insert into libros values('1988', 'harper ee');
-- Indentity no lee numeros, el sistema se encarga de administrar los valores de ese
campo
select *from libros;
```

Cadena Espinoza Braulio

```
-- Si borramos un campo con identify se eliminara toda la fila, y el contador
iniciara siempre despues
-- del numero borrado
delete from libros where codigo= 2;
drop table libros;

---Si no queremos que inicie a contar de uno en uno podemos indicarle que inicie
--modificandolo asi
create table libros(
codigo int identity (10,1), -- Aqui asigna que empiece de 10 en 10, y con el uno le
digo que aumente
-- de uno en uno, (10,11,12,13) y pongo 5 seria (10,15,20,25).
titulo varchar(60) not null,
autor varchar(60) not null
);
insert into libros values('1988', 'harper ee');

select  ident_seed('libros'); --Con este codigo podemos visualizar el parametro
inicial
select ident_incr('libros'); --Nos dice el rango de incremento de cada valor(De
cuanto en cuanto va)

set identity_insert libros on; --Con este condigo te permite insertar valores que
antes
-- no te permitia // para activarlo solo cambiamos a OF
-- FOREING KEY O LLAVE FORANEA
--ES UNA RESTRICCION PARA PREVENIR ACCIONES  O DAÑOS EN LAS
--RELACIONES DE REGISTROS O TABLAS
-- SE REALIZA PARA ENLAZAR ENTRE LLAVE PRIMARIA Y LLAVE FORANEA
--LAS TABLAS QUE TENGAN LLAVES FORANEAS SE LES LLAMAN TABLAAS HIJAS O SECUNDARIAS
-- LASPRIMARY KEY SON LAS MAADRES O PRINCIPALES
--EVITAN QIE SE INSERTEN DATOS NO COMPATIBLES CON EL CAMPO DE LA LLAVE PRIMARIA
--(LOS DATOS QUE SE INDERTAN EN UNA TABLA FORANEA DEBEN SER LOS MISMOS DE UNA
PRIMARIA


create table clientes (
id_cliente int,
nombre varchar(20) not null,
apellido varchar(30) not null,
edad  int not null,
constraint Pk_clientes primary key (id_cliente)
);
-- la llave foranea esta en la tabla ordenes por lo tanto si escribimos en tiene que
tener
--los mismos datos de id cliente


create table ordenes(
id_orden int not null,
articulo varchar(59) not null,
id_cliente int
constraint FK_ordenes_clientes foreign key references clientes(id_cliente)
);
---si queremos borrar campos de las dos tablas foraneas y secundarias tenemos que
-- colocar (on delete cascade) asi....
```

```
constraint FK_ordenes_clientes foreign key references clientes(id_cliente) on delete
cascade

-----------------------------------------------
insert into clientes values (1,'juan','perez',30);
----en este insert no va funcionar por que tiene un 5 cuando en id_cliente llave
primaria tiene un 1 y
-- se aplica nuestra regla
insert into ordenes values(1,'martillo',5);
------Ahora si funciona
insert into ordenes values(1,'martillo',1);

select *from clientes;
select * from ordenes;

------------------
--Aqui si me deja eliminar por que la tanbla ordenes no dependen de clientes

delete from ordenes where id_cliente= 1;
----------------------
--No me deja borrrar por que la tabla clientes tiene datos relacionados
--con ordenes, de los datos clientes dependen los de ordenes
delete from clientes id_cliente=1;
---------------------------------------
---Si queremos actualizar el id clientes desde ordenes no nos va dejar
update clientes set id_cliente=5
where id_cliente =1;
----------------
-- Tampoco podemos borrar la tabla primaria, el foreigth key protege la tabla
primaria
--para borrar la primera tabla tenemos que borrar primero la llave foranea (segunda
tabla)
drop table clientes;
-----------------------------
--Para borrar el foreingn key soolo lo hacemos asi con drop constraint
alter table ordenes
drop constraint FK_ordenes_clientes;
-------------------

---Vistas
---lAS VISTAS SON PARA AGILIZAR Y NO ESCRIBIR TANTO CODIGO

--Aqui creamos una vista con mayores de 30  y solo cuando queramos
--verla escriviriamos la vista no la condicion
create view Mayores_30
as
select nombre, apellido, telefono, edad
from clientes where edad>30;
select *from Mayores_30;
----------
--Para alterar la vista es alter view y el codigo
alter view Mayores_30
as select nombre, apellido, telefono, edad, fecha_nacimiento
from clientes where edad>30;
-- y para borrarlo es drop view y borramos
drop view Mayores_30;
```

```sql
--Indices

--estructuras utilizadas para mejorar el rendimiento de las consultas
-- nos ayuda a acelerar la busqueda y recuperacion de datos..
--- Existen dos tipos
-- Clustered: Definen el orden de los datos
-- NonClustered: No definen dicho datos
----- Son faciles  de borrar
--Podemos tener varios indices en una sola tabla
---Son recomendados para tablas que son muy consultadas
-- Como maximo dos indices por que puede realentizar la tabla
select * from Empleados;
use Principal;
-------------------------------------
----En este codigo hacemos un indice para la tabla empleados va agilizar la busqueda
de mi tabla empleados
create clustered index I_idempleado
on empleados (idempleado);
-----------------------------------------
--En este codigo es para un nonclustered
create nonclustered  index I_Edad_empleado
on empleados(edad);
-------------------------------------------
--- Con este codigo  cambiamos el nombre del indice
exec sp_rename  'Empleados.I_idempleado','I_id','INDEX';
-----------------------------------------------
----Esto es para eliminar el indice
drop index I_id on empleados;
```

```sql
---DISTINCT
--Utilizada para seleccionar valores unicos de una columna
--Debuelve solo los valores distintos y elimina las repeticiones de datos en el
resultado


select * from clientes;
-- Muestra registros repetidos ejemplo 3 alemanias 4 mexicos etc
select distinct pais from clientes;
-------------
--Este codigo muestra valores repetidos en este caso con la condicion
-- donnde en pais haya mas de 1 argentina
select distinct id_cliente from clientes
where pais = 'argentina';
----------------------------------------
---tambien sirve para saber valores nulos
---Usamos el distinct para saber que paises no han comprado
--Aqui muestra los paises que no tienen compras
select distinct (pais) from clientes
where compras is null;
```

Apuntes propios sql en (SQL SERVER)

```
--ALIAS EN UN BASE DE DATOS
--Un  alias es un nombre especifico aque se le da a un campo
---al momento de hacer una consulta en una tabla
-- Nombres de campos personalizados
-------------------
--En este codigo seleccionamos vampos y los renombramos(Temporalmente) con AS
--para poderlos mostrar con otro nombre y tambien condiconamos en donde salario
-- de la tabla empleados solo muestra  los valores con salario dem <3000

select idempleado as Indentificador, Nombre as Primer_Nombre,
Apellido as primer_apellido, salario from empleados where salario <3000;
--SI NO QUEREMOS QUE MUESTRE UN _ GUION BAJO SOLO PONEMOS SERIA ENTRE COMILLAS
EJEMPLO :"Primer Nombre"
-------------------------------------------------------------------------
---CONCATENAR VALORES
--SIFNIFICA UNIR VALORES, UNIR REGISTROS
-----------------------------------------------------
--Esta es una consulta normal donde los datos se muestran separados
Select nombre, apellido, Salario from Empleados;
-------------------------------------------------
--Esta es una consulta que une datos
select nombre + apellido from  empleados;
-- ES REMENDABLE USARLO ASI +''+ PARA QUE HAYA UN ESPACIO
-- El signo de + UNE
-- SIN '' SE VERIA BraulioCadena
---Con '' Se veria Braulio Cadena
---SOLO SE CONCATENAN DATOS IGUALES ES DECIR VALORES IGUALES NUMEROS CON NUMEROS Y
LETRAS CON LETRAS ETC
-----------------------------------------------------------------
--En este codigo concatenemos NUMEROS Y LETRAS edad lo coonvertimos a varchar 2
facil
--por que no hay edad que supera dos valores
select nombre + apellido + cast (edad as varchar(2))from  empleados;
```

```
--Operadores Matematicos
--sirven para hacer operaciones aritmeticas en las consultas

select (10+10);-- Ejemplo basico de un operador

select * from articulos;
------------------------
--Este codigo muestra nombre seleccion y precio actual mas el 10% con un alias
---que es nuevo precio de la tabla articulos
select nombre, descripcion, precio + (precio *0.1)as "Nuevo precio" from articulos;
---- Con este codigo vemos el 10% menos a nuestro precio de la tabla articulos
select nombre, descripcion, precio -(precio*0.1)as "Descuento " from articulos;
-----------------------------------------
--En este codigo vemos el costo total de los inventarios de la tabla articulos
-- ejemplo  en laptop hacer tenemos un costo total de 8999 sumadas todas las acer
select nombre, descripcion, precio*cantidad as total from articulos;
-------------------------------------------------------------------------
--En este codigo nos enseña el total en ventas de cada articulo
select nombre, descripcion,
precio* vendidos as "Ventas" from articulos;
-----------------------------------------------------
---En este codigo podemos ver las existencias que quedan de cada articulo
select nombre, descripcion,
cantidad - vendidos as Existencia from articulos;
```

Cadena Espinoza Braulio

```sql
--Esquemas de sql
-- Es una estructura logica para organizar y agrupar objetos
--- Tablas procedimientos, funciones, sinonimos...
--ACTUA COMO CONTENEDOR LOGICO Y NOS PERMITE AISLAR Y CONTROLAR EL ACCESO A ESOS
DATOS
---En este codigo creamos un esquema llamdo crobros--
create schema Cobros;
-----------------------------
--En este codigo creamos una tabla en nuestro esquema ventas, y esa tabla se llama
clientes
create table ventas.clientes(
idclientes int,
nombre int,
direcion varchar(30));
---Este codigo visualiza la tabla clientes del esquema ventas..
select * from ventas.clientes;
```

```sql
--Order by
--para organizar Nuestras consultas de manera ascendente y decende
use Principal;
select * from Empleados;
---En la siguiente consulta se ordena el campo id empleados de menor a mayor, de la
tabla empleados
--(ASCENDENTE ES POR DEFECTO no hay que colocar asc) en no enteros o letras
select * from Empleados order by idempleado;
-----------------------------------------
--En este codgio ordenamos el campo edad de manera ascendente (mayor a menor)
---- de la tabla empleado
select * from Empleado order by edad asc;
-------------------------------------------------------------
---En este codigo ordenamos de activo a no activo (LETRAS o varchar) y como primero
esta la N en el alfabeto
--por lo tanto al algonizarlo colocara primero los NO y despues los SI
----OJo pero aqui pusimos desc asi que van primero los si
select * from Empleado order by activo desc;
-------------------------------------------------------------
-- En este codigo hacemos una consulta de la tabla empleados y ordenamos el campo
edad
-- con 25 y va desendiendo en 30 35 etc, no se coloca asc por defecto.
select * from Empleados
where Edad > 25 order by Edad;
-------------------------------------------------------------
--Organizar datos por varios campos
-- En este codigo se consultan los campos NOMBRE , EDAD, SALARIO de la tabla nombres
-- y los ordena de menor a mayor por defecto ascendente, y empieza con edad y
despues salario...
-- Tambien podemos ordenarle escribiendo asc o desc si es que queremos algo en
especifico
select nombre, edad, Salario
from Empleados
order by Edad, salario;
-------------------------------------------------------------------
```

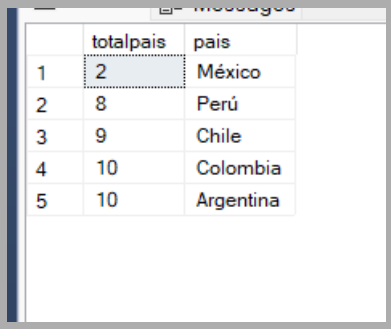
```sql
---MAX Y MIN VALORES SQL

select * from  articulos;

--En este codigo seleccionamos del campo precio de la tabla empleados, el valor
maximo y lo
-- llamamos lo_ caro
select MAX(precio)as Lo_caro from articulos;
--------------------------
--En este codigo seleccionamos del campo precio de la tabla empleados, el valor
minimo y lo
-- llamamos lo_ mas barato
select Min(precio) as Lo_mas_barato from articulos;
-------------------------------------------------------------
---Aqui solo  unimos las consultas de arriba, consultamos lo caro y lo barato
--- de la tabla articulos
select MAX(precio) as mas_caro, MIN(precio) as mas_barato from articulos;
```

```sql
--FUNCIONES DE AGRUPACION SQL SERVER -COUNT -SUM -AVG
--COUNT Cuenta la cantidad de registros en nuestras tablas
--SUM Nos permite realizar una sumatoria de registros
--AVG Nos permite realizar el promedio de registros

select * from Empleados;
---------------------------
select COUNT(*)as Cantidad from Empleados;
--------------------------------}
--En este codigo contamos de la tabla empleados cuantos asistentes del campo puesto
tienen sueldo
select COUNT(sueldo)as from Empleados
where puesto = asistente;
---------------------------------
--En este codigo hacemos una suma del campo sueldo "Es como el total de nominas de
un empresa"
-- de la tabla empleados
select SUM (sueldo) as total from Empleados;
-------------------------------------------------------------
--En este codigo nos da el total de sueldo de todos los asistentes de empleados.
select SUM (sueldo) as total from Empleados
where puesto= 'asistente'

--------------------------------------
--En este codigo solo vemos la suma de los sueldos distintos de la tabla empleados
select SUM (distinct sueldo) from Empleados;
-----------------------------------------------
---Funcion AVG

select * from Empleados;
--En este campo vemos el promedio de nuestro campo empleados de nuestra tabla
empleados
select AVG (sueldo) from Empleados;
------------------------
--En esta consulta vemos el promedio del campo sueldo de los generentes de la tabla
empleados
select AVG (sueldo) from Empleados
```

Cadena Espinoza Braulio

```
where puesto ='Gerente';
-----------------------------------------
--Aqui solo vemos el promedio distinto de sueldo de la tabla empleados
select AVG (distinct sueldo) from Empleados;
-------------------------------------------------

--DIFICIL
--En este coonsulta se muestra el total de todos los registros, el promedo del campo
sueldo y la suma de
--el campo cantidad de hijos, esto solo en las filas 1,2,45,8
---todo esto que esta en la tabla empleados

select COUNT(*), AVG(sueldo), SUM(cant_hijos) from Empleados
where id_empleado in (1,3,4,5,8);

--Operadores AND OR Y NOT
--Filtran registros en funcion de mas de una condicion

--End nos da una consulta siempre y cuando encuentre una coincidencia tanto en la
condicion uno
--como en la condicion dos
--END NO SE PUEDE UTILIZAR VARIAS VECES
-----------------------------------------------------
--En esta consulta vemos los campos de clientes, solo los que tengan pais con italia
y ciudad con roma
--- en pocas palabras puros italias y romas
select * from clientes
where pais= 'Italia'and ciudad ='roma';
-------------------------------------------------
--En esta consulta estamos consultando si hay registros en el campo de ciudad roma
-- o ciudad venecia y podemos agregar mas operadores como florencia
select * from clientes
where ciudad='roma' or ciudad='venecia'
or ciudad = 'florencia';
-----------------------------------------------------
--Traeme todos los datos de la tabla  clientes del campo pais que sea italia y
ciudad de roma y venecia
select * from clientes
where pais ='italia' and ciudad in ('Roma','Venecia');
-----------------------------------------------------------------------
Esta es una consulta donde consultamos la tabla cientes pero meno los del campo pais
alemania
select* from clientes
where not pais = 'alemania';
```

```sql
--CLausula BETWEEN
--Nos permite consultar valores especificos
--En este codigo de la tabla empleados vamos a mostrar las filas del campo
ideempleado
---1 al 7 , sabemos que podemos usar la clausa IN pero en ella seria 1,2,3,4 etc y
en
-- between tomamos todo el rango de numero a numero
select* from empleados; where idempleado
between 1 and 7;
-------------------------------------------------
--DIFICIL
--En esta consulta vemos de la tabla empleados  solo a los que tengan id 1 al 7 en
el campo id_empleado
---- o tengan en el campo cant hijos de 1 a 3...
select * from  Empleados where id_empleado
between 1 and 7 or cant_hijos between 1 al 3;
-----------------------------------------------------------------------
-- En esta consulta consultamos de la tabla empleados cque los sueldos
-- sean de 2000  y 4000 }, de los cuales solo mostrara quienes en puesto no seas
dearroladores...
select* from empleados
where sueldo between 2000 and 4000
and puesto not in ('Desarrollador');
-------------------------------------------
--Operador like
--ES unsaado cuando buscamos patrones especificos en una tabla

--En esta consulta de caampo nombre de la tabla clientes solo se va a mostrar
-- los nombres que inicien con A
select * from clientes where nombre like '%A';

-- si colocamos %A solo mostrara los nombres que tengas una A en la segunda letra de
su nombre
-- si colocamos un _%A solo mostrara los nombres que en la tercera letra tengan una
A
-- si colocamos un Pe% solo mostrara nombres con pe al principio como pepe pedro
perry
--- TODO FUNCIONA IGUAL CON NOT LIKE PERO POR LO CONTRARIO NO MUESTRA LAS CLAUSALAS
QUE LE PNGAMOS

--Operador union}
--Usado para combinar un conjunto de resultados de dos o mas columnas
---Regla debe tener el mismo numero de columnas en cada consulta
--- Debe tener datos iguales
select * from clientes;
select * from suplidores;
--En este codigo o consulta nos dice el campo de definicion en la columna tipo donde
venga contacto, ciudad y pais
-- y lo mismo en union con suplidor .........
select 'cliente' as tipo, contacto, ciudad, pais from clientes
union
select 'Suplidor' as tipo, contacto, ciudad, pais from suplidores;
```



	totalpais	pais
1	2	México
2	8	Perú
3	9	Chile
4	10	Colombia
5	10	Argentina

```
--FUNCIOON DE AGRUPACION GROUP BY
-- AGRUPA FILAS QUE TIENEN LOS MISMOS VALORES EN EL RESULTADO
DE UNA CONSULTA
select * from clientes;
--En esta consulta contamos los id cliente (clientes) de cada
pais de la tabla clientes
--- en grupos de pais ejemplo ... 29 de argentina ... 9 de
colombia, el order solo lo ordena y el AS renombra
--el campo a total pais
-----
select COUNT (idcliente) as totalpais, pais
from clientes
group  by  pais order by totalpais ;
-----FUNCIONA CON MAX, MIN, AVG,
------------------------
--loas ve. o. etc son alias
--seleccioname de mi tabla vendedor (ve) el nombre y cuenta la cantidad de
-- registros de mi tabla ordenes en un campo que se llame cantidad de ordenes
--tomando como tabla de la izquierda la tabla ordenes
-- y uniendola a la tabla vendedor donde el campo de la tabla vendedor
---id vendedor sea igual al campo vendedor de la tabla ordenes, todo esto agrupalo
en el campo ordenes

select ve.nombre, COUNT (o.idorden) as "Cantidad de ordenes"
from  ordenes o
left join vendedor ve on ve.id_vendedor = o.id_vendedor
group by ve.nombre
--Función de agrupacion having
--REALIZA FILTOS A CONSULTAS ES CASI LO  MISMO A LA FUNCION WHERE
--AL UTILIZA AGRUPAMIENTO NO SE PUEDE UTILIZAR WHERE SOLO HAVING

select * from clientes;
--En este codigo seleccionamos la cantidad de id cliente y pais de la tabla clientes
-- agrupamosen un campo llamdo pais teniendo que mostrar la cantidad de clientes con
mas de 5 ,
-- paises con mas de 5 clientes y esto lo voy ordenar el resultado en forma
descendente
select COUNT (idcliente), pais
from clientes group by pais
having COUNT (idcliente) > 5
order by COUNT (idcliente) desc;
-----------------------------------------------------
--En esta consulta tomamos el alias v de nombre  con la cantidad o de id orden  en
un campo llamado
-- cantidad de ordenes  de la tabla ordenes que tengan el alias o , que seria
idorden (ordenes o  todos los campos del select que tengan
--una o junto significa que pertenece a la tabla ordenes,
-- a eso lo unimos con vendedor,(todos los campos que tengan una v opertenecen a la
tabla vendedor)
---luego coloamos los campos de enlace o.idvendedor(es decir mi campo ide vendedor
de mi tablar aorden)
-- con el campo v.id vendedor
--agrupo todo en el campo nombre
-- con el where solo traemos el monbre ana y pedro
--ahora le decimos al sistema que todos los datos los traiga siempre y cuando sean
mayor a dos
```

```sql
select v.nombre, COUNT (o.idorden) as "Cantidad  ordenes"
from ordenes o
inner join vendedor v on o.id_vendedor = v.id_vendedor
where nombre like '%Ana%' or nombre like '%Pedro%'
group by nombre
having COUNT (o.idorden) >2;



----------------------------  INNER JOIN
select * from Paciente;
select * from TurnoPaciente;

select * from paciente p
inner join TurnoPaciente t

on t.idPaciente = p.idPaciente

----------------------------------LEDFT JOIN

select * from Paciente p
right join TurnoPaciente T
on T.idpaciente = P.idpaciente
```