

# **UNIVERSIDAD CÉSAR VALLEJO**

## **FACULTAD DE INGENIERÍA**

### **Escuela de Ingeniería de Sistemas**

#### **Práctica MARKETPERU - Gestión de Datos S11**

#### **AUTOR**

Quispe Olano, Marlon Yber ([0000-0002-3668-207X](mailto:0000-0002-3668-207X))

#### **DOCENTE**

Saavedra Jimenez, Robert Roy

#### **SECCIÓN**

**A1**

**PERÚ - 2023**

# INTRODUCCIÓN

El presente informe tiene como propósito detallar y analizar una base de datos en SQL Server, así como realizar consultas utilizando el lenguaje de consulta estructurado (SQL) y documentar cada paso a través de capturas y explicaciones de los códigos utilizados.

En el ámbito de la gestión de bases de datos, SQL Server es ampliamente reconocido como un sistema robusto y confiable. Su capacidad para almacenar, administrar y manipular grandes volúmenes de datos lo convierte en una herramienta fundamental en el campo de la informática y la gestión de la información. A través de este informe, explicaremos diferentes aspectos de SQL Server y su aplicación práctica en la realización de consultas.

Durante el desarrollo del informe, se mostrarán ejemplos de consultas utilizando código SQL específico, acompañados de capturas de pantalla que ilustran el resultado obtenido en cada caso. Además, se proporcionarán explicaciones detalladas de cada línea de código, destacando su propósito y función dentro del contexto de la base de datos.

El objetivo principal de este informe es proporcionar una comprensión clara y concisa del proceso de consulta en SQL Server, así como de las capacidades y características que ofrece este sistema de gestión de bases de datos. A través de la ejecución y análisis de las consultas, buscamos fortalecer nuestros conocimientos en la manipulación de datos, la extracción de información relevante y la comprensión de las estructuras y relaciones presentes en la base de datos.

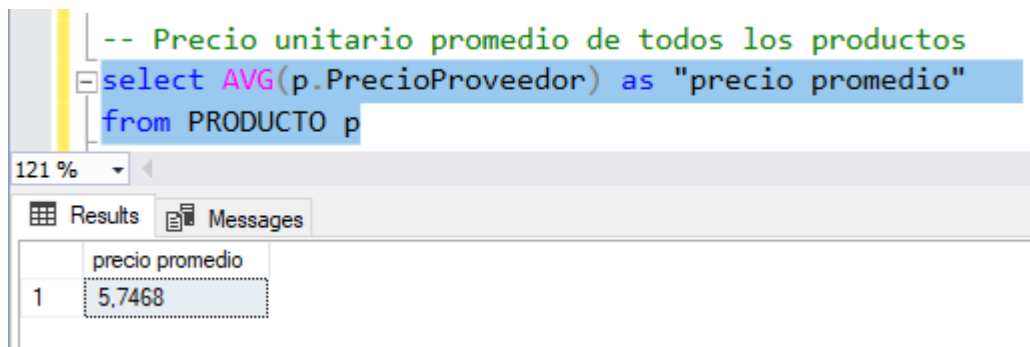
# I. DESARROLLO

## “Adjuntando Evidencias”

### 1. Primer Video

#### 1.1 Función AVG

- La consulta selecciona el promedio de los precios de proveedor de la tabla PRODUCTO y lo muestra como "precio promedio".



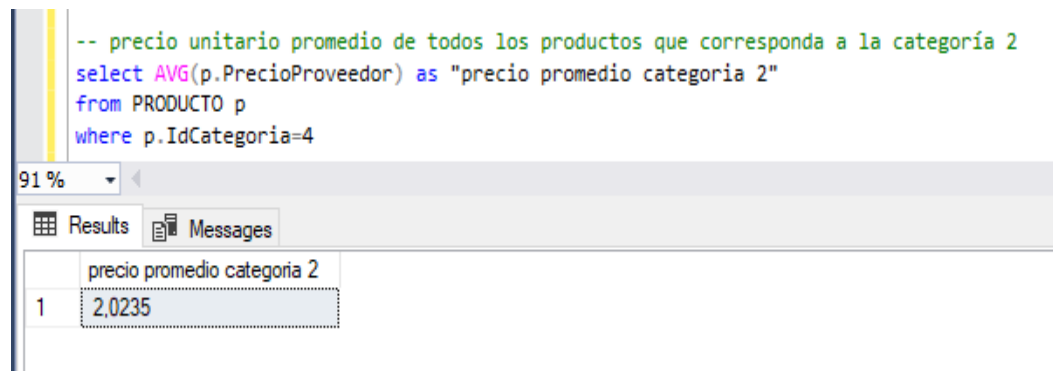
```
-- Precio unitario promedio de todos los productos
select AVG(p.PrecioProveedor) as "precio promedio"
from PRODUCTO p
```

121 %

Results Messages

	precio promedio
1	5,7468

- La consulta calcula el promedio de los precios de proveedor de los productos de la categoría con el identificador 4 y lo muestra como "precio promedio categoría 2".



```
-- precio unitario promedio de todos los productos que corresponda a la categoría 2
select AVG(p.PrecioProveedor) as "precio promedio categoría 2"
from PRODUCTO p
where p.IdCategoria=4
```

91 %

Results Messages

	precio promedio categoría 2
1	2,0235

- La consulta calcula el promedio del stock actual de los productos cuyo precio de proveedor es mayor a 10 y lo muestra como "Promedio stock".

```
-- stock de los promedios de los productos cuyo precio unit es mayor a 10
select avg(p.StockActual) as "Promedio stock"
from PRODUCTO p
where p.PrecioProveedor > 10
go
```

91 %

Results Messages

	Promedio stock
1	139

## 1.2 Función COUNT y DISTINCT

- La consulta cuenta la cantidad de registros en la tabla PRODUCTO y lo muestra como "cantidad producto".

```
-- Cuenta de los productos registrados en la tabla producto
select COUNT(*) as "cantidad producto"
from PRODUCTO
```

91 %

Results Messages

	cantidad producto
1	138

- La consulta selecciona todas las columnas de la tabla GUIA\_DETALLE y muestra todos los registros.

```
-- Cuenta los productos despechados a los diferentes locales de la empresa
select *
from GUIA_DETALLE g
```

91 %

Results Messages

	IdGuia	IdProducto	PrecioVenta	Cantidad
1	1	1	2,25	20
2	1	2	1,50	20
3	1	3	2,25	20
4	1	4	1,95	20
5	1	6	2,70	30
6	1	7	3,30	25
7	1	8	2,40	25

```
-- Cuenta los productos despatchados a los diferentes locales de la empresa
select distinct(g.IdProducto)
from GUIA_DETALLE g
```

91 %

Results Messages

	IdProducto
1	92
2	115
3	29
4	132
5	9
6	109
7	3
8	95
9	118

```
-- Cuenta los productos despatchados a los diferentes locales de la empresa
select count(distinct(g.IdProducto)) as despachos
from GUIA_DETALLE g
go
```

91 %

Results Messages

	despachos
1	66

### 1.3 Función MIN, MAX y AVG

- La consulta calcula el precio más bajo, el precio más alto y el precio promedio de los proveedores de los productos y los muestra con las etiquetas "Precio Barato", "Precio Caro" y "Precio Promedio" respectivamente.

```
-- precio mas alto, mas bajo y el promedio
select
    min(p.PrecioProveedor) as "Precio Barato",
    max(p.PrecioProveedor) as "Precio caro",
    avg(p.PrecioProveedor) as "Precio Promedio"
from PRODUCTO p
```

91 %

Results Messages

	Precio Barato	Precio caro	Precio Promedio
1	0,40	24,40	5,7468

- La consulta selecciona la fecha más reciente y la fecha más antigua de la columna FechaSalida de la tabla guia y las muestra con las etiquetas "Fecha reciente" y "Fecha antigua", respectivamente.

```
-- guias de remision mas reciente y mas antigua
select
    max(g.FechaSalida) as "Fecha reciente",
    min(g.FechaSalida) as "Fecha antigua"
from guia g
go
```

	Fecha reciente	Fecha antigua
1	2023-06-15 09:27:56.370	2023-06-05 09:27:56.077

- La consulta calcula el monto total multiplicando el PrecioVenta por la Cantidad en cada registro de la tabla GUIA\_DETALLE y muestra el resultado como "Monto Total".

```
-- monto total de los productos saldios de alamce
select sum(g.PrecioVenta*g.Cantidad) as "Monto Total"
from GUIA_DETALLE g
go
```

	Monto Total
1	378095,00

- La consulta selecciona el nombre del producto (columna Nombre) y calcula el valorizado multiplicando el PrecioProveedor por el StockActual de cada producto en la tabla producto, y lo muestra como "valorizado".

```
-- stock valorado
select p.Nombre,p.PrecioProveedor*p.StockActual as valorizado
from producto p
go
```

	Nombre	valorizado
1	CARAMELOS BASTON VIENA ARCOR	300,00
2	CARAMELOS SURTIDO DE FRUTAS	300,00
3	CARAMELOS FRUTAS SURTIDA ARCOR	375,00
4	CARAMELOS FRUTAS MASTICABLES	325,00
5	CHUPETES LOLY AMBROSOLI	180,00
6	FRUNA SURTIDA DONOFRIO	900,00
7	CHOCOLATE DOÑA PEPA FIELD	1100,00
8	CHOCOLATE CUA CUA FIELD	800,00

- La consulta calcula la suma total de la cantidad despachada (columna Cantidad) en la tabla GUIA\_DETALLE, donde el IdProducto es igual a

7, y muestra el resultado como "Despachados unidades".

```
-- Total de unidades despchadas del producto 7
select SUM(g.Cantidad) as "Despachados unidades"
from GUIA_DETALLE g
where g.IdProducto=7
go
```

91 %

Results Messages

	Despachados unidades
1	500

#### 1.4 Cláusula GROUP BY

- La consulta proporciona un resumen de la cantidad de productos registrados en cada categoría. Utilizando la cláusula GROUP BY junto con la función COUNT, se agrupan los productos por categoría y se cuenta la cantidad de productos en cada categoría.

```
-- Clausula group by
-- cantidad de productos registrados para cada categoría
select p.IdCategoria,
       count(p.IdProducto) as "cant productos"
from PRODUCTO p
group by p.IdCategoria
go
```

91 %

Results Messages

	IdCategoria	cant productos
1	1	25
2	2	25
3	3	41
4	4	17
5	5	15
6	6	15

- La consulta selecciona la columna IdCategoria como "categoria", la columna IdProveedor como "proveedor" y utiliza la función de agregación COUNT para contar la cantidad de productos (columna IdProducto) en cada combinación de categoría y proveedor.

```
-- cantidad de productos por proveedor para la categoría de 2 y 4
select
    p.IdCategoria as categoria,
    p.IdProveedor as proveedor,
    COUNT(p.IdProducto) as "cant productos"
from PRODUCTO p
where p.IdCategoria IN (2,4)
group by p.IdCategoria, p.IdProveedor
order by p.IdCategoria asc
```

	categoria	proveedor	cant productos
1	2	1	4
2	2	2	10
3	2	3	4
4	2	4	4
5	2	12	3
6	4	1	6
7	4	2	6
8	4	13	5

- La consulta selecciona la columna IdProducto y utiliza la función de agregación SUM para calcular el monto total multiplicando el PrecioVenta por la Cantidad en cada registro de la tabla GUIA\_DETALLE. Los resultados se agrupan por el IdProducto.

```
--Monto total despachado por producto
select g.IdProducto, sum(g.PrecioVenta*g.Cantidad) as "Monto total"
from GUIA_DETALLE g
group by g.IdProducto
order by "Monto total" desc
go
```

	IdProducto	Monto total
1	130	27000,00
2	124	22500,00
3	125	22500,00
4	129	21150,00
5	64	21000,00
6	127	20250,00
7	126	16875,00
8	132	15750,00
9	66	15000,00
10	65	15000,00

- La consulta selecciona la columna IdProducto y utiliza la función de agregación SUM para calcular el monto total multiplicando el PrecioVenta por la Cantidad en cada registro de la tabla GUIA\_DETALLE. Los resultados se agrupan por el IdProducto. Luego, la cláusula HAVING se utiliza para filtrar los resultados y mostrar solo aquellos registros cuyo monto total despachado es mayor a 20000.



```
-- Productos cuyo monto total despachado es mayor a 20000
select g.IdProducto,
       sum(g.PrecioVenta*g.Cantidad) as "Monto total"
from GUIA_DETALLE g
group by g.IdProducto
having sum(g.PrecioVenta*g.Cantidad)>20000
order by "Monto total" desc
go
```

91 %

Results Messages

	IdProducto	Monto total
1	130	27000,00
2	124	22500,00
3	125	22500,00
4	129	21150,00
5	64	21000,00
6	127	20250,00

- La consulta realiza una combinación interna (inner join) entre las tablas PRODUCTO y CATEGORIA utilizando la condición de igualdad entre las columnas IdCategoria de ambas tablas. Selecciona la columna IdCategoria de la tabla PRODUCTO, la columna Categoria de la tabla CATEGORIA y utiliza la función de agregación COUNT para contar la cantidad de productos (columna IdProducto) en cada combinación de categoría y categoría. Los resultados se agrupan por el IdCategoria y la Categoria.

```
-- mostrando el nombre del producto
select p.IdCategoria, c.Categoria,
       count(p.IdProducto) as "cant productos"
from PRODUCTO p inner join CATEGORIA c on (p.IdCategoria=c.IdCategoria)
group by p.IdCategoria, c.Categoria
go
```

110 %

Results Messages

	IdCategoria	Categoria	cant productos
1	1	GOLOSINAS	25
2	2	EMBUTIDOS	25
3	3	HIGIENE PERSONAL	41
4	4	LACTEOS	17
5	5	LICORES Y GASEOSAS	15
6	6	LIMPIEZA	15

- La consulta combina internamente (inner join) las tablas GUIA y GUIA\_DETALLE utilizando la condición de igualdad entre las columnas IdGuia de ambas tablas. Selecciona las columnas IdGuia, IdLocal y FechaSalida de la tabla GUIA. También calcula la columna monto sumando la multiplicación de la Cantidad por el PrecioVenta en cada registro de la tabla GUIA\_DETALLE. Los resultados se agrupan por

IdGuia, IdLocal y FechaSalida.

```
-- Escriba una consulta que muestre los datos de la cabecera de la
-- guía de remision número 27 y además su monto total

select g.IdGuia, g.IdLocal, g.FechaSalida,
       monto=sum(x.Cantidad * x.PrecioVenta)
from GUIA g inner join GUIA_DETALLE x on (g.IdGuia = x.IdGuia)
group by g.IdGuia, g.IdLocal, g.FechaSalida
having g.IdGuia=27
go
```

IdGuia	IdLocal	FechaSalida	monto
27	2	2023-06-05 09:27:56.127	10515,00

- La consulta combina internamente (inner join) las tablas LOCAL, GUIA y GUIA\_DETALLE utilizando las condiciones de igualdad entre las columnas IdLocal de LOCAL y GUIA, y entre las columnas IdGuia de GUIA y GUIA\_DETALLE. Selecciona la columna Direccion de la tabla LOCAL. También calcula la columna monto sumando la multiplicación de PrecioVenta por Cantidad en cada registro de la tabla GUIA\_DETALLE. Los resultados se agrupan por la columna Direccion de la tabla LOCAL.

```
-- Escriba una consulta que muestre el monto total despachado a cada local

select l.Direccion,
       monto=sum(g.PrecioVenta*g.Cantidad)
from LOCAL l inner join GUIA x on (l.IdLocal=x.IdLocal)
              inner join GUIA_DETALLE g on (x.IdGuia=g.IdGuia)
group by l.Direccion
go
```

Direccion	monto
AV. BOLIVAR 1789	76363,75
AV. ESPAÑA 775	75122,50
AV. LA PAZ 659	76363,75
AV. SAENZ PEÑA 590	75122,50
PANAMERICANA NORTE KM. 17.5	75122,50

- La consulta combina internamente (inner join) las tablas GUIA y GUIA\_DETALLE utilizando la condición de igualdad entre las columnas IdGuia de ambas tablas. Selecciona el año (YEAR) y mes (MONTH) de la columna FechaSalida de la tabla GUIA. También calcula la columna "Total Unidades" sumando la columna Cantidad en cada registro de la tabla GUIA\_DETALLE. Se aplica una cláusula WHERE para filtrar los resultados y mostrar solo aquellos registros donde el IdProducto sea igual a 27. Los resultados se agrupan por el año y mes de la

FechaSalida. Luego, se ordenan por el año y mes en orden ascendente.

```
select
    YEAR(g.FechaSalida) as Año,
    MONTH(g.FechaSalida) as Mes,
    sum(x.Cantidad) as "Total Unidades"
from GUIA g inner join GUIA_DETALLE x on(g.IdGuia=x.IdGuia)
where x.IdProducto=27
group by YEAR(g.FechaSalida), MONTH(g.FechaSalida)
order by Año, Mes
go
```

- La consulta combina internamente (inner join) las tablas GUIA, GUIA\_DETALLE y PRODUCTO utilizando las condiciones de igualdad entre las columnas IdGuia de GUIA y GUIA\_DETALLE, y entre las columnas IdProducto de GUIA\_DETALLE y PRODUCTO. Selecciona la columna Nombre de la tabla PRODUCTO, y el año (YEAR) y mes (MONTH) de la columna FechaSalida de la tabla GUIA. También calcula la columna "Total Unidades" sumando la columna Cantidad en cada registro de la tabla GUIA\_DETALLE. Los resultados se agrupan por el nombre del producto, año y mes de la FechaSalida. Luego, se ordenan por el nombre del producto, año y mes en orden ascendente.

```
-- Escriba una consulta que muestre el total de unidades mensuales despachadas de cada grupo
-- la consultas debe mostrar el nombre del producto
-- unidades mensuales despachadas de cada producto
select p.Nombre,
    YEAR(g.FechaSalida) as Año,
    MONTH(g.FechaSalida) as Mes,
    sum(x.Cantidad) as "Total Unidades"
from GUIA g inner join GUIA_DETALLE x on(g.IdGuia=x.IdGuia)
    inner join PRODUCTO p on (x.IdProducto=p.IdProducto)
group by p.Nombre, YEAR(g.FechaSalida), MONTH(g.FechaSalida)
order by p.Nombre, Año, Mes
go
```

	Nombre	Año	Mes	Total Unidades
1	7 UP DESCARTABLE	2023	6	1500
2	ACEITE BABY JOHNSONS	2023	6	200
3	ACEITE BABY JOHNSONS C/ALOE Y VIT. E	2023	6	400
4	ACEITE PIBEDES CHICCO	2023	6	200
5	CARAMELOS BASTON VIENA ARCOR	2023	6	400
6	CARAMELOS FRUTAS MASTICABLES	2023	6	400
7	CARAMELOS FRUTAS SURTIDA ARCOR	2023	6	400

## 1.5 Sub-Consultas

- La consulta selecciona el IdProducto, Nombre, PrecioProveedor y la columna "diferencia" de la tabla PRODUCTO. La columna "diferencia" se calcula restando el PrecioProveedor de cada producto al precio promedio de todos los productos. El precio promedio se obtiene utilizando una subconsulta que selecciona la media (AVG) del PrecioProveedor de la tabla PRODUCTO.

```
-- Diferencia entre el precio promedio y el precio de cada producto
select p.IdProducto, p.Nombre, p.PrecioProveedor,
       diferencia=p.PrecioProveedor - (select AVG(p.PrecioProveedor)
from PRODUCTO p)
from PRODUCTO p
```

133 %

Results Messages

	IdProducto	Nombre	PrecioProveedor	diferencia
40	40	HOT DOG CERDE?A	8.00	2.2532
41	41	HOTDOG AMERICANO BRAEDT	9.50	3.7532
42	42	SALCHICHA DE HUACHO	10.50	4.7532
43	43	HOT DOG EXTRA SAN FERNANDO	9.50	3.7532
44	44	CHORIZO PARRILLERO LAIVE	11.50	5.7532
45	45	CHORIZO PARRILLERO OTTO KUNZ	10.50	4.7532
46	46	CHORIZO PARRILLERO BRAEDT	15.50	9.7532
47	47	CHORIZO PARRILLERO CERDE?A	11.50	5.7532
48	48	CHORIZO ITALIANO OTTO KUNZ	15.00	9.2532
49	49	CHORIZO NURENBERG BRAEDT	12.00	6.2532
50	50	CHORIZO PARRILLERO CATALANES	10.50	4.7532

- La primera consulta selecciona la fecha máxima (última fecha) de la columna FechaSalida en la tabla GUIA utilizando la función MAX. La segunda consulta combina internamente (inner join) las tablas GUIA, GUIA\_DETALLE y PRODUCTO utilizando las condiciones de igualdad entre las columnas IdGuia de GUIA y GUIA\_DETALLE, y entre las columnas IdProducto de GUIA\_DETALLE y PRODUCTO. Selecciona el IdProducto distinto, el Nombre del producto y la FechaSalida formateada en formato dd/mm/aaaa utilizando la función CONVERT y el estilo 103. Se aplica una cláusula WHERE para filtrar los resultados y mostrar solo aquellos registros donde la FechaSalida coincida con la fecha máxima obtenida en la primera consulta.

```
-- Fecha de la última salida
select max(g.FechaSalida) from guia g
-- Productos que se despacharon en la fecha de la consulta anterior
select distinct x.IdProducto,
               p.Nombre,
               CONVERT(char(10), g.FechaSalida, 103) as Fecha
from GUIA g inner join GUIA_DETALLE x on(g.IdGuia=x.IdGuia)
            inner join PRODUCTO p on (x.IdProducto=p.IdProducto)
where CONVERT(char(10), g.FechaSalida, 103)=(
select CONVERT(char(10), max(g.FechaSalida), 103) from guia g)
go
```

146 %

Results Messages Client Statistics

(No column name)

1 2023-06-16 00:29:10.690

	IdProducto	Nombre	Fecha
1	1	CARAMELOS BASTON VIENA ARCOR	16/06/2023
2	2	CARAMELOS SURTIDO DE FRUTAS	16/06/2023
3	3	CARAMELOS FRUTAS SURTIDA ARCOR	16/06/2023
4	4	CARAMELOS FRUTAS MASTICABLES	16/06/2023
5	6	FRUNA SURTIDA DONOFRIO	16/06/2023
6	7	CHOCOLATE DO?A PEPA FIELD	16/06/2023
7	8	CHOCOLATE CUA CUA FIELD	16/06/2023
8	9	MELLOWS FAMILIAR FIELD	16/06/2023
9	10	WAFER CHOCOLATE FIELD	16/06/2023
10	11	CHOCOLATE BARRA REGULAR	16/06/2023
11	12	CHOCOLATE MOSTRO FIELD	16/06/2023

## 2. Segundo Video

### 2.1 Create database Informe2023

- Se inicia la creación de una tabla llamada "USUARIO" con

tres columnas: "cod\_user" (entero), "nam\_user" (cadena de caracteres) y "status\_user" (carácter). La columna "cod\_user" se establece como clave primaria.

```
/* INICIAMOS */
-- crear una tabla
create table USUARIO(
    cod_user int primary key not null,
    nam_user varchar(50),
    status_user char(1) --est 1:activo | 0:inactivo
)
go
```

- Se inserta un registro directamente en la tabla "USUARIO" con los valores 1, 'MYBER' y '1' para las columnas "cod\_user", "nam\_user" y "status\_user" respectivamente.

```
-- Agregando los registro de forma directa
insert into USUARIO values(1, 'MYBER', '1')
---
```

- Se realiza una consulta a la tabla "USUARIO" para mostrar todos los registros. Se insertan varios registros directamente en la tabla "USUARIO" con diferentes valores para las columnas "cod\_user", "nam\_user" y "status\_user".

```
-- Agregando los registro de forma directa
insert into USUARIO values(2, 'BERMY', '1')
insert into USUARIO values(3, 'JOSEA', '1')
insert into USUARIO values(4, 'JRICH', '1')
insert into USUARIO values(5, 'LOPEZ', '1')
insert into USUARIO values(6, 'EMANU', '1')
insert into USUARIO values(7, 'ZOTOM', '1')
---
```

- Se utiliza una forma diferente de inserción de registros en la tabla "USUARIO" especificando las columnas a las que se les asignarán los valores.

```
-- Agregando de otra forma los registro de tablas
insert into USUARIO(cod_user, nam_user, status_user)
    values(8, 'EPACHES', '1')

insert into USUARIO(cod_user, nam_user, status_user)
    values(9, 'NOTORIBY', '1')
```

- Se crea una nueva tabla llamada "USUARIOUCV" con tres columnas: "cod\_UCVuser" (entero), "nom\_UCVuser" (cadena de caracteres) y "status\_UCVuser" (carácter). La columna "cod\_UCVuser" se establece como clave primaria. Se realiza una inserción múltiple en la tabla "USUARIOUCV" seleccionando los valores de las columnas "cod\_user", "nam\_user" y "status\_user" de la tabla "USUARIO" donde el "status\_user" es igual a '1'.

```
-- Insercción Múltiple
insert into USUARIOUCV(cod_UCVuser, nom_UCVuser, status_UCVuser)
    select cod_user, nam_user, status_user
    from USUARIO
    where status_user = '1'
go
```

- Se eliminan todos los registros de la tabla "USUARIOUCV".
- Se elimina un registro específico de la tabla "USUARIO" utilizando la cláusula WHERE para filtrar por el valor de la columna "cod\_user".

```
--Eliminar registros
DELETE FROM USUARIOUCV

--Eliminar registros especificos WHERE
delete from USUARIO where cod_user=3
select * from USUARIO
```

- Se actualiza el valor de la columna "status\_user" en la tabla "USUARIO" a '0' (inactivo) para el registro donde el "cod\_user" es igual a 5.
- Se realiza una consulta a la tabla "USUARIO" para mostrar todos los registros después de la actualización.
- Se actualiza el valor de la columna "status\_UCVuser" en la tabla "USUARIOUCV" a '0' (inactivo) para todos los registros.

```
--ACTUALIZAR EL ESTADO DEL USUARIO
UPDATE USUARIO
  set status_user = '0' --inactivo
  WHERE cod_user=5
select * from usuario
```

- Se realiza una consulta a la tabla "USUARIOUCV" para mostrar todos los registros después de la actualización.

```
UPDATE USUARIOUCV
  set status_UCVuser='0'
select * from USUARIOUCV
```

- Se actualiza el valor de la columna "status\_user" en la tabla "USUARIO" a '0' (inactivo) para el registro donde el "cod\_user" es igual a 9.
- Se realiza una consulta a la tabla "USUARIO" para mostrar solo los registros con "status\_user" igual a '1' (activos).
- Se realiza una consulta a la tabla "USUARIO" para mostrar solo los registros con "status\_user" igual a '0' (inactivos).

```
UPDATE USUARIO
  set status_user = '0' --inactivo
  WHERE cod_user = 9
select * from usuario where status_user='1'-- ACTIVOS
select * from usuario where status_user='0'-- INACTIVOS
```

## 2.1 Create database bd\_escuelaucv

- Se utiliza la declaración "if DB\_ID('BD\_ESCUELAUCV') is not null" para verificar la existencia de una base de datos llamada "BD\_ESCUELAUCV". La función DB\_ID() devuelve el identificador de la base de datos si existe y no es NULL. Si la base de datos existe, se ejecuta el comando "drop database bd\_escuelaucv" para eliminarla.
- Una vez eliminada la base de datos existente (si la hay), se crea una nueva base de datos llamada "bd\_escuelaucv" mediante el comando "create database bd\_escuelaucv". Esto crea una base de datos vacía con el nombre

especificado.

```
SQLQuery2.sql - YB-...scuelaucv (sa (51))*
use master
go
---eliminando la bd si existe
if DB_ID('BD_ESCUELAUCV') is not null
    drop database bd_escuelaucv
go
--crear bd
create database bd_escuelaucv
go
---activando la bd
use bd_escuelaucv
go
```

- Se procede a crear varias tablas dentro de la base de datos "bd\_escuelaucv" utilizando el comando "create table". Cada tabla tiene un nombre y una lista de columnas con sus respectivos tipos de datos y restricciones.

- La tabla "curso" almacena información sobre los cursos, como su identificador, nombre, horas teóricas, horas prácticas, nivel y grado.

```
---crear las tablas
create table curso
( idcurso char(5) not null primary key,
  nombrecurso varchar(15) not null,
  horasteoricas numeric not null,
  horaspracticas numeric not null,
  nivel char(1) not null,
  grado char(1) not null
)
go
```

- La tabla "docente" almacena información sobre los docentes, incluyendo su identificador, nombre, apellidos, dirección, teléfono, DNI, especialidad, correo electrónico y sexo.



```
create table docente
( iddocente char(5) not null primary key,
  nombre varchar(25) not null,
  apellidos varchar(35) not null,
  direccion varchar(50) not null,
  telefono varchar(12),
  DNI varchar(8) not null,
  especialidad varchar(8) not null,
  e_mail varchar(50),
  sexo char(1) not null
)
go
```

- La tabla "ubigeo" guarda información sobre los lugares geográficos, como identificador, distrito, provincia y departamento.

```
create table ubigeo
( idubigeo char(6) not null primary key,
  distrito varchar(35) not null,
  provincia varchar(25) not null,
  departamento varchar(35) not null
)
go
```

- La tabla "alumno" almacena información sobre los alumnos, incluyendo su identificador, nombre, apellidos, fecha de nacimiento, teléfono, sexo, correo electrónico y el identificador del ubigeo al que pertenecen.

```
create table alumno
( idalumno char(5) not null primary key,
  nombre varchar(25) not null,
  apellidos varchar(35) not null,
  fechanac datetime not null,
  telefono varchar(12),
  sexo char(1) not null,
  e_mail varchar(50),
  idubigeo char(6) not null references ubigeo
)
go
```

- La tabla "promedio" registra los promedios de los

alumnos en los cursos, con su identificador, el identificador del curso y el promedio.

```
create table promedio
( idalumno char(5) not null references alumno,
  idcurso char(5) not null references curso,
  promedio real,
  primary key(idalumno,idcurso)
)
go
```

- La tabla "asignacion" guarda información sobre la asignación de docentes a cursos, con el identificador del docente, el identificador del curso y la sección.

```
create table asignacion
( iddocente char(5) not null references docente,
  idcurso char(5) not null references curso,
  seccion char(1) not null,
  primary key(iddocente,idcurso)
)
go
```

- La tabla "notas" almacena las notas de los alumnos en los cursos, con el identificador del curso, el identificador del alumno, las notas de cada bimestre y el promedio.

```
create table notas
( idcurso char(5) not null references curso,
  idalumno char(5) not null references alumno,
  b1 float not null,
  b2 float not null,
  b3 float not null,
  b4 float not null,
  promedio float not null,
  primary key(idcurso,idalumno)
)
go
```

- Después de crear las tablas, se aplican restricciones adicionales utilizando las declaraciones "ALTER TABLE". Estas restricciones incluyen valores predeterminados (DEFAULT) y restricciones de verificación (CHECK) para

garantizar la integridad de los datos.

```
SQLQuery2.sql - YB-...scuelaucv (sa (51)) *  X
-----RESTRICCIONES-----
--RESTRICCION POR DEFAULT: PREDETERMINADO
--ASIGNAR EL VALOR CERO AL B1,B2,B3,B4 DE LA TABLA NOTA
ALTER TABLE NOTAS
    ADD CONSTRAINT DF_NOTAB1 DEFAULT 0
    FOR B1

ALTER TABLE NOTAS
    ADD CONSTRAINT DF_NOTAB2 DEFAULT 0
    FOR B2

ALTER TABLE NOTAS
    ADD CONSTRAINT DF_NOTAB3 DEFAULT 0
    FOR B3

ALTER TABLE NOTAS
    ADD CONSTRAINT DF_NOTAB4 DEFAULT 0
    FOR B4

ALTER TABLE NOTAS
    ADD CONSTRAINT DF_PROMEDIO DEFAULT 0
    FOR PROMEDIO
```

- También se aplican restricciones de unicidad (UNIQUE) en ciertos campos para asegurar que los valores sean únicos en las respectivas tablas.

```
--ASIGNAR EL VALOR CERO AL CAMPO HORAS TEORICAS DE LA TABLA CURSO
ALTER TABLE CURSO
    ADD CONSTRAINT DF_HT DEFAULT 0
    FOR HORASTEORICAS
GO

--ASIGNAR EL VALOR CERO AL CAMPO HORAS PRACTICAS DE LA TABLA CURSO
ALTER TABLE CURSO
    ADD CONSTRAINT DF_HP DEFAULT 0
    FOR HORASPRACTICAS
GO

--ASIGNAR EL VALOR "NO REGISTRA" AL CAMPO EMAIL DE LA TABLA DOCENTE
ALTER TABLE DOCENTE
    ADD CONSTRAINT DF_EMAIL DEFAULT 'NO REGISTRA'
    FOR E_MAIL
GO

---->RESTRICCION CHECK
--EL CAMPO SEXO DE LA TABLA ALUMNO Y DOCENTE DEBE PERMITIR VALORES F Y M
ALTER TABLE ALUMNO
    ADD CONSTRAINT CHK_SEXO CHECK(SEXO LIKE '[FM]')
GO
```



```
--EL CAMPO SEXO DE LA TABLA DOCENTE Y DOCENTE DEBE PERMITIR VALORES F Y M
ALTER TABLE DOCENTE
ADD CONSTRAINT CHK_SEXODOC CHECK(SEXO LIKE '[FM]')
GO

--EL CODIGO DEL ALUMNO DEBE COMENZAR CON LA LETRA A
ALTER TABLE ALUMNO
ADD CONSTRAINT CHK_IDA CHECK (IDALUMNO LIKE 'A[0-9][0-9][0-9][0-9]')
GO

ALTER TABLE DOCENTE
DROP CONSTRAINT DF_EMAIL

ALTER TABLE DOCENTE
ADD CONSTRAINT DF_EMAIL DEFAULT 'NO REGISTRA'
FOR E_MAIL
GO

--EL CODIGO DEL DOCENTE DEBE COMENZAR CON LA LETRA D
ALTER TABLE DOCENTE
ADD CONSTRAINT CHK_IDD CHECK (IDDOCENTE LIKE 'D[0-9][0-9][0-9][0-9]')
GO
```

```
--EL CODIGO DEL CURSO DEBE COMENZAR CON LA LETRA C
ALTER TABLE CURSO
ADD CONSTRAINT CHK_IDC CHECK (IDCURSO LIKE 'C[0-9][0-9][0-9][0-9]')
GO

--DEBE ACEPTAR VALOR MAYORES O IGUALES A CERO
ALTER TABLE CURSO
ADD CONSTRAINT CHK_CHT CHECK (HORASTEORICAS>=0)
GO

--EL CAMPO GRADO DEBE ACEPTAR VALORES MAYORES ENTRE 1 Y 6
ALTER TABLE CURSO
ADD CONSTRAINT CHK_GRADO CHECK (GRADO LIKE '[1-6]')
GO

--EL CAMPO NIVEL DEBE ACEPTAR VALOR COMO P Y S
ALTER TABLE CURSO
ADD CONSTRAINT CHK_CNIVEL CHECK (NIVEL IN('P','S'))
GO
```

```
GO
--LOS CAMPOS B1 A B4 Y PROMEDIO DE LA TABLA NOTAS DEBE ACEPTAR
--VALORES DE 0 A 20
ALTER TABLE NOTAS
ADD CONSTRAINT CHK_NB1 CHECK (B1>=0 AND B1<=20),
CONSTRAINT CHK_NB2 CHECK (B2>=0 AND B2<=20),
CONSTRAINT CHK_NB3 CHECK (B3>=0 AND B3<=20),
CONSTRAINT CHK_NB4 CHECK (B4>=0 AND B4<=20),
CONSTRAINT CHK_NPROM CHECK (PROMEDIO>=0 AND PROMEDIO<=20)
GO
---APLICANDO UNIQUE
--EL CAMPO EMAIL DE LA TABLA DOCENTE DEBE SER UNICO
ALTER TABLE DOCENTE
ADD CONSTRAINT UQ_EMA
UNIQUE(E_MAIL)
GO
--EL CAMPO DIRECCION DE LA TABLA DOCENTE DEBE SER UNICO
ALTER TABLE DOCENTE
ADD CONSTRAINT UQ_DIRECCION
UNIQUE(DIRECCION)
GO
--EL CAMPO DNI DEBE SER UNICO
ALTER TABLE DOCENTE
ADD CONSTRAINT UQ_DNI
UNIQUE(DNI)
GO
--EL CAMPO DEPARTAMENTO DE LA TABLA UBIGEO DEBE SER UNICO
ALTER TABLE UBIGEO
ADD CONSTRAINT UQ_DEPA
UNIQUE(DEPARTAMENTO)
GO
```

- Por último, se realizan algunas inserciones de datos de ejemplo en la tabla "curso" utilizando la declaración "insert into". Se agregan dos registros, uno con todos los valores de columna especificados y otro con solo algunos valores. Estos registros representan cursos con identificadores "C0001" y "C0002" respectivamente.

```
GO
-----
insert into curso values ('C0001','java',20,30,'P',5)
select * from curso
insert into curso (idcurso,nombrecurso,nivel,grado)
values ('C0002','php','S',3)
```

## 2.2 Create database UCVBDNEGOCIO2022

- Crear la base de datos: La primera línea del script crea una base de datos llamada "UCVBDNEGOCIOS2022".

- Usar la base de datos: La línea "use UCVBDNEGOCIOS2022" establece la base de datos recién creada como la base de datos actual para las operaciones posteriores.
- Establecer el formato de fecha: La línea "set dateformat ymd" establece el formato de fecha para el sistema en "año-mes-día".
- Crear esquemas: Las líneas que comienzan con "Create Schema" crean tres esquemas en la base de datos: Ventas, Compras y RRHH. Los esquemas son contenedores lógicos que se utilizan para organizar y agrupar objetos relacionados en una base de datos.

```
SQLQuery3.sql - YB...CIOS2022 (sa (67))* X SQLQuery3
create database UCVBDNEGOCIOS2022
go

use UCVBDNEGOCIOS2022
go

set dateformat ymd
go

Create Schema Ventas
go
Create Schema Compras
go
Create Schema RRHH
go
```

- 5. Crear tablas: A continuación, se crean varias tablas en los esquemas creados anteriormente. Cada tabla tiene columnas que definen su estructura y restricciones.

- La tabla "Compras.categorias" tiene columnas para el ID de categoría, el nombre de la categoría y la descripción.

```
CREATE TABLE Compras.categorias (  
    IdCategoria int primary key ,  
    NombreCategoria varchar(15) not null,  
    Descripcion text  
)  
go  
  
INSERT INTO Compras.categorias VALUES(1, 'Bebidas', 'Gaseosas, cafe, te, cervezas y maltas')  
INSERT INTO Compras.categorias VALUES(2, 'Condimentos', 'Salsas dulces y picantes, delicias, comida para untar y aderezos')  
INSERT INTO Compras.categorias VALUES(3, 'Reposteria', 'Postres, dulces y pan dulce')  
INSERT INTO Compras.categorias VALUES(4, 'Lacteos', 'Quesos')  
INSERT INTO Compras.categorias VALUES(5, 'Granos/Cereales', 'Pan, galletas, pasta y cereales')  
INSERT INTO Compras.categorias VALUES(6, 'Carnes', 'Carnes preparadas')  
INSERT INTO Compras.categorias VALUES(7, 'Frutas/Verduras', 'Frutas secas y queso de soja')  
INSERT INTO Compras.categorias VALUES(8, 'Pescado/Marisco', 'Pescados, mariscos y algas')  
go
```

- La tabla "Ventas.países" tiene columnas para el ID de país y el nombre del país.

```
CREATE TABLE Ventas.países (  
    Idpais char(3) primary key,  
    NombrePais varchar(40) not null  
)  
go  
  
INSERT INTO Ventas.países VALUES('001', 'Peru')  
INSERT INTO Ventas.países VALUES('002', 'Argentina')  
INSERT INTO Ventas.países VALUES('003', 'Chile')  
INSERT INTO Ventas.países VALUES('004', 'USA')  
INSERT INTO Ventas.países VALUES('005', 'España')  
INSERT INTO Ventas.países VALUES('006', 'Francia')  
INSERT INTO Ventas.países VALUES('007', 'Colombia')  
INSERT INTO Ventas.países VALUES('008', 'Canada')  
INSERT INTO Ventas.países VALUES('009', 'China')  
go
```

- La tabla "Ventas.clientes" tiene columnas para el ID de cliente, el nombre del cliente, la dirección, el ID de país y el teléfono del cliente.

```

CREATE TABLE Ventas.clientes (
    IdCliente varchar(5) primary key,
    NomCliente varchar(40) not null,
    DirCliente varchar(60) not null,
    idpais char(3) references Ventas.paises,
    fonoCliente varchar(25) not null
)
go

INSERT INTO Ventas.clientes VALUES('ALFKI', 'Alfreds Futterkiste', 'Obere Str. 57', '002', '030-0074321')
INSERT INTO Ventas.clientes VALUES('ANATR', 'Ana Trujillo Emparedados y helados', 'Avda. de la Constitucion 2222', '005', '(5) 555-4729')
INSERT INTO Ventas.clientes VALUES('ANTON', 'Antonio Moreno Taqueria', 'Mataderos 2312', '007', '(5) 555-3932')
INSERT INTO Ventas.clientes VALUES('AROUT', 'Around the Horn', '120 Hanover Sq.', '004', '(71) 555-7788')
INSERT INTO Ventas.clientes VALUES('BERGS', 'Berglunds snabbköp', 'Berguvsvägen 8', '006', '0921-12 34 65')
INSERT INTO Ventas.clientes VALUES('BLAUS', 'Blauer See Delikatessen', 'Forsterstr. 57', '001', '0621-08460')
INSERT INTO Ventas.clientes VALUES('BLONP', 'Blondel père et fils', '24, place Kleber Estrasburgo', '008', '88.60.15.31')
INSERT INTO Ventas.clientes VALUES('BOLID', 'Bolido Comidas preparadas', 'C/ Araquil, 67', '009', '(91) 555 91 99')
INSERT INTO Ventas.clientes VALUES('BONAP', 'Bon app', '12, rue des Bouchers', '001', '91.24.45.41')
INSERT INTO Ventas.clientes VALUES('BOTTM', 'Bottom-Dollar Markets', '23 Tsawassen Blvd.', '003', '(604) 555-3745')
INSERT INTO Ventas.clientes VALUES('BSBEV', 'B's Beverages', 'Fauntleroy Circus', '009', '(71) 555-1212')
INSERT INTO Ventas.clientes VALUES('CACTU', 'Cactus Comidas para llevar', 'Cerrito 333', '002', '(1) 135-4892')
INSERT INTO Ventas.clientes VALUES('CENTC', 'Centro comercial Moctezuma', 'Sierras de Granada 9993', '008', '(5) 555-7293')
INSERT INTO Ventas.clientes VALUES('CHOPS', 'Chop-suey Chinese', 'Hauptstr. 29', '001', '')
INSERT INTO Ventas.clientes VALUES('COMMI', 'Comercio Mineiro', 'Av. dos Lusíadas, 23', '004', '')
INSERT INTO Ventas.clientes VALUES('CONSH', 'Consolidated Holdings', 'Berkeley Gardens\r\n12 Brewery', '005', '(71) 555-2282')
INSERT INTO Ventas.clientes VALUES('DRACD', 'Drachenblut Delikatessen', 'Walserweg 21', '006', '0241-059428')
INSERT INTO Ventas.clientes VALUES('DUMON', 'Du monde entier', '67, rue des Cinquante Otages', '007', '40.67.89.89')
INSERT INTO Ventas.clientes VALUES('EASTC', 'Eastern Connection', '35 King George', '005', '(71) 555-3373')
INSERT INTO Ventas.clientes VALUES('ERNSH', 'Ernst Handel', 'Kirchgasse 6', '001', '7675-3426')
INSERT INTO Ventas.clientes VALUES('FAMTA', 'Familia Arquibaldo', 'Rua dos 92', '004', '(11) 555-9857')

```

- La tabla "Compras.proveedores" tiene columnas para el ID de proveedor, el nombre del proveedor, la dirección, el nombre y cargo del contacto, el ID de país, el teléfono y el fax del proveedor.

```

CREATE TABLE Compras.proveedores (
    IdProveedor int primary key,
    NomProveedor varchar(40) not null,
    DirProveedor varchar(60) not null,
    NomContacto varchar(80) not null,
    CargoContacto varchar(50) not null,
    idpais char(3) references Ventas.paises,
    fonoProveedor varchar(15) not null,
    FaxProveedor varchar(15) not null
)
go

INSERT INTO Compras.proveedores VALUES('1', 'Exotic Liquids', 'Charlotte Cooper', 'Gerente de compras', '49 Gilbert St.', '003', '(171) 555-2222',
INSERT INTO Compras.proveedores VALUES('2', 'New Orleans Cajun Delights', 'Shelley Burke', 'Administrador de pedidos', 'P.O. Box 78934', '008', '(10) 555-8576')
INSERT INTO Compras.proveedores VALUES('3', 'Grandma Kellys Homestead', 'Regina Murphy', 'Representante de ventas', '707 Oxford Rd.', '001', '(313) 555-5796')
INSERT INTO Compras.proveedores VALUES('4', 'Tokyo Traders', 'Yoshi Nagase', 'Gerente de marketing', '9-8 Sekimai\r\nMusashino-shi', '007', '(03) 3108 4772')
INSERT INTO Compras.proveedores VALUES('5', 'Cooperativa de Quesos Las Cabras', 'Antonio del Valle Saavedra', 'Administrador de exportaciones', 'Ca. 1000', '52-1-5052301')
INSERT INTO Compras.proveedores VALUES('6', 'Mayumis', 'Mayumi Ohno', 'Representante de marketing', '92 Setsuko\r\nChuo-ku', '004', '(06) 431-7877')
INSERT INTO Compras.proveedores VALUES('7', 'Pavlova, Ltd.', 'Ian Devling', 'Gerente de marketing', '74 Rose St.\r\nMoonie Ponds', '008', '(03) 444 3333')
INSERT INTO Compras.proveedores VALUES('8', 'Specialty Biscuits, Ltd.', 'Peter Wilson', 'Representante de ventas', '29 Kings Way', '003', '(161) 555 5555')
INSERT INTO Compras.proveedores VALUES('9', 'PB Knäckebröd AB', 'Lars Peterson', 'Agente de ventas', 'Kaloadagatan 13', '009', '031-987 65 43', '031-987 65 43')
INSERT INTO Compras.proveedores VALUES('10', 'Refrescos Americanas LTDA', 'Carlos Diaz', 'Gerente de marketing', 'Av. das Americanas 12.890', '003', '(11) 555-7890')
INSERT INTO Compras.proveedores VALUES('11', 'Heli Süßwaren GmbH & Co. KG', 'Petra Winkler', 'Gerente de ventas', 'Tiergartenstraße 5', '002', '(011) 555-0909')
INSERT INTO Compras.proveedores VALUES('12', 'Plutzer Lebensmittelgroßmärkte AG', 'Martin Bein', 'Ger. marketing internacional', 'Bogenallee 51', '004', '(49) 555 0044')
INSERT INTO Compras.proveedores VALUES('13', 'Nord-Ost-Fisch Handelsgesellschaft mbH', 'Sven Petersen', 'Coordinador de mercados', 'Frahmredder 112a', '004', '(49) 555 0044')
INSERT INTO Compras.proveedores VALUES('14', 'Formaggi Fortini s.r.l.', 'Elio Rossi', 'Representante de ventas', 'Viale Dante, 75', '006', '(0544) 054400')
INSERT INTO Compras.proveedores VALUES('15', 'Norske Meierier', 'Beate Vileid', 'Gerente de marketing', 'Hatleveggen 5', '006', '(02) 953010', '')
INSERT INTO Compras.proveedores VALUES('16', 'Bigfoot Breweries', 'Cheryl Saylor', 'Repr. de cuentas regional', '3400 - 8th Avenue\r\nSuite 210', '004', '(416) 555-5555')
INSERT INTO Compras.proveedores VALUES('17', 'Svensk Sjöföda AB', 'Michael Björn', 'Representante de ventas', 'Brovallavägen 231', '005', '08-123 4567')

```

- La tabla "Compras.productos" tiene columnas para el ID de producto, el nombre del producto, el ID de proveedor, el ID de categoría, la cantidad por unidad, el precio por unidad, las unidades en existencia y las unidades en pedido.



```

CREATE TABLE Compras.productos (
    IdProducto int primary key,
    NomProducto varchar(40) not null,
    IdProveedor int References Compras.proveedores,
    IdCategoria int References Compras.categorias,
    CantxUnidad varchar(20) not null,
    PrecioUnidad decimal(10,0) not null,
    UnidadesEnExistencia smallint not null,
    UnidadesEnPedido smallint not null)
go

INSERT INTO Compras.productos VALUES('1', 'Te Dharamsala', '1', '1', '10 cajas x 20 bolsas', '18', '39', '0')
INSERT INTO Compras.productos VALUES('2', 'Cerveza tibetana Barley', '1', '1', '24 - bot. 12 l', '19', '17', '40')
INSERT INTO Compras.productos VALUES('3', 'Siropo de regaliz', '1', '2', '12 - bot. 550 ml', '10', '13', '70')
INSERT INTO Compras.productos VALUES('4', 'Especias Cajun del chef Anton', '2', '2', '48 - frascos 6 l', '22', '53', '0')
INSERT INTO Compras.productos VALUES('5', 'Mezcla Gumbo del chef Anton', '2', '2', '36 cajas', '21', '0', '0')
INSERT INTO Compras.productos VALUES('6', 'Mermelada de grosellas de la abuela', '3', '2', '12 - frascos 8 l', '25', '120', '0')
INSERT INTO Compras.productos VALUES('7', 'Peras secas organicas del tio Bob', '3', '7', '12 - paq. 1 kg', '30', '15', '0')
INSERT INTO Compras.productos VALUES('8', 'Salsa de arandanos Northwoods', '3', '2', '12 - frascos 12 l', '40', '6', '0')
INSERT INTO Compras.productos VALUES('9', 'Buey Mishi Kobe', '4', '6', '18 - paq. 500 g', '97', '29', '0')
INSERT INTO Compras.productos VALUES('10', 'Pez espada', '4', '8', '12 - frascos 200 ml', '31', '31', '0')
INSERT INTO Compras.productos VALUES('11', 'Queso Cabrales', '5', '4', 'paq. 1 kg', '21', '22', '30')
INSERT INTO Compras.productos VALUES('12', 'Queso Manchego La Pastora', '5', '4', '10 - paq. 500 g', '38', '86', '0')
INSERT INTO Compras.productos VALUES('13', 'Algas Konbu', '6', '8', 'caja 2 kg', '6', '24', '0')
INSERT INTO Compras.productos VALUES('14', 'Cuajada de judias', '6', '7', '40 - paq. 100 g', '23', '35', '0')
INSERT INTO Compras.productos VALUES('15', 'Salsa de soja baja en sodio', '6', '2', '24 - bot. 250 ml', '15', '39', '0')
INSERT INTO Compras.productos VALUES('16', 'Postre de merengue Pavlova', '7', '3', '32 - cajas 500 g', '17', '29', '0')

```

- La tabla "RRHH.Cargos" tiene columnas para el ID de cargo y la descripción del cargo.
- La tabla "RRHH.Distritos" tiene columnas para el ID de distrito y el nombre del distrito.
- La tabla "RRHH.empleados" tiene columnas para el ID de empleado, el apellido, el nombre, la fecha de nacimiento, la dirección, el ID de distrito, el teléfono, el ID de cargo y la fecha de contratación.

```

CREATE TABLE RRHH.Cargos(
    idcargo int primary key,
    desCargo varchar(30) not null
)
go

insert RRHH.Cargos values(1,'Representante de Ventas')
insert RRHH.Cargos values(2,'Ejecutivo de Ventas')
insert RRHH.Cargos values(3,'Supervisor de Ventas')
insert RRHH.Cargos values(4,'Auxiliar de Ventas')
go

CREATE TABLE RRHH.Distritos(
    idDistrito int primary key,
    nomDistrito varchar(50) not null
)
go

insert RRHH.Distritos values(1,'Lima')
insert RRHH.Distritos values(2,'Rimac')
insert RRHH.Distritos values(3,'Ate')
insert RRHH.Distritos values(4,'San Miguel')
go

```

```

CREATE TABLE RRHH.empleados (
    IdEmpleado int primary key,
    ApeEmpleado varchar(50) not null,
    NomEmpleado varchar(50) not null,
    FecNac datetime not null,
    DirEmpleado varchar(60) not null,
    idDistrito int references RRHH.Distritos,
    fonoEmpleado varchar(15) NULL,
    idCargo int references RRHH.Cargos,
    FecContrata datetime not null
)
go

INSERT INTO RRHH.empleados VALUES(1, 'Davolio', 'Nancy', '1968-12-08 00:00:00', 'Calle Las Magnolias 123', 2, '6573344', 4, '2010-02-15')
INSERT INTO RRHH.empleados VALUES(2, 'Fuller', 'Andrew', '1952-02-19 00:00:00', 'Av Abancay 234', 1, '98788766', 1, '2011-02-09')
INSERT INTO RRHH.empleados VALUES(3, 'Leverling', 'Janet', '1963-08-30 00:00:00', 'Av. Riva Agüero 233', 4, '5664555', 3, '2011-03-10')
INSERT INTO RRHH.empleados VALUES(4, 'Peacock', 'Margaret', '1958-09-19 00:00:00', 'Calle las flores 411', 2, '980523344', 1, '2010-01-01')
INSERT INTO RRHH.empleados VALUES(5, 'Buchanan', 'Steven', '1955-03-04 00:00:00', 'Av. Brasil 222', 3, '6776566', 3, '2011-04-05')
INSERT INTO RRHH.empleados VALUES(6, 'Suyama', 'Michael', '1963-07-02 00:00:00', 'Calle Zafiro 344', 1, '8997877', 4, '2011-04-10')
INSERT INTO RRHH.empleados VALUES(7, 'King', 'Robert', '1960-05-29 00:00:00', 'Jr Puni 322', 1, '99876677', 2, '2011-05-04')
INSERT INTO RRHH.empleados VALUES(8, 'Callahan', 'Laura', '1958-01-09 00:00:00', 'Dinthilac 123', 2, '90098999', 4, '2011-06-10')
INSERT INTO RRHH.empleados VALUES(9, 'Dodsworth', 'Anne', '1969-07-02 00:00:00', 'Av Los fresnos 2334', 3, '98877888', 1, '2011-07-15')
go

```

- Insertar datos en las tablas: Después de crear las tablas, se insertan datos de ejemplo en algunas de ellas utilizando la declaración "INSERT INTO". Se agregan filas a las tablas "Compras.categorias", "Ventas.países", "Ventas.clientes", "Compras.proveedores", "Compras.productos", "RRHH.Cargos" y "RRHH.empleados".
- Crear la tabla "Ventas.pedidoscabe": Se crea una nueva tabla llamada "Ventas.pedidoscabe" con varias columnas para almacenar información sobre los pedidos, como el ID de pedido, el ID de cliente, el ID de empleado, las fechas de pedido, entrega y envío, la cantidad de pedido, el destinatario, la dirección de entrega y el estado del pedido.

```

CREATE TABLE [Ventas].[pedidoscabe](
    [IdPedido] [int] NOT NULL,
    [IdCliente] [varchar](5) NULL,
    [IdEmpleado] [int] NULL,
    [FechaPedido] [datetime] NOT NULL,
    [FechaEntrega] [datetime] NULL,
    [FechaEnvio] [datetime] NULL,
    [EnvioPedido] [char](1) NULL,
    [CantidaPedido] [int] NULL,
    [Destinatario] [varchar](40) NULL,
    [DirDestinatario] [varchar](60) NULL,
    [CiuDestinatario] [varchar](60) NULL,
    [RefDestinatario] [varchar](60) NULL,
    [DepDestinatario] [varchar](60) NULL,
    [PaiDestinatario] [varchar](60) NULL,
    PRIMARY KEY CLUSTERED
    (
        [IdPedido] ASC
    )
    WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

```

```

SET ANSI_PADDING OFF
GO

ALTER TABLE [Ventas].[pedidoscabe] WITH CHECK ADD FOREIGN KEY([IdCliente])
REFERENCES [Ventas].[clientes] ([IdCliente])
GO

ALTER TABLE [Ventas].[pedidoscabe] WITH CHECK ADD FOREIGN KEY([IdEmpleado])
REFERENCES [RRHH].[empleados] ([IdEmpleado])
GO

ALTER TABLE [Ventas].[pedidoscabe] ADD DEFAULT (getdate()) FOR [FechaPedido]
GO

ALTER TABLE [Ventas].[pedidoscabe] ADD DEFAULT ('0') FOR [EnvioPedido]
GO

```

- Crear la tabla "Ventas.pedidosdetalle": Se crea otra tabla llamada "Ventas.pedidosdetalle" con columnas para el ID de pedido, el ID de producto, la cantidad de producto y el precio unitario.

```

CREATE TABLE [Ventas].[pedidosdeta](
    [IdPedido] [int] NULL,
    [IdProducto] [int] NULL,
    [PrecioUnidad] [decimal](10, 0) NOT NULL,
    [Cantidad] [smallint] NOT NULL,
    [Descuento] [float] NOT NULL
) ON [PRIMARY]
GO

ALTER TABLE [Ventas].[pedidosdeta] WITH CHECK ADD FOREIGN KEY([IdPedido])
REFERENCES [Ventas].[pedidoscabe] ([IdPedido])
GO

ALTER TABLE [Ventas].[pedidosdeta] WITH CHECK ADD FOREIGN KEY([IdProducto])
REFERENCES [Compras].[productos] ([IdProducto])
GO

```

```

INSERT INTO ventas.pedidoscabe VALUES('10248', 'WILMK', '5', '1996-07-04 00:00:00', '1996-08-01 00:00:00', '1996-07-16 00:00:00', '1', '32', 'Wilman Kala', 'Keskuskatu')
INSERT INTO ventas.pedidoscabe VALUES('10249', 'TOMSP', '6', '1996-07-05 00:00:00', '1996-08-16 00:00:00', '1996-07-10 00:00:00', '1', '11', 'Toms Spezialitäten', 'Luis')
INSERT INTO ventas.pedidoscabe VALUES('10250', 'HANAR', '4', '1996-07-08 00:00:00', '1996-08-05 00:00:00', '1996-07-12 00:00:00', '1', '65', 'Hanari Carnes', 'Rua do Pa')
INSERT INTO ventas.pedidoscabe VALUES('10251', 'VICTE', '3', '1996-07-08 00:00:00', '1996-08-05 00:00:00', '1996-07-15 00:00:00', '1', '41', 'Victuailles en stock', '2')
INSERT INTO ventas.pedidoscabe VALUES('10252', 'SUPRD', '4', '1996-07-09 00:00:00', '1996-08-06 00:00:00', '1996-07-11 00:00:00', '1', '51', 'Suprêmes delicés', 'Boulev')
INSERT INTO ventas.pedidoscabe VALUES('10253', 'HANAR', '3', '1996-07-10 00:00:00', '1996-07-24 00:00:00', '1996-07-16 00:00:00', '1', '58', 'Hanari Carnes', 'Rua do Pa')
INSERT INTO ventas.pedidoscabe VALUES('10254', 'CHOPS', '5', '1996-07-11 00:00:00', '1996-08-08 00:00:00', '1996-07-23 00:00:00', '1', '22', 'Chop-suey Chinese', 'Haupt')
INSERT INTO ventas.pedidoscabe VALUES('10255', 'RICSU', '9', '1996-07-12 00:00:00', '1996-08-09 00:00:00', '1996-07-15 00:00:00', '1', '148', 'Richter Supermarkt', 'Sta')
INSERT INTO ventas.pedidoscabe VALUES('10256', 'WELLI', '3', '1996-07-15 00:00:00', '1996-08-12 00:00:00', '1996-07-17 00:00:00', '1', '13', 'Wellington Importadora', '1')
INSERT INTO ventas.pedidoscabe VALUES('10257', 'HILAA', '4', '1996-07-16 00:00:00', '1996-08-13 00:00:00', '1996-07-22 00:00:00', '1', '81', 'HILARIO Abastos', 'Carren')
INSERT INTO ventas.pedidoscabe VALUES('10258', 'ERNSH', '1', '1996-07-17 00:00:00', '1996-08-14 00:00:00', '1996-07-23 00:00:00', '1', '140', 'Ernst Handel', 'Kirchgass')
INSERT INTO ventas.pedidoscabe VALUES('10259', 'CENTC', '4', '1996-07-18 00:00:00', '1996-08-15 00:00:00', '1996-07-25 00:00:00', '1', '3', 'Centro comercial Moctezuma', '1')
INSERT INTO ventas.pedidoscabe VALUES('10260', 'OTTIK', '4', '1996-07-19 00:00:00', '1996-08-16 00:00:00', '1996-07-29 00:00:00', '1', '55', 'Ottilies Käseladen', 'Mehn')
INSERT INTO ventas.pedidoscabe VALUES('10261', 'QUEDE', '4', '1996-07-19 00:00:00', '1996-08-16 00:00:00', '1996-07-30 00:00:00', '1', '3', 'Que Delicia', 'Rua da Panif')
INSERT INTO ventas.pedidoscabe VALUES('10262', 'RATTCT', '8', '1996-07-22 00:00:00', '1996-08-19 00:00:00', '1996-07-25 00:00:00', '1', '48', 'Rattlesnake Canyon Grocery', '1')
INSERT INTO ventas.pedidoscabe VALUES('10263', 'ERNSH', '9', '1996-07-23 00:00:00', '1996-08-20 00:00:00', '1996-07-31 00:00:00', '1', '146', 'Ernst Handel', 'Kirchgass')
INSERT INTO ventas.pedidoscabe VALUES('10264', 'FOLKO', '6', '1996-07-24 00:00:00', '1996-08-21 00:00:00', '1996-08-23 00:00:00', '1', '3', 'Folk och få HB', 'Åkerгатan')
INSERT INTO ventas.pedidoscabe VALUES('10265', 'BLONP', '2', '1996-07-25 00:00:00', '1996-08-22 00:00:00', '1996-08-12 00:00:00', '1', '55', 'Blondel père et fils', '24')
INSERT INTO ventas.pedidoscabe VALUES('10266', 'WARTH', '3', '1996-07-26 00:00:00', '1996-08-06 00:00:00', '1996-07-31 00:00:00', '1', '25', 'Wartian Herkku', 'Torikatu')
INSERT INTO ventas.pedidoscabe VALUES('10267', 'FRANK', '4', '1996-07-29 00:00:00', '1996-08-26 00:00:00', '1996-08-06 00:00:00', '1', '208', 'Frankenversand', 'Berline')
INSERT INTO ventas.pedidoscabe VALUES('10268', 'GROSR', '8', '1996-07-30 00:00:00', '1996-08-27 00:00:00', '1996-08-02 00:00:00', '1', '66', 'GROSELLA-Restaurante', '59')
INSERT INTO ventas.pedidoscabe VALUES('10269', 'WHITC', '5', '1996-07-31 00:00:00', '1996-08-14 00:00:00', '1996-08-09 00:00:00', '1', '4', 'White Clover Markets', '102')
INSERT INTO ventas.pedidoscabe VALUES('10270', 'WARTH', '1', '1996-08-01 00:00:00', '1996-08-29 00:00:00', '1996-08-02 00:00:00', '1', '136', 'Wartian Herkku', 'Torikat')
INSERT INTO ventas.pedidoscabe VALUES('10271', 'SPLIR', '6', '1996-08-01 00:00:00', '1996-08-29 00:00:00', '1996-08-30 00:00:00', '1', '4', 'Split Rail Beer & Ale', 'P')
INSERT INTO ventas.pedidoscabe VALUES('10272', 'RATTCT', '6', '1996-08-02 00:00:00', '1996-08-30 00:00:00', '1996-08-06 00:00:00', '1', '98', 'Rattlesnake Canyon Grocery', '1')

```

```
INSERT INTO ventas.pedidosdeta VALUES('11075', '46', '12', '30', '0.15')
INSERT INTO ventas.pedidosdeta VALUES('11075', '76', '18', '2', '0.15')
INSERT INTO ventas.pedidosdeta VALUES('11076', '6', '25', '20', '0.25')
INSERT INTO ventas.pedidosdeta VALUES('11076', '14', '23', '20', '0.25')
INSERT INTO ventas.pedidosdeta VALUES('11076', '19', '9', '10', '0.25')
INSERT INTO ventas.pedidosdeta VALUES('11077', '2', '19', '24', '0.2')
INSERT INTO ventas.pedidosdeta VALUES('11077', '3', '10', '4', '0')
INSERT INTO ventas.pedidosdeta VALUES('11077', '4', '22', '1', '0')
INSERT INTO ventas.pedidosdeta VALUES('11077', '6', '25', '1', '0.02')
INSERT INTO ventas.pedidosdeta VALUES('11077', '7', '30', '1', '0.05')
INSERT INTO ventas.pedidosdeta VALUES('11077', '8', '40', '2', '0.1')
INSERT INTO ventas.pedidosdeta VALUES('11077', '10', '31', '1', '0')
INSERT INTO ventas.pedidosdeta VALUES('11077', '12', '38', '2', '0.05')
INSERT INTO ventas.pedidosdeta VALUES('11077', '13', '6', '4', '0')
INSERT INTO ventas.pedidosdeta VALUES('11077', '14', '23', '1', '0.03')
INSERT INTO ventas.pedidosdeta VALUES('11077', '16', '17', '2', '0.03')
INSERT INTO ventas.pedidosdeta VALUES('11077', '20', '81', '1', '0.04')
INSERT INTO ventas.pedidosdeta VALUES('11077', '23', '9', '2', '0')
INSERT INTO ventas.pedidosdeta VALUES('11077', '32', '32', '1', '0')
INSERT INTO ventas.pedidosdeta VALUES('11077', '39', '18', '2', '0.05')
INSERT INTO ventas.pedidosdeta VALUES('11077', '41', '9', '3', '0')
INSERT INTO ventas.pedidosdeta VALUES('11077', '46', '12', '3', '0.02')
INSERT INTO ventas.pedidosdeta VALUES('11077', '52', '7', '2', '0')
INSERT INTO ventas.pedidosdeta VALUES('11077', '55', '24', '2', '0')
INSERT INTO ventas.pedidosdeta VALUES('11077', '60', '34', '2', '0.06')
INSERT INTO ventas.pedidosdeta VALUES('11077', '64', '33', '2', '0.03')
INSERT INTO ventas.pedidosdeta VALUES('11077', '66', '17', '1', '0')
INSERT INTO ventas.pedidosdeta VALUES('11077', '73', '15', '2', '0.01')
INSERT INTO ventas.pedidosdeta VALUES('11077', '75', '7', '4', '0')
INSERT INTO ventas.pedidosdeta VALUES('11077', '77', '13', '2', '0')
go
```

- Utilizando el operador \*: Esta consulta muestra todas las filas y columnas de la tabla "compras.proveedores". El operador \* se utiliza para seleccionar todas las columnas en la salida.
- Especificando columnas: En esta consulta, se seleccionan columnas específicas de la tabla "compras.productos", incluyendo "idproducto", "NomProducto", "CantxUnidad" y "PrecioUnidad". Solo se mostrarán estas columnas en la salida.

```
--1. Utilizando el operador * //Trae todas las filas y columnas
select * from compras.proveedores

--2. Especificando columnas
select idproducto, NomProducto, CantxUnidad, PrecioUnidad
from compras.productos
```

- Utilizar alias: En esta consulta, se utiliza el alias "p" para abreviar

el nombre de la tabla "compras.productos" en la consulta. Esto simplifica la escritura y lectura de la consulta.

```
--3.Utilizar alias
select p.IdProducto,p.NomProducto,p.PrecioUnidad,p.UnidadesEnExistencia
from compras.productos p
```

- Colocar nombres a la cabecera: Esta consulta utiliza alias para asignar nombres personalizados a las columnas en la salida. Por ejemplo, "p.IdProducto as CODIGO" asigna el nombre "CODIGO" a la columna "IdProducto" en la salida resultante.

```
--4. Colocar nombres a la cabecera
select p.IdProducto as CODIGO,
       p.NomProducto as PRODUCTO,
       p.PrecioUnidad as 'PRECIO VENTA',
       p.UnidadesEnExistencia as 'CANTIDAD UNIDADES'
from Compras.productos p
```

- USANDO la función DISTINCT: La cláusula DISTINCT se utiliza para eliminar duplicados en el resultado de la consulta. En estas consultas, se selecciona la columna "IdProveedor" y "IdCategoria" de la tabla "compras.productos" eliminando cualquier duplicado.

```
from Compras.productos p
--5.USANDO la funcion DISTINCT
select DISTINCT p.IdProveedor
from compras.productos p

select DISTINCT p.IdCategoria
from compras.productos p
```

- ORDER BY ASC y DESC: La cláusula ORDER BY se utiliza para ordenar los resultados de la consulta según una columna específica en orden ascendente (ASC) o descendente (DESC). Estas consultas ordenan los resultados de la tabla "compras.productos" por la columna "NomProducto" en orden descendente y ascendente respectivamente.

```
--6.ORDER BY ASC Y DESC
select p.*
from compras.productos p
order by p.NomProducto desc

select p.*
from compras.productos p
order by p.NomProducto asc
```

- CLAUSULA WHERE: Esta consulta selecciona todas las filas de la tabla "compras.productos" donde el valor de la columna "IdCategoria" es igual a 3.

```
--CLAUSULA WHERE
--7.-Seleccionar un codigo de categoria de la tabla producto
select p.*
from compras.productos p
where p.IdCategoria=3
```

- Consulta que muestra todos los productos cuya unidad en existencia es igual a 39.

```
--8.-Consulta que muestre todos los productos que su unidad en existencia sea igual a 39
select p.*
from compras.productos p
where p.UnidadesEnExistencia=39
```

- Consulta que muestra todos los productos cuyo precio por unidad es mayor que 100.

```
--9.-Consulta que muestre todos los productos que el precio de unidad sea mayor que 100
--Operadores relaciones: =,>,<,>=,<=
select p.*
from compras.productos p
where p.PrecioUnidad>100
```

- Consulta que muestra los productos de la categoría 1 y cuya unidad en existencia sea mayor a 50.

```
--Operadores logicos and,or, not
--10 Consulta que muestre la categoria 1 y la unidad de existencia sea mayor a 50
select p.*
from compras.productos p
where p.IdCategoria=1 and p.UnidadesEnExistencia>50
```

- Consulta que muestra los productos de las categorías "lacteos" o "carnes".



```
--11 Consulta que muestre la categoria lacteos o carnes
select * from compras.categorias
select p.*
from compras.productos p
where p.IdCategoria=4 or p.IdCategoria=6
```

- Consulta que muestra los productos cuya unidad en existencia está entre 50 y 100, o cuyo precio está entre 5 y 30.

```
--12. consulta para mostrar todos los productos que su unidad de existencia
--este entre 50 y 100 o su precio este entre 5 y 30
select p.*
from compras.productos p
where (p.UnidadesEnExistencia>=50 and p.UnidadesEnExistencia<= 100)
or (p.preciounidad>=5 and p.preciounidad<=30)
```

- Operador LIKE: Esta consulta muestra los productos cuya descripción comienza con la letra "C", seguida de cualquier letra.

```
--13. operador like
--mostrar todos los productos que su descripcion comience con C,
--seguido de la letra e y el resto cualesquiera
select p.*
from compras.productos p
where p.NomProducto like 'Ce%'
----que la tercera letras sea j(investigar)
```

- Esta consulta muestra los productos cuya descripción tiene la letra "e" en la segunda posición.

```
----que la tercera letras sea j(investigar)
--14. mostrar todos los productos que la descripcion del producto
--tenga la letra e en la segunda posicion
select p.*
from compras.productos p
where p.NomProducto like '_e%'
```

- Consulta que muestra los productos cuyo precio está entre 10 y 25 utilizando el operador BETWEEN.

```
--15. mostrar todos los productos que su precio este entre 10 y 25
--between :recupera registros segun el intervalo de valores un campo
select p.*
from compras.productos p
where p.PrecioUnidad BETWEEN 10 AND 25
```

- Operador IN: Esta consulta muestra los productos cuyo precio por unidad es 30, 35 o 100, utilizando el operador IN para verificar si el precio está en la lista especificada.

```
--16. operador IN
--IN: devuelve aquellos registros cuyo campo indicado coincide con alguno
--de los valores que se encuentra en la lista
--mostrar todos los productos de su precio sea 30,35,100
select p.*
from Compras.productos p
where p.PrecioUnidad IN (30,35 ,100)
```

- Función DAY: Esta consulta utiliza la función DAY para obtener el día correspondiente de la fecha de pedido en la tabla "Ventas.pedidoscabe". También se muestran otras partes de la fecha, como el mes, el año y la fecha completa.

```
--17. funcion DAY
--mostrar el día correspondiente de a la fecha de pedido
select day(getdate())
select day(x.FechaPedido) as Dia,
       month(x.FechaPedido) as Mes,
       year(x.FechaPedido) as Año,
       x.FechaPedido as [Fecha Completa]
from Ventas.pedidoscabe x
```

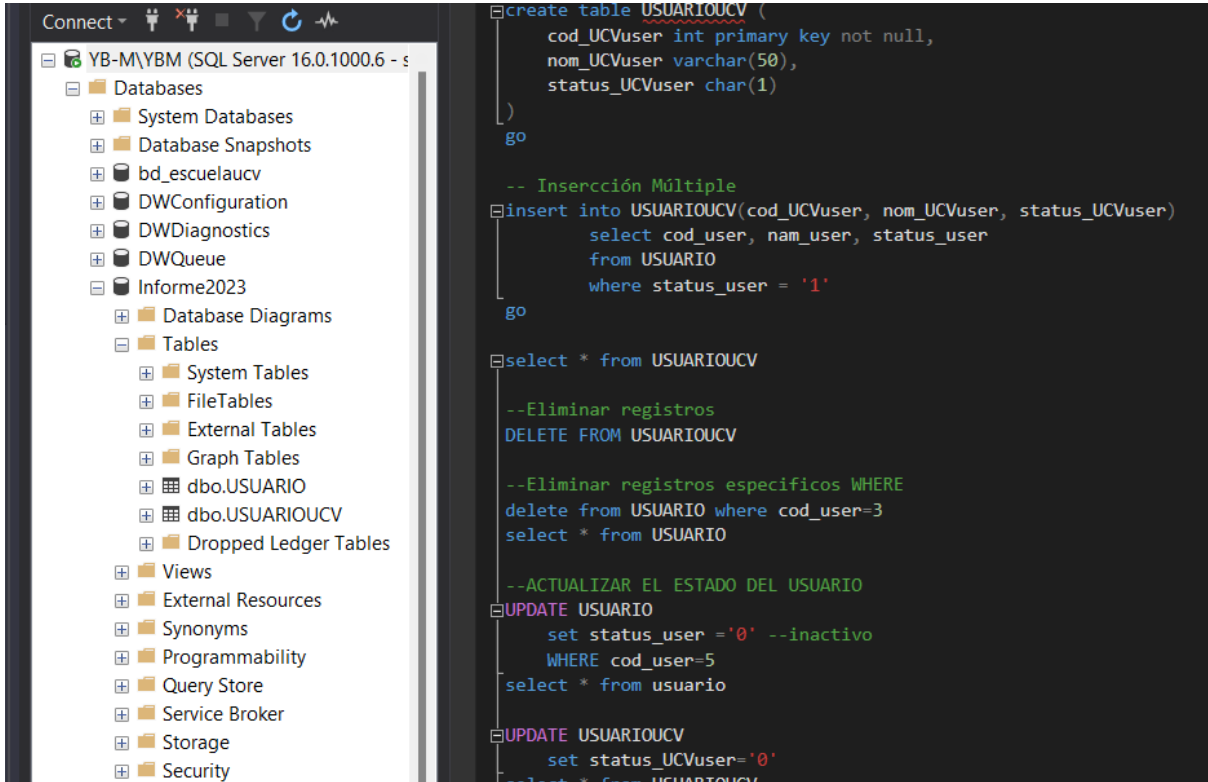
- Función DATEPART: Esta consulta utiliza la función DATEPART para obtener partes específicas de la fecha de pedido, como el día, el mes y el año. Los resultados se formatean para mostrar la fecha de pedido en un formato personalizado.

```
--mostrar el día de una determinada fecha obtenida a partir de una variable local
declare @fecha date='2022/05/31'
select day(@fecha) as día
--función DATEPART
--Devuelve un valor numerico entero que representa una parte especifica de una determinada fecha
--mostrar la fecha de pedido separado por día , mes y año
select cast (datepart(DD,x.FechaPedido) as varchar(2))+ '/' +
       cast(datepart(MM, x.FechaPedido) as varchar(2))+ '/' +
       cast(datepart(YYYY,x.FechaPedido) as varchar(4)) as [FECHA DE PEDIDO],
       x.FechaPedido
from ventas.pedidoscabe x
```



## Anexos

- Base de datos “Informe2023”



The screenshot shows the SQL Server Enterprise Manager interface on the left and a SQL Query window on the right. The Enterprise Manager displays the 'Informe2023' database under the 'Databases' folder. The SQL Query window contains the following T-SQL script:

```
create table USUARIOUCV (
    cod_UCVuser int primary key not null,
    nom_UCVuser varchar(50),
    status_UCVuser char(1)
)
go

-- Inserción Múltiple
insert into USUARIOUCV(cod_UCVuser, nom_UCVuser, status_UCVuser)
select cod_user, nam_user, status_user
from USUARIO
where status_user = '1'
go

select * from USUARIOUCV

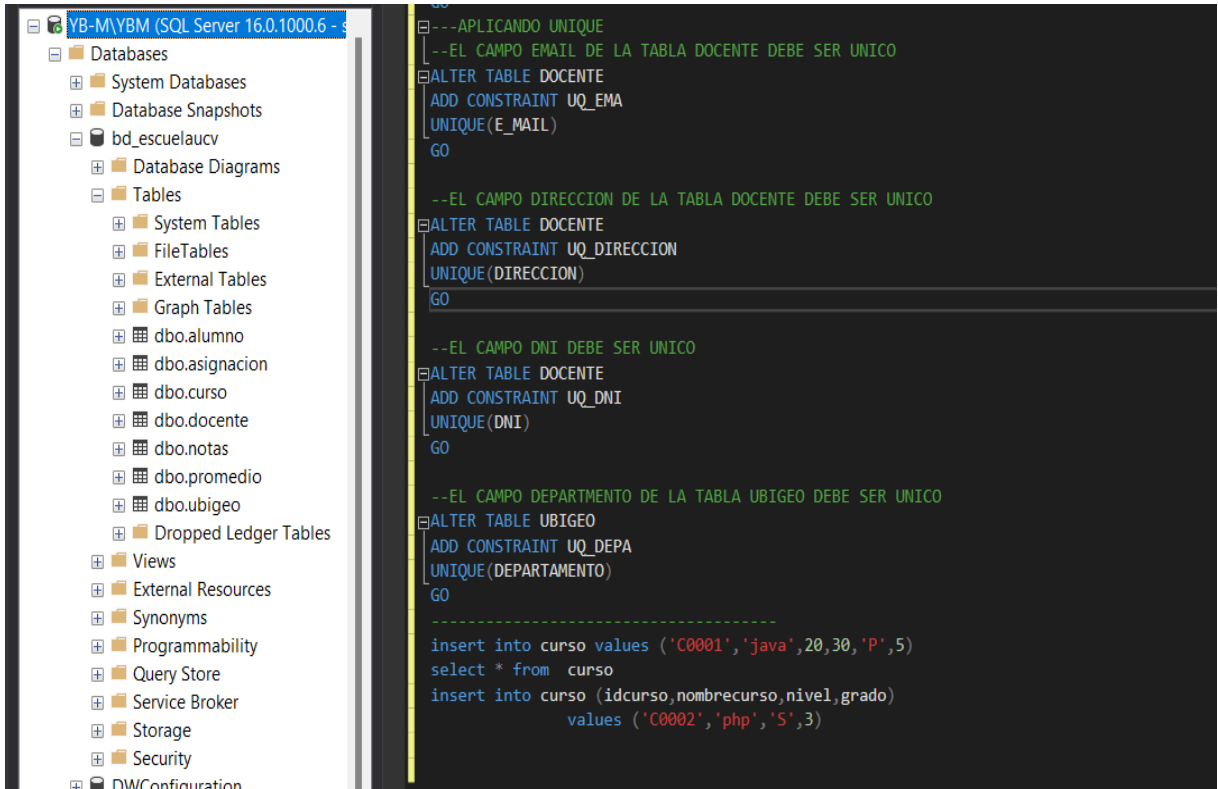
--Eliminar registros
DELETE FROM USUARIOUCV

--Eliminar registros especificos WHERE
delete from USUARIO where cod_user=3
select * from USUARIO

--ACTUALIZAR EL ESTADO DEL USUARIO
UPDATE USUARIO
    set status_user = '0' --inactivo
    WHERE cod_user=5
select * from usuario

UPDATE USUARIOUCV
    set status_UCVuser='0'
select * from USUARIOUCV
```

- Base de datos “bd\_escualaucv”



The screenshot shows the SQL Server Enterprise Manager interface on the left, displaying the database structure for 'bd\_escuelaucv'. The right pane shows a SQL query window with the following code:

```

GO
---APLICANDO UNIQUE
--EL CAMPO EMAIL DE LA TABLA DOCENTE DEBE SER UNICO
ALTER TABLE DOCENTE
ADD CONSTRAINT UQ_EMAIL
UNIQUE(E_MAIL)
GO

--EL CAMPO DIRECCION DE LA TABLA DOCENTE DEBE SER UNICO
ALTER TABLE DOCENTE
ADD CONSTRAINT UQ_DIRECCION
UNIQUE(DIRECCION)
GO

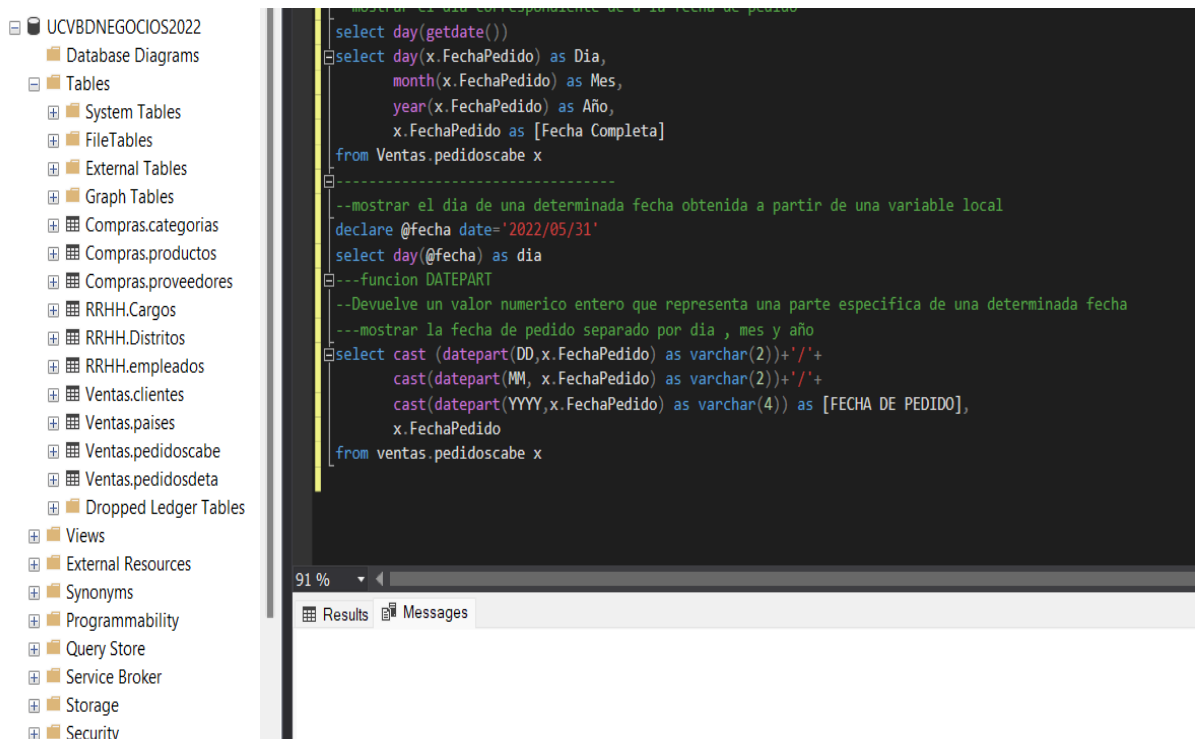
--EL CAMPO DNI DEBE SER UNICO
ALTER TABLE DOCENTE
ADD CONSTRAINT UQ_DNI
UNIQUE(DNI)
GO

--EL CAMPO DEPARTAMENTO DE LA TABLA UBIGEO DEBE SER UNICO
ALTER TABLE UBIGEO
ADD CONSTRAINT UQ_DEPA
UNIQUE(DEPARTAMENTO)
GO

-----
insert into curso values ('C0001','java',20,30,'P',5)
select * from curso
insert into curso (idcurso,nombrecurso,nivel,grado)
values ('C0002','php','S',3)

```

- Base de datos “UCVBDNEGOCIOS2022”



The screenshot shows the SQL Server Enterprise Manager interface on the left, displaying the database structure for 'UCVBDNEGOCIOS2022'. The right pane shows a SQL query window with the following code:

```

--mostrar el día correspondiente de a la fecha de pedido
select day(getdate())
select day(x.FechaPedido) as Dia,
month(x.FechaPedido) as Mes,
year(x.FechaPedido) as Año,
x.FechaPedido as [Fecha Completa]
from Ventas.pedidoscabe x

-----
--mostrar el día de una determinada fecha obtenida a partir de una variable local
declare @fecha date='2022/05/31'
select day(@fecha) as dia

---funcion DATEPART
--Devuelve un valor numerico entero que representa una parte especifica de una determinada fecha
--mostrar la fecha de pedido separado por día , mes y año
select cast (datepart(DD,x.FechaPedido) as varchar(2))+ '/' +
cast(datepart(MM, x.FechaPedido) as varchar(2))+ '/' +
cast(datepart(YYYY,x.FechaPedido) as varchar(4)) as [FECHA DE PEDIDO],
x.FechaPedido
from ventas.pedidoscabe x

```