

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

	clave	descripcion	precio	impuesto
1	1.000	Varilla 3/16	100	10
2	1.010	Varilla 4/32	115	11,5
3	1.020	Varilla 3/17	130	13
4	1.030	Varilla 4/33	145	14,5
5	1.040	Varilla 3/18	160	16
6	1.050	Varilla 4/34	175	17,5
7	1.060	Varilla 3/19	190	19
8	1.070	Varilla 4/35	205	20,5
9	1.080	Ladrillos rojos	50	5
10	1.090	Ladrillos grises	35	3,5
11	1.100	Block	30	3
12	1.110	Megablock	40	4
13	1.120	Sillar rosa	100	10
14	1.130	Sillar gris	110	11
15	1.140	Cantera blanca	200	20

Selección

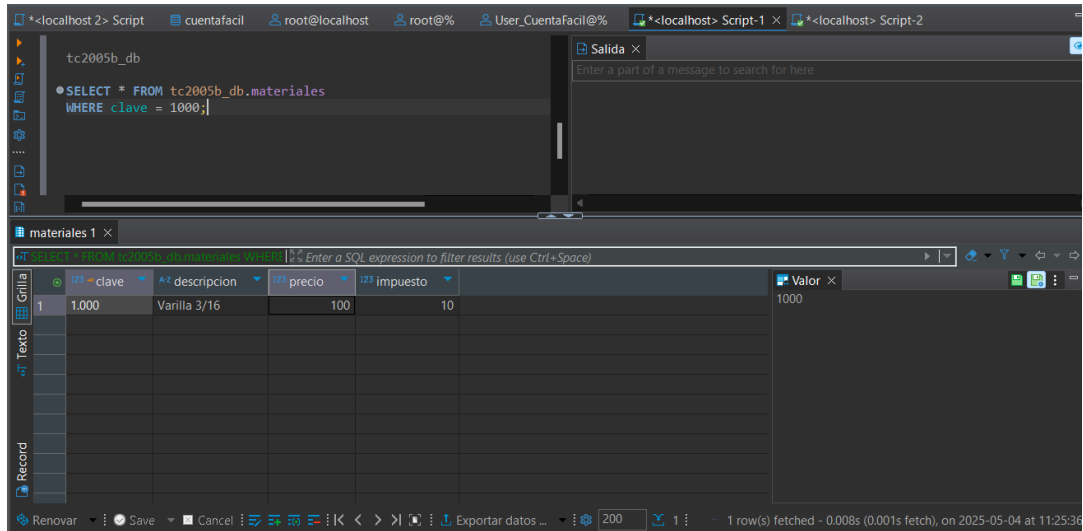
Algebra relacional.

$SL_{\{clave=1000\}}(materiales)$

SQL

select * from materiales

where clave=1000



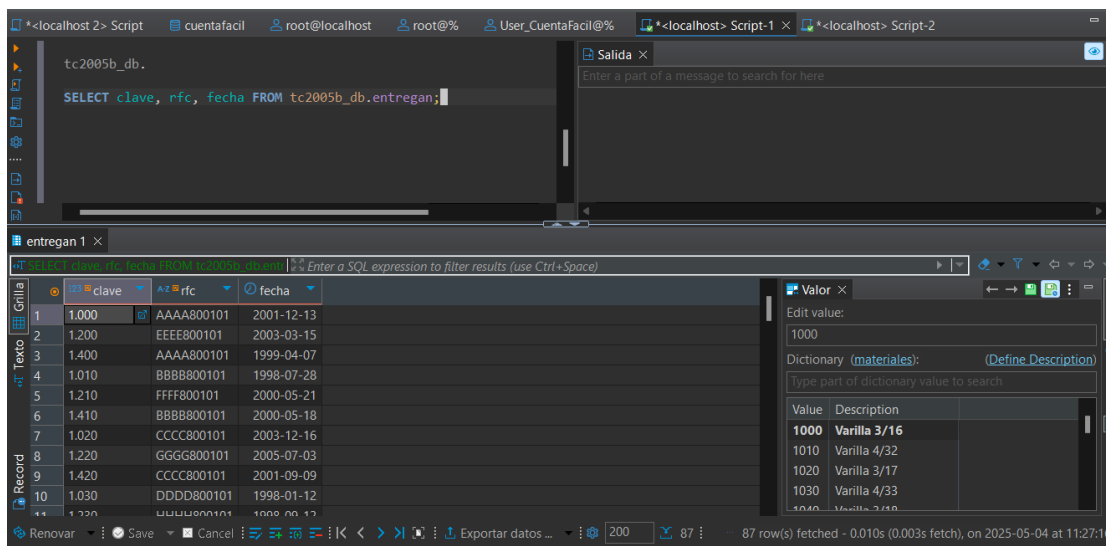
Proyección

Algebra relacional.

$PR_{\{clave,rfc,fecha\}}(entregan)$

SQL

select clave,rfc,fecha from entregan



Reunión natural

Algebra relacional.

entregan JN materiales

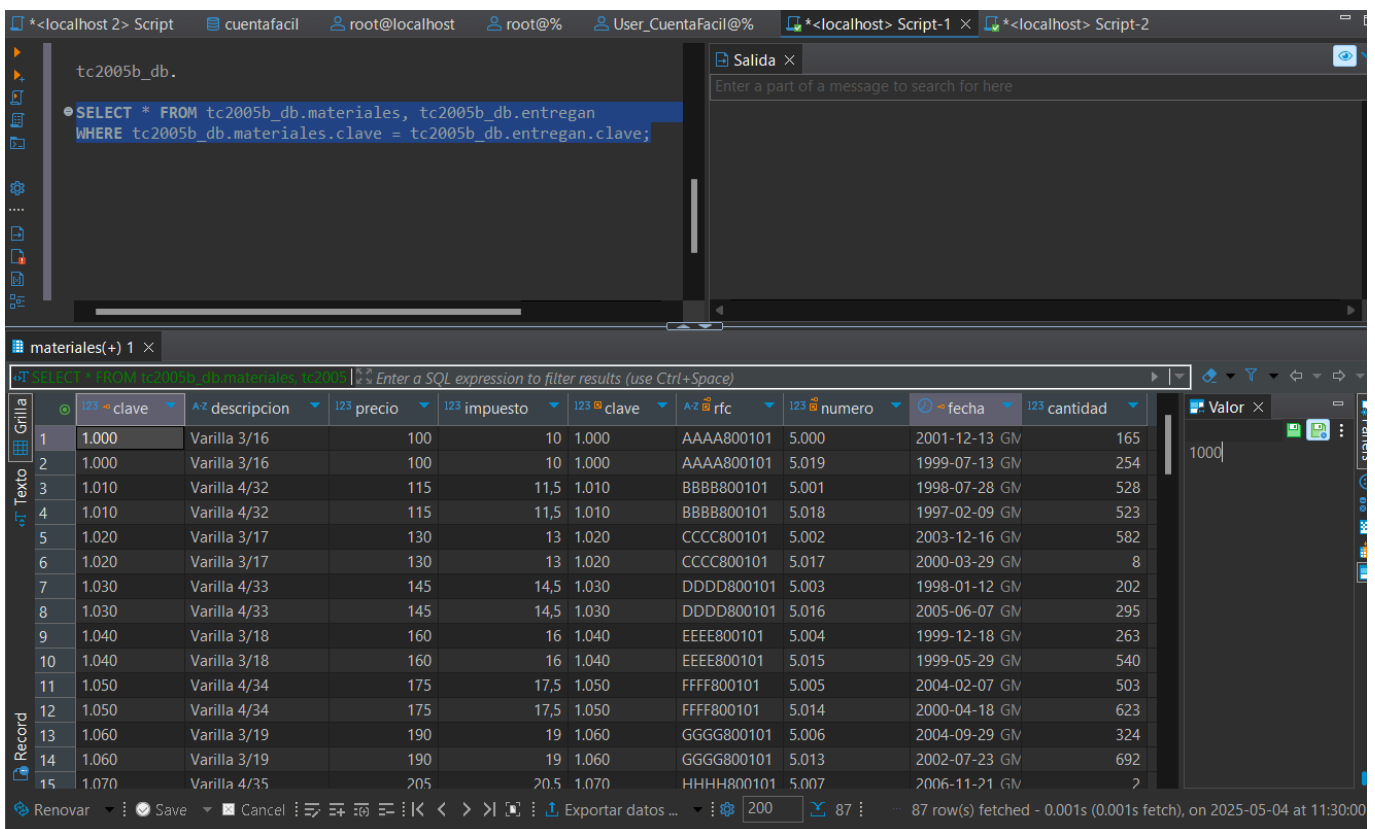
SQL

```
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*.



The screenshot shows a database client interface with a script editor and a results pane. The script editor contains the following SQL query:

```
tc2005b_db.  
SELECT * FROM tc2005b_db.materiales, tc2005b_db.entregan  
WHERE tc2005b_db.materiales.clave = tc2005b_db.entregan.clave;
```

The results pane displays a table with 15 rows and 12 columns. The columns are: clave, descripcion, precio, impuesto, clave, rfc, numero, fecha, cantidad, and Valor. The data is as follows:

	clave	descripcion	precio	impuesto	clave	rfc	numero	fecha	cantidad	Valor
1	1.000	Varilla 3/16	100	10	1.000	AAAA800101	5.000	2001-12-13 GV	165	
2	1.000	Varilla 3/16	100	10	1.000	AAAA800101	5.019	1999-07-13 GV	254	
3	1.010	Varilla 4/32	115	11,5	1.010	BBBB800101	5.001	1998-07-28 GV	528	
4	1.010	Varilla 4/32	115	11,5	1.010	BBBB800101	5.018	1997-02-09 GV	523	
5	1.020	Varilla 3/17	130	13	1.020	CCCC800101	5.002	2003-12-16 GV	582	
6	1.020	Varilla 3/17	130	13	1.020	CCCC800101	5.017	2000-03-29 GV	8	
7	1.030	Varilla 4/33	145	14,5	1.030	DDDD800101	5.003	1998-01-12 GV	202	
8	1.030	Varilla 4/33	145	14,5	1.030	DDDD800101	5.016	2005-06-07 GV	295	
9	1.040	Varilla 3/18	160	16	1.040	EEEE800101	5.004	1999-12-18 GV	263	
10	1.040	Varilla 3/18	160	16	1.040	EEEE800101	5.015	1999-05-29 GV	540	
11	1.050	Varilla 4/34	175	17,5	1.050	FFFF800101	5.005	2004-02-07 GV	503	
12	1.050	Varilla 4/34	175	17,5	1.050	FFFF800101	5.014	2000-04-18 GV	623	
13	1.060	Varilla 3/19	190	19	1.060	GGGG800101	5.006	2004-09-29 GV	324	
14	1.060	Varilla 3/19	190	19	1.060	GGGG800101	5.013	2002-07-23 GV	692	
15	1.070	Varilla 4/35	205	20,5	1.070	HHHH800101	5.007	2006-11-21 GV	2	

The status bar at the bottom indicates: 87 row(s) fetched - 0.001s (0.001s fetch), on 2025-05-04 at 11:30:00.

Reunión con criterio específico

Algebra relacional.

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

SQL

select * from entregan,proyectos

where entregan.numero <= proyectos.numero

The screenshot displays a database management interface with a script editor at the top and a results grid below. The script editor contains the following SQL query:

```
tc2005b_db.  
SELECT * FROM tc2005b_db.entregan, tc2005b_db.proyectos  
WHERE tc2005b_db.entregan.numero <= tc2005b_db.proyectos.numero;
```

The results grid shows 15 records with the following columns: clave, rfc, numero, fecha, cantidad, numero, and denominacion. The data is as follows:

	clave	rfc	numero	fecha	cantidad	numero	denominacion
1	1.000	AAAA800101	5.000	2001-12-13 GM	165	5.000	Vamos Mexico
2	1.200	EEEE800101	5.000	2003-03-15 GM	177	5.000	Vamos Mexico
3	1.400	AAAA800101	5.000	1999-04-07 GM	382	5.000	Vamos Mexico
4	1.000	AAAA800101	5.000	2001-12-13 GM	165	5.001	Aztecon
5	1.200	EEEE800101	5.000	2003-03-15 GM	177	5.001	Aztecon
6	1.400	AAAA800101	5.000	1999-04-07 GM	382	5.001	Aztecon
7	1.010	BBBB800101	5.001	1998-07-28 GM	528	5.001	Aztecon
8	1.210	FFFF800101	5.001	2000-05-21 GM	43	5.001	Aztecon
9	1.410	BBBB800101	5.001	2000-05-18 GM	601	5.001	Aztecon
10	1.000	AAAA800101	5.000	2001-12-13 GM	165	5.002	CIT Campeche
11	1.200	EEEE800101	5.000	2003-03-15 GM	177	5.002	CIT Campeche
12	1.400	AAAA800101	5.000	1999-04-07 GM	382	5.002	CIT Campeche
13	1.010	BBBB800101	5.001	1998-07-28 GM	528	5.002	CIT Campeche
14	1.210	FFFF800101	5.001	2000-05-21 GM	43	5.002	CIT Campeche
15	1.410	BBBB800101	5.001	2000-05-18 GM	601	5.002	CIT Campeche

On the right side, a 'Valor' panel shows a search for '1000' in a dictionary, displaying a list of values and descriptions:

Value	Description
1000	Varilla 3...
1010	Varilla 4/...
1020	Varilla 3/...
1030	Varilla 4/...
1040	Varilla 3/...
1050	Varilla 4/...
1060	Varilla 3/...
1070	Varilla 4/...
1080	Ladrillos r...

Unión (se ilustra junto con selección)

Algebra relacional.

$SL\{clave=1450\}(entregan) \cup SL\{clave=1300\}(entregan)$

SQL

(select * from entregan where clave=1450)

union

(select * from entregan where clave=1300)

The screenshot shows a SQL IDE with a script editor containing the following query:

```
tc2005b_db.  
• (SELECT * FROM tc2005b_db.entregan WHERE clave = 1450)  
UNION  
(SELECT * FROM tc2005b_db.entregan WHERE clave = 1300);
```

The results pane shows a table with 5 columns: clave, rfc, numero, fecha, and cantidad. The results are as follows:

	clave	rfc	numero	fecha	cantidad
1	1.300	GGGG800101	5.005	2004-02-28	521
2	1.300	GGGG800101	5.010	2001-02-10	119

¿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;

The screenshot shows a SQL IDE with a script editor containing the following query:

```
tc2005b_db.  
• SELECT * FROM tc2005b_db.entregan  
WHERE clave = 1450 OR clave = 1300;
```

The results pane shows a table with 5 columns: clave, rfc, numero, fecha, and cantidad. The results are as follows:

	clave	rfc	numero	fecha	cantidad
1	1.300	GGGG800101	5.005	2004-02-28	521
2	1.300	GGGG800101	5.010	2001-02-10	119

Intersección (se ilustra junto con selección y proyección)

Algebra relacional.

$PR\{clave\}(SL\{numero=5001\}(entregan)) \cap PR\{clave\}(SL\{numero=5018\}(entregan))$

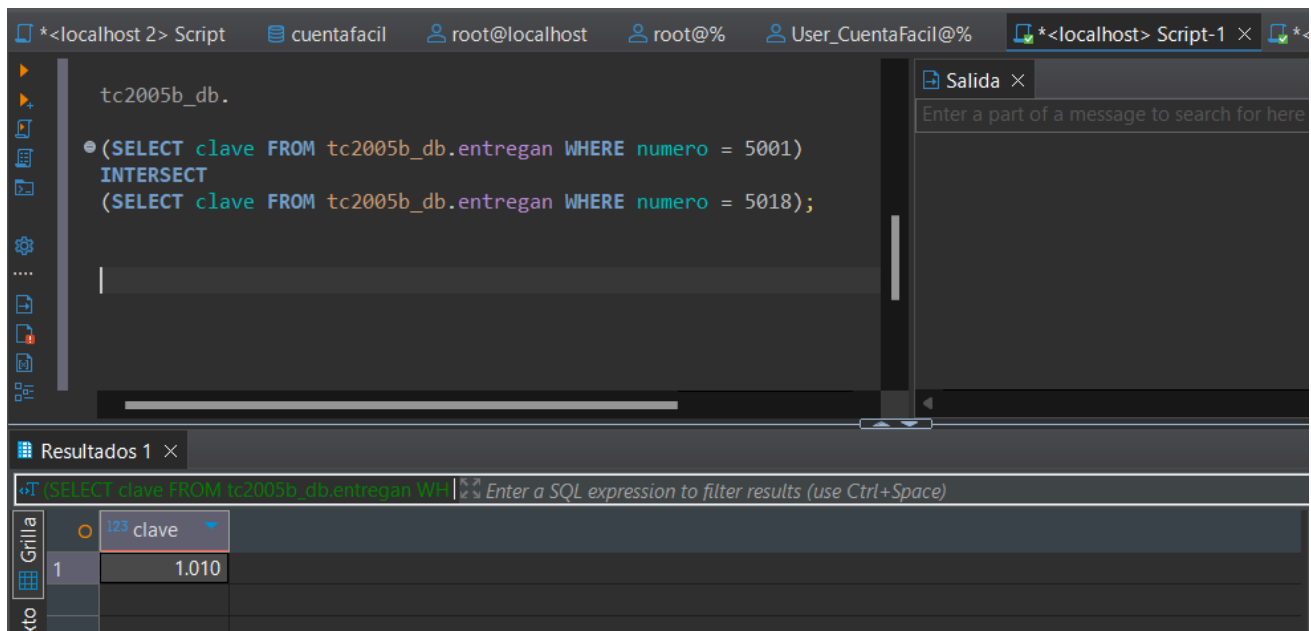
SQL

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.

The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a query script for the 'tc2005b_db' database:

```
SELECT * FROM tc2005b_db.entregan
WHERE clave NOT IN (
    SELECT clave FROM tc2005b_db.entregan WHERE clave = 1000
);
```

The bottom pane shows the results of the query in a grid format. The columns are: clave, rfc, numero, fecha, and cantidad. The results show 14 rows of data, with the first row having a clave of 1.010 and a cantidad of 528.

	clave	rfc	numero	fecha	cantidad
1	1.010	BBBB800101	5.001	1998-07-28 GM	528
2	1.010	BBBB800101	5.018	1997-02-09 GM	523
3	1.020	CCCC800101	5.002	2003-12-16 GM	582
4	1.020	CCCC800101	5.017	2000-03-29 GM	8
5	1.030	DDDD800101	5.003	1998-01-12 GM	202
6	1.030	DDDD800101	5.016	2005-06-07 GM	295
7	1.040	EEEE800101	5.004	1999-12-18 GM	263
8	1.040	EEEE800101	5.015	1999-05-29 GM	540
9	1.050	FFFF800101	5.005	2004-02-07 GM	503
10	1.050	FFFF800101	5.014	2000-04-18 GM	623
11	1.060	GGGG800101	5.006	2004-09-29 GM	324
12	1.060	GGGG800101	5.013	2002-07-23 GM	692
13	1.070	HHHH800101	5.007	2006-11-21 GM	2
14	1.070	HHHH800101	5.012	2004-11-27 GM	503

The right pane shows a 'Valor' dialog box with the value '1010' entered. The bottom status bar indicates that 85 rows were fetched in 0.013s on 2025-05-04 at 11:43.

Producto cartesiano

Algebra relacional.
entregan X materiales

SQL

select * from entregan,materiales

The screenshot shows a SQL IDE interface with a script editor and a results pane. The script editor contains the following SQL query:

```
tc2005b_db.  
  
SELECT * FROM tc2005b_db.entregan, tc2005b_db.materiales;
```

The results pane displays a table with 13 rows and 9 columns. The columns are: clave, rfc, numero, fecha, cantidad, clave, descripcion, and precio. The data represents a Cartesian product of the 'entregan' and 'materiales' tables.

	clave	rfc	numero	fecha	cantidad	clave	descripcion	precio
1	1.000	AAAA800101	5.000	2001-12-13	GV	165	1.000 Varilla 3/16	10
2	1.000	AAAA800101	5.000	2001-12-13	GV	165	1.010 Varilla 4/32	11
3	1.000	AAAA800101	5.000	2001-12-13	GV	165	1.020 Varilla 3/17	13
4	1.000	AAAA800101	5.000	2001-12-13	GV	165	1.030 Varilla 4/33	14
5	1.000	AAAA800101	5.000	2001-12-13	GV	165	1.040 Varilla 3/18	16
6	1.000	AAAA800101	5.000	2001-12-13	GV	165	1.050 Varilla 4/34	17
7	1.000	AAAA800101	5.000	2001-12-13	GV	165	1.060 Varilla 3/19	19
8	1.000	AAAA800101	5.000	2001-12-13	GV	165	1.070 Varilla 4/35	20
9	1.000	AAAA800101	5.000	2001-12-13	GV	165	1.080 Ladrillos rojos	5
10	1.000	AAAA800101	5.000	2001-12-13	GV	165	1.090 Ladrillos grises	3
11	1.000	AAAA800101	5.000	2001-12-13	GV	165	1.100 Block	3
12	1.000	AAAA800101	5.000	2001-12-13	GV	165	1.110 Megablock	4
13	1.000	AAAA800101	5.000	2001-12-13	GV	165	1.120 Sillar rosa	10

The status bar at the bottom indicates: 200 row(s) fetched - 0.010s, on 2025-05-04 at 11:44:51.

¿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.

The screenshot shows a database IDE with a SQL query editor and a results pane. The query is as follows:

```
USE TC2005B_DB;  
  
select m.description  
from materiales m, entregan e  
where m.clave = e.clave and e.fecha between '2000-01-01' and '2000-12-31';
```

The results pane displays a table with two columns: 'description' and 'Valor'. The 'description' column lists 12 items, with 'Varilla 3/17' appearing at the top. The 'Valor' column shows the value 'Varilla 3/17' for the first row. The status bar at the bottom indicates '12 row(s) fetched - 0.101s, on 2025-05-04 at 12:01:09'.

	description	Valor
1	Varilla 3/17	Varilla 3/17
2	Varilla 4/34	
3	Block	
4	Sillar gris	
5	Sillar gris	
6	Cantera blanca	
7	Recubrimiento P1028	
8	Tubería 3.6	
9	Pintura C1010	
10	Pintura B1021	
11	Pintura B1021	
12	Pintura B1022	

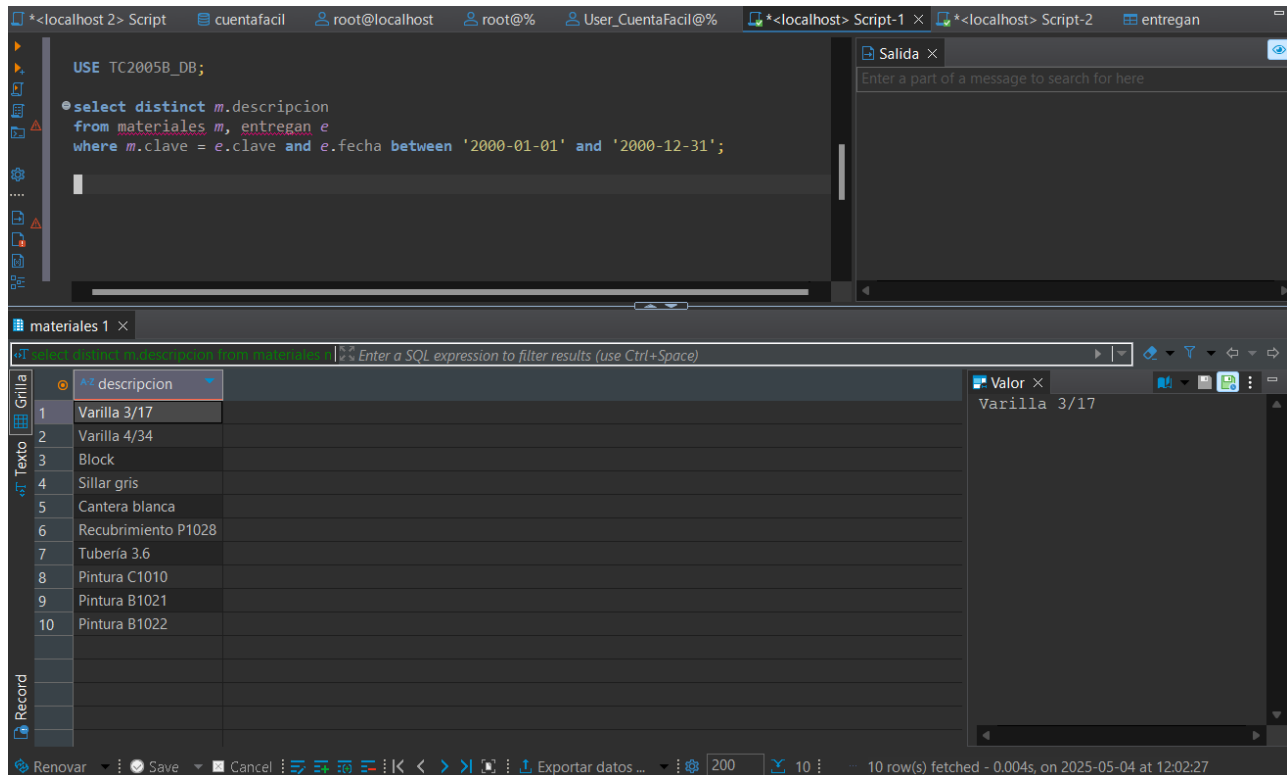
¿Por qué aparecen varias veces algunas descripciones de material?

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.



The screenshot shows a database IDE interface. The top pane contains a SQL script with the following query:

```
USE TC2005B_DB;  
  
select distinct m.descripcion  
from materiales m, entregan e  
where m.clave = e.clave and e.fecha between '2000-01-01' and '2000-12-31';
```

The bottom pane displays the results in a grid view. The first column is labeled 'descripcion' and contains 10 rows of material descriptions. The second column is labeled 'Valor' and contains the value 'Varilla 3/17' for the first row.

descripcion	Valor
Varilla 3/17	Varilla 3/17
Varilla 4/34	
Block	
Sillar gris	
Cantera blanca	
Recubrimiento P1028	
Tubería 3.6	
Pintura C1010	
Pintura B1021	
Pintura B1022	

The status bar at the bottom indicates '10 row(s) fetched - 0.004s, on 2025-05-04 at 12:02:27'.

¿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.

The screenshot shows a SQL IDE interface with a query editor and a results pane. The query editor contains the following SQL statement:

```
USE TC2005B_DB;

select p.numero, p.denominacion, e.fecha, e.cantidad
from proyectos p, entregan e
where p.numero = e.numero
order by p.numero, e.fecha desc;
```

The results pane displays a table with 15 rows and 4 columns: numero, denominacion, fecha, and cantidad. The data is sorted by project number and then by date in descending order.

	numero	denominacion	fecha	cantidad
1	5.000	Vamos Mexico	2003-03-15	177
2	5.000	Vamos Mexico	2001-12-13	165
3	5.000	Vamos Mexico	1999-04-07	382
4	5.001	Aztecon	2000-05-21	43
5	5.001	Aztecon	2000-05-18	601
6	5.001	Aztecon	1998-07-28	528
7	5.002	CIT Campeche	2005-07-03	24
8	5.002	CIT Campeche	2003-12-16	582
9	5.002	CIT Campeche	2001-09-09	603
10	5.003	Mexico sin ti no estamos completos	2005-04-30	576
11	5.003	Mexico sin ti no estamos completos	1998-09-12	530
12	5.003	Mexico sin ti no estamos completos	1998-01-12	202
13	5.004	Educando en Coahuila	2002-11-14	453
14	5.004	Educando en Coahuila	1999-12-18	263
15	5.004	Educando en Coahuila	(NULL)	152

The status bar at the bottom indicates that 87 row(s) were fetched in 0.075s (0.005s fetch) on 2025-05-04 at 12:17:3.

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.

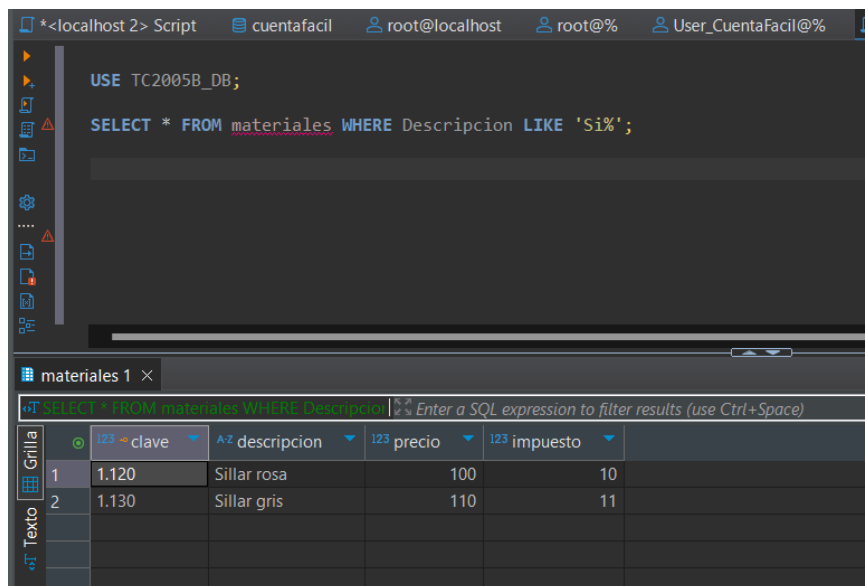
Operadores de cadena

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



The screenshot shows a SQL IDE window with a script editor and a results grid. The script editor contains the following SQL query:

```
USE TC2005B_DB;  
  
SELECT * FROM materiales WHERE Descripcion LIKE 'Si%';
```

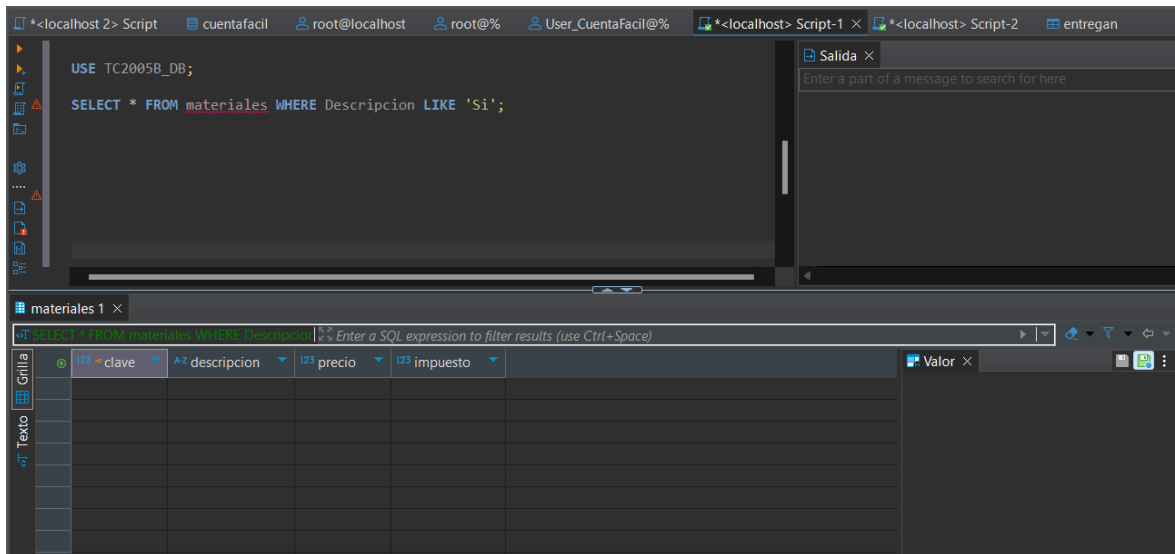
The results grid, titled "materiales 1", displays the following data:

	clave	descripcion	precio	impuesto
1	1.120	Sillar rosa	100	10
2	1.130	Sillar gris	110	11

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);
```

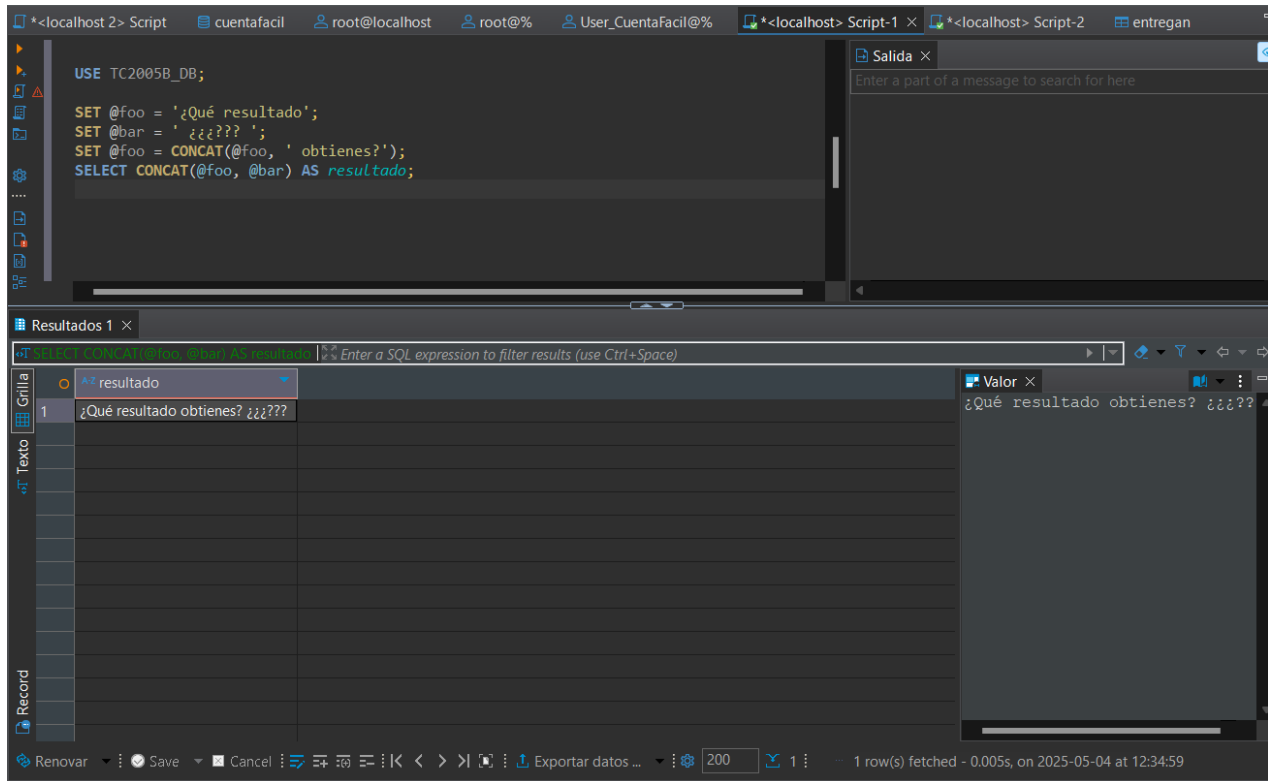
```
SET @foo = '¿Que resultado?';
```

```
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.



The screenshot shows a SQL Server Enterprise Manager window with a script editor and a results pane. The script editor contains the following T-SQL code:

```
USE TC2005B_DB;  
  
SET @foo = '¿Qué resultado?';  
SET @bar = '¿¿¿¿?';  
SET @foo = CONCAT(@foo, ' obtienes?');  
SELECT CONCAT(@foo, @bar) AS resultado;
```

The results pane shows a single row with the value '¿Qué resultado obtienes? ¿¿¿¿?' in the column 'resultado'.

resultado
¿Qué resultado obtienes? ¿¿¿¿?

¿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.

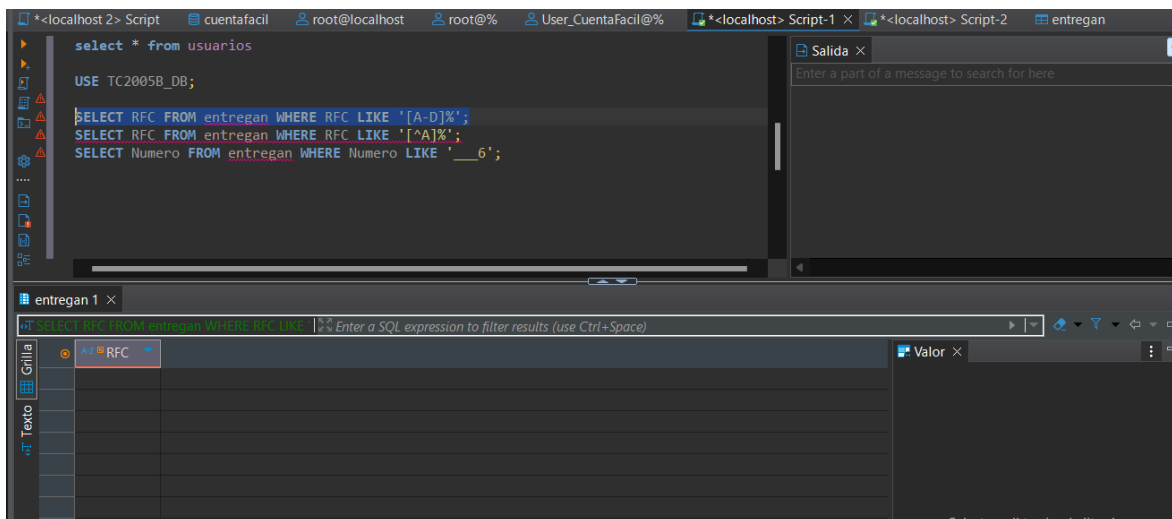
[^] - En contra parte, este operador coincide con cualquier caracter que no se encuentre dentro del intervalo o del conjunto especificado.

_ - 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

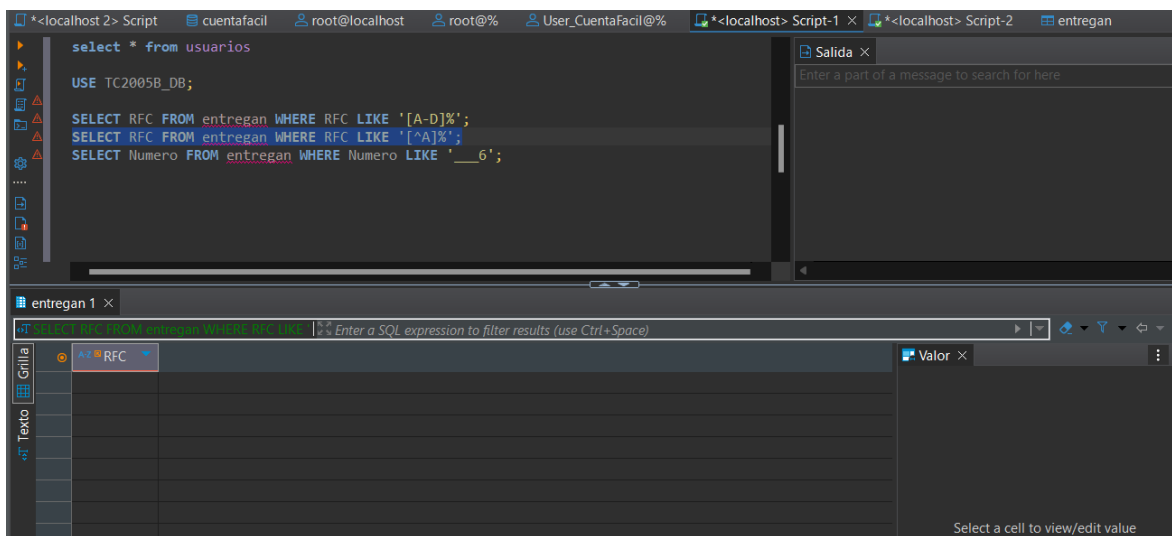
consultas:

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



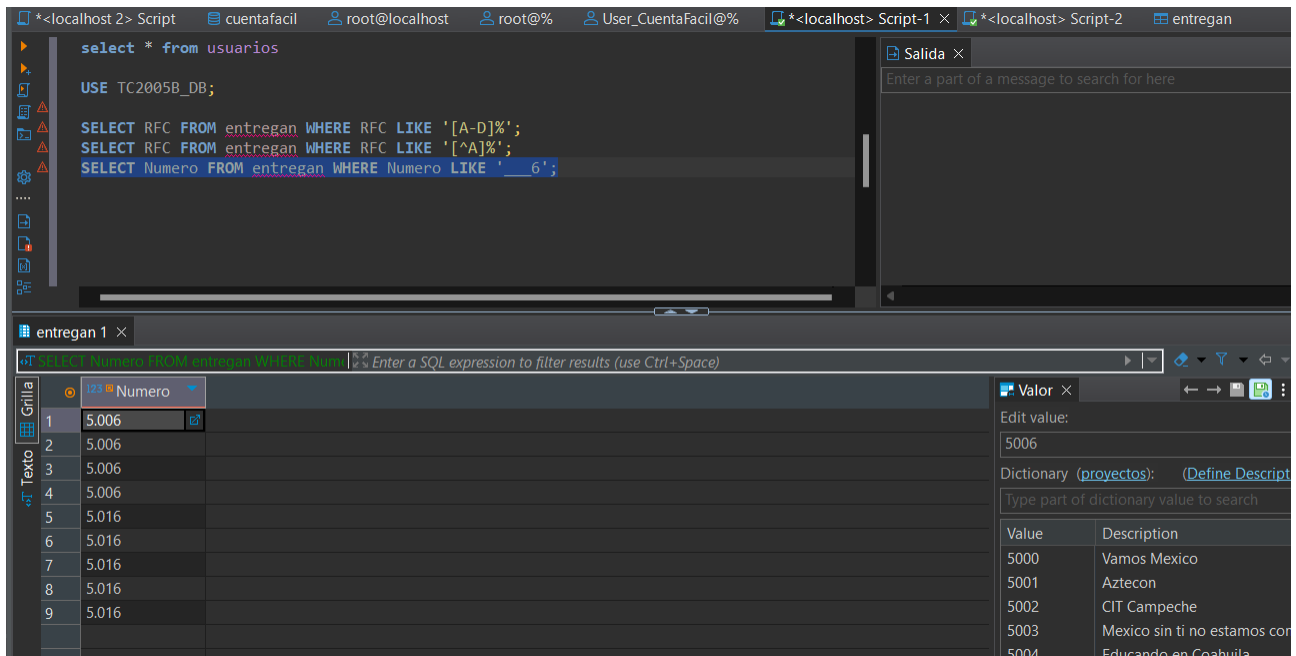
Comportamiento y función: Selecciona todos los RFCs de entregan que comienzan con una letra de la A a la D.

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';
```

The screenshot shows a database application interface with a dark theme. At the top, there are tabs for different scripts: '*<localhost 2> Script', 'cuentaafacil', 'root@localhost', 'root@%', 'User_CuentaFacil@%', '*<localhost> Script-1 x', '*<localhost> Script-2', and 'entregan'. The main window displays a SQL query in a text editor:

```
select * from usuarios
USE TC2005B_DB;
SELECT Clave, RFC, Numero, Fecha, Cantidad
FROM entregan
WHERE Fecha BETWEEN '2000-01-01' AND '2000-12-31';
```

Below the query editor, there is a table titled 'entregan 1 x'. The table has columns: Clave, RFC, Numero, Fecha, and Cantidad. The data is displayed in a grid with 12 rows. To the right of the table, there is a 'Valor x' panel with a search bar and a list of values and descriptions.

Clave	RFC	Numero	Fecha	Cantidad
1.020	CCCC800101	5.017	2000-03-29 GM	8
1.050	FFFF800101	5.014	2000-04-18 GM	623
1.100	CCCC800101	5.009	2000-12-07 GM	466
1.130	FFFF800101	5.006	2000-04-13 GM	562
1.130	FFFF800101	5.013	2000-02-09 GM	63
1.140	GGGG800101	5.005	2000-06-30 GM	583
1.210	FFFF800101	5.001	2000-05-21 GM	43
1.310	HHHH800101	5.011	2000-09-14 GM	72
1.360	EEEE800101	5.014	2000-02-04 GM	265
1.390	HHHH800101	5.019	2000-11-10 GM	107
1.410	BBBB800101	5.001	2000-05-18 GM	601
1.430	DDDD800101	5.007	2000-03-02 GM	13

The 'Valor x' panel on the right shows a search bar with the value '1020' entered. Below the search bar, there is a list of values and descriptions:

Value	Description
1000	Varilla 3/16
1010	Varilla 4/32
1020	Varilla 3/17
1030	Varilla 4/33
1040	Varilla 3/18
1050	Varilla 4/34
1060	Varilla 3/19
1070	Varilla 4/35
1080	Ladrillos rojos

At the bottom of the interface, there is a status bar with the text: '12 row(s) fetched - 0.116s, on 2025-05-04 at 13:40:56'.

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] )
```

The screenshot shows a SQL IDE interface. The top pane displays a SQL query using the EXISTS operator. The bottom pane shows the results of the query in a grid view. The grid has columns for RFC, Cantidad, Fecha, and Numero. The results show 12 rows of data. On the right side, there is a 'Salida' (Output) window and a 'Dictionary' window showing a list of values and their descriptions.

SQL Query:

```
USE TC2005B_DB;
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
);
```

Results Grid:

	RFC	Cantidad	Fecha	Numero
1	AAAA8001	165	2001-12-13	5.000
2	CCCC800101	582	2003-12-16	5.002
3	AAAA800101	86	2005-04-03	5.008
4	CCCC800101	466	2000-12-07	5.009
5	CCCC800101	699	2001-11-19	5.010
6	AAAA800101	152	[NULL]	5.004
7	CCCC800101	460	2001-04-09	5.006
8	CCCC800101	631	2001-07-28	5.009
9	AAAA800101	382	1999-04-07	5.000
10	AAAA800101	116	2005-04-21	5.010
11	CCCC800101	603	2001-09-09	5.002
12	CCCC800101	278	1999-05-05	5.008

Dictionary (proveedores):

Value	Description
AAAA800101	La fragua
BBBB800101	Oviedo
CCCC800101	La Ferre
DDDD800101	Cecoferre
EEEE800101	Alvin
FFFF800101	Comex
GGGG800101	Tabiquera del centro
HHHH800101	Tubasa

¿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
```

```
);
```

The screenshot shows a SQL IDE window with a script titled "Script-1". The script contains the following SQL query:

```
USE TC2005B_DB;  
  
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  
);
```

Below the script editor, the results of the query are displayed in a table. The table has 5 columns: RFC, Cantidad, Fecha, and Numero. The results are as follows:

	RFC	Cantidad	Fecha	Numero
1	AAAA8001	165	2001-12-13	5.000
2	CCCC800101	582	2003-12-16	5.002
3	AAAA800101	86	2005-04-03	5.008
4	CCCC800101	466	2000-12-07	5.009
5	CCCC800101	699	2001-11-19	5.010
6	AAAA800101	152	[NULL]	5.004
7	CCCC800101	460	2001-04-09	5.006
8	CCCC800101	631	2001-07-28	5.009
9	AAAA800101	382	1999-04-07	5.000
10	AAAA800101	116	2005-04-21	5.010
11	CCCC800101	603	2001-09-09	5.002
12	CCCC800101	278	1999-05-05	5.008

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%'
```

```
);
```

The screenshot shows a MySQL IDE window with a script titled "Script-1" containing the following SQL query:

```
USE TC2005B_DB;  
  
SELECT RFC, Cantidad, Fecha, Numero  
FROM Entregan  
WHERE Numero BETWEEN 5000 AND 5010  
AND RFC IN (  
    SELECT RFC  
    FROM Proveedores  
    WHERE RazonSocial LIKE 'La%'  
);
```

Below the query editor, the results of the query are displayed in a table grid. The table has 5 columns: RFC, Cantidad, Fecha, and Numero. The results are as follows:

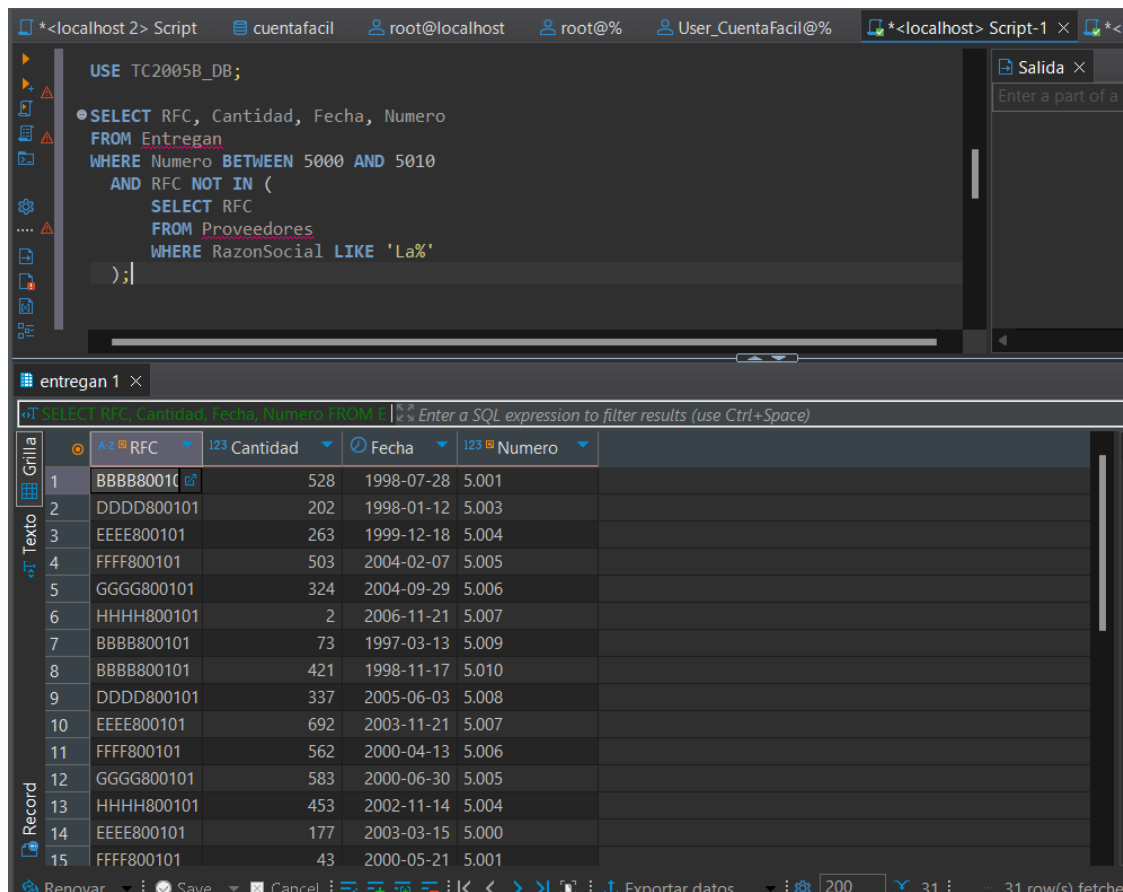
	RFC	Cantidad	Fecha	Numero
1	AAAA8001	165	2001-12-13	5.000
2	CCCC800101	582	2003-12-16	5.002
3	AAAA800101	86	2005-04-03	5.008
4	CCCC800101	466	2000-12-07	5.009
5	CCCC800101	699	2001-11-19	5.010
6	AAAA800101	152	[NULL]	5.004
7	CCCC800101	460	2001-04-09	5.006
8	CCCC800101	631	2001-07-28	5.009
9	AAAA800101	382	1999-04-07	5.000
10	AAAA800101	116	2005-04-21	5.010
11	CCCC800101	603	2001-09-09	5.002
12	CCCC800101	278	1999-05-05	5.008

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%'
);
```



The screenshot shows a SQL IDE interface with a query editor and a results pane. The query editor contains the following SQL code:

```
USE TC2005B_DB;
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%'
);
```

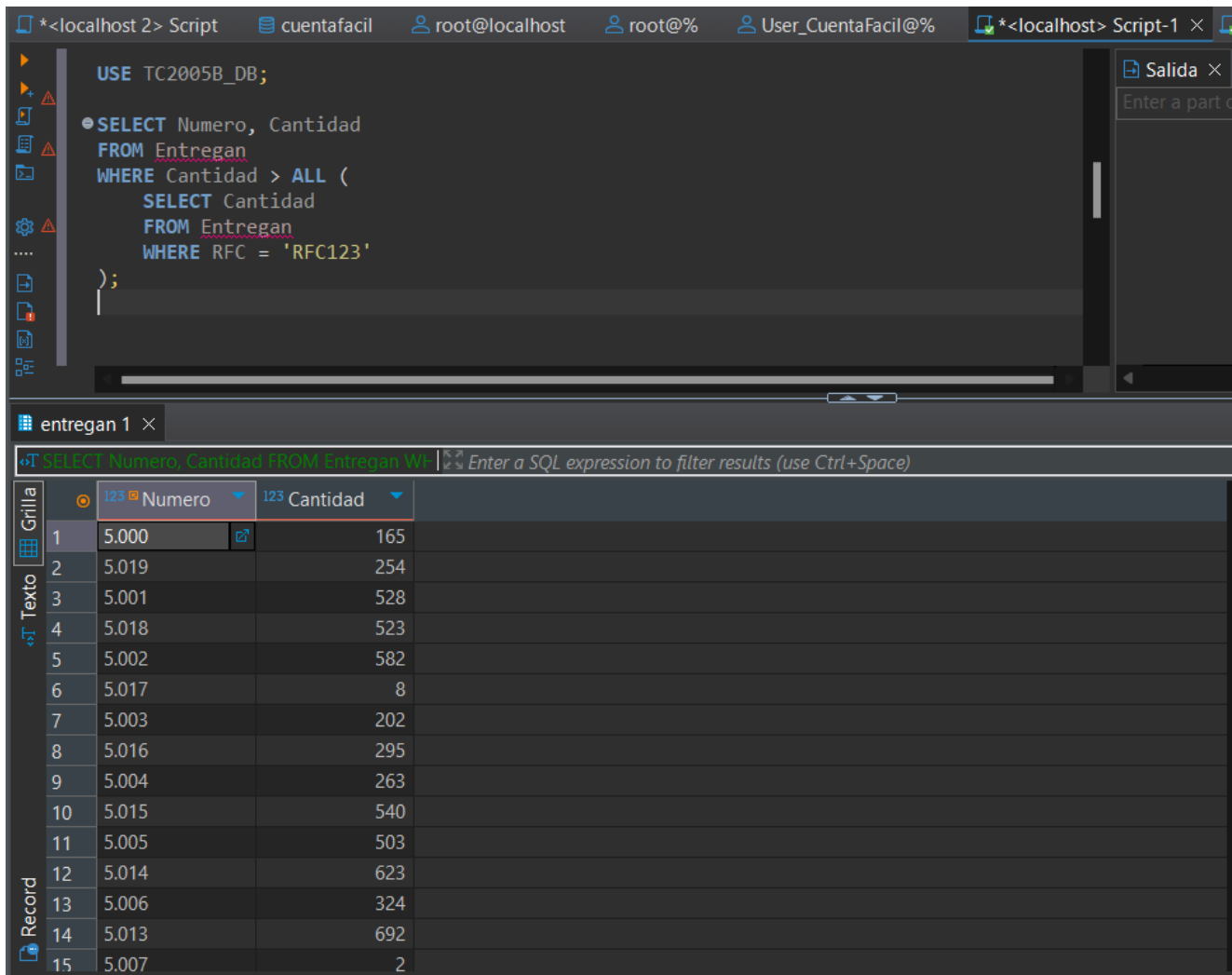
The results pane displays a table with 15 rows and 5 columns: RFC, Cantidad, Fecha, and Numero. The data is as follows:

	RFC	Cantidad	Fecha	Numero
1	BBBB800101	528	1998-07-28	5.001
2	DDDD800101	202	1998-01-12	5.003
3	EEEE800101	263	1999-12-18	5.004
4	FFFF800101	503	2004-02-07	5.005
5	GGGG800101	324	2004-09-29	5.006
6	HHHH800101	2	2006-11-21	5.007
7	BBBB800101	73	1997-03-13	5.009
8	BBBB800101	421	1998-11-17	5.010
9	DDDD800101	337	2005-06-03	5.008
10	EEEE800101	692	2003-11-21	5.007
11	FFFF800101	562	2000-04-13	5.006
12	GGGG800101	583	2000-06-30	5.005
13	HHHH800101	453	2002-11-14	5.004
14	EEEE800101	177	2003-03-15	5.000
15	FFFF800101	43	2000-05-21	5.001

The interface also shows a sidebar with icons for various database operations and a bottom status bar indicating 31 row(s) fetched.

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'
);
```



The screenshot shows a SQL IDE interface. The top panel displays a script with the following SQL query:

```
USE TC2005B_DB;
SELECT Numero, Cantidad
FROM Entregan
WHERE Cantidad > ALL (
    SELECT Cantidad
    FROM Entregan
    WHERE RFC = 'RFC123'
);
```

The bottom panel shows the results of the query in a table view. The table has two columns: 'Numero' and 'Cantidad'. The results are as follows:

	Numero	Cantidad
1	5.000	165
2	5.019	254
3	5.001	528
4	5.018	523
5	5.002	582
6	5.017	8
7	5.003	202
8	5.016	295
9	5.004	263
10	5.015	540
11	5.005	503
12	5.014	623
13	5.006	324
14	5.013	692
15	5.007	2

El Operador TOP, es un operador que recorre la entrada, un query, y sólo devuelve el primer número o porcentaje específico 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 primeros dos números o porcentajes 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.

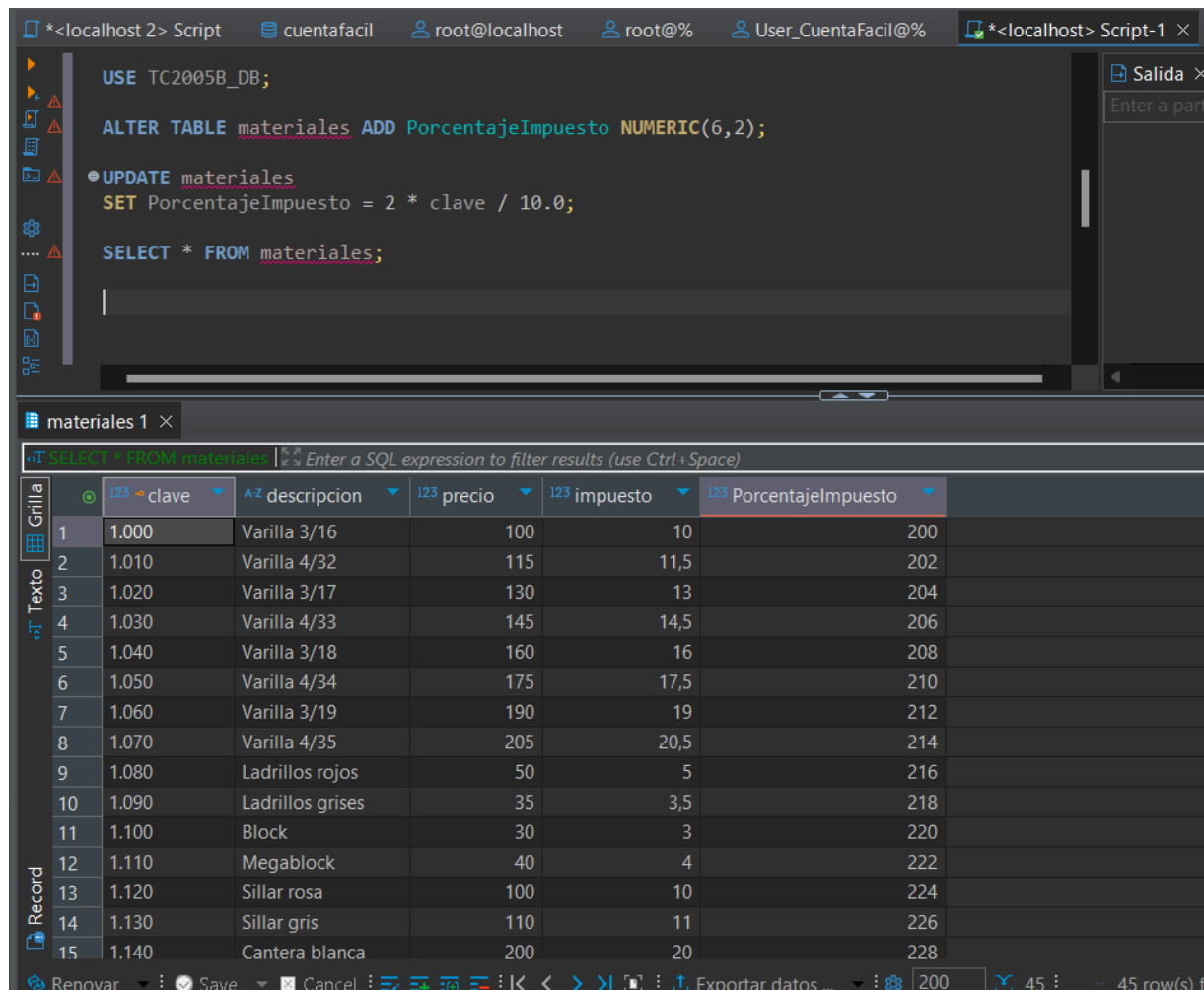
Listo, ya la tabla y la update fueron creadas usando el siguiente script:

```
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:



The screenshot shows a MySQL IDE window with the following SQL commands executed:

```
USE TC2005B_DB;
ALTER TABLE materiales ADD PorcentajeImpuesto NUMERIC(6,2);
UPDATE materiales
SET PorcentajeImpuesto = 2 * clave / 10.0;
SELECT * FROM materiales;
```

Below the SQL editor, the 'materiales' table is displayed with 15 rows. The columns are: clave, descripcion, precio, impuesto, and PorcentajeImpuesto.

	clave	descripcion	precio	impuesto	PorcentajeImpuesto
1	1.000	Varilla 3/16	100	10	200
2	1.010	Varilla 4/32	115	11,5	202
3	1.020	Varilla 3/17	130	13	204
4	1.030	Varilla 4/33	145	14,5	206
5	1.040	Varilla 3/18	160	16	208
6	1.050	Varilla 4/34	175	17,5	210
7	1.060	Varilla 3/19	190	19	212
8	1.070	Varilla 4/35	205	20,5	214
9	1.080	Ladrillos rojos	50	5	216
10	1.090	Ladrillos grises	35	3,5	218
11	1.100	Block	30	3	220
12	1.110	Megablock	40	4	222
13	1.120	Sillar rosa	100	10	224
14	1.130	Sillar gris	110	11	226
15	1.140	Cantera blanca	200	20	228

¿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;

```

	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%'
);

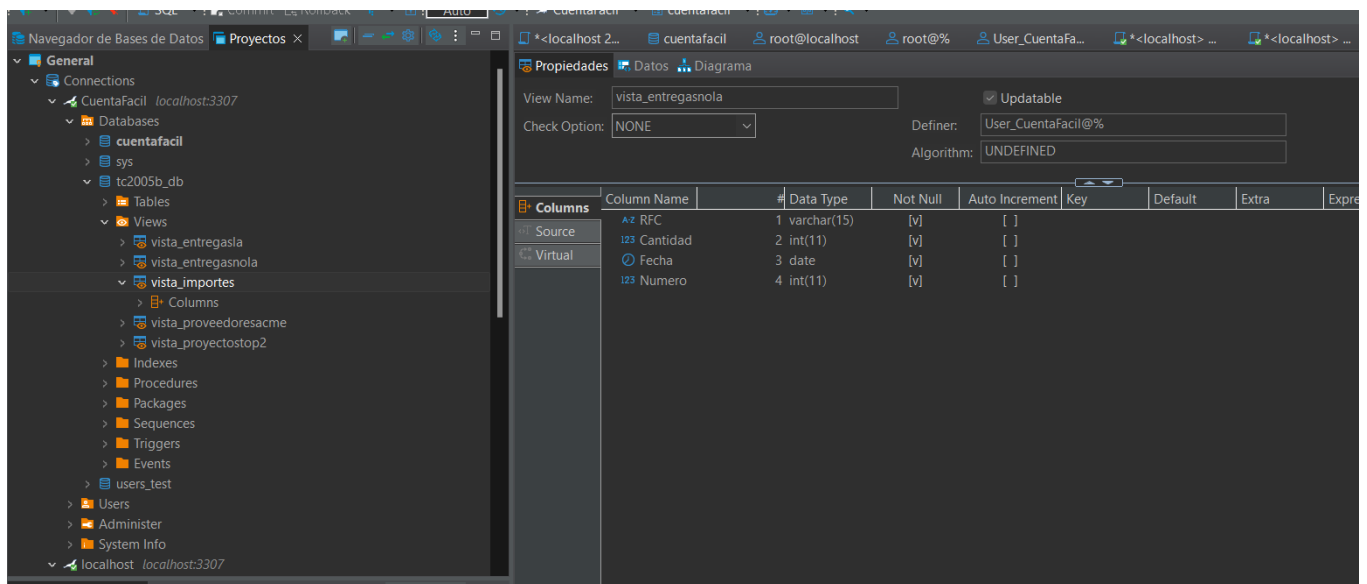
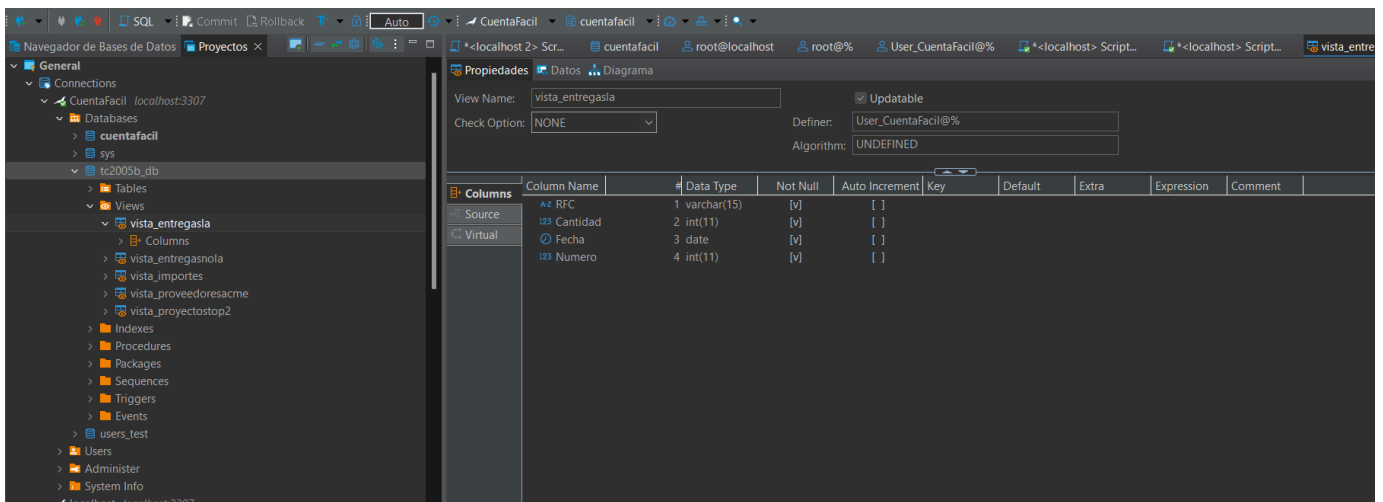
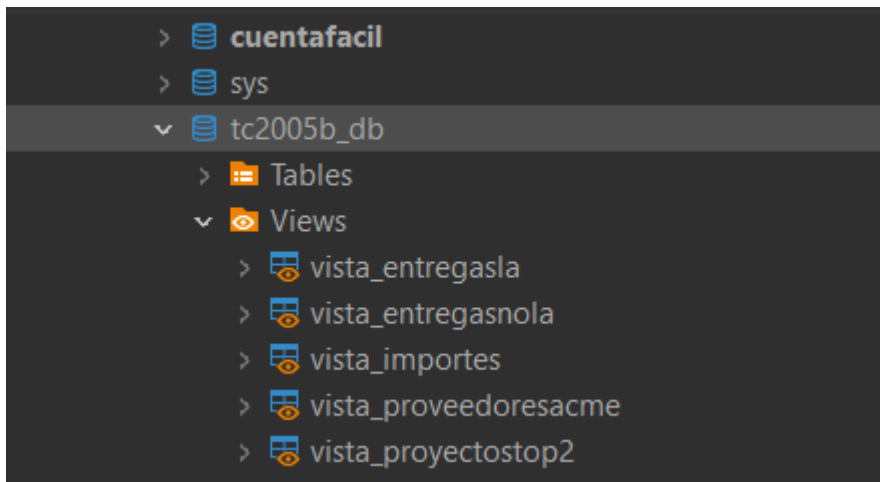
CREATE VIEW Vista_EntregasNoLa AS
SELECT RFC, Cantidad, Fecha, Numero
FROM Entregan
WHERE Numero BETWEEN 5000 AND 5010
AND RFC NOT IN (
    SELECT RFC
    FROM Proveedores
    WHERE RazonSocial LIKE 'La%'
);

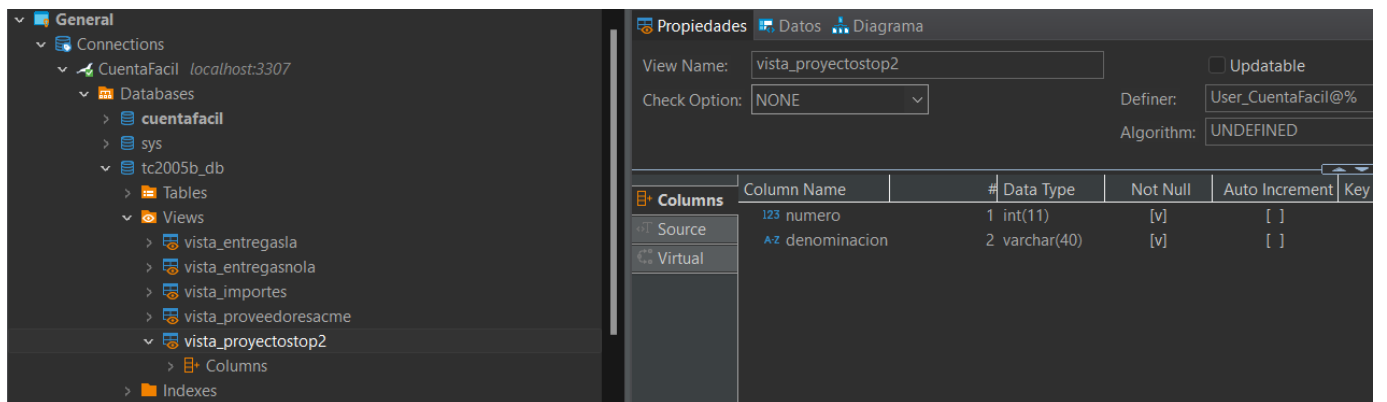
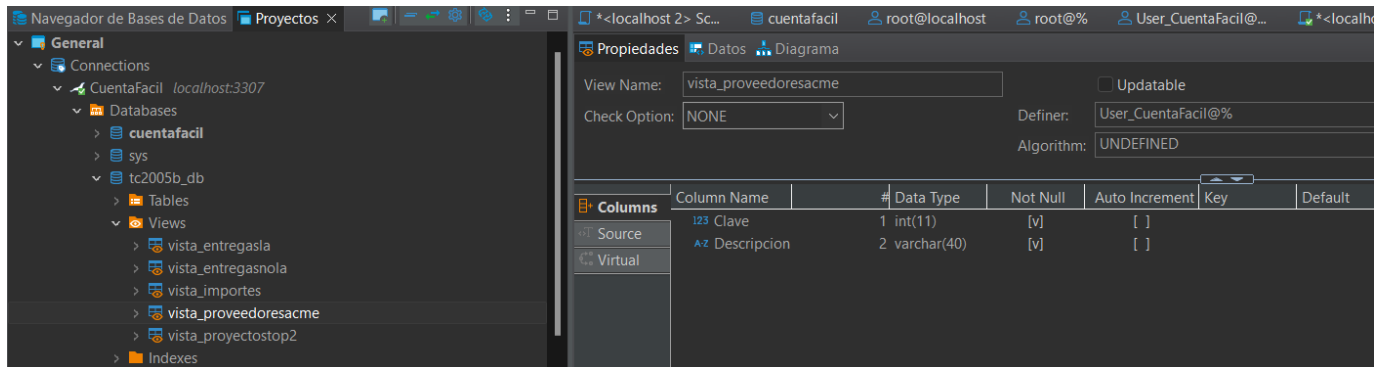
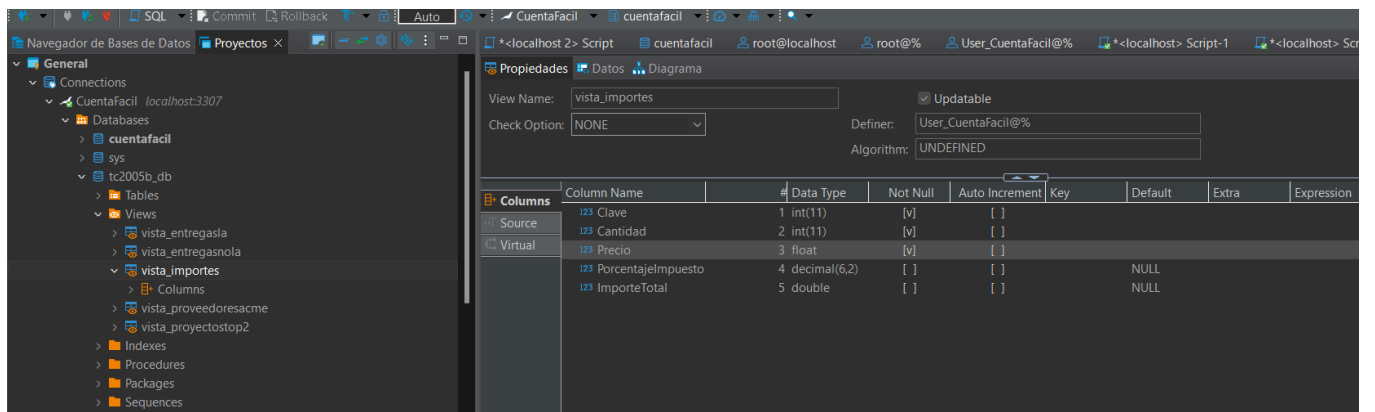
CREATE VIEW Vista_Importes AS
SELECT e.Clave, e.Cantidad, m.Precio, m.PorcentajeImpuesto,
       e.Cantidad * m.Precio * (1 + m.PorcentajeImpuesto / 100) AS ImporteTotal
FROM Entregan e
JOIN Materiales m ON e.Clave = m.Clave;

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:





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.

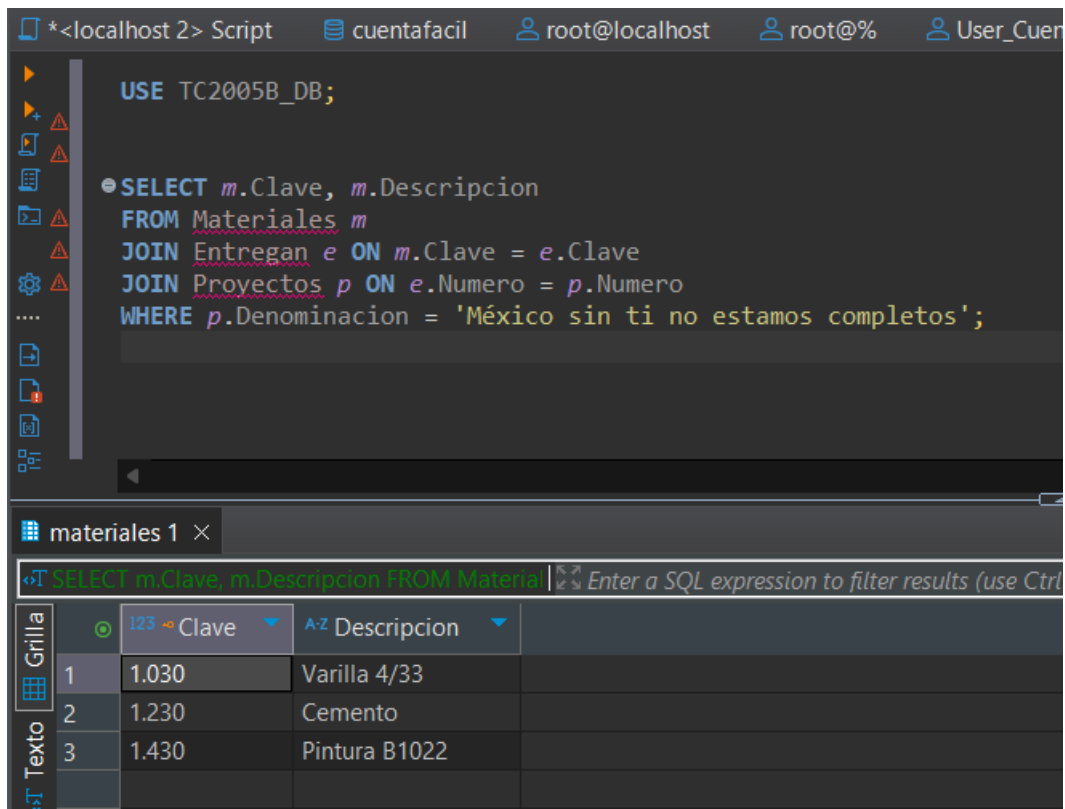
Importante: Las vistas no pueden incluir la cláusula order by.

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';
```



The screenshot shows a SQL IDE interface. The top panel displays a script with the following SQL query:

```
USE TC2005B_DB;

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';
```

The bottom panel shows the results of the query in a table view titled "materiales 1". The table has two columns: "Clave" and "Descripcion". The results are as follows:

	Clave	Descripcion
1	1.030	Varilla 4/33
2	1.230	Cemento
3	1.430	Pintura B1022

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';
```

The screenshot shows a SQL IDE interface. The top pane displays a script with the following SQL query:

```
USE TC2005B_DB;  
  
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';
```

The bottom pane shows the results of the query in a table grid. The table has two columns: 'Clave' and 'Descripcion'. The 'Clave' column is sorted by numerical value (123), and the 'Descripcion' column is sorted alphabetically (A-Z). The results are currently empty.

Clave	Descripcion
-------	-------------

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

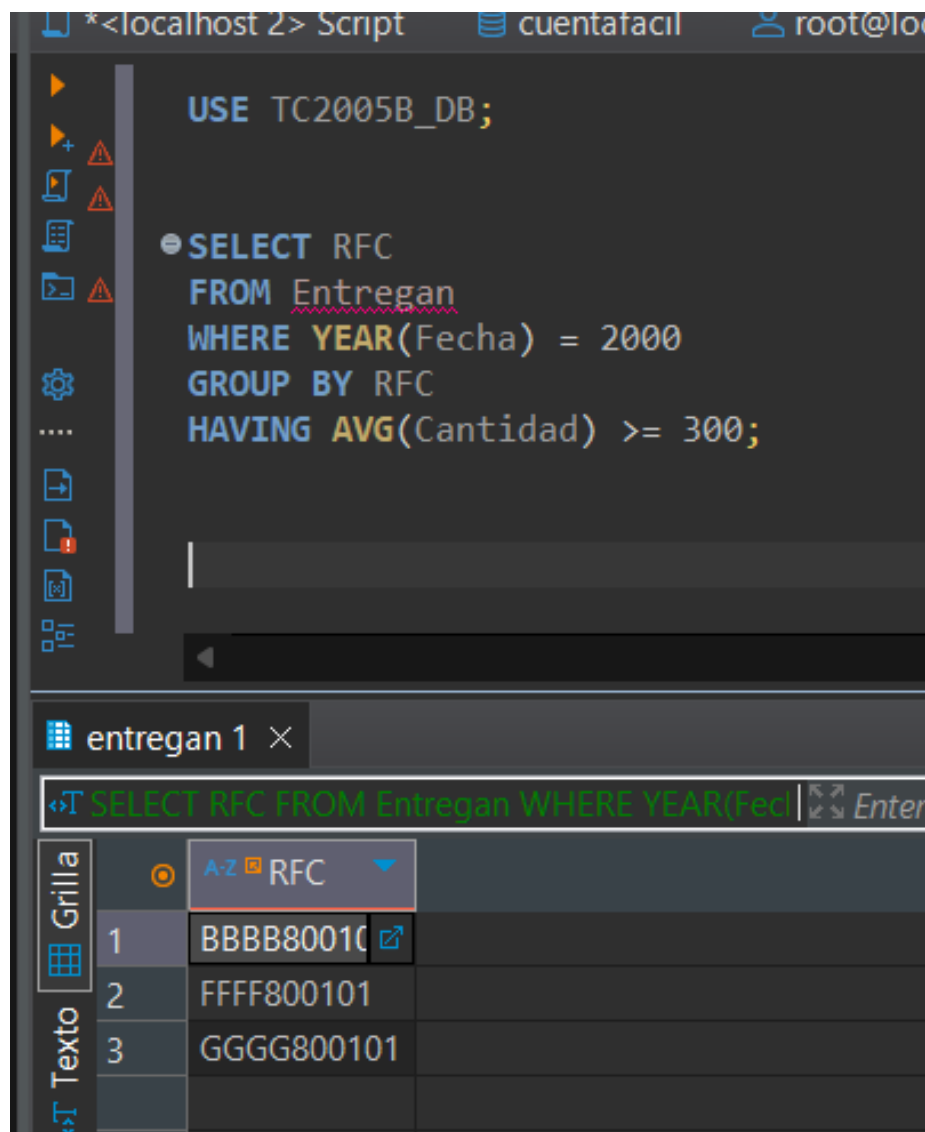
SELECT RFC

FROM Entregan

WHERE YEAR(Fecha) = 2000

GROUP BY RFC

HAVING AVG(Cantidad) >= 300;



The screenshot shows a MySQL command-line interface. The top part displays the SQL query being executed:

```
USE TC2005B_DB;  
  
SELECT RFC  
FROM Entregan  
WHERE YEAR(Fecha) = 2000  
GROUP BY RFC  
HAVING AVG(Cantidad) >= 300;
```

Below the query, the results are shown in a table format. The table has two columns: an index and the RFC values. The results are:

	RFC
1	BBBB80010
2	FFFF800101
3	GGGG800101

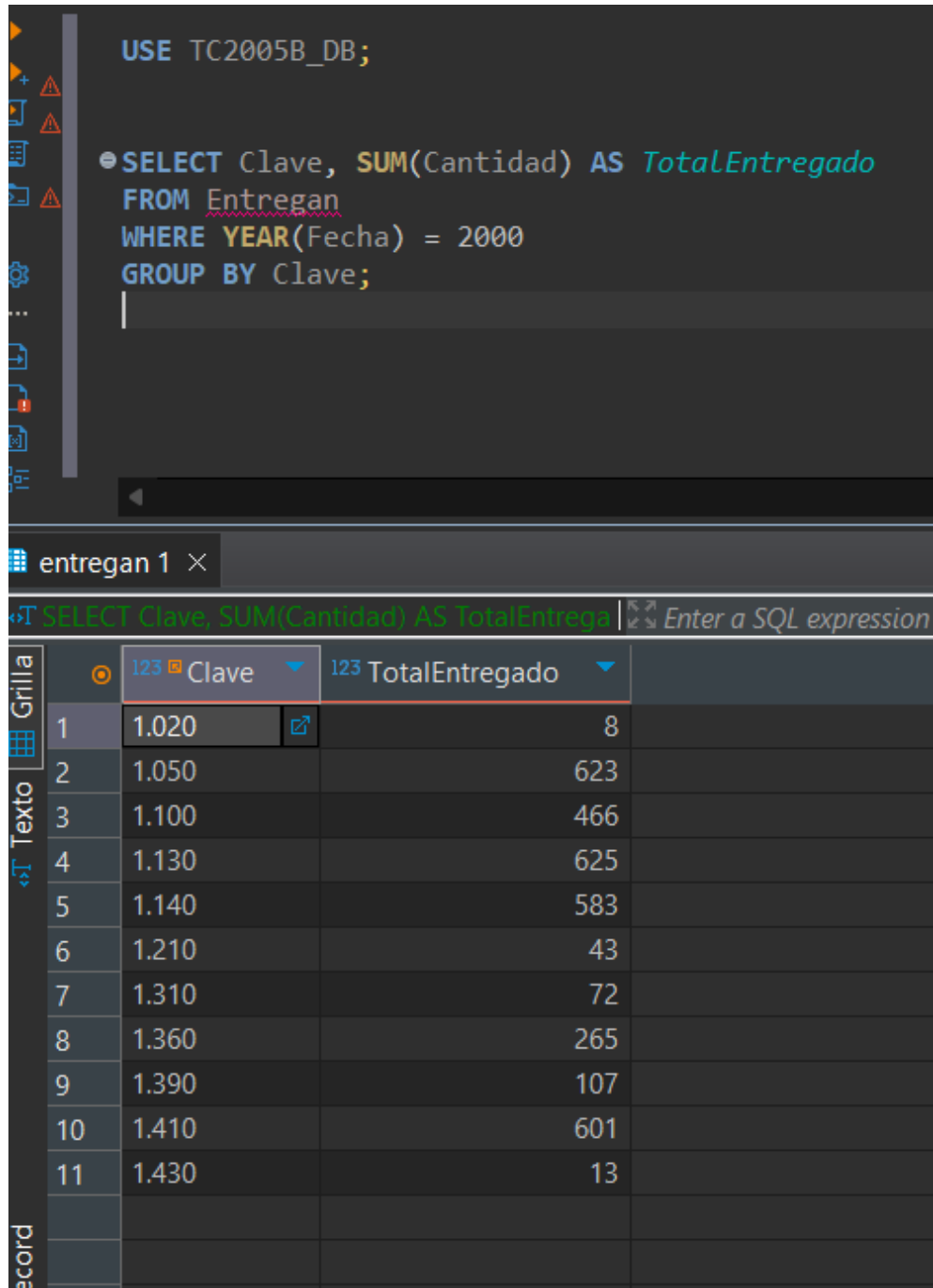
El Total entregado por cada material en el año 2000.

SELECT Clave, SUM(Cantidad) AS TotalEntregado

FROM Entregan

WHERE YEAR(Fecha) = 2000

GROUP BY Clave;



```
USE TC2005B_DB;

SELECT Clave, SUM(Cantidad) AS TotalEntregado
FROM Entregan
WHERE YEAR(Fecha) = 2000
GROUP BY Clave;
```

	123 Clave	123 TotalEntregado
1	1.020	8
2	1.050	623
3	1.100	466
4	1.130	625
5	1.140	583
6	1.210	43
7	1.310	72
8	1.360	265
9	1.390	107
10	1.410	601
11	1.430	13

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;
```

The screenshot shows a SQL IDE interface with the following components:

- Script Editor:** Contains two SQL queries. The first query creates a view named `VistaVentas2001` by aggregating data from the `Entregan` table for the year 2001. The second query selects the `Clave` from `VistaVentas2001`, ordered by `Total` in descending order, with a `LIMIT 1`.
- Query Results:** A tab labeled `entregan 1` is active, showing a grid with the results of the second query. The grid has a header row with a dropdown menu set to `123 Clave` and a data row with the value `1.260`.

	123 Clave
1	1.260

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

SELECT *

FROM Materiales

WHERE Descripcion LIKE '%ub%';

The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, a search icon, and a settings icon. The main editor area contains the following SQL query:

```
USE TC2005B_DB;  
  
SELECT *  
FROM Materiales  
WHERE Descripcion LIKE '%ub%';
```

Below the editor, a tab labeled "materiales 1" is active. The results pane shows the output of the query, which is a table with 7 columns: "clave", "descripcion", "precio", "impuesto", and "PorcentajeImpuesto". The table contains 12 rows of data, all of which include the substring "ub" in the description.

	clave	descripcion	precio	impuesto	PorcentajeImpuesto
1	1.180	Recubrimiento P1001	200	20	236
2	1.190	Recubrimiento P1010	220	22	238
3	1.200	Recubrimiento P1019	240	24	240
4	1.210	Recubrimiento P1028	250	25	242
5	1.220	Recubrimiento P1037	280	28	244
6	1.290	Tubería 3.5	200	20	258
7	1.300	Tubería 4.3	210	21	260
8	1.310	Tubería 3.6	220	22	262
9	1.320	Tubería 4.4	230	23	264
10	1.330	Tubería 3.7	240	24	266
11	1.340	Tubería 4.5	250	25	268
12	1.350	Tubería 3.8	260	26	270

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
```

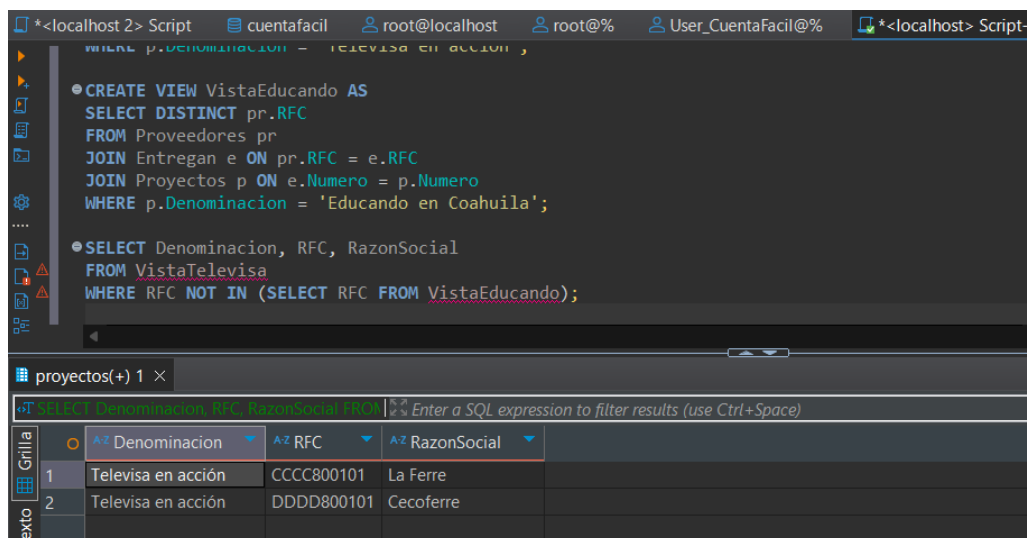
```
JOIN Proyectos p ON e.Numero = p.Numero
```

```
WHERE p.Denominacion = 'Educando en Coahuila';
```

```
SELECT Denominacion, RFC, RazonSocial
```

```
FROM VistaTelevisa
```

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



```
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);
```

	Az Denominacion	Az RFC	Az RazonSocial
1	Televisa en acción	CCCC800101	La Ferre
2	Televisa en acción	DDDD800101	Cecoferre

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'
```

```
);
```

The screenshot shows a SQL IDE interface with a script editor and a results pane. The script editor contains the following SQL query:

```
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'
);
```

The results pane shows a table with 4 columns: Denominacion, RFC, RazonSocial, and an empty column. The table contains 2 rows of data:

	A-Z Denominacion	A-Z RFC	A-Z RazonSocial	
1	Televisa en acción	CCCC800101	La Ferre	
2	Televisa en acción	DDDD800101	Cecoferre	

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 (

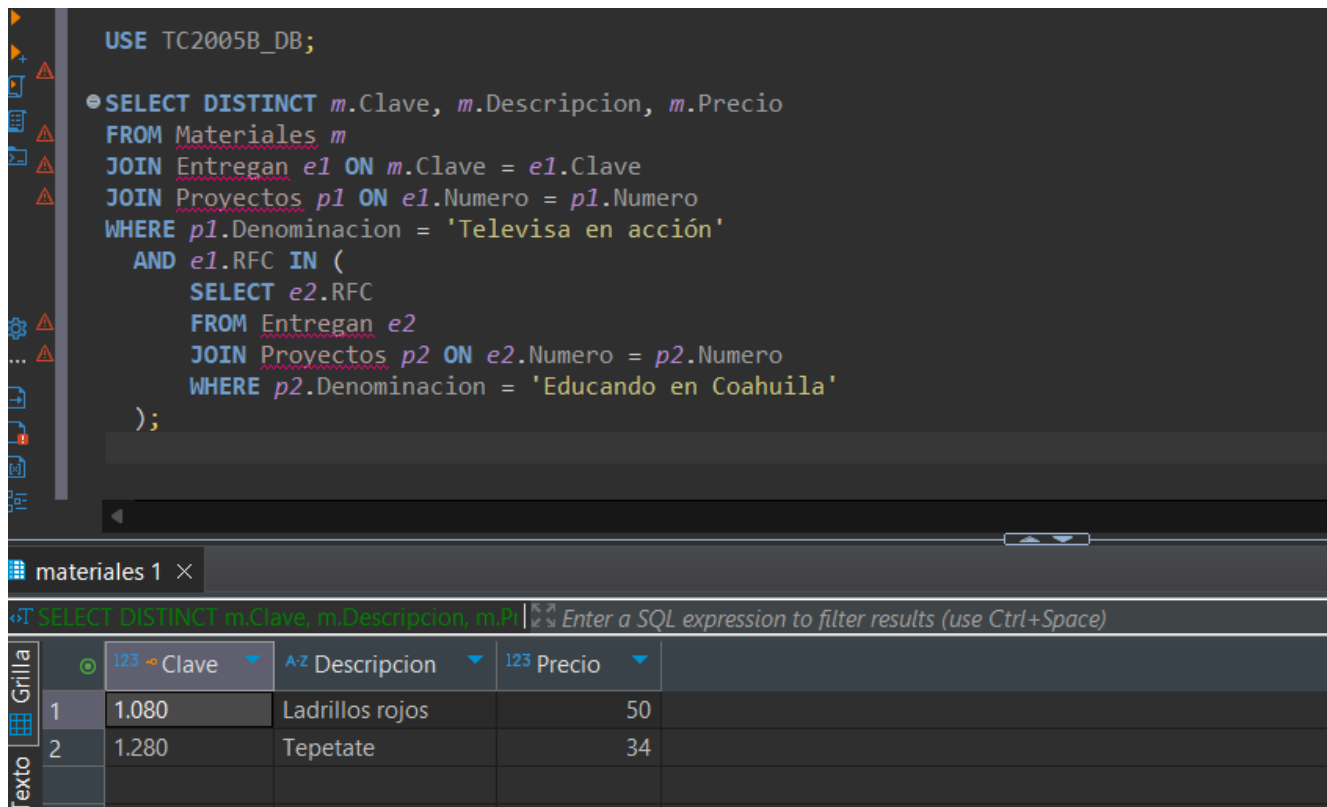
    SELECT e2.RFC

    FROM Entregan e2

    JOIN Proyectos p2 ON e2.Numero = p2.Numero

    WHERE p2.Denominacion = 'Educando en Coahuila'

);
```



The screenshot shows a SQL IDE interface. The top pane displays a SQL query that filters materials based on their delivery records for two specific projects. The bottom pane shows the results of the query in a table format.

```
USE TC2005B_DB;

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 IN (
    SELECT e2.RFC
    FROM Entregan e2
    JOIN Proyectos p2 ON e2.Numero = p2.Numero
    WHERE p2.Denominacion = 'Educando en Coahuila'
);
```

materials 1 x

Enter a SQL expression to filter results (use Ctrl+Space)

	Clave	Descripcion	Precio
1	1.080	Ladrillos rojos	50
2	1.280	Tepetate	34

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'
);
```

The screenshot shows a SQL IDE interface with a dark theme. The top toolbar includes icons for file operations and a 'Salida' (Output) button. The main editor displays the SQL query from the previous block. Below the editor, a tab labeled 'materiales 1' is active, showing a table with the following columns: 'Clave', 'Descripcion', and 'Precio'. The table is currently empty, with only the header row visible. The status bar at the bottom indicates the table has 123 rows and 3 columns.

Clave	Descripcion	Precio
-------	-------------	--------