PRACTICAS CURSO DE TRANSACT SQL 2005

VISTAS:

1- **Empresa**: Una vista para obtener el número de pedidos que realiza cada compañía de envío y añadir una columna calculada que nos indique lo siguiente:
   a. 150 a 250 → Simple, 250 a 300 → Normal, 300 a 350 → Super
   b. Se verá algo así:

1	248	Simple
2	327	Super
3	255	Normal

2- **Empresa y Academia**: Escribir una vista que realice una unión entre todos los nombre de los proveedores (Bd.Empresa) y los nombres de los alumnos (Bd.Academia), además añadir una columna concepto que nos indique de que BD viene el registro:

	Contacto	Concepto
1	Ana María	Academia
2	Andrés	Academia
3	Anne Heikkonen	Empresa
4	Antonio	Academia
5	Antonio del Valle Saavedra	Empresa
6	Beate Vileid	Empresa
7	Belén	Academia
8	Carlos Diaz	Empresa
9	Chandra Leka	Empresa
10	Chantal Goulet	Empresa

3- **Ventas**: A partir de la BD de Ventas queremos obtener la siguiente vista con totales y subtotales por vendedor y producto. **Rollup**

	Vendedor	Producto	total
1	ANA	ACCESS 2000	1
2	ANA	Total	1
3	ANDRES	WINDOWS XP	2
4	ANDRES	Total	2
5	ANGEL	PEINE	1
6	ANGEL	Total	1
7	ANTONIO	ACCESS 2000	1
8	ANTONIO	EXCEL 2000	1
9	ANTONIO	Total	2
10	EVA	COPA	1
11	EVA	SECADOR	2
12	EVA	Total	3
13	JOSE LUIS	COPA	1
14	JOSE LUIS	Total	1
15	MARIO	VASO	1
16	MARIO	Total	1
17	MARTA	COPA	1
18	MARTA	Total	1
19	SANDRA	VAQUEROS	1
20	SANDRA	Total	1
21	Total	Total	13

4- **Ventas**: A partir de la BD de Ventas queremos obtener la siguiente vista con totales y subtotales por vendedor y producto y cruces. **Cube**

	Vendedor	Producto	total
1	ANA	ACCESS 2000	1
2	ANA	Total	1
3	ANDRES	WINDOWS XP	2
4	ANDRES	Total	2
5	ANGEL	PEINE	1
6	ANGEL	Total	1
7	ANTONIO	ACCESS 2000	1
8	ANTONIO	EXCEL 2000	1
9	ANTONIO	Total	2
10	EVA	COPA	1
11	EVA	SECADOR	2
12	EVA	Total	3
13	JOSE LUIS	COPA	1
14	JOSE LUIS	Total	1
15	MARIO	VASO	1
16	MARIO	Total	1
17	MARTA	COPA	1
18	MARTA	Total	1
19	SANDRA	VAQUEROS	1
20	SANDRA	Total	1
21	Total	Total	13
22	Total	ACCESS 2000	2
23	Total	COPA	3
24	Total	EXCEL 2000	1
25	Total	PEINE	1
26	Total	SECADOR	2
27	Total	VAQUEROS	1
28	Total	VASO	1
29	Total	WINDOWS XP	2

FUNCIONES:

1- Cadenas: Definir una variable, se le asigna un nombre (nombre apellido1 apellido2), escribir un bloque de código T-SQL que reparta cada parte en una variable distinta. Después imprimirlas por pantalla. Siempre existen los dos espacios en blanco.

2- Fechas, while: A partir de dos variables: mes y año de tipo int generar las fechas correspondientes de ese mes. Imprimirlas, teniendo en cuenta el número de días del mes, si el año es bisiesto, etc.

3- Rand, while: Escribir un script que genere 10 cuentas en la Base de datos **Banco**. Tenemos que utilizar números aleatorios para generar el saldo que debe estar entre 1000 y 5000 €. El número de cuenta se generará con 4 letras aleatorias y un número aleatorio de 6 cifras. Grabar los registros en la tabla. Podemos utilizar las funciones:

   a. `print char(65)` → 'A', `print char(90)` → 'Z'
   b. `print char(48)` → '0', `print char(57)` → '9'

FUNCIONES DEFINIDAS POR EL USUARIO:

1- **Empresa**: Escribir una función que reciba el id de un pedido y devuelva el importe total de un pedido. Tener en cuenta el cargo del pedido y la suma de todos los detalles de pedido, aplicando los descuentos cuando proceda. Escribir un código de prueba para probarlo, se debería emitir un mensaje por pantalla con este formato:

   a. `El importe del pedido 10248 es 206.38`

2- **Empresa**: Escribir una función que compruebe si existe un socio en la base de datos. La función recibirá el login y las password por parámetro, devolverá un 0 cuando el usuario exista y la password sea correcta, 1 cuando solo exista el login y un 2 en caso contrario no existe el socio.

3- **Empresa**: (con **Cursor**) Función que devuelve una tabla con la siguiente información. Necesitamos hacer un pedido a los proveedores con los productos que no tenemos suficientes existencias para servir todos los pedidos que nos han hecho. Ver la tabla de detalles de pedido y productos. La tabla generada será como la que se adjunta. Probar a ejecutar la función desde otra ventana.

	idproducto	total_en_pedido	existencias	A_pedir	Proveedor
1	1	828	39	789	Exotic Liquids
2	2	1057	17	1040	Exotic Liquids
3	3	328	13	315	Exotic Liquids
4	4	453	53	400	New Orleans Cajun Delights
5	5	298	0	298	New Orleans Cajun Delights
6	6	301	120	181	Grandma Kelly's Homestead
7	7	763	15	748	Grandma Kelly's Homestead
8	8	372	6	366	Grandma Kelly's Homestead
9	9	95	29	66	Tokyo Traders
10	10	742	31	711	Tokyo Traders

4- **Academia**: Calcular el sueldo final de cada profesor sabiendo que tienen 12 pagas + 500 € por beneficios. El cálculo se realiza en una función. La función recibirá el código del profesor, si no existe devolverá -1. Después en un script escribir una consulta que utilice la función.

5- **Academia:** Calcular el sueldo de los profesores de la siguiente manera: si el sueldo es mayor o igual que 1000 se aumenta un 5% y si es menor 8%. El cálculo se realiza en una función, si no existe devolverá -1. Después en un script escribir una consulta que utilice la función.

6- **Ventas:** Sobre la tabla de Ventas, tenemos 4 fechas:
    a. Fecha Primer Contacto
    b. Fecha Pedido
    c. Fecha Entrega
    d. Fecha Cancelación
    e. Mostrar todos los datos de las ventas y añadir una función que calcule lo siguiente:
    f. Cuando el número de días que se ha tardado en expedir el pedido desde la fecha del primer contacto sea inferior al número de días que se ha entregado el pedido desde su realización, se devolverá el mensaje: "a revisar por Almacén" cuando sea superior: "a revisar por comercial". En caso que haya una cancelación no se aplicará este criterio y se devolverá: "cancelación". Si no existe la venta que devuelva '' (cadena vacía).
    g. Escribir un script para probarlo.

PROCEDIMIENTOS ALMACENADOS:

1- Implementar un procedimiento almacenado que recibe el nombre completo de una persona y lo divide en nombre, apellido1 y apellido2. Escribir el código T-SQL para comprobar que funciona. El procedimiento debe controlar que recibe los datos correctos.

2- **Empresa**: Implementar un procedimiento almacenado que borre un socio a partir de un parámetro, antes de borrarlo comprobará si existe, si no existe imprimirá un mensaje por pantalla. En el caso de que exista lo copiará en una tabla histórica. En el caso de que esta tabla no exista la creará (utilizando una consulta de select into) y en caso contrario lo insertará (insert into). Si consigue borrarlo debe de emitir un mensaje por la consola.

3- **Banco**: Implementar un procedimiento que añada un movimiento a la tabla de movimientos, parametrizado mediante importe, fecha y número de cuenta. Se debe comprobar que la cuenta exista y que cuando grabe el movimiento que actualice la cuenta correspondiente. Por otra parte tener en cuenta que si la cantidad a retirar supera el saldo de la cuenta tampoco se permitirá sacar dinero (se emitirá un mensaje con el saldo disponible), las retiradas de dinero se indican mediante una cantidad negativa.
    a. Para probarlo, escribir un script de T-SQL que añada 6 movimientos a la primera cuenta de la tabla y 8 movimientos a la última cuenta creada. Con una fecha y las cantidades serán -100, 200, -300, 400, etc.

4- **Empresa**: Implementar un procedimiento almacenado para añadir un detalle a un determinado pedido. Se indicará por parámetro el idpedido, idproducto y cantidad. El descuento será 0. Considerar:
    a. El idpedido y el idproducto deben de existir.
    b. El precio lo toma de la tabla de productos.
    c. Comprobar si hay cantidad suficiente en existencias.
    d. Comprobar si hay ya un apunte del idpedido y del idproducto indicado. Si es así tiene que incrementar las unidades pedidas.
    e. Si está todo ok, crea el detalle y mediante el trigger2 se actualizarán las unidades en existencia en la tabla de productos.

TRIGGERS:

1- **Banco**: Modificar el proc3 (se llamará proc3_v2), sólo añade el movimiento, en caso de que exista la cuenta y la cantidad a retirar no supere el saldo. Ahora la actualización se va a realizar mediante un Trigger. Para probarlo utilizar el script creado para el proc3, pero llamando al proc3_v2.

2- **Empresa**: Asociado al proc4 se crea un trigger que será el encargado de actualizar las unidades en existencia de la tabla de productos al insertar el nuevo detalle.

3- **Empresa**: Asociado al proc4 se crea un trigger que será el encargado de actualizar las unidades en existencia de la tabla de productos al ACTUALIZAR un detalle ya existente.

CURSORES:

1- **Empresa**: Escribir un script y declarar un cursor para recorrer la tabla de Categorías e imprimir el nombre de la categoría por la consola. Y que imprima al final el número de filas.

TRANSACCIONES:

1- **Empresa**: Reescribir mediante un script el código del proc4, el trigger2 y el trigger3 se pretende que tanto la inserción del detalle y la actualización de la tabla de productos se realice todo junto como un única operación, controlando los posibles errores.