Nombre: Jorge Manuel Oyoqui Aguilera
Matrícula: A01711783
Fecha: 04/05/2025
Materia: Construcción de Software y Toma de Decisiones
Grupo: 501
Profesores: Enrique Alfonso Calderón Balderas,
Denisse L. Maldonado Flores, Alejandro Fernández Vilchis

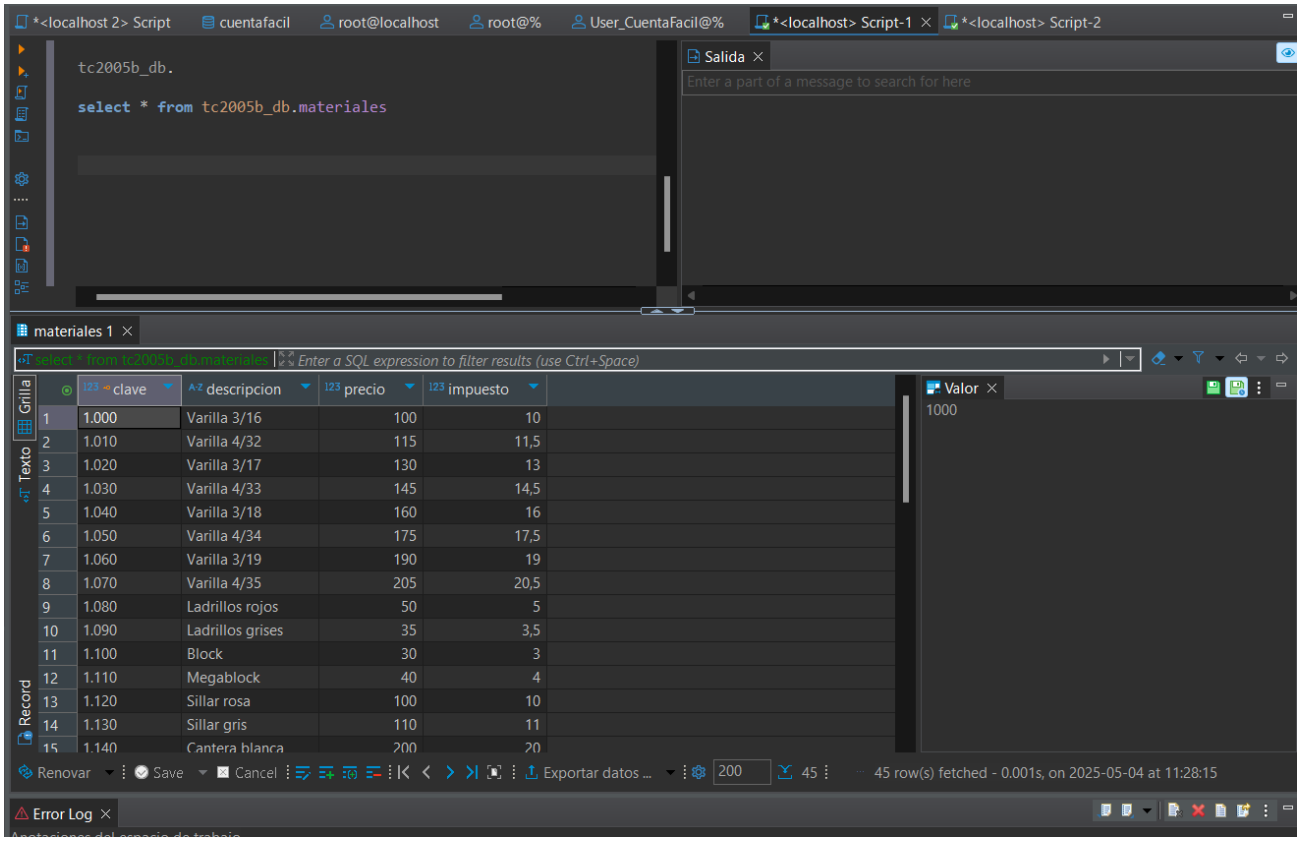
Laboratorio 20

Consulta de una tabla completa

Algebra relacional.

materiales

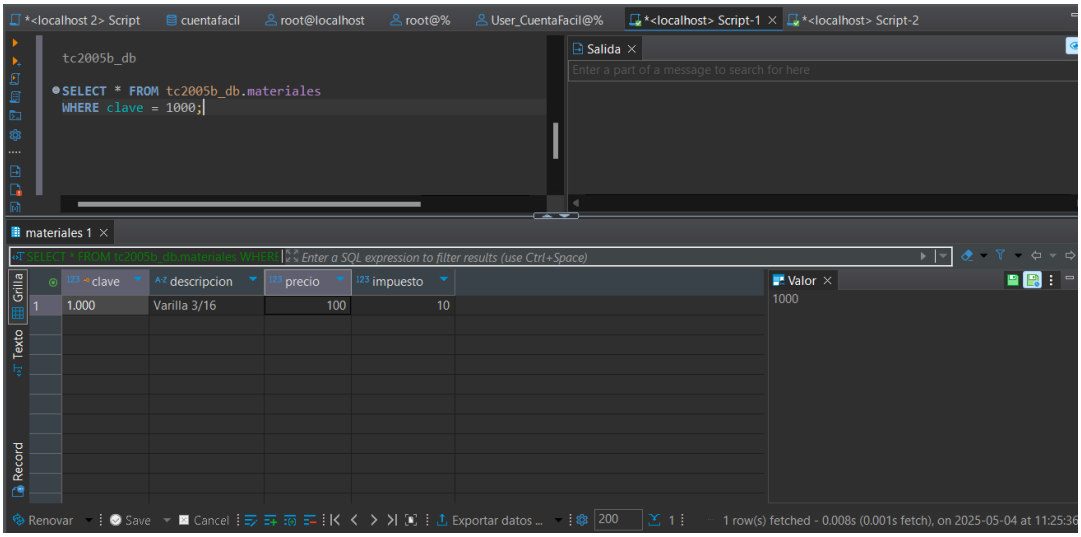
SQL
select * from materiales



Selección

Algebra relacional.
SL{clave=1000}(materiales)

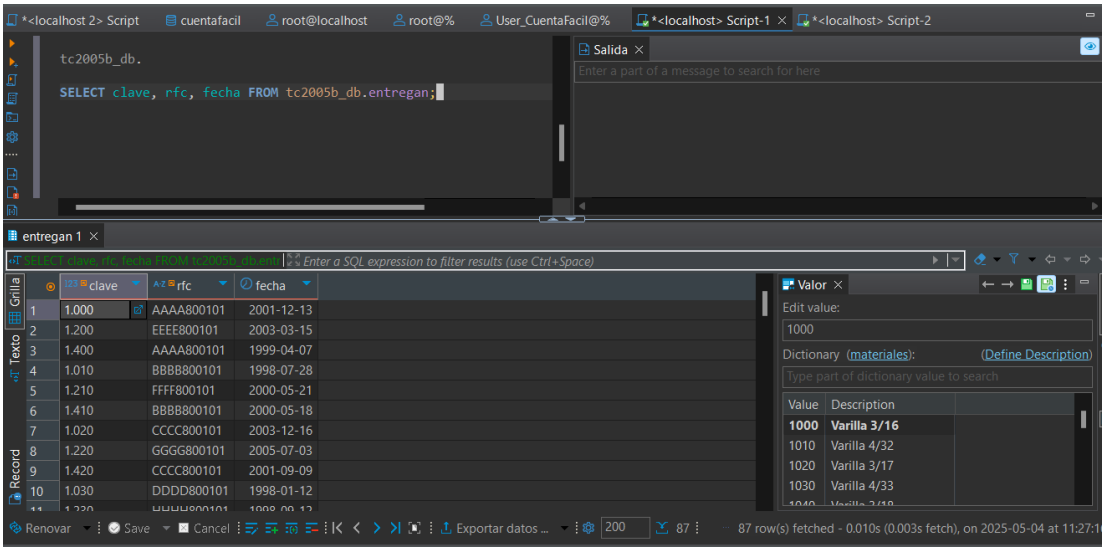
SQL
select * from materiales
where clave=1000



Proyección

Algebra relacional.
PR{clave,rfc,fecha} (entregan)

SQL
select clave,rfc,fecha from entregan



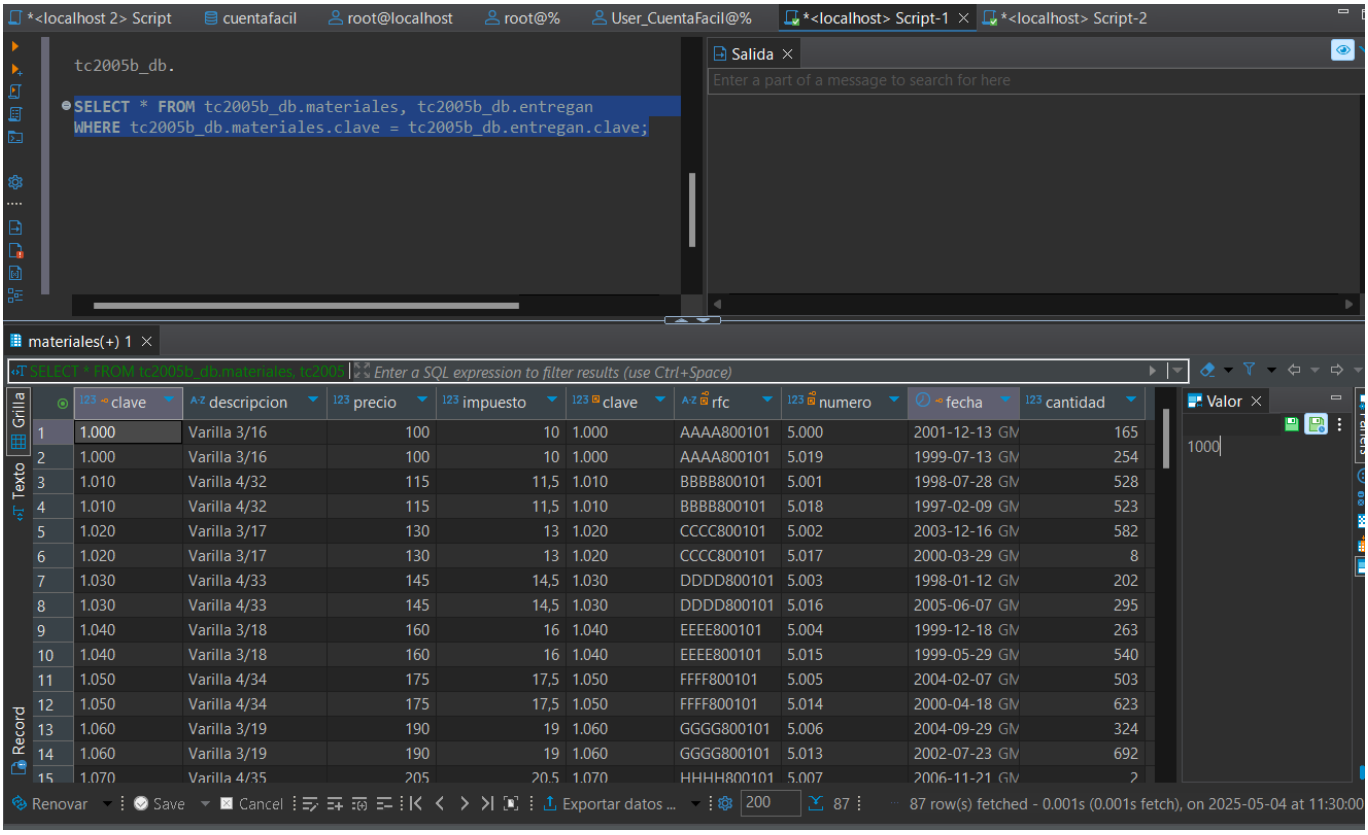
Reunión natural

Algebra relacional.
entregan JN materiales

SQL
select * from materiales,entregan
where materiales.clave = entregan.clave

Si algún material no ha se ha entregado ¿Aparecería en el resultado de esta consulta?

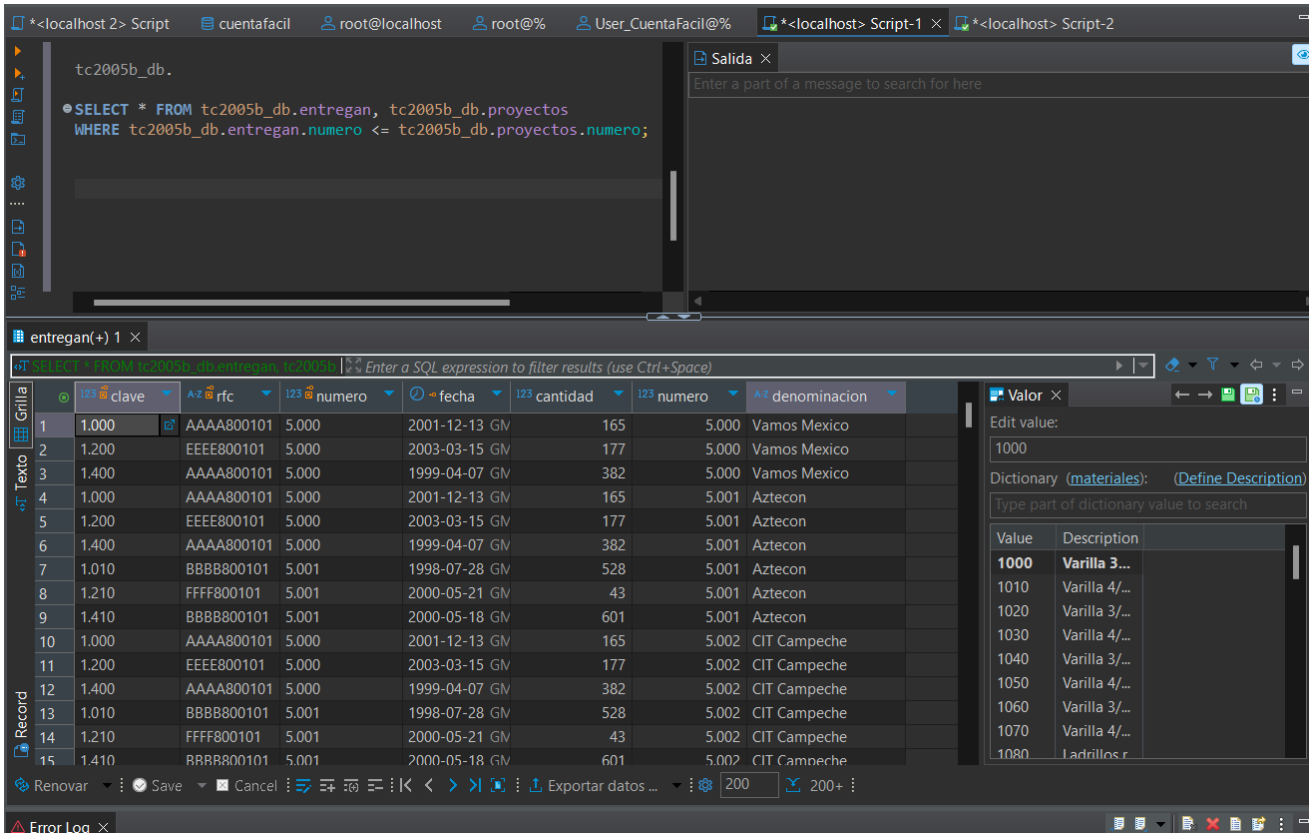
No, porque se usa un JOIN implícito que funciona como un INNER JOIN. Entonces, solo aparecen los materiales que tienen al menos una coincidencia en *entregan*.



Reunión con criterio específico

Algebra relacional.
entregan JN{entregan.numero <= proyectos.numero} proyectos

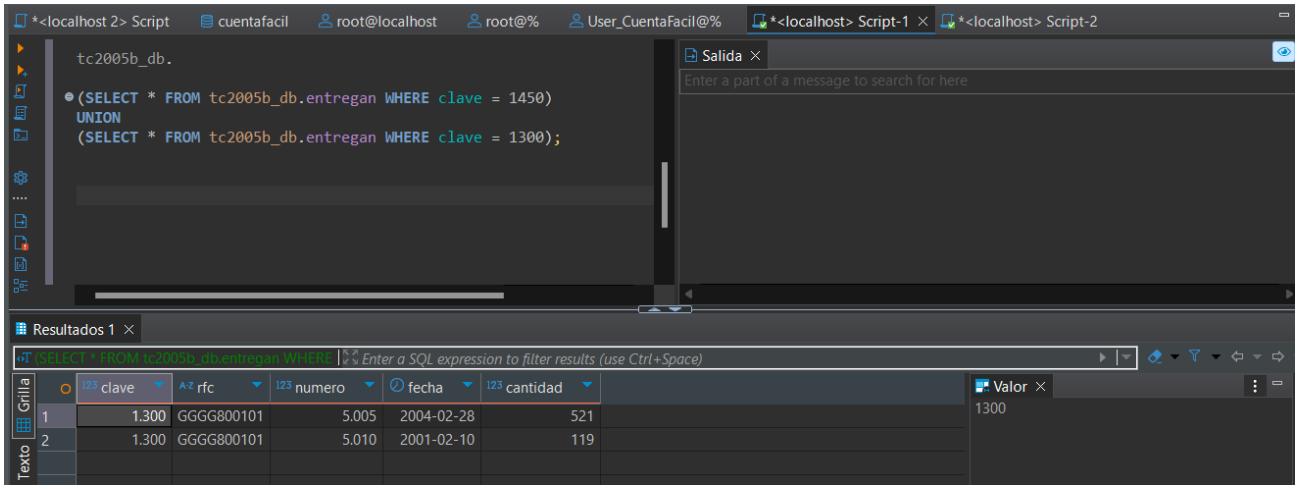
SQL
select * from entregan,proyectos
where entregan.numero < = proyectos.numero



Unión (se ilustra junto con selección)

Algebra relacional.
SL{clave=1450}(entregan) UN SL{clave=1300}(entregan)

SQL
(select * from entregan where clave=1450)
union
(select * from entregan where clave=1300)

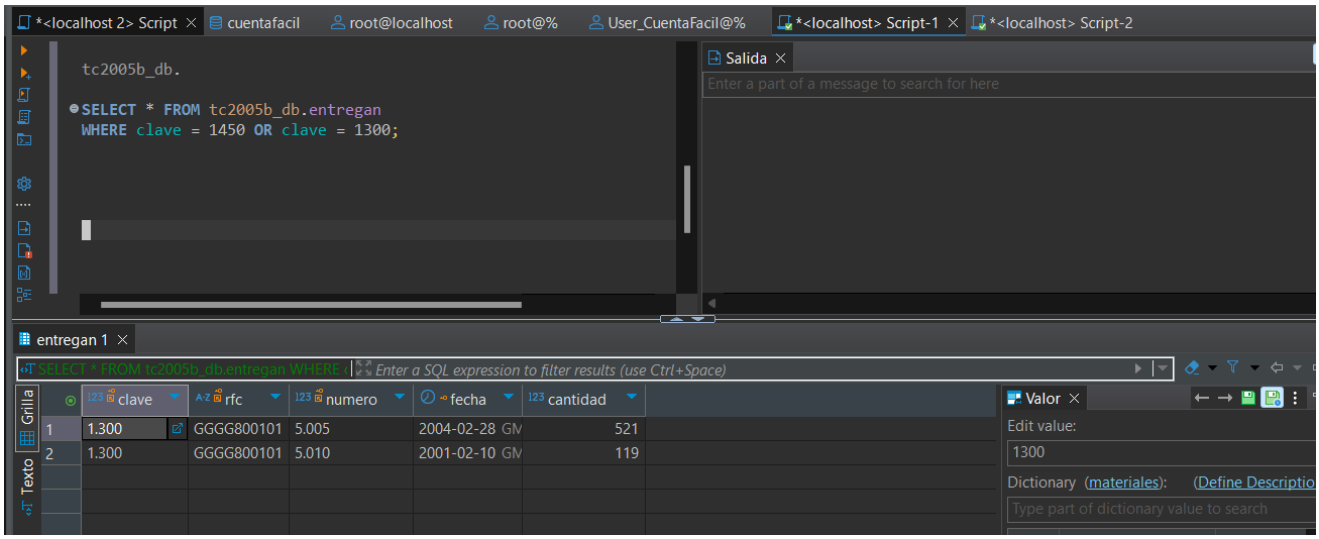


¿Cuál sería una consulta que obtuviera el mismo resultado sin usar el operador Unión? Compruébalo.

La consulta que obtendría el mismo resultado pero sin usar el operador JOIN sería la siguiente:

SELECT * FROM entregan

WHERE clave = 1450 OR clave = 1300;

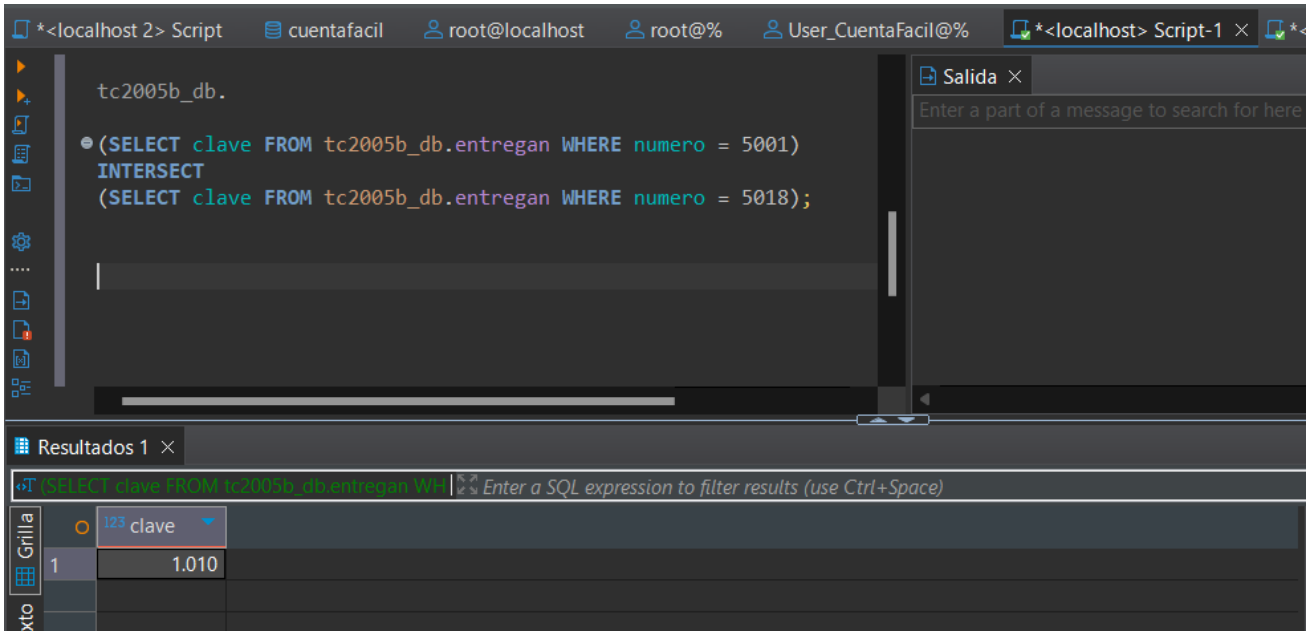


Intersección (se ilustra junto con selección y proyección)

Algebra relacional.
PR{clave}(SL{numero=5001}(entregan)) IN PR{clave}(SL{numero=5018}(entregan))

SQL
Nota: Debido a que en SQL server no tiene definida alguna palabra reservada que nos permita hacer esto de una manera entendible, veremos esta sección en el siguiente laboratorio con el uso de Subconsultas. Un ejemplo de un DBMS que si tiene la implementación de una palabra reservada para esta función es Oracle, en él si se podría generar la consulta con una sintaxis como la siguiente:

(select clave from entregan where numero=5001)
intersect
(select clave from entregan where numero=5018)

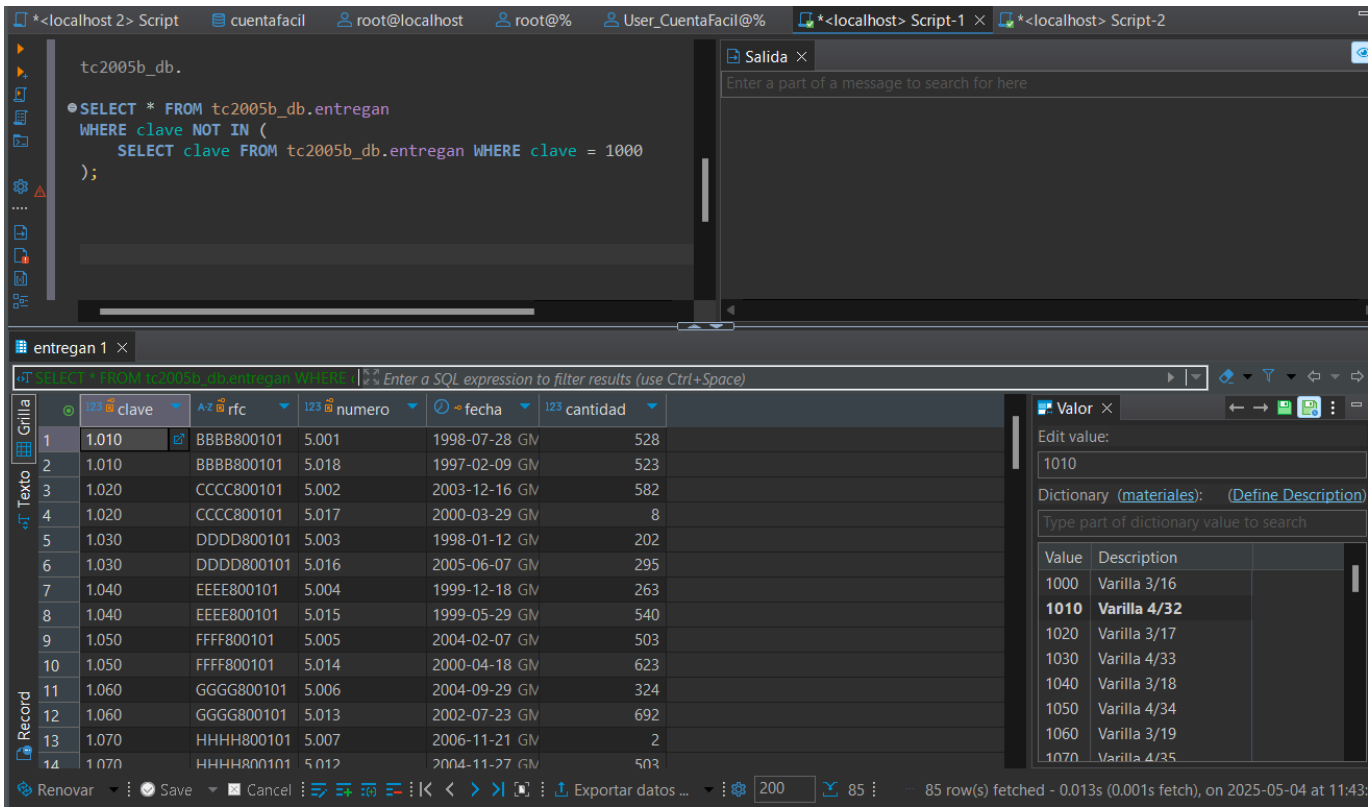


Diferencia (se ilustra con selección)

Algebra relacional.
entregan - SL{clave=1000}(entregan)

SQL
(select * from entregan)
minus
(select * from entregan where clave=1000)

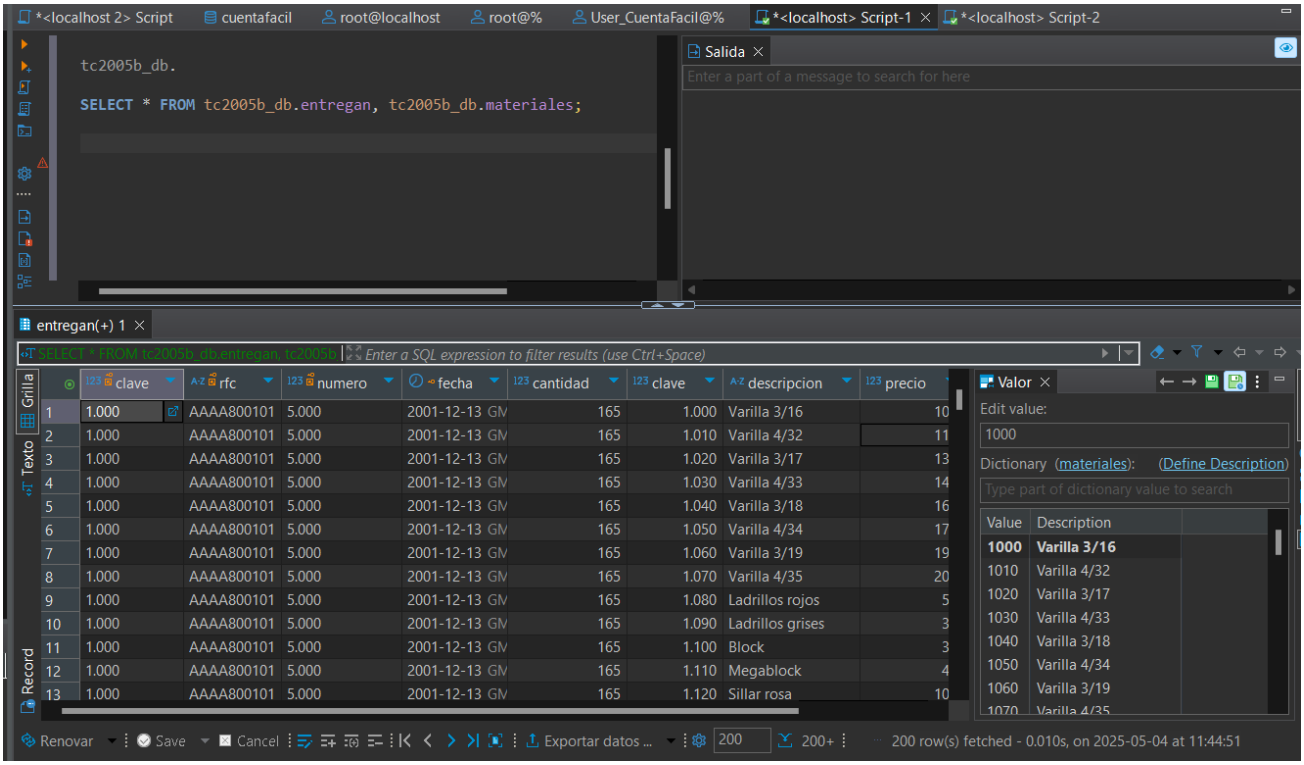
Nuevamente, "minus" es una palabra reservada que no está definida en SQL Server, define una consulta que regrese el mismo resultado.



Producto cartesiano

Algebra relacional.
entregan X materiales

SQL
select * from entregan,materiales



¿Cómo está definido el número de tuplas de este resultado en términos del número de tuplas de entregan y de materiales?

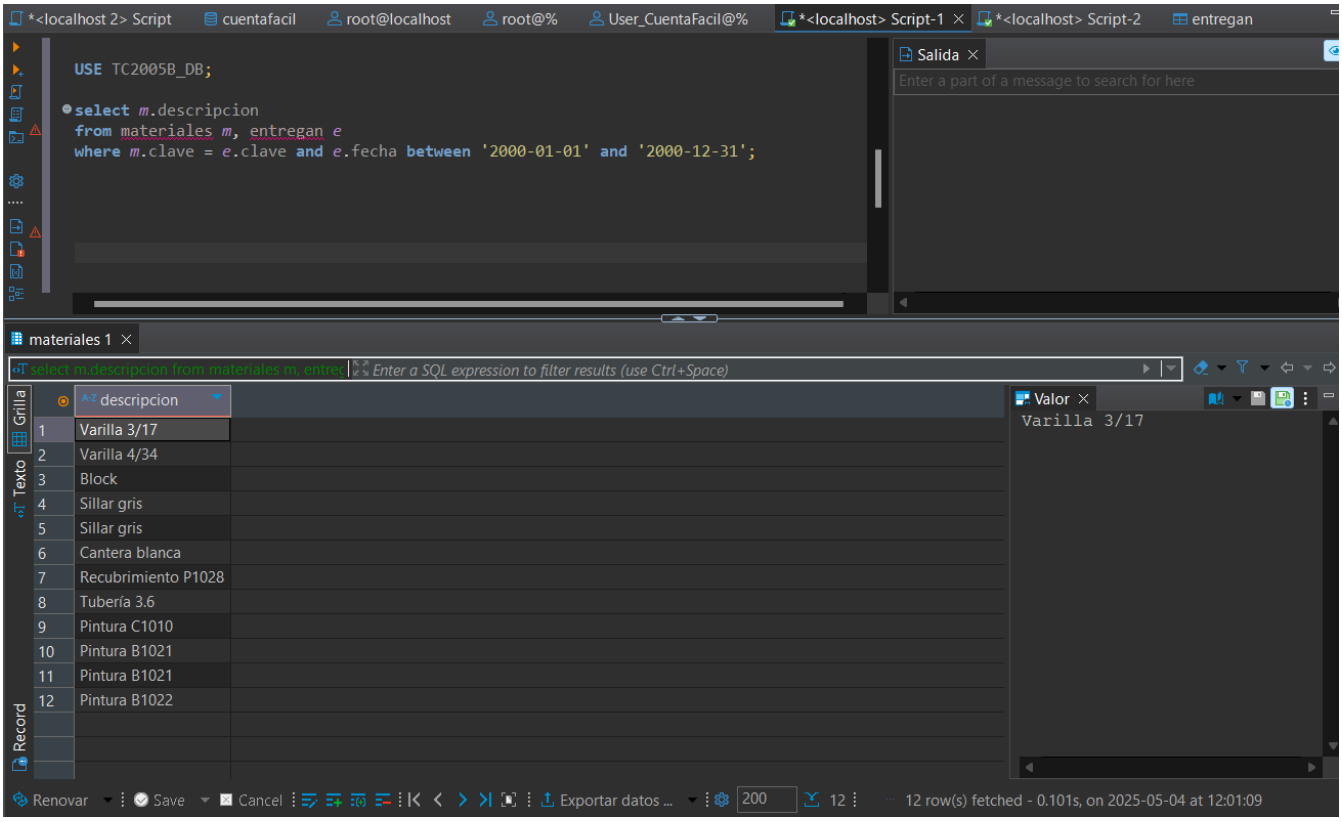
La cantidad de tuplas del resultado es el producto entre la cantidad de tuplas de *entregan* multiplicado por la cantidad de tuplas en *materiales*.

Construcción de consultas a partir de una especificación

Plantea ahora una consulta para obtener las descripciones de los materiales entregados en el año 2000.

Recuerda que la fecha puede indicarse como '01-JAN-2000' o '01/01/00'.

Importante: Recuerda que cuando vayas a trabajar con fechas, antes de que realices tus consultas debes ejecutar la instrucción "set dateformat dmy". Basta con que la ejecutes una sola vez para que el manejador sepa que vas a trabajar con ese formato de fechas.



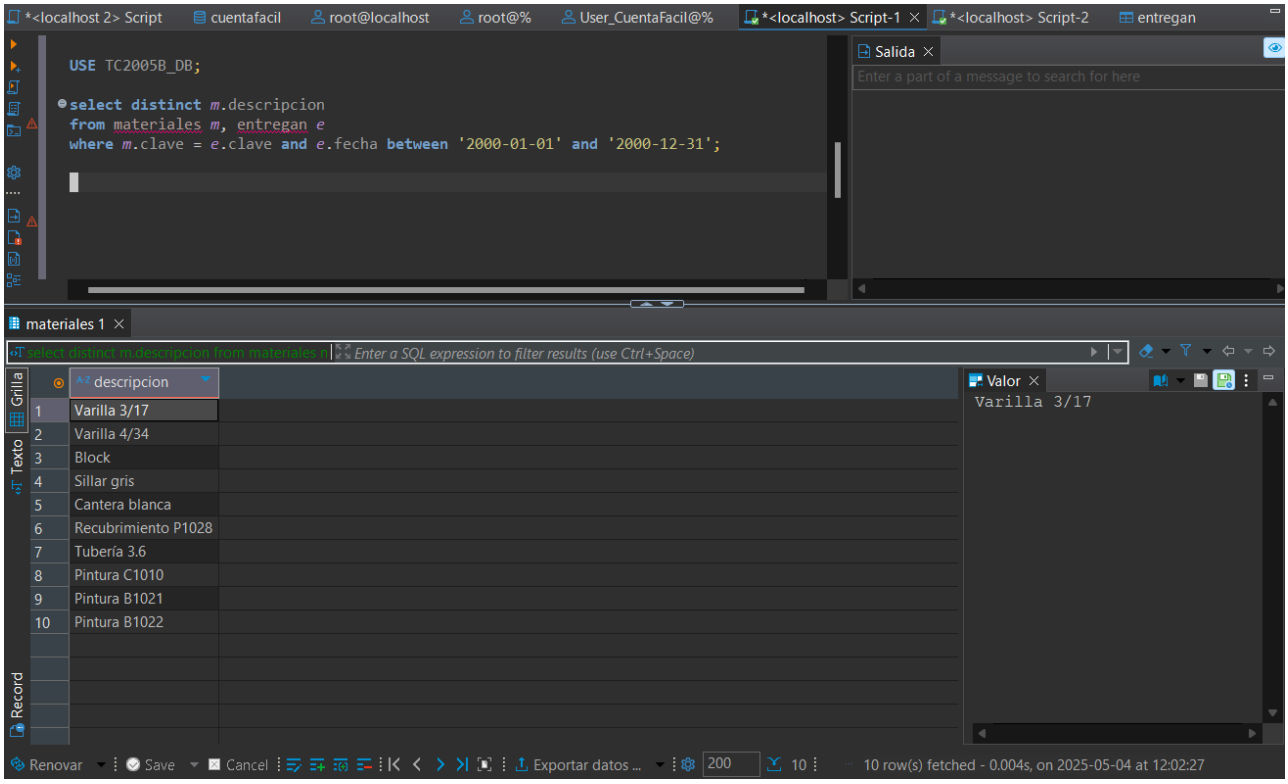
¿Por qué aparecen varias veces algunas descripciones de material?

Hay descripciones que aparecen varias veces justo por que un mismo material se pudo verer entregado en distintas fechas a lo largo del tiempo, pero eso se corrige al usar DISTINCT, de modo que cualquier descripción duplicada se borra, tal y como vemos en la siguiente sección de Uso del calificador distinct.

Uso del calificador distinct

En el resultado anterior, observamos que una misma descripción de material aparece varias veces.

Agrega la palabra distinct inmediatamente después de la palabra select a la consulta que planteaste antes.



¿Qué resultado obtienes en esta ocasión?

En este caso, se borran las descripciones de material que se repiten (que si mal no veo, se borra una descripción Sillar gris y una Pintura B1021), dejando sólo una tabla con las descripciones sin que se repita ninguna.

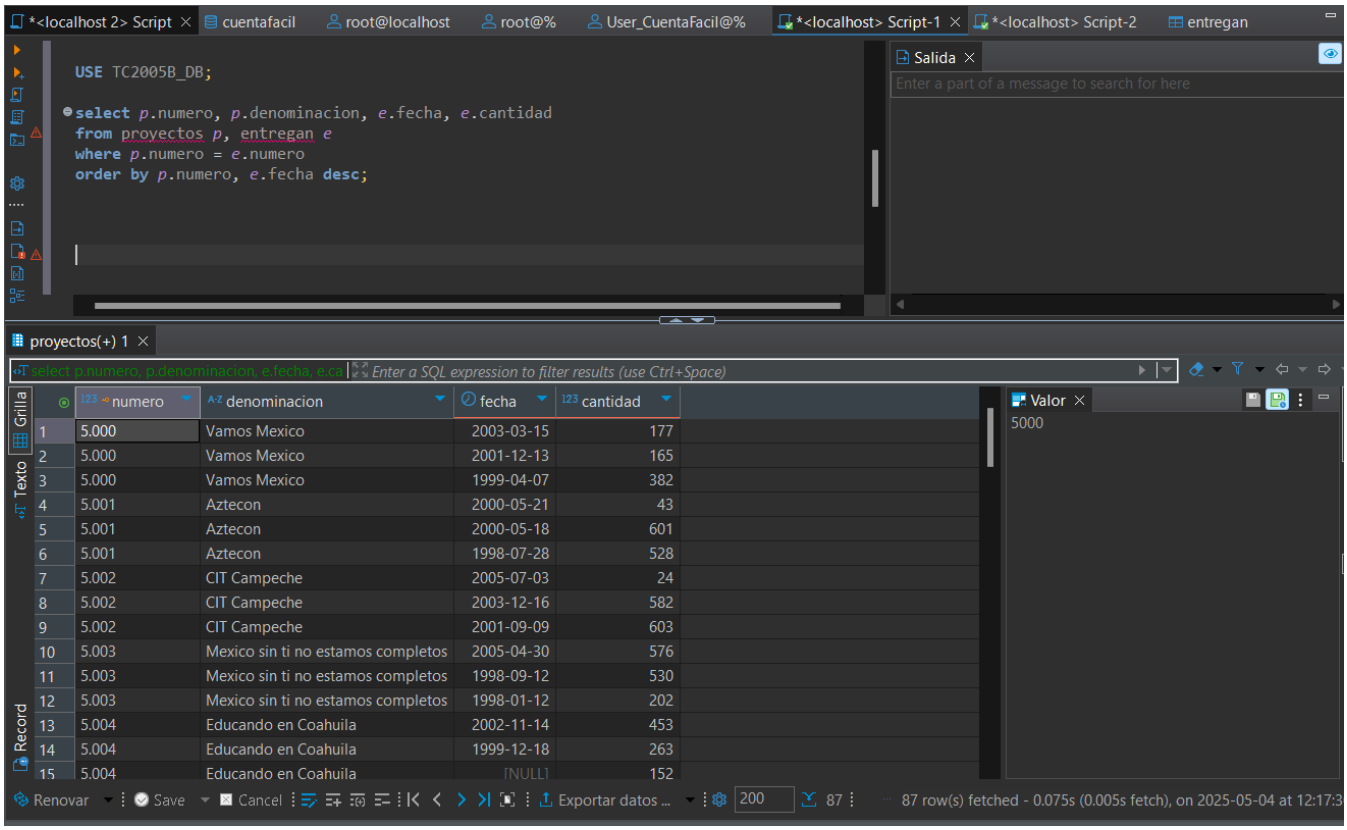
Ordenamientos.

Si al final de una sentencia select se agrega la cláusula

order by campo [desc] [,campo [desc] ...]

donde las partes encerradas entre corchetes son opcionales (los corchetes no forman parte de la sintaxis), los puntos suspensivos indican que pueden incluirse varios campos y la palabra desc se refiere a descendente. Esta cláusula permite presentar los resultados en un orden específico.

Obtén los números y denominaciones de los proyectos con las fechas y cantidades de sus entregas, ordenadas por número de proyecto, presentando las fechas de la más reciente a la más antigua.



Uso de expresiones.

En álgebra relacional los argumentos de una proyección deben ser columnas. Sin embargo, en una sentencia SELECT es posible incluir expresiones aritméticas o funciones que usen como argumentos de las columnas de las tablas involucradas o bien constantes. Los operadores son:

+ Suma
- Resta
* Producto
/ División

Las columnas con expresiones pueden renombrarse escribiendo después de la expresión un alias que puede ser un nombre arbitrario; si el alias contiene caracteres que no sean números o letras (espacios, puntos etc.) debe encerrarse entre comillas dobles (" nuevo nombre"). Para SQL Server también pueden utilizarse comillas simples.

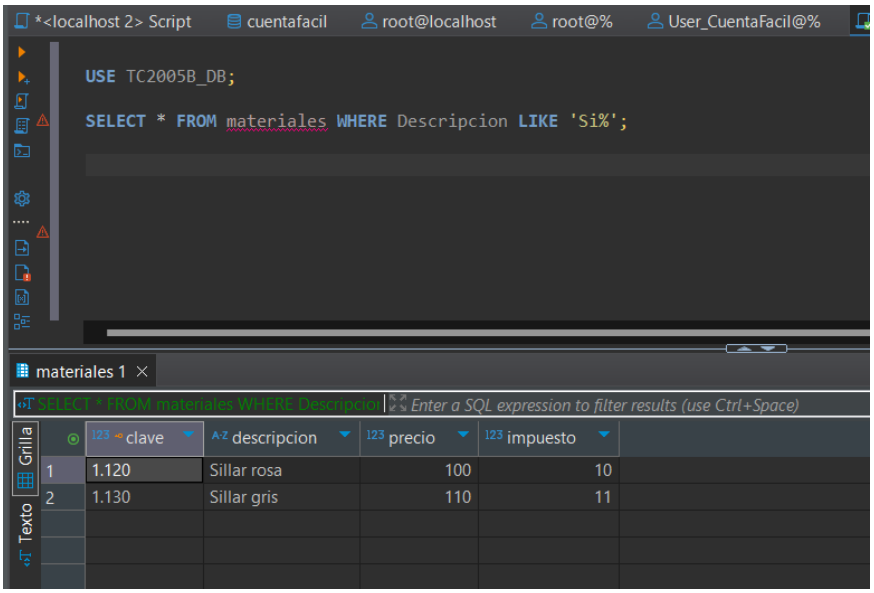
Operadores de cadena

El operador LIKE se aplica a datos de tipo cadena y se usa para buscar registros, es capaz de hallar coincidencias dentro de una cadena bajo un patrón dado.

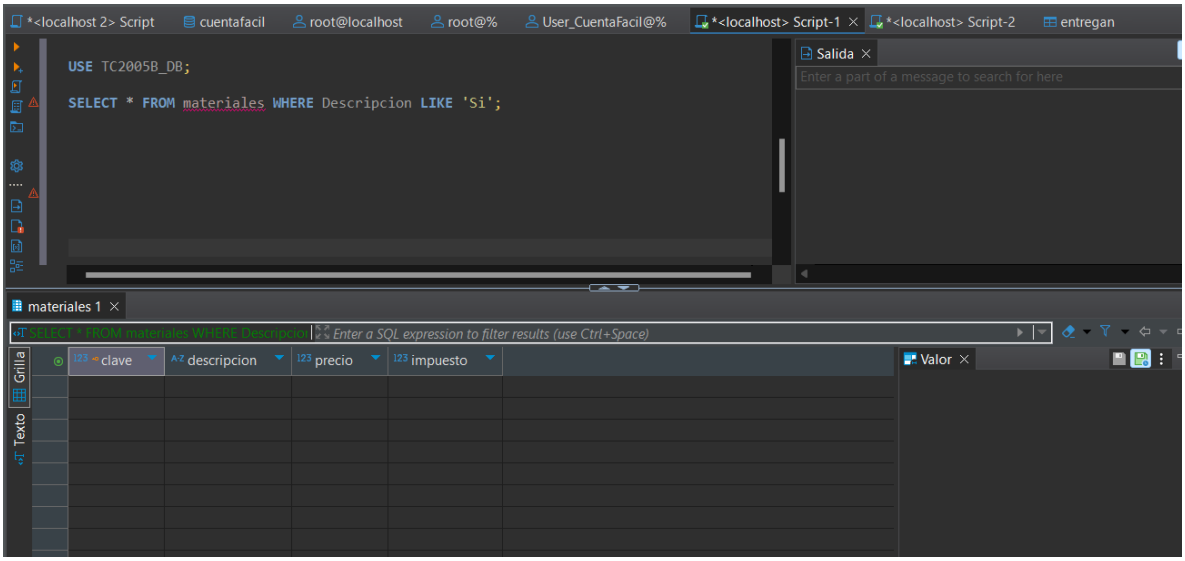
También contamos con el operador comodín (%), que coincide con cualquier cadena que tenga cero o más caracteres. Este puede usarse tanto de prefijo como sufijo.

SELECT * FROM productos where Descripcion LIKE 'Si%'

¿Qué resultado obtienes? Que se devuelven sólo las descripciones que empiecen por Si



Explica que hace el símbolo '%'. % es como un comodín que representa cualquier número de caracteres incluyendo el 0. En este caso, representa el resto de la cadena de la descripción que empiece por Si.
¿Qué sucede si la consulta fuera: LIKE 'Si'? En ese caso, se devolverían las consultas cuya descripción fuera exactamente Si, sin nada antes o después, sólo Si.
¿Qué resultado obtienes? No se devuelve nada porque no hay ninguna descripción cuyo valor sea únicamente Si.

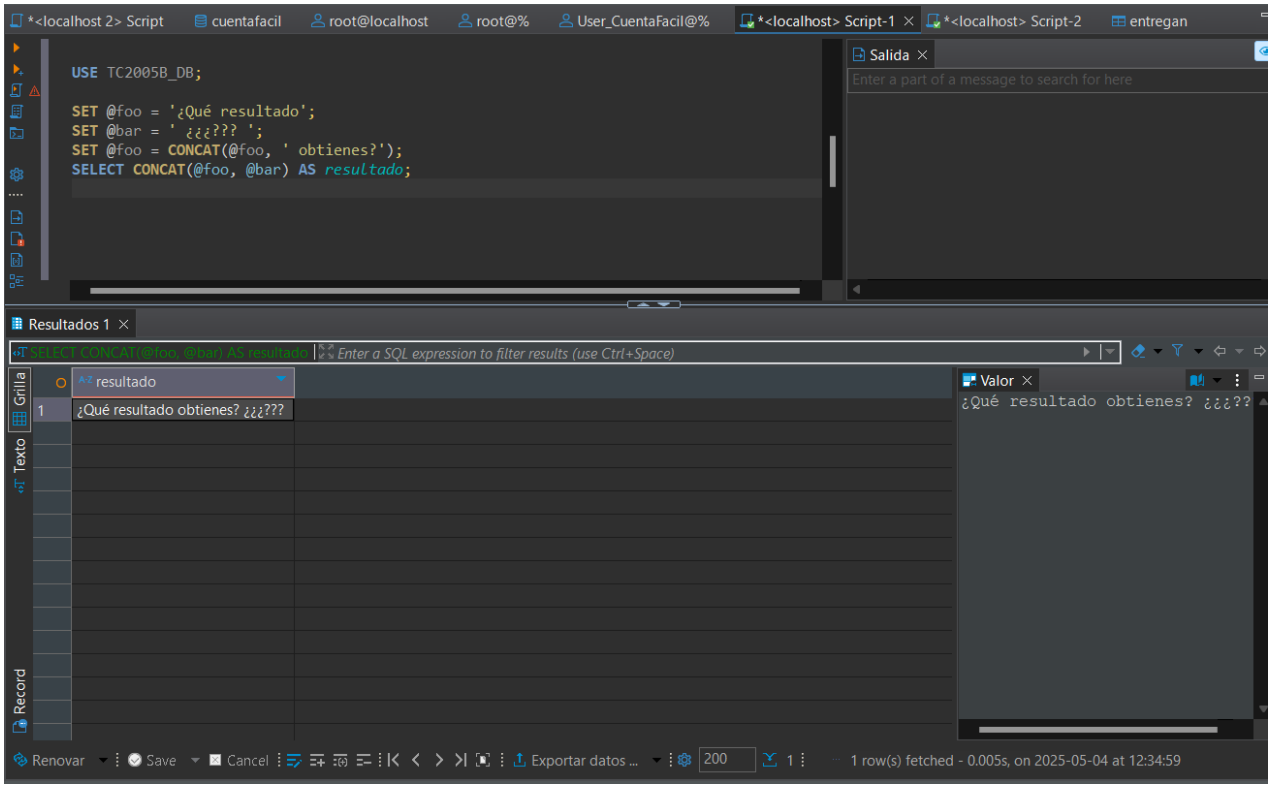


Explica a qué se debe este comportamiento. Como ya mencioné, se debe a que % representa el resto de la cadena de la descripción que empiece con Si, pero al ya no haber esa continuación, entonces la descripción debe ser solo Si, pero como no hay ninguna descripción dentro de materiales cuyo valor sea sólo Si, entonces no se devuelve nada.

Otro operador de cadenas es el de concatenación, (+, +=) este operador concatena dos o más cadenas de caracteres.
Su sintaxis es : Expresión + Expresión.
Un ejemplo de su uso, puede ser: Un ejemplo de su uso, puede ser:
SELECT (Apellido + ', ' + Nombre) as Nombre FROM Personas;

```
DECLARE @foo varchar(40);
DECLARE @bar varchar(40);
SET @foo = '¿Que resultado';
SET @bar = ' ¿¿¿??? '
SET @foo += ' obtienes?';
PRINT @foo + @bar;
```

¿Qué resultado obtienes de ejecutar el siguiente código? Obtengo una concatenación de los strings @foo y @bar en una tabla que llamé resultado.



¿Para qué sirve DECLARE? Aunque en este caso no lo usé porque me marcaba errores, DECLARE se usa para declarar variables locales en SQL Server.
¿Cuál es la función de @foo? @foo se limita a ser una variable que contiene una cadena de texto.
¿Que realiza el operador SET? SET asigna un valor a una variable, en este caso, a las variables @foo y @bar.

Sin embargo, tenemos otros operadores como [ ] , [^] y _.

[ ] - Busca coincidencia dentro de un intervalo o conjunto dado. Estos caracteres se pueden utilizar para buscar coincidencias de patrones como sucede con LIKE.

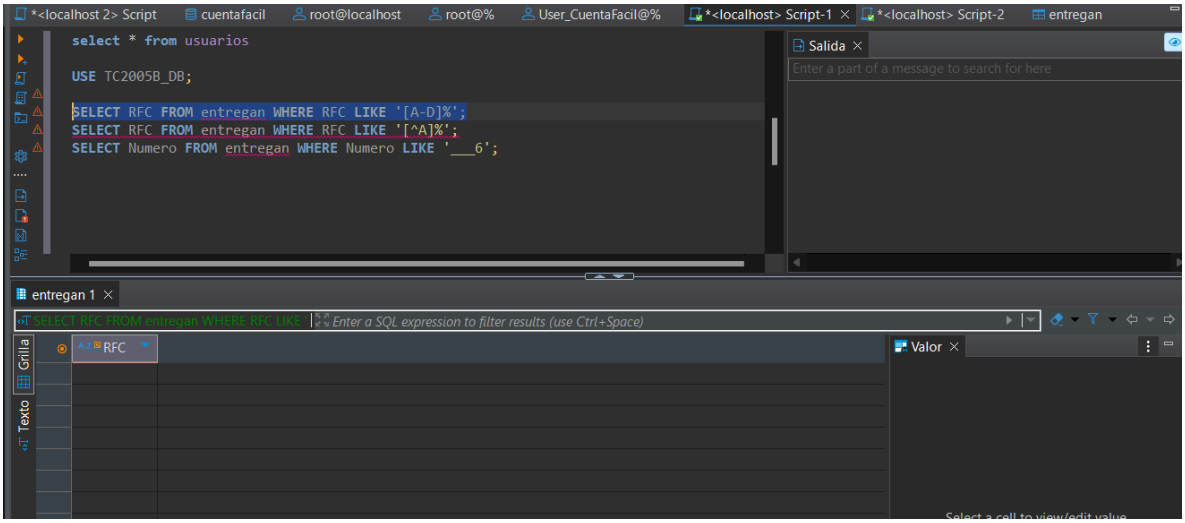
[^] - En contra parte, este operador coincide con cualquier caracter que no se encuentre dentro del intervalo o del conjunto especificado.

_ - El operador _ o guion bajo, se utiliza para coincidir con un caracter de una comparación de cadenas.

Ahora explica el comportamiento, función y resultado de cada una de las siguientes

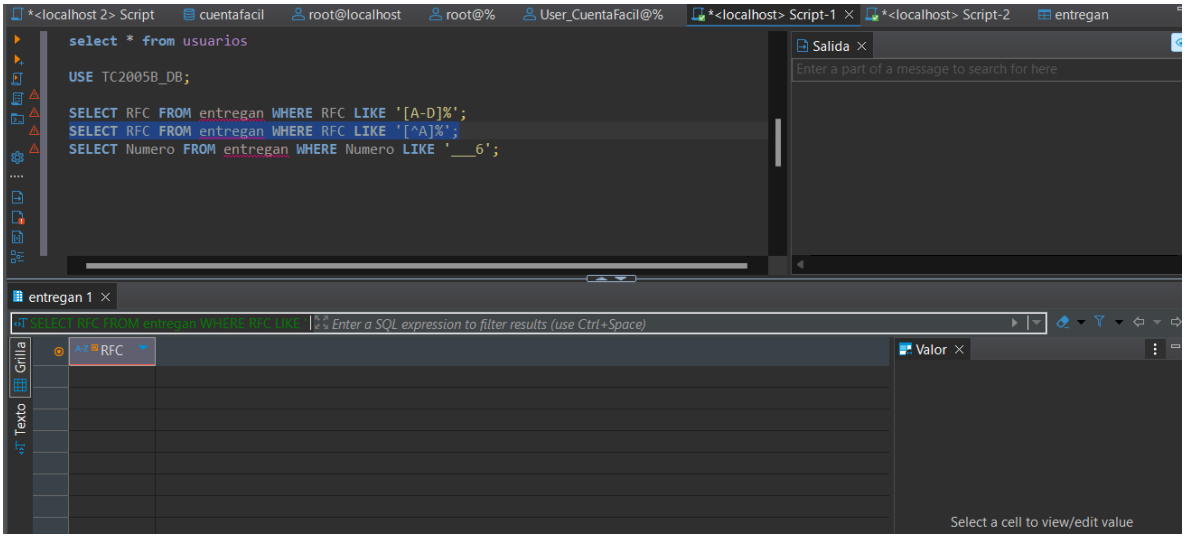
consultas:

SELECT RFC FROM Entregan WHERE RFC LIKE '[A-D]%';



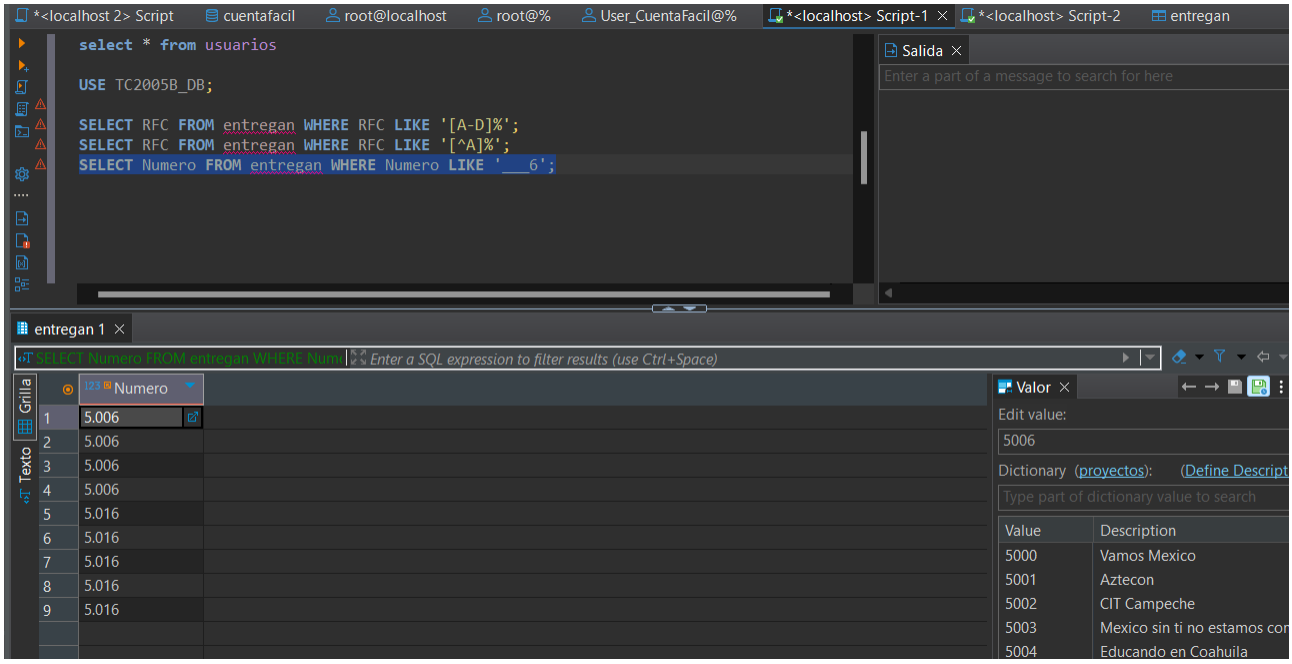
Comportamiento y función: Selecciona todos los RFCs de entregan que comienzan con una letra de la A a la D.

SELECT RFC FROM Entregan WHERE RFC LIKE '[^A]%';



Comportamiento y función: Selecciona todos los RFCs que NO comienzan con la letra A.

SELECT Numero FROM Entregan WHERE Numero LIKE '___6';



Comportamiento y función: busca números de cuatro dígitos donde el último sea 6. Cabe aclarar que cada "_" representa un carácter exacto, por eso se sabe que son cuatro dígitos, porque son tres guiones bajos y un 6 como último carácter,

Operadores compuestos.

Los operadores compuestos ejecutan una operación y establecen un valor.
+ = (Suma igual)
- = (Restar igual)
* = (Multiplicar igual)
/ = (Dividir igual)
% = (Módulo igual)

Operadores Lógicos.

Los operadores lógicos comprueban la verdad de una condición, al igual que los operadores de comparación, devuelven un tipo de dato booleano (True, false o unknown).

ALL Es un operador que compara un valor numérico con un conjunto de valores representados por un subquery. La condición es verdadera cuando todo el conjunto cumple la condición.

ANY o SOME Es un operador que compara un valor numérico con un conjunto de valores. La condición es verdadera cuando al menos un dato del conjunto cumple la condición.

La sintaxis para ambos es: valor_numerico {operador de comparación} subquery

BETWEEN Es un operador para especificar intervalos. Una aplicación muy común de dicho operador son intervalos de fechas.

SELECT Clave,RFC,Numero,Fecha,Cantidad
FROM Entregan
WHERE Numero Between 5000 and 5010;

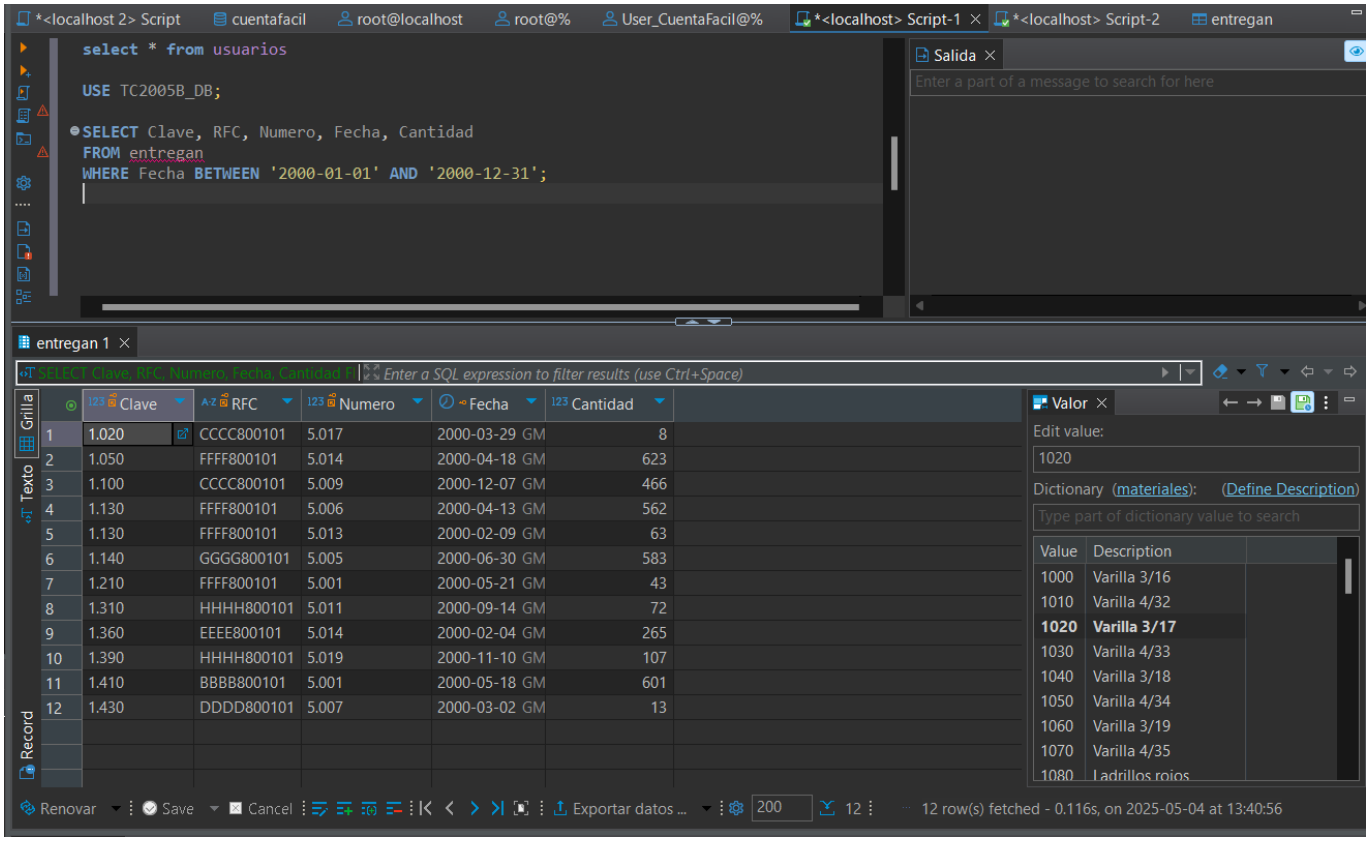
¿Cómo filtrarías rangos de fechas?

Lo filtraría usando BETWEEN, específicamente, este script:

SELECT Clave, RFC, Numero, Fecha, Cantidad

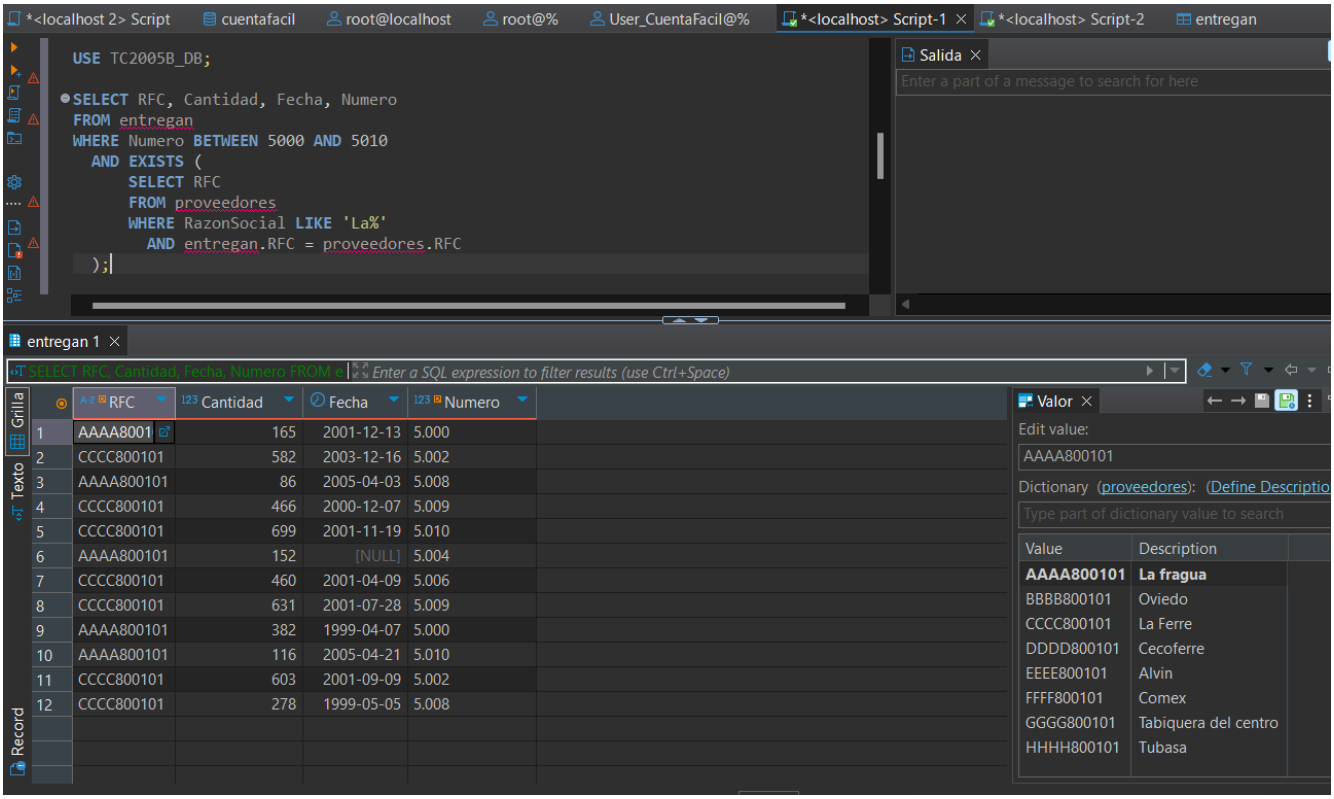
FROM Entregan

WHERE Fecha BETWEEN '2000-01-01' AND '2000-12-31';



EXISTS Se utiliza para especificar dentro de una subconsulta la existencia de ciertas filas.

SELECT RFC,Cantidad, Fecha,Numero
FROM [Entregan]
WHERE [Numero] Between 5000 and 5010 AND
Exists (SELECT [RFC]
FROM [Proveedores]
WHERE RazonSocial LIKE 'La%' and [Entregan].[RFC] = [Proveedores].[RFC])



¿Qué hace la consulta? Devuelve las entregas de la tabla Entregan cuyo número esté entre 5000 y 5010**,** y además**,** que el proveedor correspondiente tenga una razón social que comience con “La”**.**

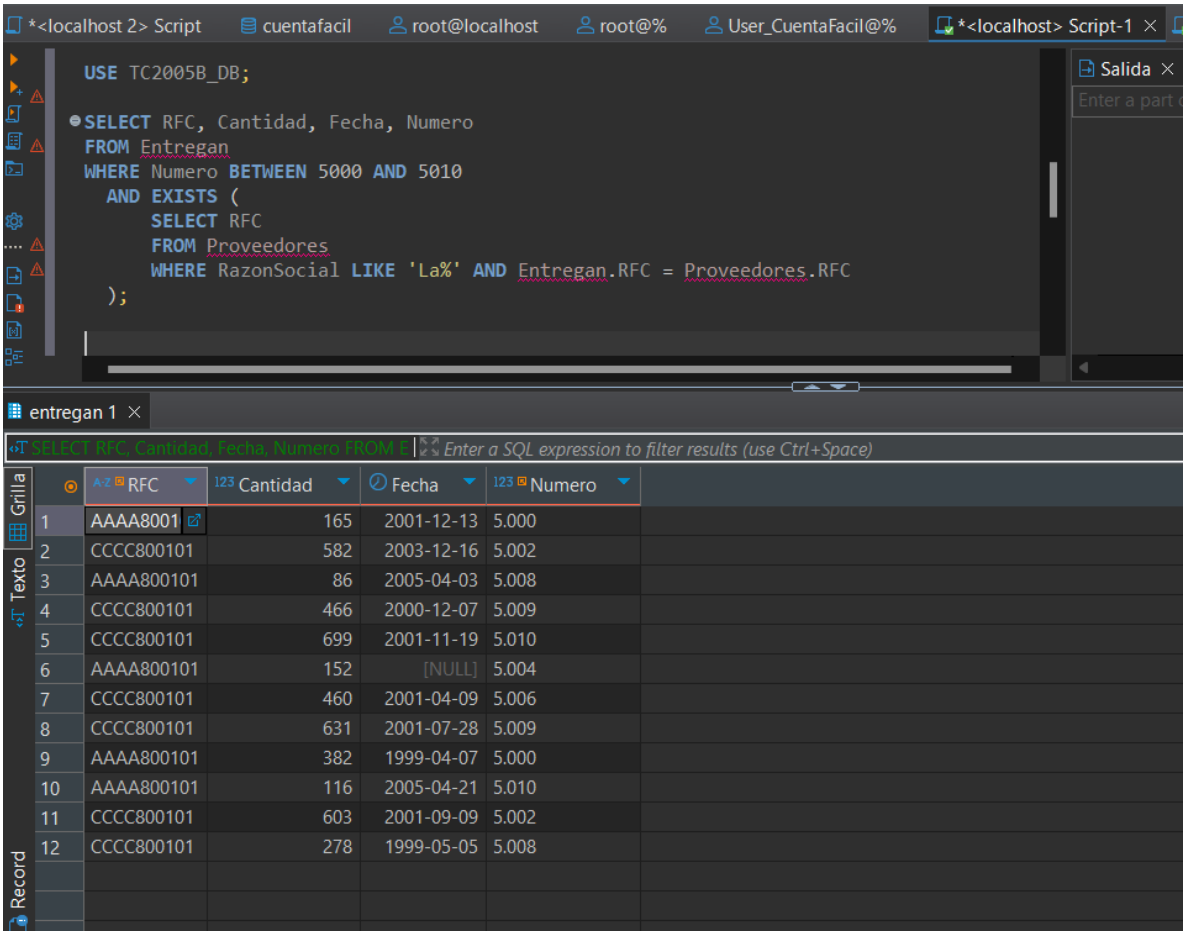
¿Qué función tiene el paréntesis ( ) después de EXISTS? El paréntesis contiene una subconsulta. EXISTS evalúa si esa subconsulta devuelve al menos una fila**.** Si es así, la condición es verdadera.

IN Especifica si un valor dado tiene coincidencias con algún valor de una subconsulta. NOTA: Se utiliza dentro del WHERE pero debe contener un parametro. Ejemplo: Where proyecto.id IN Lista_de_Proyectos_Subquery

Tomando de base la consulta anterior del EXISTS, realiza el query que devuelva el mismo resultado, pero usando el operador IN

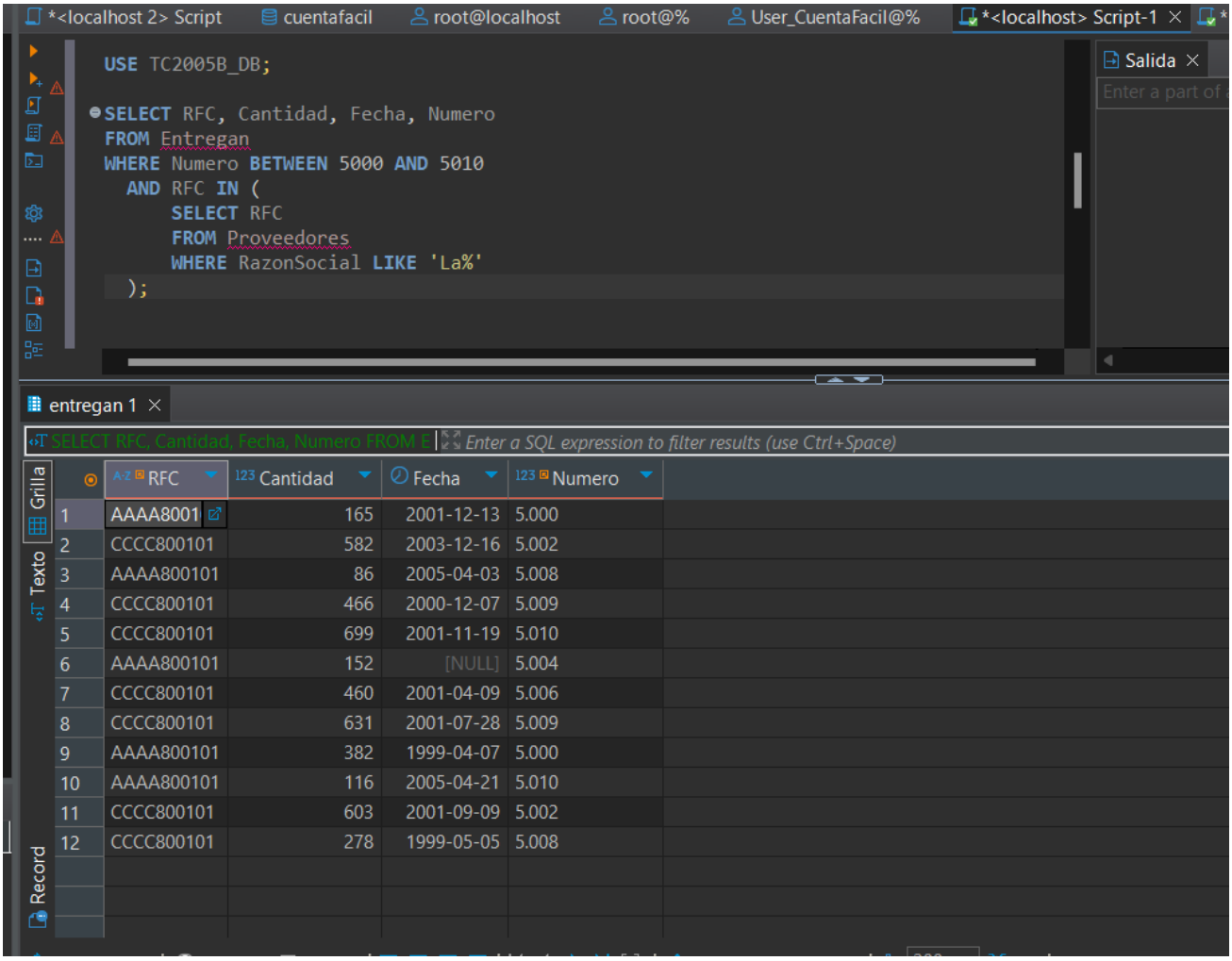
Un ejemplo podría ser la siguiente consulta:

```
SELECT RFC, Cantidad, Fecha, Numero

FROM Entregan

WHERE Numero BETWEEN 5000 AND 5010

  AND EXISTS (

      SELECT RFC

      FROM Proveedores

      WHERE RazonSocial LIKE 'La%' AND Entregan.RFC = Proveedores.RFC

  );
```



Que cambio para que trabaje con IN en lugar de EXISTS

```
SELECT RFC, Cantidad, Fecha, Numero
FROM Entregan
WHERE Numero BETWEEN 5000 AND 5010
  AND RFC IN (
      SELECT RFC
      FROM Proveedores
      WHERE RazonSocial LIKE 'La%'
  );
```

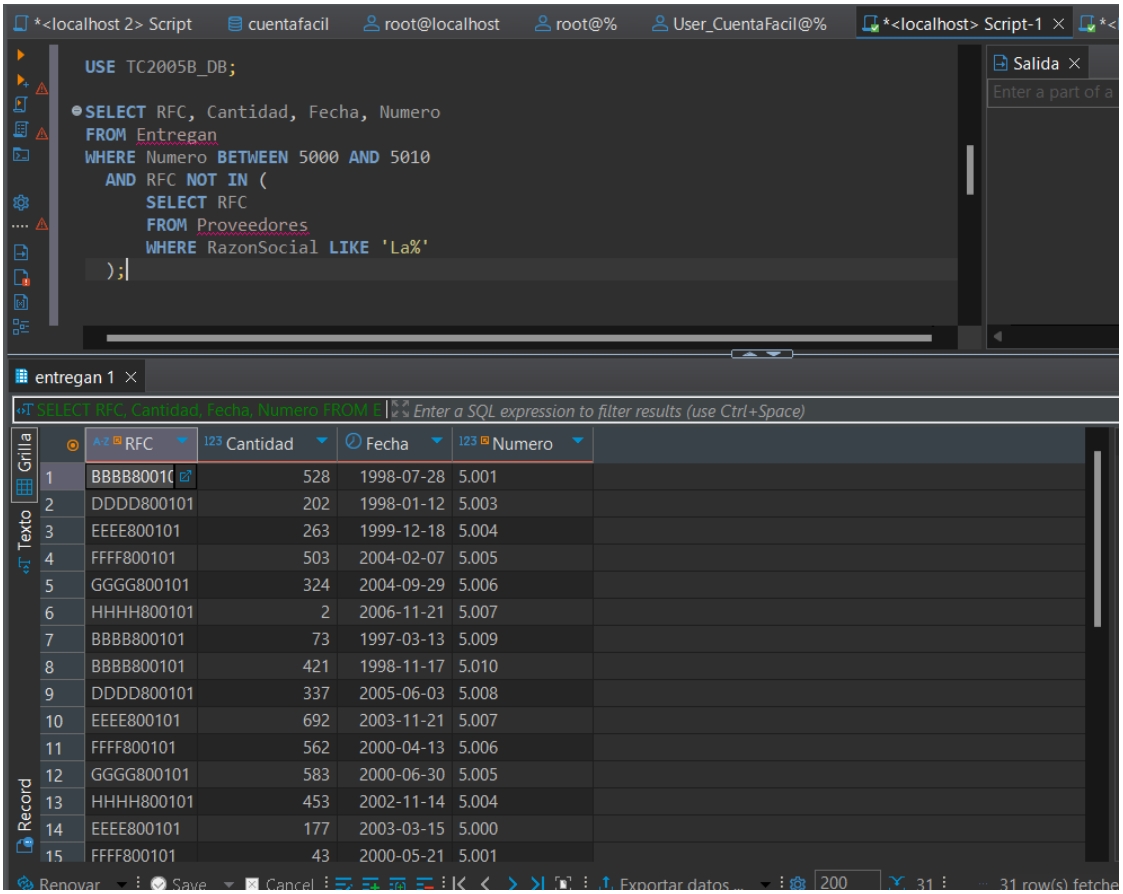


NOT Simplemente niega la entrada de un valor booleano.

Tomando de base la consulta anterior del EXISTS, realiza el query que devuelva el mismo resultado, pero usando el operador NOT IN Realiza un ejemplo donde apliques algún operador : ALL, SOME o ANY.

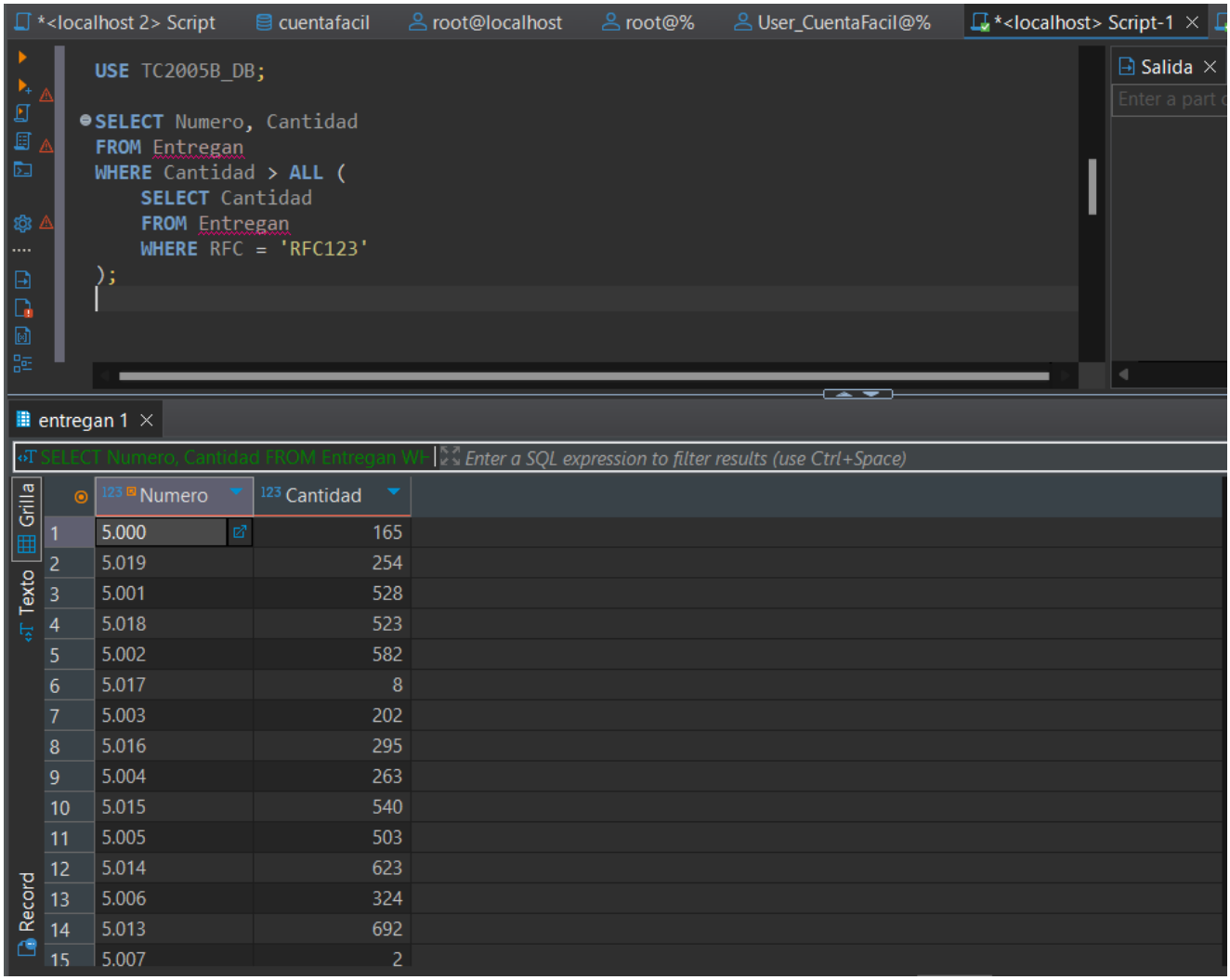
En el caso de usar NOT IN, se podría usar para que NO tome a los proveedores que inician con La, por ejemplo:

```
SELECT RFC, Cantidad, Fecha, Numero
FROM Entregan
WHERE Numero BETWEEN 5000 AND 5010
  AND RFC NOT IN (
      SELECT RFC
      FROM Proveedores
      WHERE RazonSocial LIKE 'La%'
  );
```



Ahora, en el caso de aplicar un ejemplo usando también ALL,SOME o ANY en el NOT IN, se podría usar, por ejemplo, para obtener proyectos cuya cantidad entregada es mayor a todas las entregas hechas por el proveedor con RFC = 'RFC123':

```
SELECT Numero, Cantidad
FROM Entregan
WHERE Cantidad > ALL (
    SELECT Cantidad
    FROM Entregan
    WHERE RFC = 'RFC123'
);
```



El Operador TOP, es un operador que recorre la entrada, un query, y sólo devuelve el primer número o porcentaje especifico de filas basado en un criterio de ordenación si es posible.

¿Qué hace la siguiente sentencia? Explica por qué. Devuelve las dos primeras filas de la tabla Proyectos. Esto porque se le define TOP 2, o sea que quiere que devuelva sólo los priemros dos números o porcentjaes específicos de la tabla. Aunque eso sí, requiere un orden definido para que el resultado tenga sentido lógico. (Posdata, no hay imagen de eso por que se supone que Top es exclusivo para SQL Server y no es compatible con MariaDB).

SELECT TOP 2 * FROM Proyectos

¿Qué sucede con la siguiente consulta? Explica por qué. Marca error porque TOP espera un número literal (como jun 1, 5 o un 10 por ejemplo) o una expresión constante**,** no un nombre de columna**.**

SELECT TOP Numero FROM Proyectos

Modificando la estructura de un tabla existente.

Agrega a la tabla materiales la columna PorcentajeImpuesto con la instrucción:
ALTER TABLE materiales ADD PorcentajeImpuesto NUMERIC(6,2);
A fin de que los materiales tengan un impuesto, les asignaremos impuestos ficticios basados en sus claves con la instrucción:
UPDATE materiales SET PorcentajeImpuesto = 2*clave/1000;
esto es, a cada material se le asignará un impuesto igual al doble de su clave dividida entre diez.

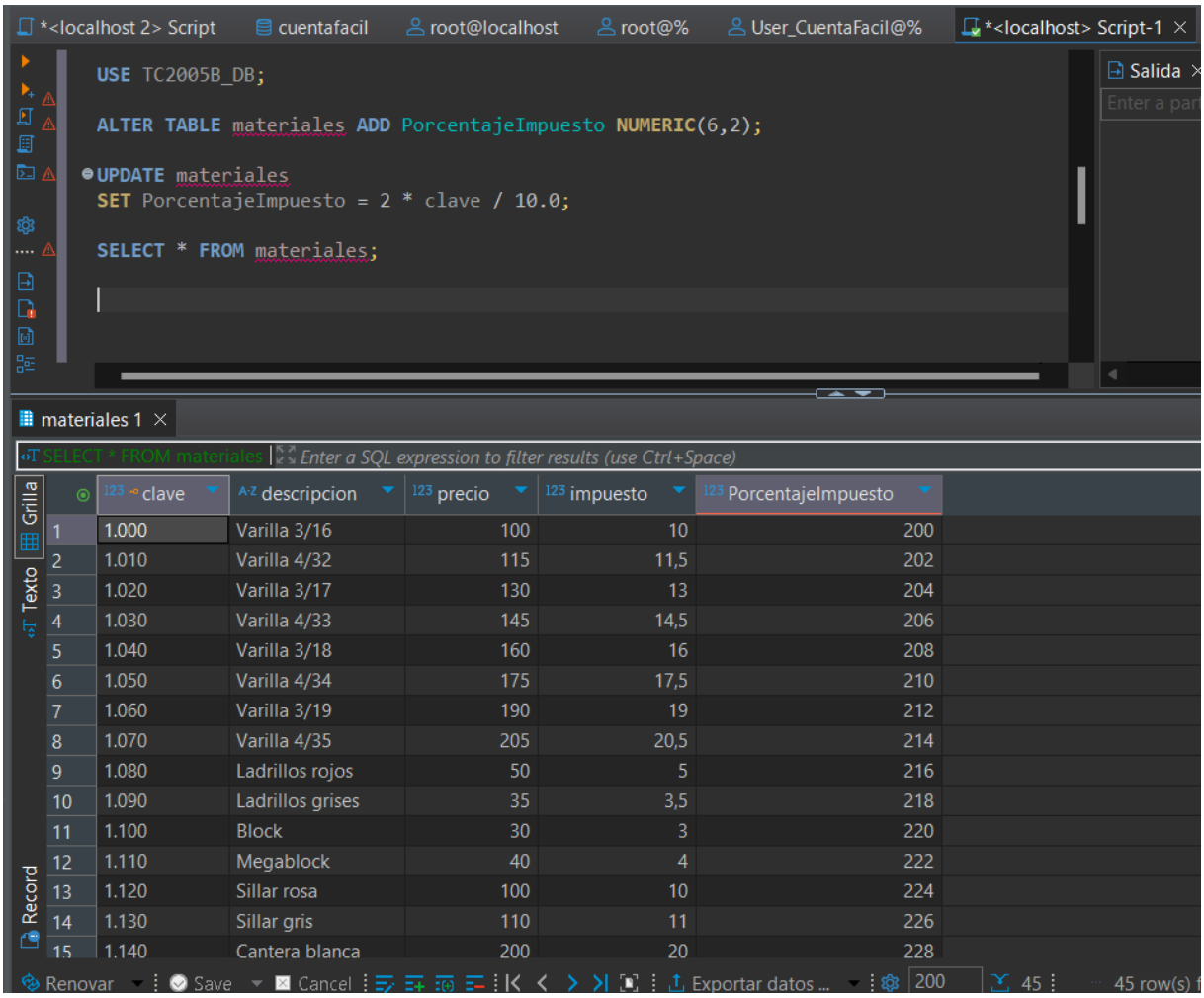
Listo, ya la tabla y la update fueron creadas usando el siguiente script:

ALTER TABLE materiales ADD PorcentajeImpuesto NUMERIC(6,2);

UPDATE materiales

SET PorcentajeImpuesto = 2 * clave / 10.0;

Revisa la tabla de materiales para que compruebes lo que hicimos anteriormente. El resultado al revisar la tabla fue:



¿Qué consulta usarías para obtener el importe de las entregas es decir, el total en dinero de lo entregado, basado en la cantidad de la entrega y el precio del material y el impuesto asignado? Usaría la siguiente consulta:

```
SELECT e.clave, m.descripcion, e.cantidad, m.precio, m.porcentajeImpuesto,
    e.cantidad * m.precio * (1 + m.porcentajeImpuesto / 100) AS ImporteTotal
FROM entregan e
JOIN materiales m ON e.clave = m.clave;
```

```
USE TC2005B_DB;

SELECT e.clave, m.descripcion, e.cantidad, m.precio, m.porcentajeImpuesto,
       e.cantidad * m.precio * (1 + m.porcentajeImpuesto / 100) AS ImporteTotal
FROM entregan e
JOIN materiales m ON e.clave = m.clave;
```

entregan(+) 1

	clave	descripcion	cantidad	precio	porcentajeImpuesto	ImporteTotal
1	1.000	Varilla 3/16	165	100	200	49.500
2	1.000	Varilla 3/16	254	100	200	76.200
3	1.010	Varilla 4/32	528	115	202	183.374,4
4	1.010	Varilla 4/32	523	115	202	181.637,9
5	1.020	Varilla 3/17	582	130	204	230.006,4
6	1.020	Varilla 3/17	8	130	204	3.161,6
7	1.030	Varilla 4/33	202	145	206	89.627,4
8	1.030	Varilla 4/33	295	145	206	130.891,5
9	1.040	Varilla 3/18	263	160	208	129.606,4
10	1.040	Varilla 3/18	540	160	208	266.112
11	1.050	Varilla 4/34	503	175	210	272.877,5
12	1.050	Varilla 4/34	623	175	210	337.977,5
13	1.060	Varilla 3/19	324	190	212	192.067,2
14	1.060	Varilla 3/19	692	190	212	410.217,6
15	1.070	Varilla 4/35	2	205	214	1.287,4

Creación de vistas

La sentencia:

Create view nombrevista (nombrecolumna1 , nombrecolumna2 ,..., nombrecolumna3)
as select...

Permite definir una vista. Una vista puede pensarse como una consulta etiquetada con un nombre, ya que en realidad al referirnos a una vista el DBMS realmente ejecuta la consulta asociada a ella, pero por la cerradura del álgebra relacional, una consulta puede ser vista como una nueva relación o tabla, por lo que es perfectamente válido emitir la sentencia:

select * from nombrevista

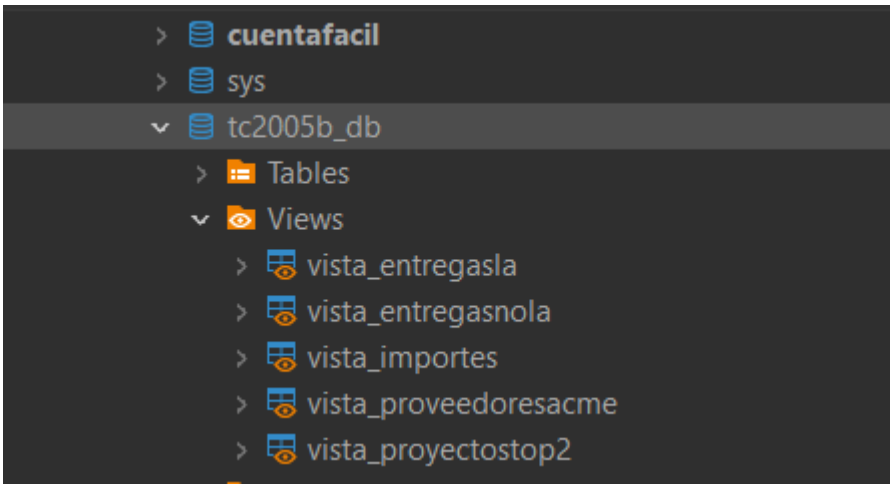
¡Como si nombrevista fuera una tabla!

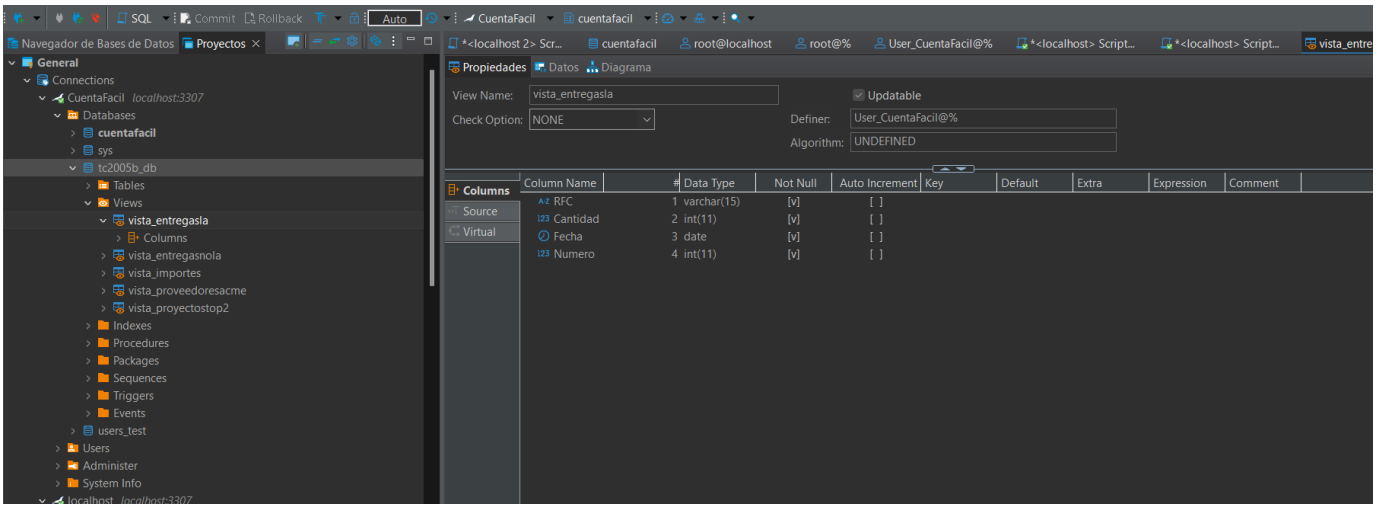
Comprueba lo anterior, creando vistas para cinco de las consultas que planteaste anteriormente en la práctica . Posteriormente revisa cada vista creada para comprobar que devuelve el mismo resultado.

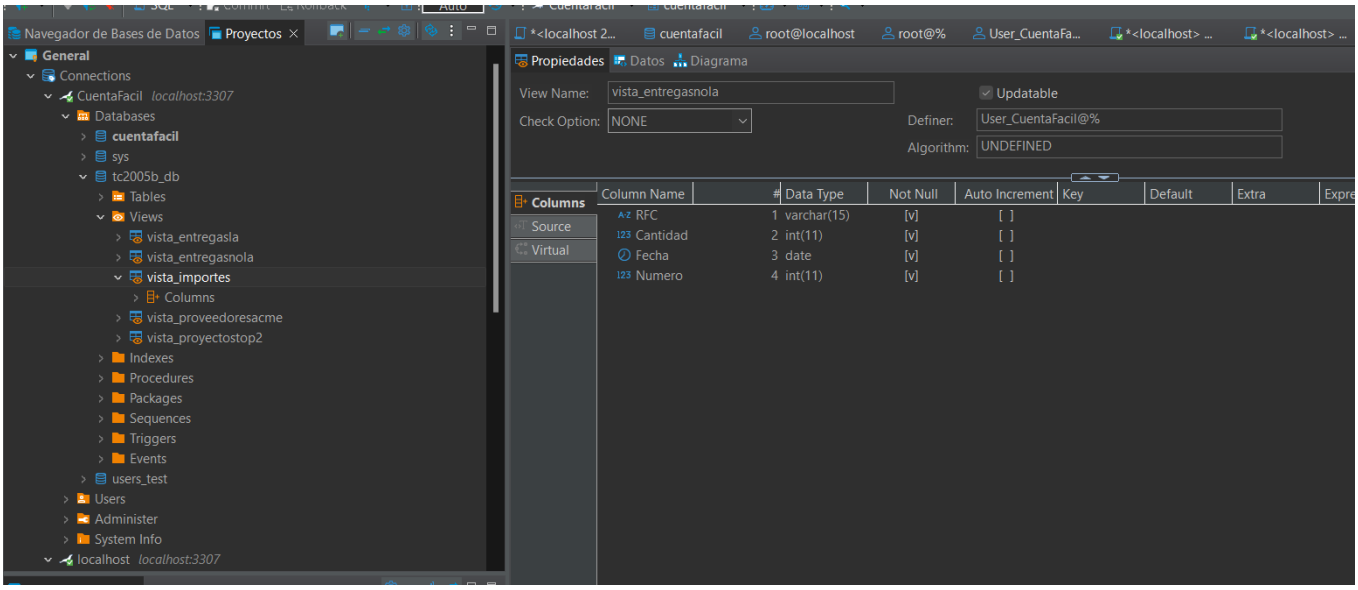
Las vistas las cree de la siguiente manera:

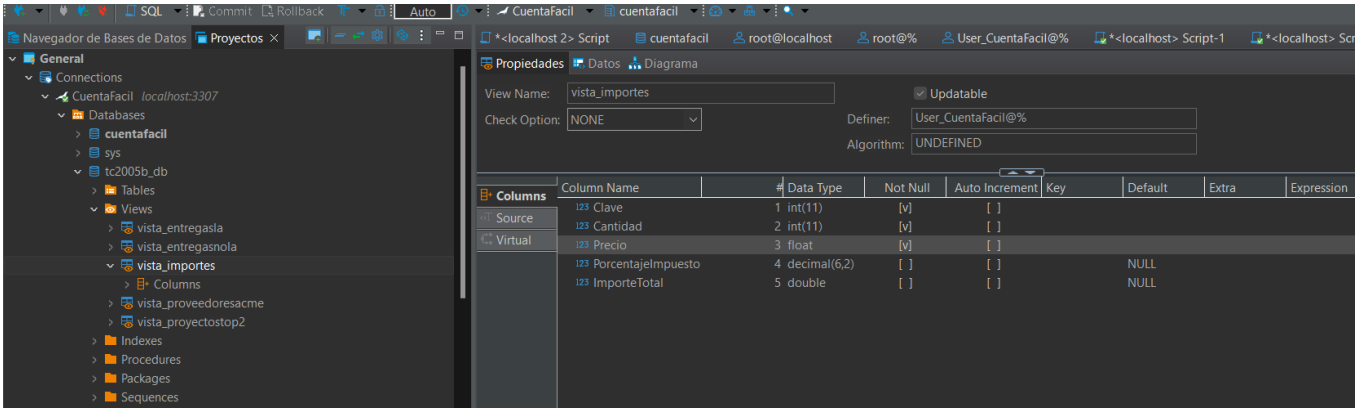
```
USE TC2005B_DB;

CREATE VIEW Vista_EntregasLa AS
SELECT RFC, Cantidad, Fecha, Numero
FROM Entregan
WHERE Numero BETWEEN 5000 AND 5010
  AND RFC IN (
      SELECT RFC
      FROM Proveedores
      WHERE RazonSocial LIKE 'La%'
  );

CREATE VIEW Vista_EntregasNoLa AS
SELECT RFC, Cantidad, Fecha, Numero
FROM Entregan
WHERE Numero BETWEEN 5000 AND 5010
  AND RFC NOT IN (
      SELECT RFC
      FROM Proveedores
      WHERE RazonSocial LIKE 'La%'
  );

CREATE VIEW Vista_Importes AS
SELECT e.Clave, e.Cantidad, m.Precio, m.PorcentajeImpuesto,
       e.Cantidad * m.Precio * (1 + m.PorcentajeImpuesto / 100) AS ImporteTotal
FROM Entregan e
JOIN Materiales m ON e.Clave = m.Clave;

CREATE VIEW Vista_ProyectosTop2 AS
SELECT *
FROM Proyectos
LIMIT 2;

CREATE VIEW Vista_ProveedoresAcme AS
SELECT DISTINCT m.Clave, m.Descripcion
FROM Materiales m
JOIN Entregan e ON m.Clave = e.Clave
JOIN Proveedores p ON e.RFC = p.RFC
WHERE p.RazonSocial = 'Acme tools';
```

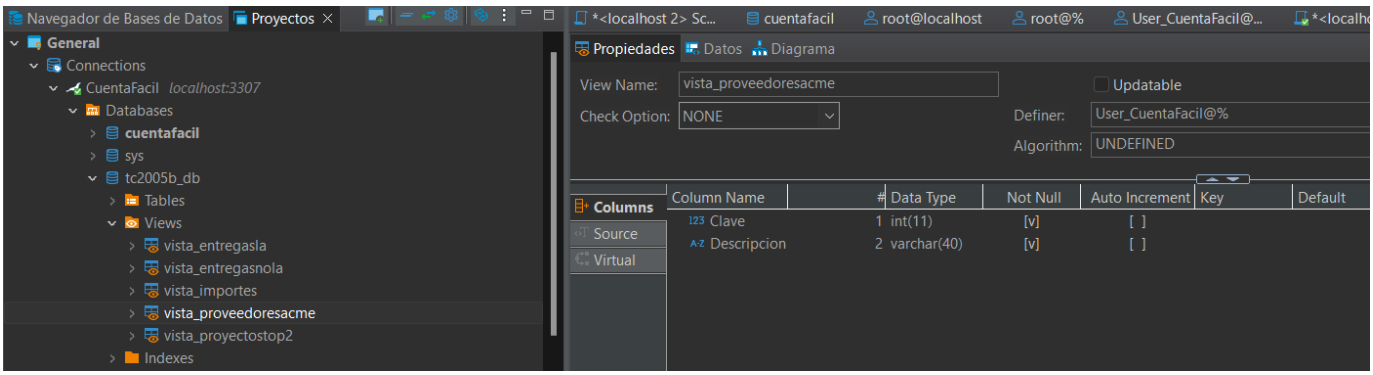
Y los resultados fueron:

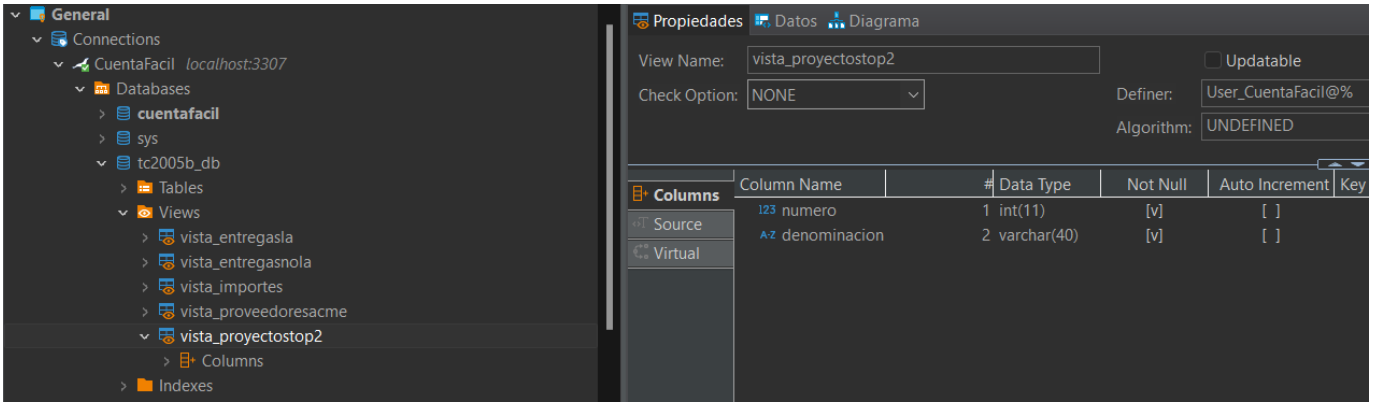

cuentafacil
sys
tc2005b_db
Tables
Views
vista_entregasla
vista_entregasnola
vista_importes
vista_proveedoresacme
vista_proyectostop2


Navegador de Bases de Datos
Proyectos
General
Connections
CuentaFacil localhost:3307
Databases
cuentafacil
sys
tc2005b_db
Tables
Views
vista_entregasla
Columns
vista_entregasnola
vista_importes
vista_proveedoresacme
vista_proyectostop2
Indexes
Procedures
Packages
Sequences
Triggers
Events
users_test
Users
Administer
System Info
Propiedades
Datos
Diagrama
View Name: vista_entregasla
Check Option: NONE
Updatable
Definer: User_CuentaFacil@%
Algorithm: UNDEFINED
Columns
Source
Virtual
Column Name # Data Type Not Null Auto Increment Key Default Extra Expression Comment
RFC 1 varchar(15) [v] [ ]
Cantidad 2 int(11) [v] [ ]
Fecha 3 date [v] [ ]
Numero 4 int(11) [v] [ ]


Navegador de Bases de Datos
Proyectos
General
Connections
CuentaFacil localhost:3307
Databases
cuentafacil
sys
tc2005b_db
Tables
Views
vista_entregasla
vista_entregasnola
vista_importes
Columns
vista_proveedoresacme
vista_proyectostop2
Indexes
Procedures
Packages
Sequences
Triggers
Events
users_test
Users
Administer
System Info
localhost localhost:3307
Propiedades
Datos
Diagrama
View Name: vista_entregasnola
Check Option: NONE
Updatable
Definer: User_CuentaFacil@%
Algorithm: UNDEFINED
Columns
Source
Virtual
Column Name # Data Type Not Null Auto Increment Key Default Extra
RFC 1 varchar(15) [v] [ ]
Cantidad 2 int(11) [v] [ ]
Fecha 3 date [v] [ ]
Numero 4 int(11) [v] [ ]







La parte (nombrecolumna1,nombrecolumna2,.de la sentencia create view puede ser omitida si no hay ambigüedad en los nombres de las columnas de la sentencia select asociada.

Importante: Las vistas no pueden incluir la cláusula order by.

A continuación se te dan muchos enunciados de los cuales deberás generar su correspondiente consulta.

En el reporte incluye la sentencia, una muestra de la salida (dos o tres renglones) y el número de renglones que SQL Server reporta al final de la consulta.

Los materiales (clave y descripción) entregados al proyecto "México sin ti no estamos completos".

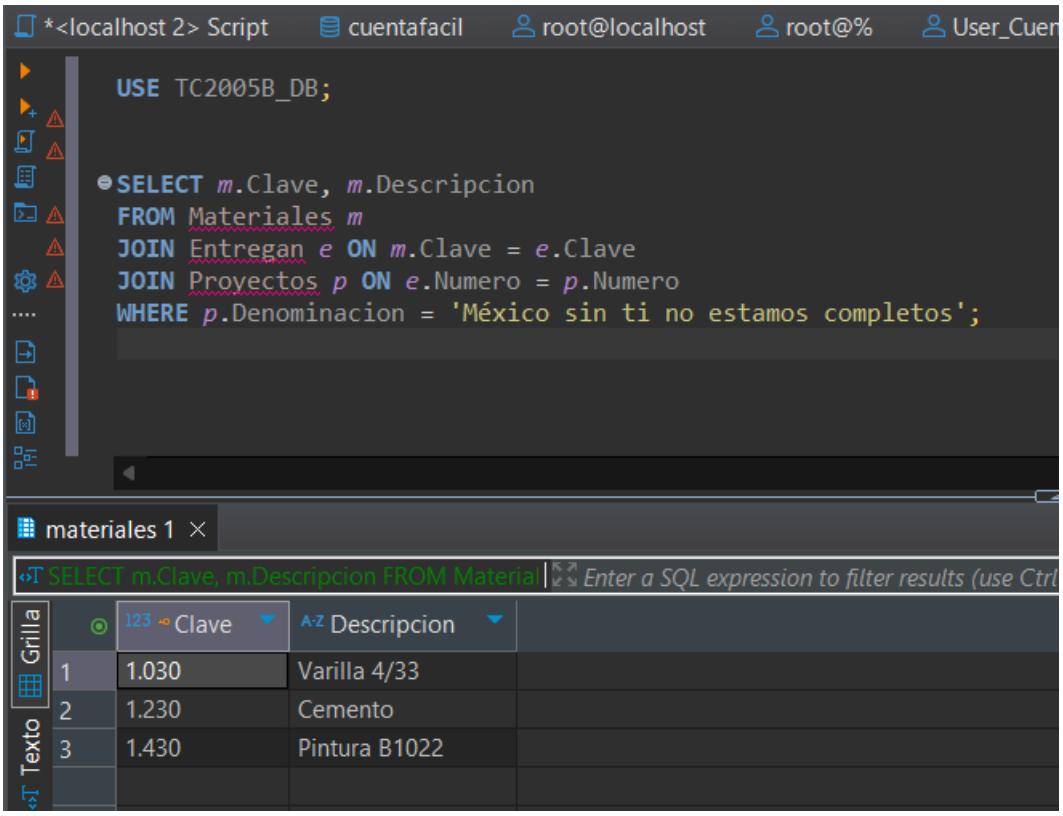
SELECT m.Clave, m.Descripcion

FROM Materiales m

JOIN Entregan e ON m.Clave = e.Clave

JOIN Proyectos p ON e.Numero = p.Numero

WHERE p.Denominacion = 'México sin ti no estamos completos';



Los materiales (clave y descripción) que han sido proporcionados por el proveedor "Acme tools".

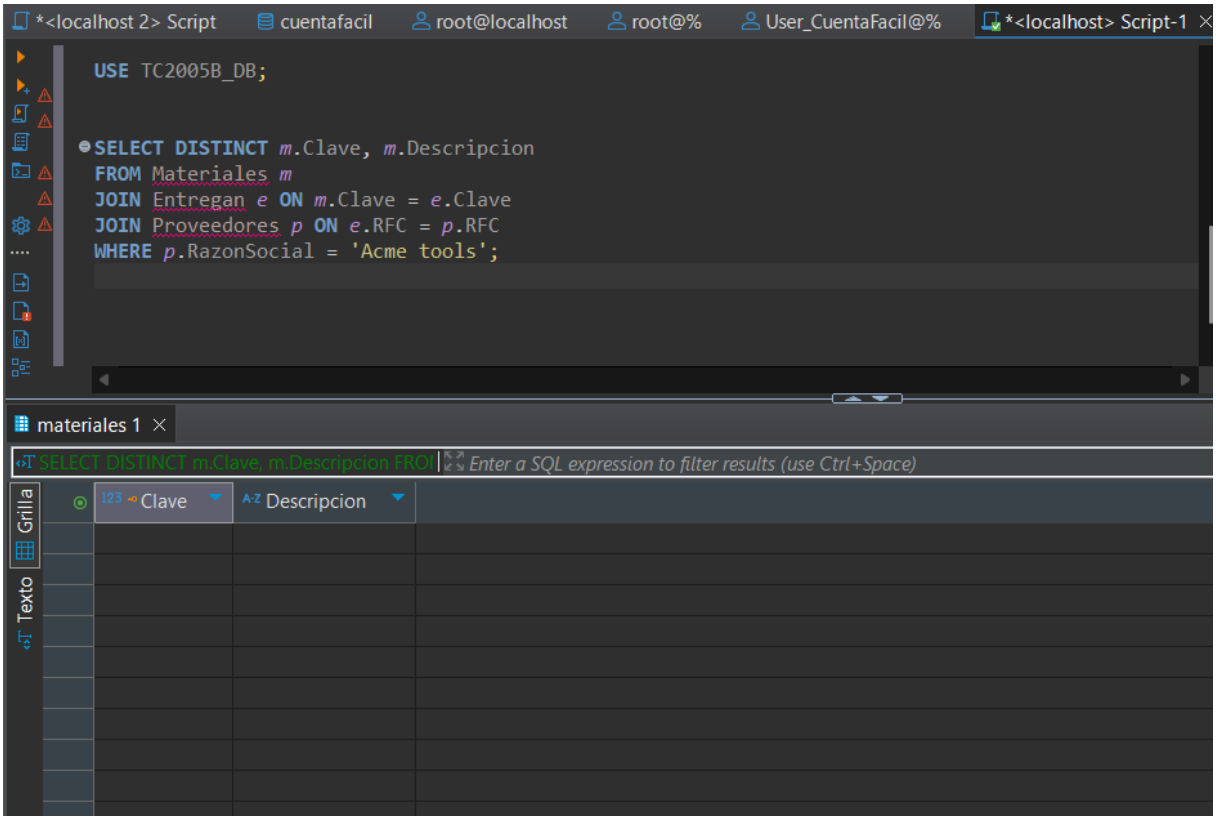
SELECT DISTINCT m.Clave, m.Descripcion

FROM Materiales m

JOIN Entregan e ON m.Clave = e.Clave

JOIN Proveedores p ON e.RFC = p.RFC

WHERE p.RazonSocial = 'Acme tools';



El RFC de los proveedores que durante el 2000 entregaron en promedio cuando menos 300 materiales.

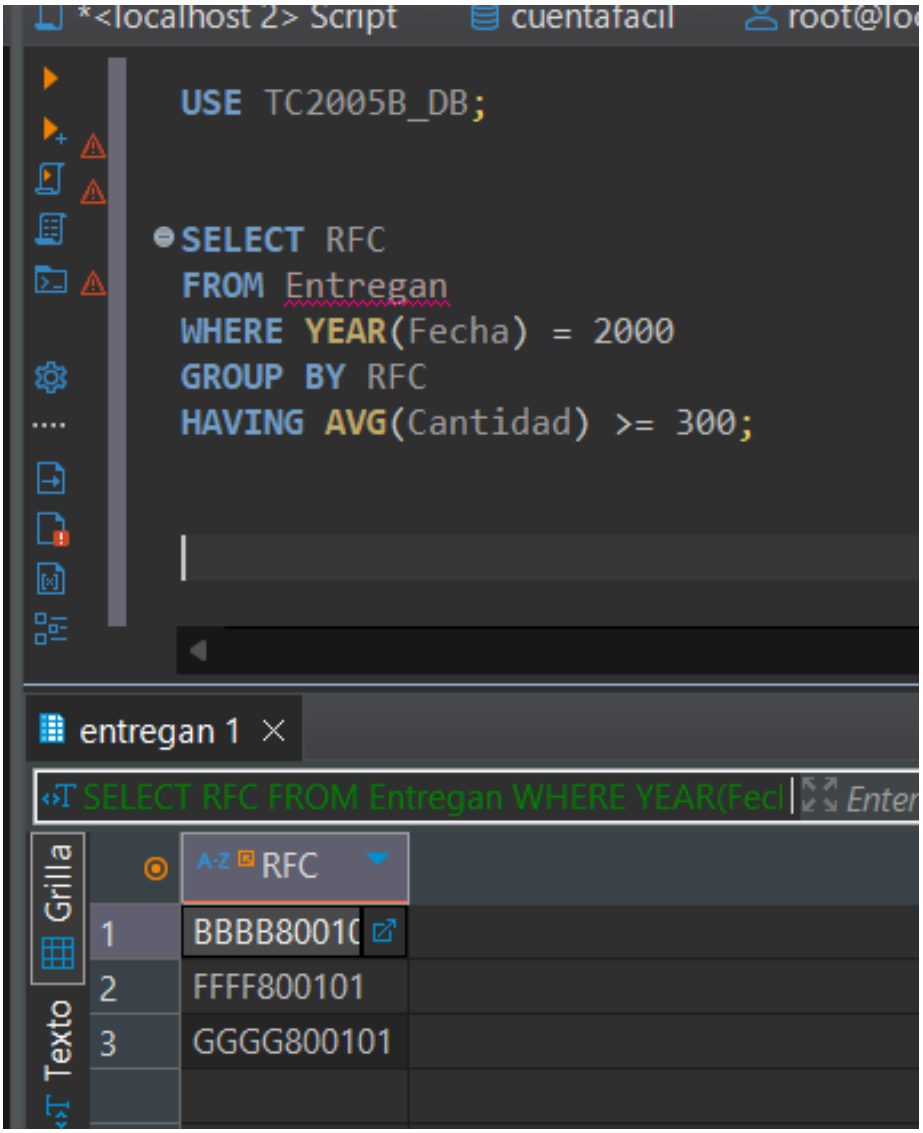
SELECT RFC

FROM Entregan

WHERE YEAR(Fecha) = 2000

GROUP BY RFC

HAVING AVG(Cantidad) >= 300;



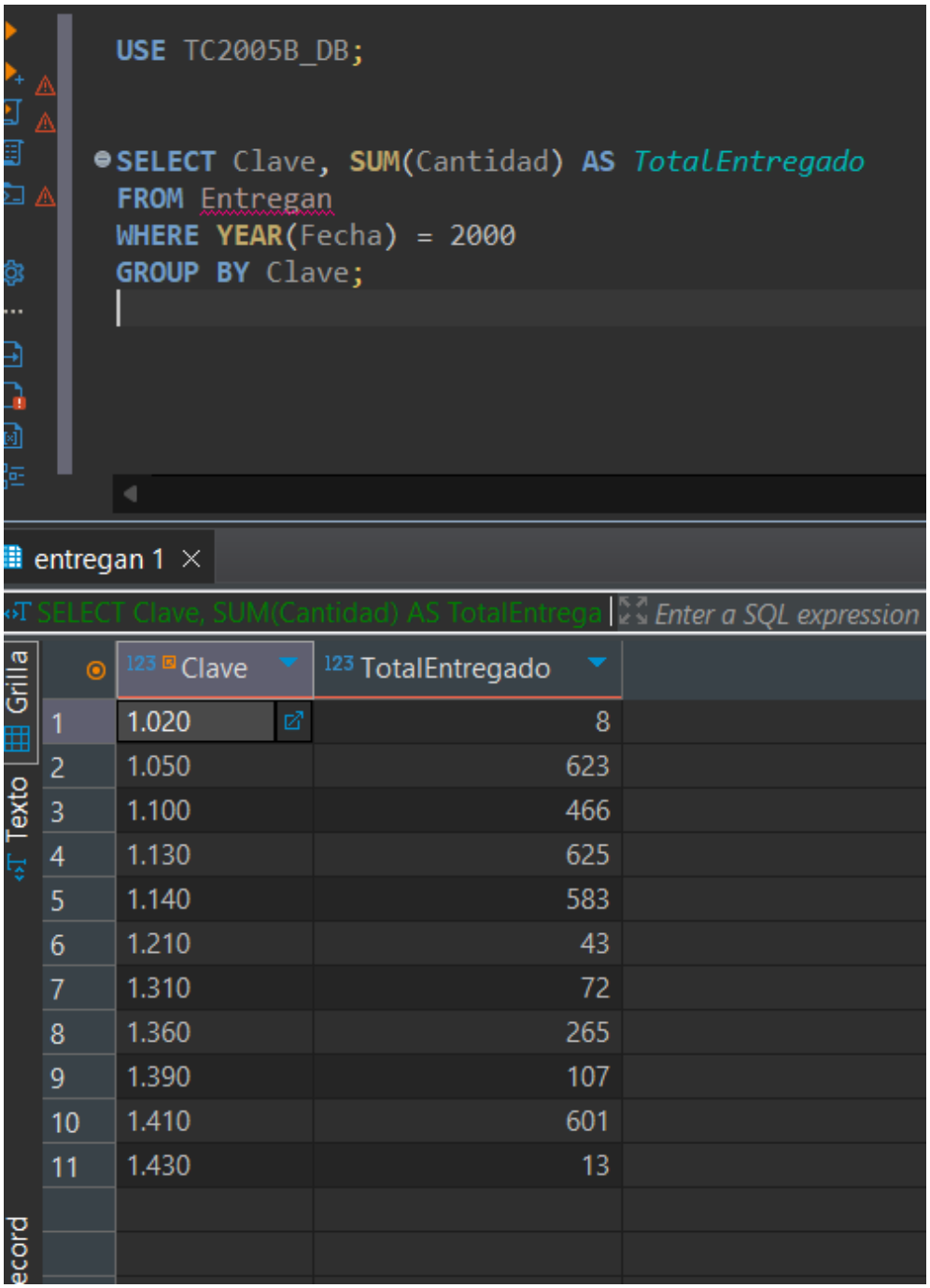
El Total entregado por cada material en el año 2000.

SELECT Clave, SUM(Cantidad) AS TotalEntregado

FROM Entregan

WHERE YEAR(Fecha) = 2000

GROUP BY Clave;



La Clave del material más vendido durante el 2001. (se recomienda usar una vista intermedia para su solución)

CREATE VIEW VistaVentas2001 AS

SELECT Clave, SUM(Cantidad) AS Total

FROM Entregan

WHERE YEAR(Fecha) = 2001

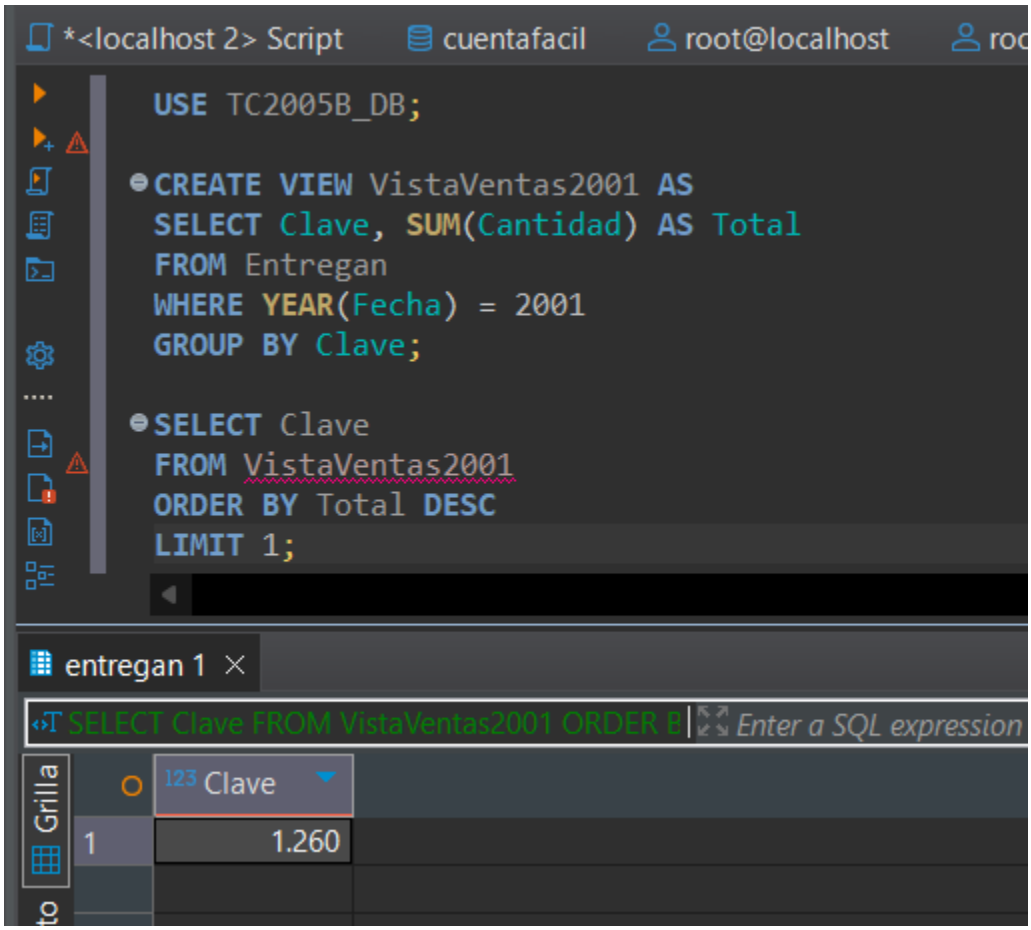
GROUP BY Clave;

SELECT Clave

FROM VistaVentas2001

ORDER BY Total DESC

LIMIT 1;

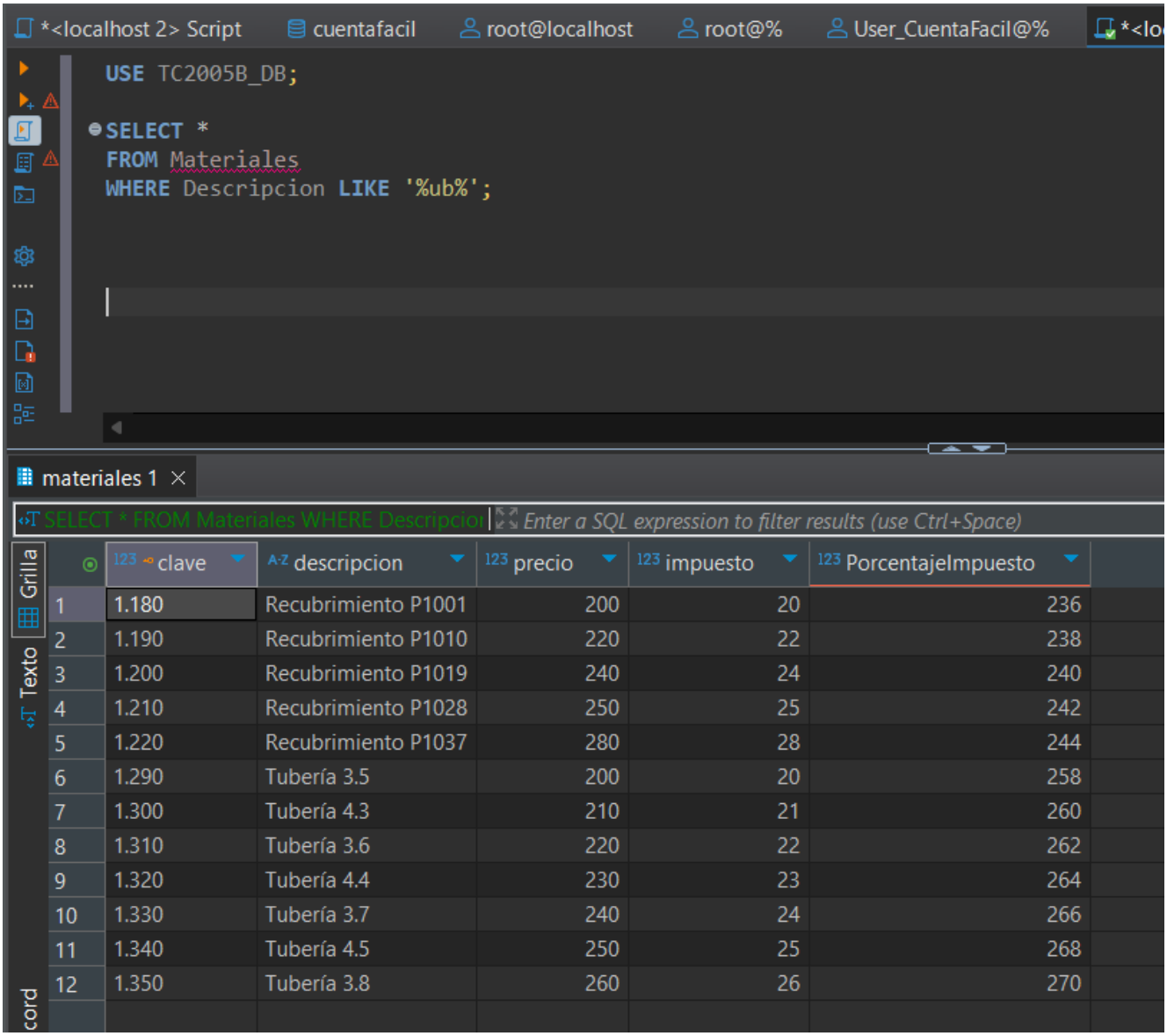


Productos que contienen el patrón 'ub' en su nombre.

SELECT *

FROM Materiales

WHERE Descripcion LIKE '%ub%';



Denominación y suma del total a pagar para todos los proyectos.

SELECT p.Denominacion,

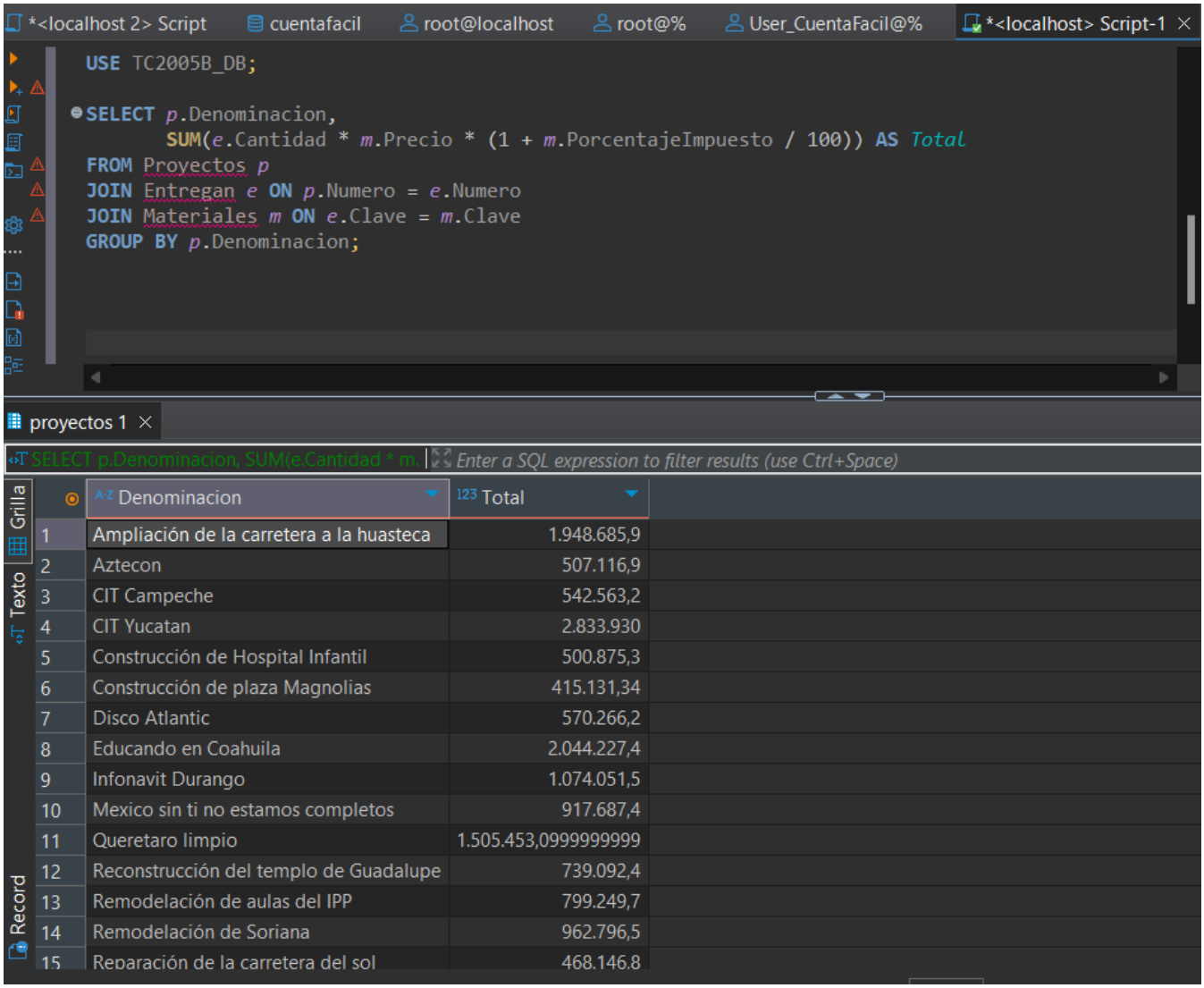
SUM(e.Cantidad * m.PrecioUnitario * (1 + m.PorcentajeImpuesto / 100)) AS Total

FROM Proyectos p

JOIN Entregan e ON p.Numero = e.Numero

JOIN Materiales m ON e.Clave = m.Clave

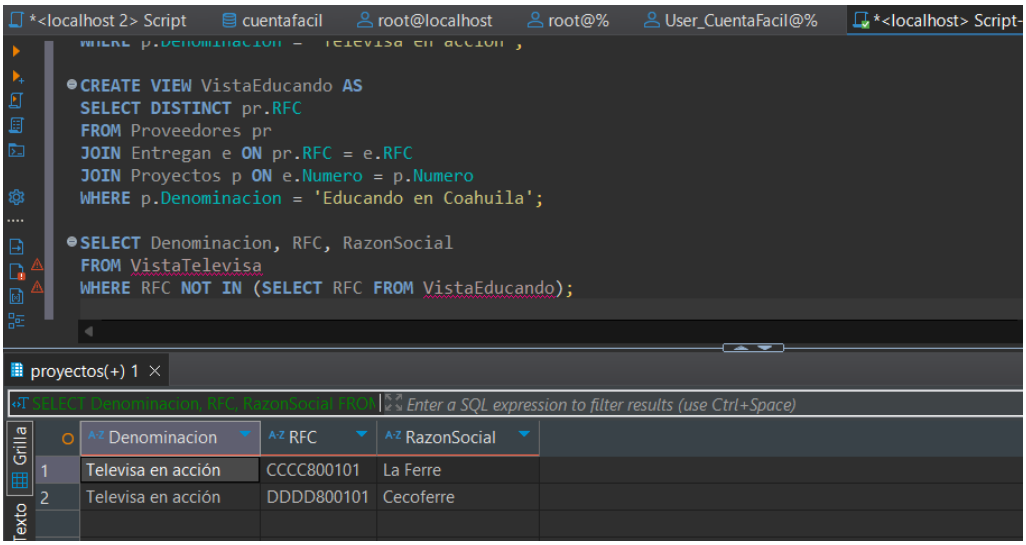
GROUP BY p.Denominacion;



	Denominacion	Total
1	Ampliación de la carretera a la huasteca	1.948.685,9
2	Aztecon	507.116,9
3	CIT Campeche	542.563,2
4	CIT Yucatan	2.833.930
5	Construcción de Hospital Infantil	500.875,3
6	Construcción de plaza Magnolias	415.131,34
7	Disco Atlantic	570.266,2
8	Educando en Coahuila	2.044.227,4
9	Infonavit Durango	1.074.051,5
10	Mexico sin ti no estamos completos	917.687,4
11	Queretaro limpio	1.505.453,0999999999
12	Reconstrucción del templo de Guadalupe	739.092,4
13	Remodelación de aulas del IPP	799.249,7
14	Remodelación de Soriana	962.796,5
15	Reparación de la carretera del sol	468.146,8

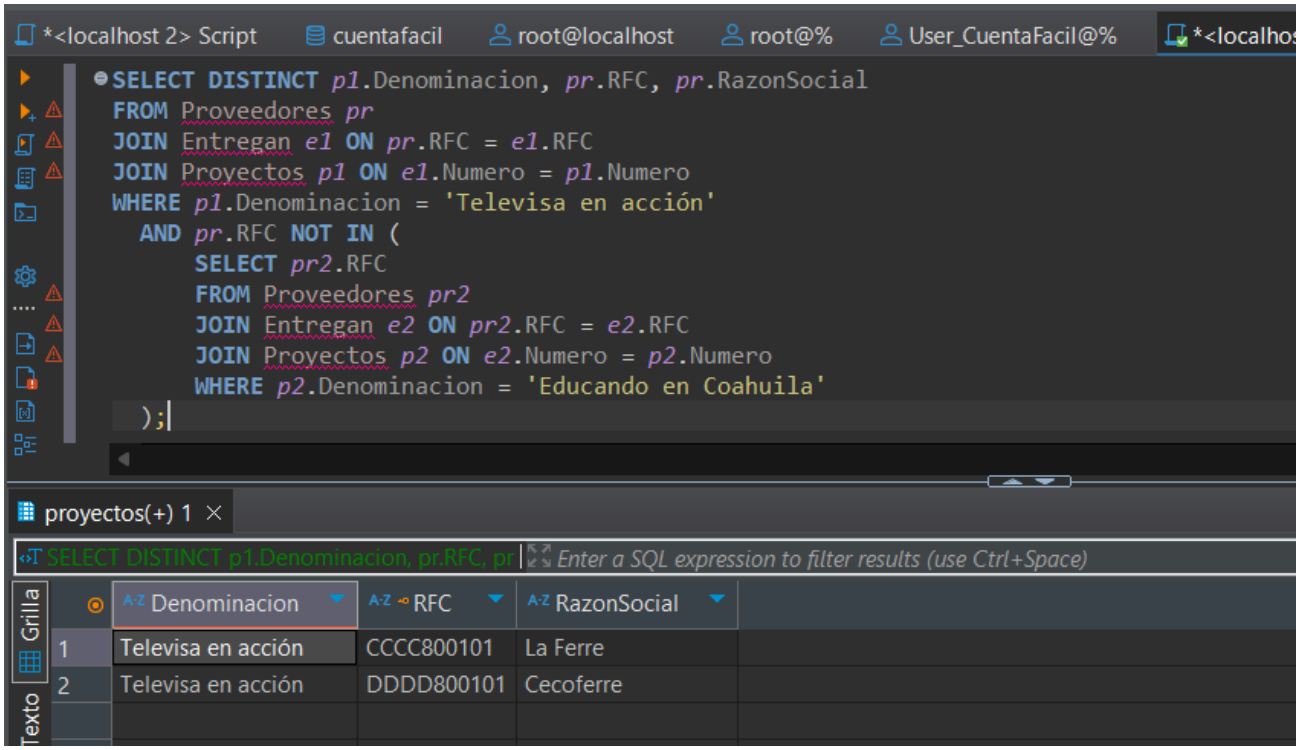
Denominación, RFC y RazonSocial de los proveedores que se suministran materiales al proyecto Televisa en acción que no se encuentran apoyando al proyecto Educando en Coahuila (Solo usando vistas).

```
CREATE VIEW VistaTelevisa AS

SELECT DISTINCT p.Denominacion, pr.RFC, pr.RazonSocial

FROM Proveedores pr

JOIN Entregan e ON pr.RFC = e.RFC

JOIN Proyectos p ON e.Numero = p.Numero

WHERE p.Denominacion = 'Televisa en acción';


CREATE VIEW VistaEducando AS

SELECT DISTINCT pr.RFC

FROM Proveedores pr

JOIN Entregan e ON pr.RFC = e.RFC

JOIN Proyectos p ON e.Numero = p.Numero

WHERE p.Denominacion = 'Educando en Coahuila';


SELECT Denominacion, RFC, RazonSocial

FROM VistaTelevisa

WHERE RFC NOT IN (SELECT RFC FROM VistaEducando);
```



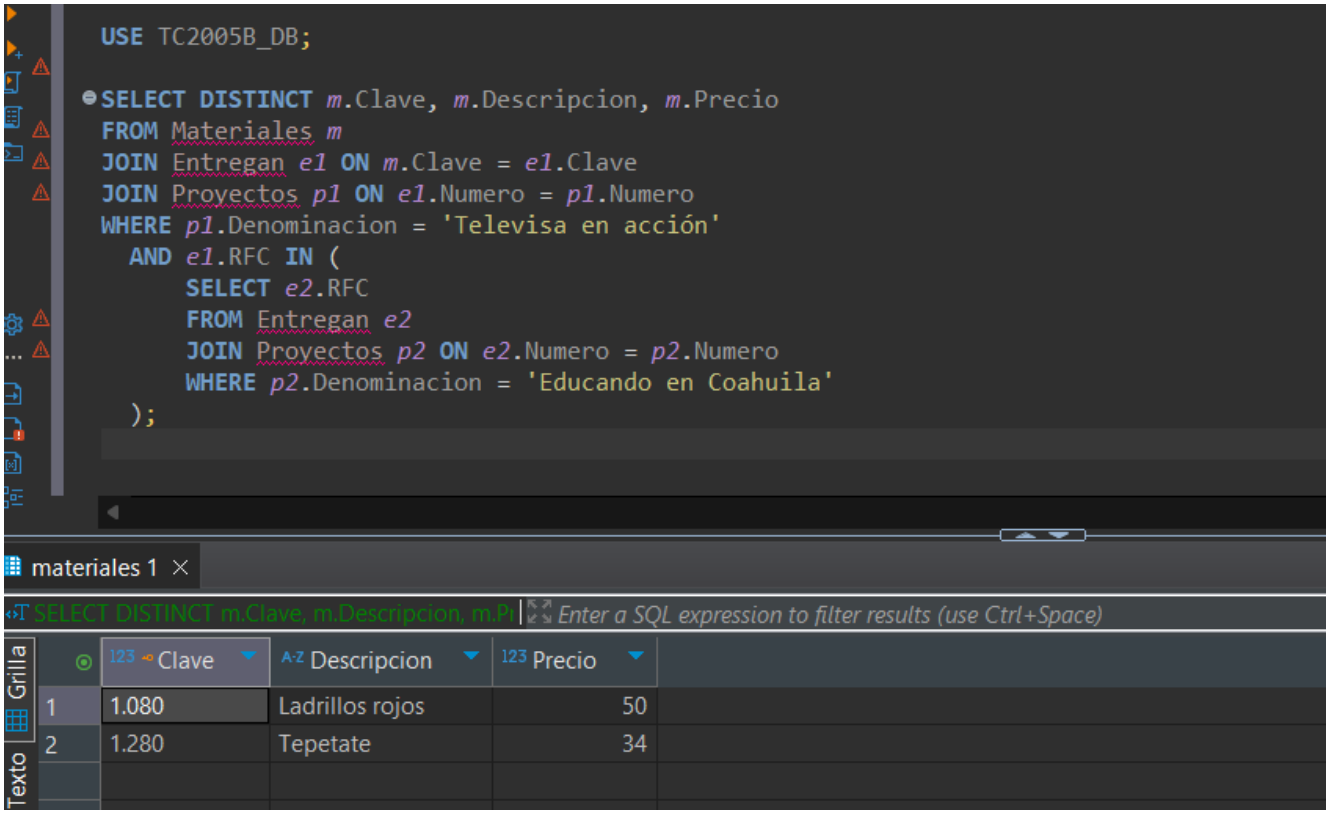
Denominación, RFC y RazonSocial de los proveedores que se suministran materiales al proyecto Televisa en acción que no se encuentran apoyando al proyecto Educando en Coahuila (Sin usar vistas, utiliza not in, in o exists).

```
SELECT DISTINCT p1.Denominacion, pr.RFC, pr.RazonSocial
FROM Proveedores pr
JOIN Entregan e1 ON pr.RFC = e1.RFC
JOIN Proyectos p1 ON e1.Numero = p1.Numero
WHERE p1.Denominacion = 'Televisa en acción'
  AND pr.RFC NOT IN (
      SELECT pr2.RFC
      FROM Proveedores pr2
      JOIN Entregan e2 ON pr2.RFC = e2.RFC
      JOIN Proyectos p2 ON e2.Numero = p2.Numero
      WHERE p2.Denominacion = 'Educando en Coahuila'
  );
```

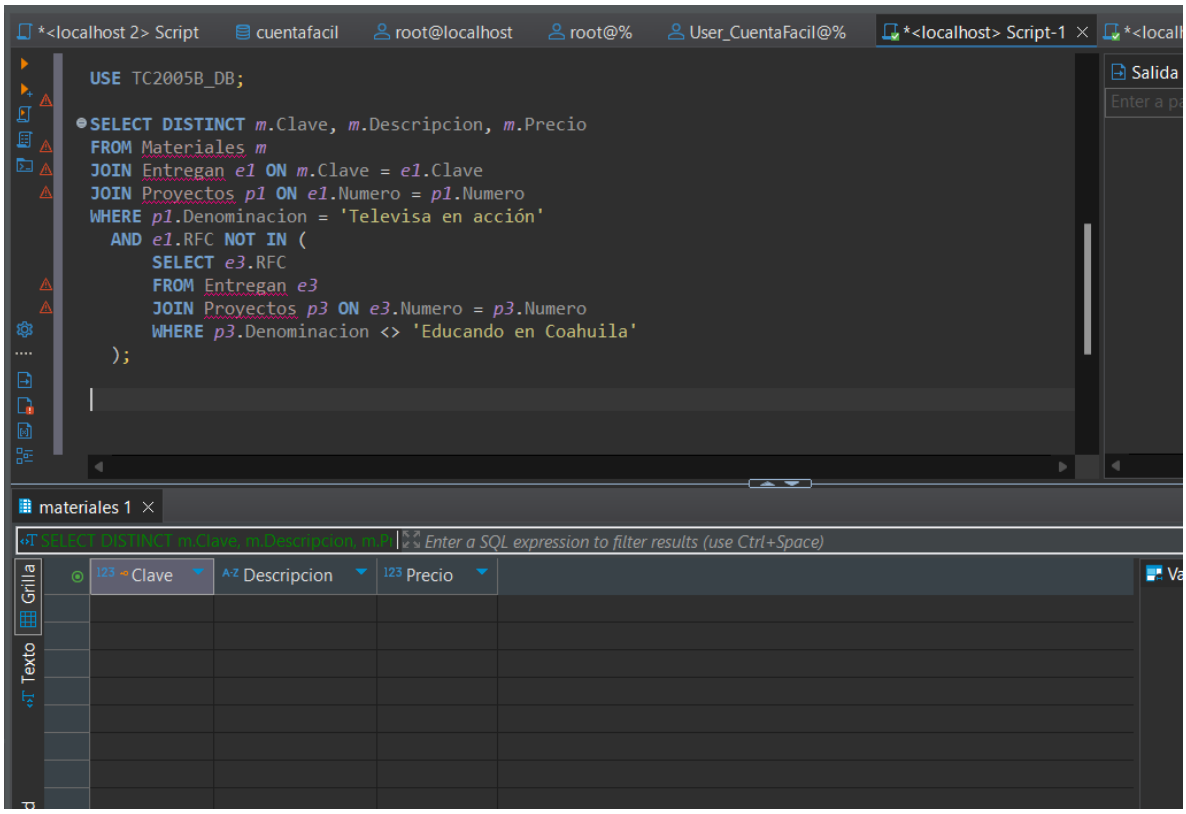


Costo de los materiales y los Materiales que son entregados al proyecto Televisa en acción cuyos proveedores también suministran materiales al proyecto Educando en Coahuila.

```
SELECT DISTINCT m.Clave, m.Descripcion, m.PrecioUnitario
FROM Materiales m
JOIN Entregan e1 ON m.Clave = e1.Clave
JOIN Proyectos p1 ON e1.Numero = p1.Numero
WHERE p1.Denominacion = 'Televisa en acción'
  AND e1.RFC IN (
      SELECT e2.RFC
      FROM Entregan e2
      JOIN Proyectos p2 ON e2.Numero = p2.Numero
      WHERE p2.Denominacion = 'Educando en Coahuila'
  );
```



Reto: Usa solo el operador NOT IN en la consulta anterior (No es parte de la entrega).

Nombre del material, cantidad de veces entregados y total del costo de dichas entregas por material de todos los proyectos.

```
SELECT DISTINCT m.Clave, m.Descripcion, m.Precio
FROM Materiales m
JOIN Entregan e1 ON m.Clave = e1.Clave
JOIN Proyectos p1 ON e1.Numero = p1.Numero
WHERE p1.Denominacion = 'Televisa en acción'
  AND e1.RFC NOT IN (
      SELECT e3.RFC
      FROM Entregan e3
      JOIN Proyectos p3 ON e3.Numero = p3.Numero
      WHERE p3.Denominacion <> 'Educando en Coahuila'
  );
```