Laboratorio 15

## Reporte

* Incluye la sentencia
* Una muestra de la salida (dos o tres renglones)
* El número de renglones que SQL Server reporta al final de la consulta.

Convenio: para evitar las letras griegas originales del álgebra relacional, en esta lectura se utiliza la siguiente notación:

|  |  |
| --- | --- |
| Operador | Significado |
| SL{condición} | selección con el criterio condición. |
| PR{lista de columnas} | proyección de lista de columnas. |
| JN | reunión natural (natural join). |
| JN{condición} | reunión con el criterio condición (teta join). |
| UN | unión. |
| IN | intersección. |
| - | diferencia |
| X | Producto cartesiano |

# Consulta de una tabla completa

Algebra relacional.

materiales

SQL

select \* from materiales

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | SELECT \* FROM Materiales |
| **Salida** | 1000 Varilla 3/16 100.00  1010 Varilla 4/32 115.00 |
| **Número de renglones reportados** | 44 rows |

# Selección

Algebra relacional.

SL{clave=1000}(materiales)

SQL

select \* from materiales

where clave=1000

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | SELECT \* FROM Materiales  WHERE Clave = 1000 |
| **Salida** | 1000 Varilla 3/16 100.00 |
| **Número de renglones reportados** | 1 rows |

# Proyección

Algebra relacional.

PR{clave,rfc,fecha} (entregan)

SQL

select clave,rfc,fecha from entregan

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | SELECT Clave, RFC, Fecha FROM Entregan |
| **Salida** | 1000 AAAA800101 1998-07-08 00:00:00.000  1000 AAAA800101 1999-08-08 00:00:00.000 |
| **Número de renglones reportados** | 132 rows |

# Reunión Natural

Algebra relacional.

entregan JN materiales

SQL

select \* from materiales,entregan

where materiales.clave = entregan.clave

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | SELECT \* FROM Materiales, Entregan  WHERE Materiales.Clave = Entregan.Clave |
| **Salida** | 1000 Varilla 3/16 100.00 1000 AAAA800101 5000 1998-07-08 00:00:00.000 165.00  1000 Varilla 3/16 100.00 1000 AAAA800101 5019 1999-08-08 00:00:00.000 254.00 |
| **Número de renglones reportados** | 132 rows |

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

No, puesto que al hacer el join natural, este no sería contemplado

# Reunión con criterio específico

Algebra relacional.

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

SQL

select \* from entregan,proyectos

where entregan.numero < = proyectos.numero

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | SELECT \* FROM Entregan, Proyectos  WHERE Entregan.Numero <= Proyectos.Numero |
| **Salida** | 1000 AAAA800101 5000 1998-07-08 00:00:00.000 165.00 5000 Vamos Mexico  1200 EEEE800101 5000 2000-03-05 00:00:00.000 177.00 5000 Vamos Mexico |
| **Número de renglones reportados** | 1188 rows |

# 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)

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | (SELECT \* FROM Entregan WHERE Clave=1450)  UNION  (SELECT \* FROM Entregan WHERE Clave=1300) |
| **Salida** | 1300 GGGG800101 5005 2002-06-10 00:00:00.000 521.00  1300 GGGG800101 5005 2003-02-02 00:00:00.000 457.00 |
| **Número de renglones reportados** | 3 rows |

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

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | SELECT \* FROM Entregan WHERE Clave=1300 |
| **Salida** | 1300 GGGG800101 5005 2002-06-10 00:00:00.000 521.00  1300 GGGG800101 5005 2003-02-02 00:00:00.000 457.00 |
| **Número de renglones reportados** | 3 rows |

# 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)

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | (SELECT Clave FROM Entregan WHERE Numero=5001)  INTERSECT  (SELECT Clave FROM Entregan WHERE Numero=5018) |
| **Salida** | 1010 |
| **Número de renglones reportados** | 1 rows |

# Diferencia (se ilustra con selección)

Algebra relacional.

entregan - SL{clave=1000}(entregan)

SQL

(select \* from entregan)

minus

(select \* from entregan where clave=1000)

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | SELECT \* FROM Entregan  EXCEPT  SELECT \* FROM Entregan WHERE Clave = 1000; |
| **Salida** | 1010 BBBB800101 5001 2000-05-03 00:00:00.000 528.00  1010 BBBB800101 5018 2000-11-10 00:00:00.000 667.00  1010 BBBB800101 5018 2002-03-29 00:00:00.000 523.00 |
| **Número de renglones reportados** | 129 rows |

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

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | SELECT \* FROM Entregan, Materiales |
| **Salida** | 1000 AAAA800101 5000 1998-07-08 00:00:00.000 165.00 1000 Varilla 3/16 100.00  1000 AAAA800101 5019 1999-08-08 00:00:00.000 254.00 1000 Varilla 3/16 100.00 |
| **Número de renglones reportados** | 5808 rows |

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

Cardinalidad Entregan \* Cardinalidad Materiales = Número de tuplas del producto cartesiano

132 \* 44 = 5808

# 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.

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | SELECT Descripcion  FROM Materiales, Entregan  WHERE Entregan.Fecha BETWEEN '2000/01/01' AND '2000/31/12' AND Materiales.Clave=Entregan.Clave  SET DATEFORMAT dmy |
| **Salida** | Recubrimiento P1019  Recubrimiento P1019 |
| **Número de renglones reportados** | 28 rows |

**¿Por qué aparecen varias veces algunas descripciones de material?**

Porque un mismo material ha sido entregado más de una vez

# 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.

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | SELECT DISTINCT Descripcion  FROM Materiales, Entregan  WHERE Entregan.Fecha BETWEEN '2000/01/01' AND '2000/31/12' AND Materiales.Clave=Entregan.Clave  SET DATEFORMAT dmy |
| **Salida** | Arena  Block |
| **Número de renglones reportados** | 22 rows |

**¿Qué resultado obtienes en esta ocasión?**

La descripción de los mismos materiales que consulté anteriormente pero ahora sin repeticiones y ordenados ascendentemente de forma alfabético respecto a su descripción.

# 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.

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | SELECT P.Numero, P.Denominacion, E.Fecha, E.Cantidad  FROM Proyectos AS P, Materiales AS M, Entregan AS E  WHERE P.Numero=E.Numero AND M.Clave=E.Clave  ORDER BY Numero DESC  SET DATEFORMAT dmy |
| **Salida** | 5019 Queretaro limpio 1999-08-08 00:00:00.000 254.00  5019 Queretaro limpio 2000-04-06 00:00:00.000 7.00 |
| **Número de renglones reportados** | 132 rows |

# 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%'

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | SELECT \* FROM Materiales WHERE Descripcion LIKE 'Si%' |
| **Salida** | 1120 Sillar rosa 100.00  1130 Sillar gris 110.00 |
| **Número de renglones reportados** | 2 rows |

**¿Qué resultado obtienes?**

Silla rosa y silla gris

**Explica que hace el símbolo '%'.**

Es un comodín que representa una cadena infinita de caracteres, por ejemplo, si% representará todas aquellas cadenas (descripción de los materiales en este caso) que empiecen con si y sin importar la cantidad de caracteres que le prosigan.

**¿Qué sucede si la consulta fuera : LIKE 'Si' ?**

Buscaría aquellos materiales cuya descripción sea exactamente Si

**¿Qué resultado obtienes?**

0 rows

**Explica a qué se debe este comportamiento.**

No hay materiales cuya descripción sea exactamente Si

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;

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | DECLARE @foo varchar(40);  DECLARE @bar varchar(40);  SET @foo = '¿Que resultado';  SET @bar = ' ¿¿¿??? '  SET @foo += ' obtienes?';  PRINT @foo + @bar; |
| **Salida** | ¿Que resultado obtienes? ¿¿¿??? |
| **Número de renglones reportados** | 0 rows |

**¿Qué resultado obtienes de ejecutar el siguiente código?**

¿Que resultado obtienes? ¿¿¿???

**¿Para qué sirve DECLARE?**

Declarar variables creadas por mí

**¿Cuál es la función de @foo?**

Es mi variable

**¿Que realiza el operador SET?**

Establece el valor de la variable

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]%';

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | SELECT RFC FROM Entregan WHERE RFC LIKE '[A-D]%'; |
| **Salida** | AAAA800101  AAAA800101 |
| **Número de renglones reportados** | 72 rows |

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

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | SELECT RFC FROM Entregan WHERE RFC LIKE '[^A]%'; |
| **Salida** | BBBB800101  BBBB800101 |
| **Número de renglones reportados** | 114 rows |

SELECT Numero FROM Entregan WHERE Numero LIKE '\_\_\_6';

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | SELECT Numero FROM Entregan WHERE Numero LIKE '\_\_\_6'; |
| **Salida** | 5016  5016 |
| **Número de renglones reportados** | 14 rows |

# 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;

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | SELECT Clave,RFC,Numero,Fecha,Cantidad  FROM Entregan  WHERE Numero BETWEEN 5000 AND 5010; |
| **Salida** | 1000 AAAA800101 5000 1998-07-08 00:00:00.000 165.00  1010 BBBB800101 5001 2000-05-03 00:00:00.000 528.00 |
| **Número de renglones reportados** | 60 rows |

¿Cómo filtrarías rangos de fechas?

SELECT Clave,RFC,Numero,Fecha,Cantidad

FROM Entregan

WHERE Fecha BETWEEN ‘yyyy/mm/dd’ AND ‘yyyy/mm/dd’;

NOTA: Fecha1 < Fecha2

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] )

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | 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] ) |
| **Salida** | AAAA800101 165.00 1998-07-08 00:00:00.000 5000  CCCC800101 582.00 2001-07-29 00:00:00.000 5002 |
| **Número de renglones reportados** | 16 rows |

**¿Qué hace la consulta?**

Hace una primera consulta de Entregan cuyo número esté entre 5000 y 5010, posteriormente vuelve a hacer una consulta de ello respecto a los proveedores cuya razón social empieza exactamente con ‘La**’.**

**¿Qué función tiene el paréntesis ( ) después de EXISTS?**

Englobar la subconsulta (o segunda consulta, o segundo filtro por así decirlo).

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

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | SELECT RFC, Cantidad, Fecha, Numero  FROM Entregan AS E  WHERE E.RFC  IN ( SELECT RFC  FROM Proveedores AS P  WHERE RazonSocial LIKE 'La%' AND E.Numero BETWEEN 5000 AND 5010 ) |
| **Salida** | AAAA800101 165.00 1998-07-08 00:00:00.000 5000  CCCC800101 582.00 2001-07-29 00:00:00.000 5002 |
| **Número de renglones reportados** | 16 rows |

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.

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** |  |
| **Salida** |  |
| **Número de renglones reportados** |  |

Realiza un ejemplo donde apliques algún operador: ALL, SOME o ANY.

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** |  |
| **Salida** |  |
| **Número de renglones reportados** |  |

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.**

Los dos valores de hasta arriba en orden ascendente de la tabla Proyectos

SELECT TOP 2 \* FROM Proyectos

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | SELECT TOP 2 \* FROM Proyectos |
| **Salida** | 5000 Vamos Mexico  5001 Aztecón |
| **Número de renglones reportados** | 2 rows |

**¿Qué sucede con la siguiente consulta? Explica por qué.**

No se puede realizar debido a que necesito un valor

SELECT TOP Numero FROM Proyectos

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | SELECT TOP Numero FROM Proyectos |
| **Salida** | Msg 102, Level 15, State 1, Line 1  Incorrect syntax near 'Numero'. |
| **Número de renglones reportados** | 0 rows |

# Modificando la estructura de una tabla existente.

Agrega a la tabla materiales la columna PorcentajeImpuesto con la instrucción:

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

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | ALTER TABLE materiales ADD PorcentajeImpuesto NUMERIC(6,2); |
| **Salida** | Commands completed successfully. |
| **Número de renglones reportados** | 0 rows |

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;

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | UPDATE materiales SET PorcentajeImpuesto = 2\*clave/1000; |
| **Salida** | (44 rows affected) |
| **Número de renglones reportados** | 0 rows |

esto es, a cada material se le asignará un impuesto igual al doble de su clave dividida entre diez.

Revisa la tabla de materiales para que compruebes lo que hicimos anteriormente.

¿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?

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | SELECT Clave, Descripcion, Costo, PorcentajeImpuesto,  Costo + PorcentajeImpuesto AS Importe  FROM Materiales |
| **Salida** | 1000 Varilla 3/16 100.00 2.00 102.00  1010 Varilla 4/32 115.00 2.02 117.02 |
| **Número de renglones reportados** | 44 rows |

# 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.

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | CREATE VIEW clave1000 AS  SELECT Clave, Descripcion, Costo, PorcentajeImpuesto  FROM Materiales  WHERE Clave=1000 |
| **Salida** | Commands completed successfully. |
| **Número de renglones reportados** | 0 rows |

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | CREATE VIEW materiales2000 AS  SELECT Descripcion  FROM Materiales, Entregan  WHERE Entregan.Fecha BETWEEN '2000/01/01' AND '2000/31/12' AND Materiales.Clave=Entregan.Clave |
| **Salida** | Commands completed successfully. |
| **Número de renglones reportados** | 0 rows |

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | CREATE VIEW materiales\_si AS  SELECT \* FROM Materiales WHERE Descripcion LIKE 'Si%' |
| **Salida** | Commands completed successfully. |
| **Número de renglones reportados** | 0 rows |

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | CREATE VIEW materiales2000\_distintos AS  SELECT DISTINCT Descripcion  FROM Materiales, Entregan  WHERE Entregan.Fecha BETWEEN '2000/01/01' AND '2000/31/12' AND Materiales.Clave=Entregan.Clave |
| **Salida** | Commands completed successfully. |
| **Número de renglones reportados** | 0 rows |

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** | CREATE VIEW proveedoresLA5000a5010 AS  SELECT RFC, Cantidad, Fecha, Numero  FROM Entregan AS E  WHERE E.RFC  IN ( SELECT RFC  FROM Proveedores AS P  WHERE RazonSocial LIKE 'La%' AND E.Numero BETWEEN 5000 AND 5010 ) |
| **Salida** | Commands completed successfully. |
| **Número de renglones reportados** | 0 rows |

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".

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** |  |
| **Salida** |  |
| **Número de renglones reportados** |  |

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

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** |  |
| **Salida** |  |
| **Número de renglones reportados** |  |

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

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** |  |
| **Salida** |  |
| **Número de renglones reportados** |  |

El Total entregado por cada material en el año 2000.

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** |  |
| **Salida** |  |
| **Número de renglones reportados** |  |

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

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** |  |
| **Salida** |  |
| **Número de renglones reportados** |  |

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

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** |  |
| **Salida** |  |
| **Número de renglones reportados** |  |

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

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** |  |
| **Salida** |  |
| **Número de renglones reportados** |  |

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).

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** |  |
| **Salida** |  |
| **Número de renglones reportados** |  |

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).

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** |  |
| **Salida** |  |
| **Número de renglones reportados** |  |

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.

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** |  |
| **Salida** |  |
| **Número de renglones reportados** |  |

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

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** |  |
| **Salida** |  |
| **Número de renglones reportados** |  |

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

|  |  |
| --- | --- |
| **Parámetro** | **Ejemplo** |
| **Sentencia SQL** |  |
| **Salida** |  |
| **Número de renglones reportados** |  |

Muchas de estas consultas requieren la utilización de funciones agregadas...

Se recomienda que revises nuevamente la lectura.