**EXAMEN DE INGRESO**

**ÁREA DE DESARROLLO**

**A. PRUEBA TEÓRICA**

1. Defina que entiende por:
   1. Inner join

Hace referencia a la búsqueda de coincidencias que existen entre dos tablas con alguna columna que las relaciona, generalmente es usada la sentencia INNER JOIN o también JOIN, al emplearlo solo tomará estos datos y el resto de atributos que acompañes estos datos

* 1. Left join

Hace referencia a la búsqueda de coincidencias que existen entre dos tablas con alguna columna que las relaciona, pero a diferencia de JOIN, va a tomar todos los datos de la primera tabla y los que se relacionan en la segunda. Si hay datos que de la primera tabla no coinciden con la segunda, traerá un NULL de la segunda tabla

* 1. Having

Permite reemplazar la funcionalidad de WHERE con funciones agregadas de SQL que permitirán realizar cálculos específicos con columnas que contienen datos numéricos.

* 1. Substring

Permite tomar la parte de una cadena de datos almacenada, obteniendo cierta cantidad de caracteres solicitada.

* 1. Right Join

Hace referencia a la búsqueda de coincidencias que existen entre dos tablas con alguna columna que las relaciona, al igual que LEFT JOIN, solo que con la diferencia de que no va a tomar todos los datos de la primera tabla sino de la segunda tabla, y los que se relacionan en la primera tabla, si hay datos en la segunda tabla que no coinciden con la primera, traerá un NULL de la primera tabla

* 1. Union All

Permite unir dos tablas con la condición que cuenten con el mismo parámetro de columnas en el mismo orden, pero trae todos los valores contenidos en ellas, incluyendo si estos valores son duplicados.

* 1. Cast
  2. Merge

Esta función MERGE permite realizar varios pasos necesarios en una sola consulta de tablas, pasos como Insertar, actualizar o eliminar de una tabla, y así mismo verificar si se realizó ese proceso

* 1. Vista

Esta función permitirá mostrar datos de varias tablas, pero como si fuera una tabla adicional virtual que no se encuentra en la base de datos, pero si se puede acceder a ella a través de la vista.

* 1. Procedimiento almacenado (stored Procedure).

Permite declarar procedimientos de consultas Query que contienen funciones predeterminadas de algún proceso de SQL, es decir, crea una función general con una consulta SQL, agregando también parámetros generales que complementan esta función, como crear una tabla, se puede crear un Procedimiento Almacenado llamado TABLA con la función CREATE table, y mi proceso quiere crear varias tablas con diferentes columnas, entonces agrego los parámetros de creación en ese proceso: @column 1, @columna 2, etc. Así cada que se llame la función TABLA se agregan los parámetros con los nombres de cada columna: EXEC TABLA primeraColumna, ColumnaID. Y asi se creo una table con esos nombres de columnas.

1. Escribir la sintaxis para insertar un registro en una tabla.

INSERT INTO tabla(Registro)

Teniendo en cuenta que tabla tiene una columna única, y el valor que se inserta es Registro

1. Escribir la sintaxis para actualizar un registro en una tabla.

UPDATE tabla

SET Registro = New\_Registro

WHERE <Parametro de condición>

1. Escriba la sintaxis para utilizar el Having.

CREATE TABLE ejemplo(

Precio int,

Name varchar(50)

)

SELECT COUNT(Precio),

FROM ejemplo

GROUP BY Name

HAVING COUNT (Precio) > 50000;

Esta consulta permite crear la tabla ejemplo, y luego va a comparar el conteo de precios agrupando por los nombres que completan un precio mayor de 50000

1. Se tiene la siguiente tabla y sobre ella realizar las siguientes consultas SQL.

**TABLA A**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **NOMBRE** | **CÉDULA** | **TELÉFONO** | **CIUDAD** |
| 1 | LUIS ALBERTO | 1.111.111 | 3234569 | BOGOTÁ |
| 2 | CRISTIAN CAMILO | 2.222.222 | 4152637 | CALI |
| 3 | JULIAN TREJOS | 3.333.333 | 7418529 | CARTAGENA |
| 4 | CARLOS ALFREDO | 4.444.444 | 1237895 | CALI |
| 5 | JULIAN TREJOS | 3.333.333 | 7418529 | CARTAGENA |
| 6 | LUIS ALBERTO | 1.111.111 | 3234569 | BOGOTÁ |
| 7 | JUAN CAMILO | 5.555.555 | 8527492 | BOGOTA |
| 8 | SAUL ANDRES | 6.666.666 | 7891235 | MANIZALES |
| 9 | JUAN CARLOS | 7.777.777 | 4561287 | MANIZALES |
| 10 | JUAN CARLOS | 88.888.888 | 30031833320 | CARTAGENA |

1. Realizar una consulta en la que se muestren únicamente las cédulas duplicadas.

SELECT \*

FROM TABLA A

GROUP BY CÉDULA

HAVING COUNT(\*)>1

1. Realizar una consulta en la que se muestren todos los registros pero los duplicados deben aparecer una sola vez.

SELECT \*

FROM [TABLA A] [A]

UNION

SELECT \*

FROM [TABLA A] [B]

1. Realizar una consulta en la que se muestren todos los registros cuyos teléfonos contengan 12.

SELECT ID, NOMBRE, CÉDULA,

TELÉFONO, CIUDAD

FROM [TABLA A]

Where CHARINDEX(‘12’,TELÉFONO) IN TELÉFONO

1. Realizar una consulta en la que muestre el nombre del cliente y la cantidad de teléfonos que tiene organizados en orden alfabético.

SELECT NOMBRE, TELÉFONOS

FROM [TABLA A]

ORDER BY NOMBRE ASC ,TELÉFONOS ASC

1. A continuación presentamos otra tabla en la cual se encuentran varios registros que deberá cruzar con la tabla anterior y desplegar toda la información.

# TABLA B.

|  |  |  |
| --- | --- | --- |
| **CÉDULA** | **DIRECCIÓN** | **Saldo** |
| 1.111.111 | CRA 777 CLA 50 | 11.236.590 |
| 6.666.666 | CALLE 6 NRO 25-36 | 598.600 |
| 3.333.333 | CALLE 70 No 85-95 | 7.600.500 |
| 4.444.444 | CALLE 100 14 | 3.000.001 |

* 1. Realizar una consulta en la que se crucen los registros de la tabla A con tabla B de tal manera que aparezcan todos los registros de A, crucen o no crucen con la tabla B.

SELECT \*

FROM [TABLA A]

LEFT JOIN [TABLA B]

ON [TABLA A].CÉDULA = [TABLA B].CÉDULA

* 1. Realizar una consulta en la que se actualice el saldo a cada cliente, restándole 500.000 al saldo actual solo para los que se encuentren en la ciudad de Bogotá.

SELECT \*

FROM [TABLA A]

UNION ALL

SELECT \*

FROM [TABLA B]

* 1. Realizar una consulta que muestre por ciudad la cantidad de clientes y el valor total del saldo.

**7.** Suponga que tiene la siguiente clase, la cual representará el ítem de un menú de aplicación:

|  |
| --- |
| public class Item  {  public string IdItem { get; set; }  public string Titulo { get; set; }  public string IdItemPadre { get; set; }  } |

Esta clase tiene como propiedades un “IdItem” que representa el identificador de cada Ítem; un “Titulo”, que representa el título o nombre del ítem y un” IdItemPadre”, el cual indica si un ítem es un subitem (hijo de otro ítem); de este modo el “IdItemPadre” indica cual es el ítem de nivel superior. De acuerdo con lo anterior se requiere imprimir en pantalla el menú de la aplicación de la siguiente forma:

* Administración
  + Clientes
  + Productos
    - Editar Productos
    - Inventario
* Gestión
  + Despacho de Pedidos

Suponiendo que esta información se encuentra contenida en un array de objetos de tipo “Item”, ya existente y cargado con los siguientes datos:

|  |  |  |
| --- | --- | --- |
| **IdItem** | **Titulo** | **IdItemPadre** |
| 1 | Administración | Null |
| 2 | Gestion | Null |
| 3 | Clientes | 1 |
| 4 | Productos | 1 |
| 5 | Editar Productos | 4 |
| 6 | Inventario | 4 |
| 7 | Despacho de Pedidos | 2 |

De acuerdo con estos datos y suponiendo que cada fila de la tabla representa un Item del array, escriba un algoritmo con los métodos (funciones), capaces de recorrer dicho array e imprimir el menú en pantalla. Represente su algoritmo escribiendo la sintaxis en el lenguaje de programación C# o alguno de los siguientes lenguajes de programación (C#, VisualBasic, Javascript ).

public class Item

{

public string IdItem { get; set; }

public string Titulo { get; set; }

public string IdItemPadre { get; set; }

}

public Class Action

{

}