**Preguntas y respuestas SQL – Parcial**

**La siguiente sentencia proyectaria los articulos mas caros. Como modificaria la misma si no pudiese utlizar funciones de totales?**

select NroArticulo, Descripcion from articulos where precio = (select Max(precio) from articulos)

Respuesta:

select NroArticulo , Descripcion from articulos where precio = (select top 1 precio from articulos group by precio order by precio desc);

**La siguiente sentencia proyectaria las facturas que superen los $100 del total? Indicar si esta correctamente formulada, si no corregirla.**

select nroFactura, sum(precio\*cantidad) as total from itemFactura where total>100 group by NroFactura.

Respuesta:

Es incorrecta, primero no se puede usar un alias en la parte del where porque esa columna no existe y segundo al tener una clausula group by se debe usar el having. La forma correcta de hacerlo es:

select nroFactura, sum(precio\*cantidad) as total from itemFactura group by nroFactura having sum(precio\*cantidad)>100;

**¿Que proyecta la siguiente consulta? Si no es correcta explique porque**

Select nombre from clientes where nombre like ‘\_[AOE]%[ S]’

Respuesta:

Es correcta

Devuelve todos los nombres que tengan una combinación de 2 caracteres seguidos por cualquier otra cosa y que terminen con una S. El segundo de esos 2 caracteres tiene que ser una A una O e una E

**Indicar si es correcta o no la siguiente consulta. Teniendo en cuenta que el idCliente es PK en clientes y clave foranea en facturas. Indicar que proyectaria.**

select nroFactura, fecha, id\_cliente as codigo, clientes.nombre from facturas f, (select \* from clientes where….

Respuesta:

Incorrecta, no se puede utilizar el resultado de un select como si fuera una tabla. Para esto se utilizan las vistas.

**Cual es la diferencia entre la clausula WHERE y HAVING.**

Con WHERE podemos establecer una condición usando registros individuales, aquellos que cumplan con esta condición serán seleccionados (eliminados o actualizados); ahora bien, con HAVING podemos establecer una condición sobre un grupo de registros, algo muy importante es que HAVING acostumbra ir acompañado de la cláusula GROUP BY. Esto último es así dado que HAVING opera sobre los grupos que nos “retorna” GROUP BY. Entonces, WHERE actua sobre registros individuales y HAVING actua sobre grupos de registros, Ej.:

SELECT genero, director, SUM(recaudo) AS TOTAL FROM peliculas GROUP BY genero, director HAVING SUM(recaudo) > 80;

**Crear una tabla Producto con los campos código de producto (numérico), nombre, descripción, categoría (numérico), fecha de alta y precio. La descripción es opcional y tanto los códigos como los nombres no pueden repetirse.**

CREATE TABLE Producto (

codProd INT NOT NULL,

nombre VARCHAR (30) NOT NULL,

descripcion VARCHAR (100) NULL,

categoria INT NOT NULL,

fechaAlta DATE NOT NULL,

precio FLOAT NOT NULL,

CONSTRAIN pk\_producto

PRIMARY KEY (codProd),

CONSTRAIN uk\_producto

UNIQUE (nombre),

)

**Crear una tabla Oferta con los campos número de oferta, código de producto (vinculado a la tabla Producto), el porcentaje de descuento y la fecha de inicio y fin de vigencia. Además, la fecha de vigencia de inicio siempre debe ser menor a la fecha de fin.**

CREATE TABLE Oferta(

numOfer INT NOT NULL,

codProd INT NOT NULL,

porDes INT NULL,

fechaIni DATE NOT NULL,

fechaFin DATE NOT NULL,

CONSTRAIN pk\_oferta

PRIMARY KEY (numOfer),

CONSTRAIN fk\_producto

FOREIGN KEY (codProd)

REFERENCES Producto.codProd,

CONSTRAIN ck\_fechas CHECK (fechaIni < fechaFin)

)

**Insertar el producto con código 1, nombre ‘Fideos’, descripción ‘Pastas secas’, categoría 8 y precio 15$ (suponer la existencia de la función NOW() que devuelve la fecha actual)**

INSERT INTO Producto (codProd,nombre,descripcion,categoria,fechaAlta,precio) VALUES

(1,'fideos','pastas secas',8,NOW(),15);

**Actualizar el precio de los fideos, aumentándolos un 20%.**

UPDATE Producto SET precio = precio + precio \* 0,20 WHERE codProd = 1;

**Eliminar a todos los productos cuyo precio sea superior a 100$ y hayan sido dados de alta en el último año (suponer la existencia de la función YEAR() que toma un DATE y devuelve el año)**

DELETE FROM Producto WHERE precio > 100 AND YEAR(fechaAlta) = 2014;

**Obtener el nombre, descripción y precio de los productos de precio menor a 50$, ordenados por precio de mayor a menor.**

SELECT nombre,descripcion,precio FROM Producto WHERE precio < 50 BY DESC;

**Por cada categoría, obtener el máximo, mínimo y precio promedio.**

SELECT categoria,MIN (precio),MAX(precio) AVG(precio) FROM Producto GROUP BY categoria;

**Obtener los nombres de los productos que tienen ofertas vigentes, sin repetidos.**

SELECT DISTINCT p.nombre FROM Producto p, Oferta o WHERE p.codProd = o.codProd AND o.fechaFin > NOW();

**De los productos con precios menores a 100$, obtener por cada categoría cuántos productos hay en ella.**

SELECT categoria, COUNT (\*) FROM Producto WHERE precio < 100 GROUP BY categoria

**Crear una vista en SQL**

CREATE VIEW vw\_estudios\_sin\_cobertura

AS

SELECT descripcion

FROM Estudios

WHERE idEstudio not in (select idEstudio from Coberturas)

**Crear un store procedure**

CREATE PROCEDURE HumanResources.uspGetEmployeesTest2

@LastName nvarchar(50),

@FirstName nvarchar(50)

AS

SET NOCOUNT ON;

SELECT FirstName, LastName, Department

FROM HumanResources.vEmployeeDepartmentHistory

WHERE FirstName = @FirstName AND LastName = @LastName

AND EndDate IS NULL;

**Ejecutar un store procedure**

EXEC HumanResources.uspGetEmployeesTest2 @LastName = N'Ackerman', @FirstName = N'Pilar';