

PRUEBA # 1

CONCEPTOS DE PROGRAMACIÓN

Fecha: Miércoles 22 de Diciembre de 2021

Nombre y apellidos: Yeison David Quinto Bolaño

1. (1 Pto) ¿En programación orientada a objetos (POO), a que se refiere el término clase?

Una "clase" en la POO es un modelo o prototipo que define el usuario a partir del cual se crean los objetos. Este contiene las propiedades o métodos que son comunes en todos los objetos de un tipo, por lo general estas clases contienen componentes o declaraciones en orden de: Modificadores, Nombre de clase, Superclase, Interfaces y cuerpo.

2. (1 Pto) ¿En programación orientada a objetos (POO), a que se refiere el término objeto?

Un "objeto" en un lenguaje de POO hace referencia a un tipo específico, o "instancia", de una clase. Se tiene que un objeto tiene una estructura similar a otros objetos de la clase, pero se le pueden asignar características individuales. Un objeto también puede llamar a funciones o métodos específicos de ese objeto.

3. (1 Pto) ¿Cuál es la diferencia entre una clase y un objeto?

Se tiene que una en la programación orientada a objeto (POO) "la clase" y "el objeto" son los dos conceptos más importantes de dicha programación, dicho esto definimos su principal diferencia entre estas dos, es que la clase es un plano que se utiliza para crear diferentes objetos del mismo tipo.

4. (1 Pto) ¿Qué es una arquitectura basada en capas?, defina cada uno de sus niveles o capas.

La arquitectura en capas consta en dividir la aplicación en capas, con la intención de que cada capa tenga un rol muy definido, como podría ser, una capa de presentación (UI), una capa de reglas de negocio (servicios) y una capa de acceso a datos (DAO), sin embargo, este estilo arquitectónico no define cuántas capas debe tener la aplicación, sino más bien, se centra en la separación de la aplicación en capas. la mayoría de las veces este estilo arquitectónico es implementado en 4 capas, presentación, negocio, persistencia y base de datos, sin embargo, es habitual ver que la capa de negocio y persistencia se combinan en una sola capa, sobre todo cuando la lógica de persistencia está incrustada dentro de la capa de negocio.

En una arquitectura en capas, todas las capas se colocan de forma horizontal, de tal forma que cada capa solo puede comunicarse con la capa que está inmediatamente por debajo, por lo que, si una capa quiere comunicarse con otras que están mucho más abajo, tendrán que hacerlo mediante la capa que está inmediatamente por debajo.

La capa de presentación: contiene todas las clases responsables de presentar la interfaz de usuario al usuario final o enviar la respuesta al cliente (en caso de que estemos operando profundamente en el back-end).

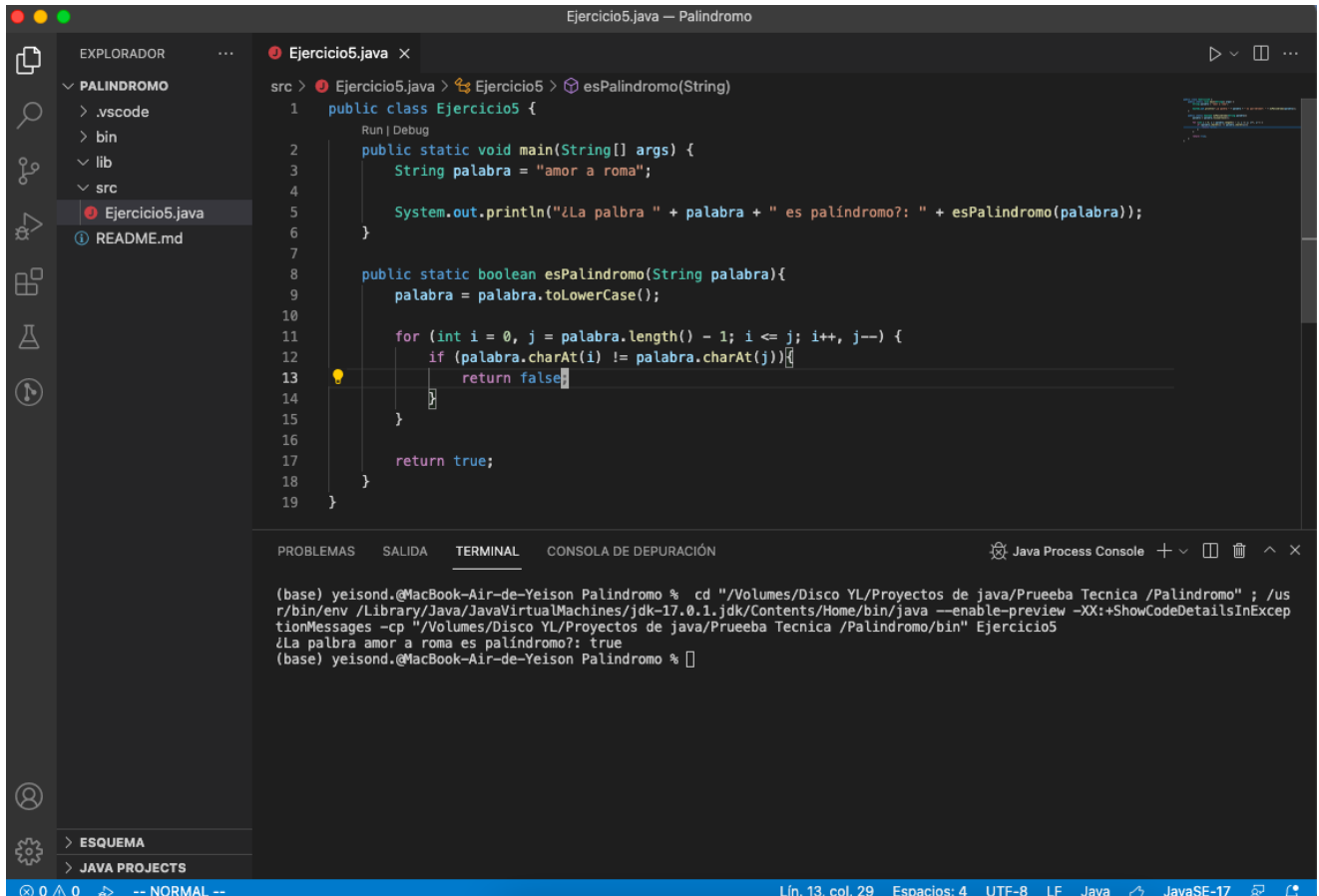
La capa de negocio: contiene toda la lógica que requiere el negocio para cumplir con sus requisitos funcionales y, al mismo tiempo, no forma parte de las reglas de persistencia. En la mayoría de los sistemas con los que he trabajado, la capa de negocio consistía en servicios que orquestaban los objetos de persistencia para cumplir con un escenario de caso de uso.

La capa de persistencia: representa la persistencia subyacente, que consta principalmente de entidades de persistencia y, en algunos casos, servicios. Las reglas comerciales, como las invariantes y los algoritmos, deben permanecer en esta capa.

La **capa de infraestructura** (también conocida como capa de persistencia) contiene todas las clases responsables de hacer las cosas técnicas, como conservar los datos en la base de datos, como DAO, repositorios o cualquier otra cosa que esté usando.

5. (2 Pto) Una palabra o frase palíndroma es aquella que se lee y escribe de igual forma al derecho y al revés, por ejemplo (arenara, oso, Ana, reconocer, amor a roma), diseñe un algoritmo (en pseudo-código o en el lenguaje programación de su preferencia) que reciba como parámetro una palabra o frase y retorne un mensaje informando al usuario si la palabra ingresada es palíndroma o no.

Nota: No se permite el uso de funciones avanzadas como reverse de java u otras.



```
src > Ejercicio5.java > Ejercicio5 > esPalindromo(String)
1 public class Ejercicio5 {
2     public static void main(String[] args) {
3         String palabra = "amor a roma";
4
5         System.out.println("La palabra " + palabra + " es palíndromo?: " + esPalindromo(palabra));
6     }
7
8     public static boolean esPalindromo(String palabra){
9         palabra = palabra.toLowerCase();
10
11         for (int i = 0, j = palabra.length() - 1; i <= j; i++, j--) {
12             if (palabra.charAt(i) != palabra.charAt(j)) {
13                 return false;
14             }
15         }
16
17         return true;
18     }
19 }
```

TERMINAL

```
(base) yeisond.@MacBook-Air-de-Yeison Palindromo % cd "/Volumes/Disco YL/Proyectos de java/Prueba Tecnica /Palindromo" ; /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-17.0.1.jdk/Contents/Home/bin/java --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp "/Volumes/Disco YL/Proyectos de java/Prueba Tecnica /Palindromo/bin" Ejercicio5
La palabra amor a roma es palíndromo?: true
(base) yeisond.@MacBook-Air-de-Yeison Palindromo %
```

6. (1 Pto) ¿Cuál es el resultado de ejecutar el siguiente pseudo-código con el valor “5”?

```
funcion metodoQueHaceAlgo(Entero valor)
    si (valor < 3) entonces
        retornar valor;
    fin si
    retornar metodoQueHaceAlgo(valor-1) * metodoQueHaceAlgo(valor-2);
fin funcion
```

Analizando el resultado tenemos que el valor que nos arroja es “8”.

7. (1 Pto) Realice un algoritmo que reciba como parámetro dos números enteros y retorne la división de ambos números.

The screenshot shows a VS Code editor with two Java files. The left file, `Ejercicio7.java`, contains a `main` method that creates a `Division` object and calls its `dividir` method with arguments 7 and 2. The right file, `Division.java`, contains a `Division` class with a `dividir` method that takes two integers and returns their quotient. The terminal at the bottom shows the execution of the program, which outputs "El resultado es 3".

```
src > Ejercicio7.java > Ejercicio7 > main(String[])
1 public class Ejercicio7{
2     public static void main(String[] args) throws Exception {
3         Division num = new Division();
4         int resul = num.dividir(7,2);
5         System.out.print("El resultado es " + resul);
6     }
7 }
8
9 }
```

```
src > Division.java > Division
1 public class Division { int n1;
2     int n2;
3
4     public int dividir(int arg1, int arg2){
5         n1= arg1;
6         n2= arg2;
7         int resul= arg1/arg2;
8         return resul;
9     }
10 }
11 }
```

PROBLEMAS SALIDA TERMINAL CONSOLA DE DEPURACIÓN

```
(base) yeisond.@MacBook-Air-de-Yeison Ejercicio7 % cd "/Volumes/Disco YL/Proyectos de java/Prueba Tecnica /Ejercicio7/Ejercicio7" ; /usr/bin/env /Library/Java/JavaVir
tualMachines/jdk-17.0.1.jdk/Contents/Home/bin/java --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp "/Volumes/Disco YL/Proyectos de java/Prueba Tecnica /Ej
ercicio7/Ejercicio7/bin" Ejercicio7
El resultado es 3
(base) yeisond.@MacBook-Air-de-Yeison Ejercicio7 % cd "/Volumes/Disco YL/Proyectos de java/Prueba Tecnica /Ejercicio7/Ejercicio7" ; /usr/bin/env /Library/Java/JavaVir
tualMachines/jdk-17.0.1.jdk/Contents/Home/bin/java --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp "/Volumes/Disco YL/Proyectos de java/Prueba Tecnica /Ej
ercicio7/Ejercicio7/bin" Ejercicio7
El resultado es 3
(base) yeisond.@MacBook-Air-de-Yeison Ejercicio7 %
```

Lín. 4, col. 34 Espacios: 4 UTF-8 LF Java JavaSE-17

8. (2 Pto) Escriba un algoritmo que imprima los números del 1 al 100. Pero para los múltiplos de 3 imprima "Fizz" en lugar del número y para los múltiplos de 5 imprima "Buzz". Para los números que son múltiplos de ambos imprima "FizzBuzz".

The screenshot shows a VS Code editor with two Java files. The left file, `Ejercicio8.java`, contains a `main` method that creates a `Numero` object and calls its `imprimirN` method. The right file, `Numero.java`, contains a `Numero` class with an `imprimirN` method that loops from 1 to 100 and prints the appropriate output based on the divisibility rules. The terminal at the bottom shows the output of the program, which prints the numbers 1 to 100, replacing multiples of 3 with "Fizz", multiples of 5 with "Buzz", and multiples of both with "FizzBuzz".

```
src > Ejercicio8.java > Ejercicio8 > main(String[])
1 public class Ejercicio8 {
2     public static void main(String[] args) throws Excep
3         Numero listaN = new Numero();
4         listaN.imprimirN();
5     }
6 }
7 }
```

```
src > Numero.java > Numero
1 public class Numero {
2     int i;
3     public void imprimirN(){
4         for (i= 1; i<=100; i++){
5             if (i % 3 == 0 && i % 5 == 0){
6                 System.out.println("FizzBuzz");
7             }else if (i%3 == 0){
8                 System.out.println("Fizz");
9             }else if ( i%5 == 0){
10                System.out.println("Buzz");
11             }else {
12                 System.out.println(i);
13             }
14         }
15     }
16 }
```

PROBLEMAS SALIDA TERMINAL CONSOLA DE DEPURACIÓN

```
92
Fizz
94
Buzz
Fizz
97
98
Fizz
Buzz
(base) yeisond.@MacBook-Air-de-Yeison EJERCICIO8 %
```

Lín. 16, col. 2 Espacios: 4 UTF-8 LF Java JavaSE-17

9. Nombre tres tendencias actuales en el área de software.

Software ‘low code’ y ‘no code’.

La automatización y la creación de aplicaciones se está volviendo más importante para todos, desde las grandes organizaciones hasta las pequeñas empresas, e incluso los profesionales y aficionados.

IA en todas partes.

A medida que más personas esperan que los sitios web y los dispositivos satisfagan todas sus nuevas necesidades, se está incorporando más Inteligencia Artificial (IA), lo que llevará a una explosión de dispositivos, aplicaciones, sitios web y herramientas con esta tecnología en 2022.

Web3 y Tecnología Blockchain aplicada.

Después de la explosión de la tecnología blockchain aplicada al mundo financiero (gracias a sus capacidades de descentralización), esta evolucionó rápidamente para dar soporte a contratos inteligentes, abriendo las puertas a ciertos nichos de mercado y nuevos casos de uso.

BASES DE DATOS

1. (1 Pto) ¿Qué es una “primary key” o clave principal o primaria?

Una clave primaria es un conjunto mínimo de atributos (columnas) en una tabla que identifica de forma única las tuplas (filas) en esa tabla.

2. (1 Pto) ¿Qué son las “foreign keys” o claves externas o foráneas?

Las claves externas son las columnas de una tabla que apuntan a la clave principal de otra tabla. Actúan como una referencia cruzada entre tablas.

3. (1 Pto) ¿Qué es un stored procedure o procedimiento almacenado?

Un procedimiento almacenado es un grupo de una o más sentencias SQL precompiladas en una unidad lógica. Se almacena como un objeto dentro del servidor de la base de datos. Es una subrutina o un subprograma en el lenguaje informático común que se ha creado y almacenado en la base de datos. Cada procedimiento en SQL Server siempre contiene un nombre, listas de parámetros y declaraciones Transact-SQL . El servidor de base de datos SQL almacena los procedimientos almacenados como objetos con nombre . Podemos invocar los procedimientos usando disparadores, otros procedimientos y aplicaciones como Java , Python , PHP , etc. Puede soportar casi todos los sistemas de bases de datos relacionales.

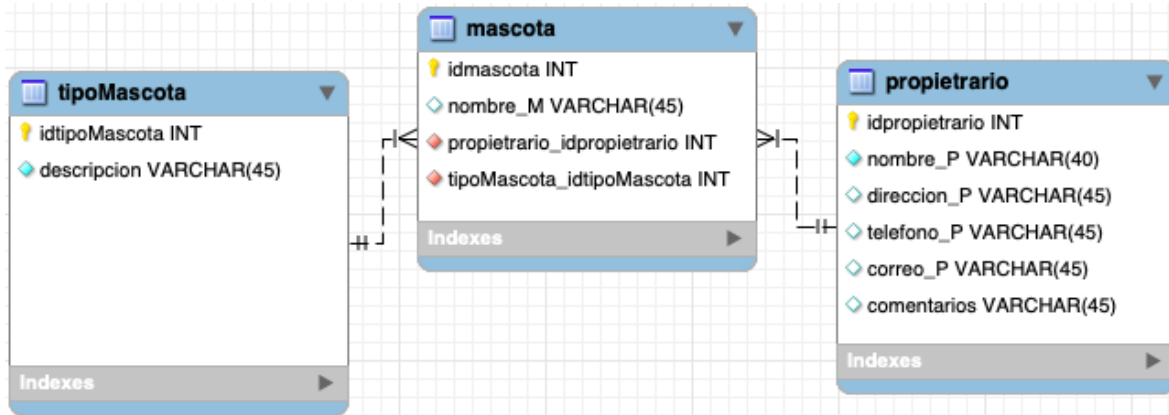
La empresa “Mascotas & Mascotas”, rescata animales que han sido abandonados, los rehabilitan y cuando están totalmente sanos, los ponen a disposición para que estos sean adoptados, se tienen las siguientes entidades:

Tabla mascotas
Código
Nombre
Tipo mascota
Código propietario

Tabla Propietarios
Código
Nombre

Tabla Tipo mascota
Código
Descripción

4. (2 Pto) Elabore el modelo entidad relación lo más detallado posible (relaciones, tipos de datos, llaves primarias y foráneas, etc...), si ud. lo considera necesario, puede adicionar más entidades o tablas.



De acuerdo al modelo ER que Ud. acaba de elaborar, escriba las siguientes instrucciones SQL:

5. (1 Pto) Listar todas las mascotas.

```
SELECT mascota. idmascota, nombre_M, tipoMascota. descripcion
FROM mymdb.mascota
INNER JOIN mymdb. tipoMascota
ON mascota. tipoMascota_idtipoMascota = tipoMascota. idmascota;
```

6. (1 Pto) Listar las mascotas que no han sido adoptadas.

```
SELECT mascota. idmascota, nombre_M, tipoMascota.descripcion
FROM mymdb.mascota
LEFT JOIN mymdb.tipoMascota
ON mascota. tipoMascota_idtipoMascota = tipoMascota. idtipoMascota
LEFT JOIN mymdb.propietario
ON mascota. propietario_idpropietario = propietario. idpropietario
WHERE mascota. propietario_idpropietario IS NULL;
```

7. (1 Pto) Listar el número de mascotas por cada tipo de mascota.

```
SELECT DISTINCT tipomascota.*, COUNT(*) AS mascotas_cnt
FROM mymdb.mascota m
INNER JOIN mymdb.tipomascota
ON m.tipomascota_id = tipomascota.id
GROUP BY tipomascota. idtipoMascota;
```

8. (1 Pto) Listar los propietarios que tengan más de una mascota.

```
SELECT DISTINCT propietario.*, COUNT(*) AS mascota_cnt
FROM mymdb.mascota m
INNER JOIN mymdb.propietario
ON m. propietario_idpropietario = propietario. idpropietario
GROUP BY propietario. idpropietario
HAVING COUNT(mascota_cnt) > 1;
```

9. (1 Pto) Listar el número de mascotas por cada tipo de mascota y por cada propietario.

```
SELECT DISTINCT tipomascota.*, propietario.*, COUNT(*) AS mascotas_cnt
```

```

FROM mymdb.mascota m
LEFT JOIN mymdb.tipoMascota
ON m. tipoMascota_idtipoMascota = tipomMscota. idtipoMascota
LEFT JOIN mymdb.propietario
ON m.propietario_idpropietario = propietario. idpropietario
WHERE m. propietario_idpropietario = propietario. idpropietario
GROUP BY tipoMascota. idtipoMascota
ORDER BY tipoMascota. idtipoMascota ASC ;

```

10. (1 Pto) Listas los propietarios que no tienen mascotas.

```

SELECT DISTINCT propietario. idpropietario, propietario. nombre_P, propietario.comentarios
FROM mymdb.mascota JOIN mymdb.propietario
ON mascota. propietario_idpropietario != propietario. idpropietario
ORDER BY propietario. idpropietario ASC;

```

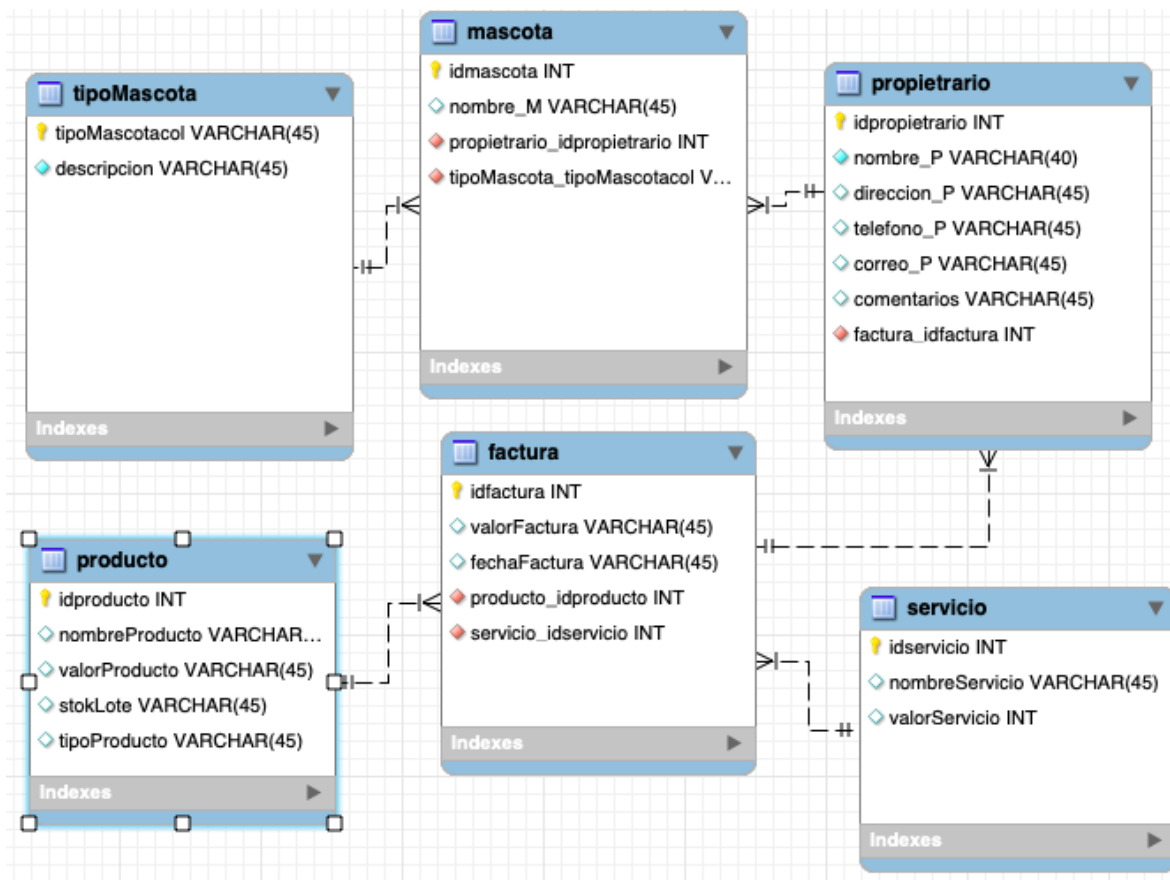
11. (1 Pto) Cree un stored procedure o procedimiento almacenado con cualquiera de las sentencias anteriores.

```

drop procedure if exists lisNumMasxMasPro;
CREATE PROCEDURE lisNumMasxMasPro()
BEGIN
SELECT DISTINCT tipomascota.*, propietario.*, COUNT(*) AS mascotas_cnt
FROM mymdb.mascota m
LEFT JOIN mymdb.tipoMascota
ON m. tipoMascota_idtipoMascota = tipomMscota. idtipoMascota
LEFT JOIN mymdb.propietario
ON m.propietario_idpropietario = propietario. idpropietario
WHERE m. propietario_idpropietario = propietario. idpropietario
GROUP BY tipoMascota. idtipoMascota
ORDER BY tipoMascota. idtipoMascota ASC ;
END //
delimiter ;
CALL lisNumMasxMasPros();

```

12. (2 Pto) La empresa “Mascotas & Mascotas”, además de prestar el servicio de adopción, también presta el servicio de veterinaria al público, la empresa desea conocer cuáles son las ventas mensuales por cada propietario. ¿Qué cambios le realizaría al modelo entidad relación para cumplir con este requisito?. Realice la sentencia SQL para obtener esta información.



13. (1 Pto) Escriba una sentencia SQL que actualice la columna “Nombre” de la tabla **mascota**, de modo que los nombres de todas las mascotas queden en mayúscula.

```
UPDATE mymdb.mascota
set nombre_M = UPPER(nombre_M)
```

14. (1 Pto) Escriba una sentencia SQL que elimine los propietarios que no tienen **mascotas**.

```
DELETE FROM mymdb.propietario
WHERE mascota. propietario_idpropietario!= propietario. idpropietario
```

15. (1 Pto) Explique las siguientes funciones de agregación:

a. COUNT()

La función COUNT devuelve el número de filas de la consulta, es decir, el número de registros que cumplen una determinada condición.

b. MAX()

La función MAX sirve para obtener el mayor valor para una columna determinada.

c. MIN()

La función MIN sirve para obtener el valor más pequeño para una columna determinada.