

Técnico Superior en Desarrollo de Aplicaciones Multimedia

Curso Académico 2021/2022 - 2º DAMS

Alumno: José Enrique Jordán Moreno

Modulo: Acceso a Datos

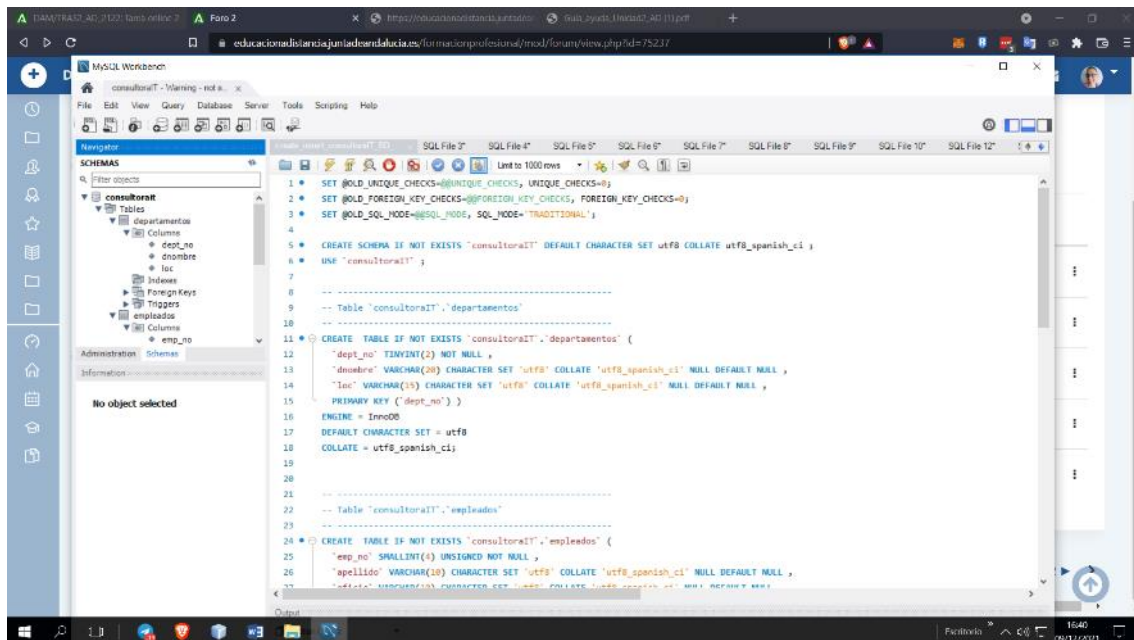
Unidad 2: Manejo de Conectores

Contenido

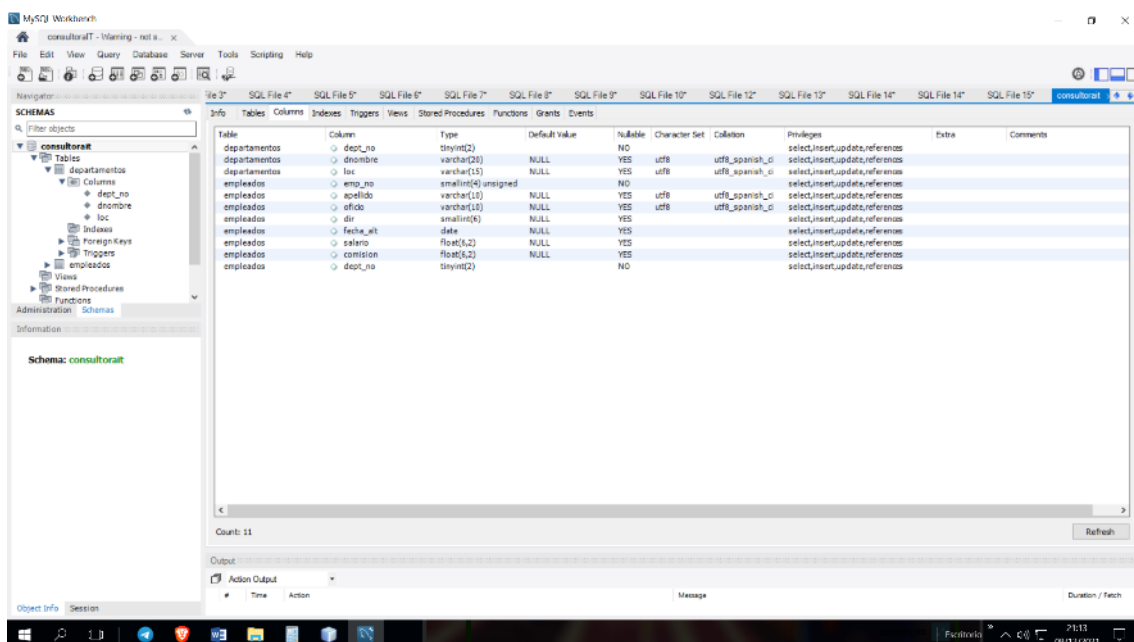
Ejercicio 1:	2
1.1 Ejecución del script SQL.	2
Ejercicio 2:	4
2.1 Consulta 1:.....	6
2.2 Consulta 2:.....	10
2.3 Consulta 3:.....	12
Ejercicio 3:	14
3.1 Procedimiento Almacenado 1:	14
3.2 Procedimiento Almacenado 2:	18
3.3 Procedimiento Almacenado 3:	20

Ejercicio 1:

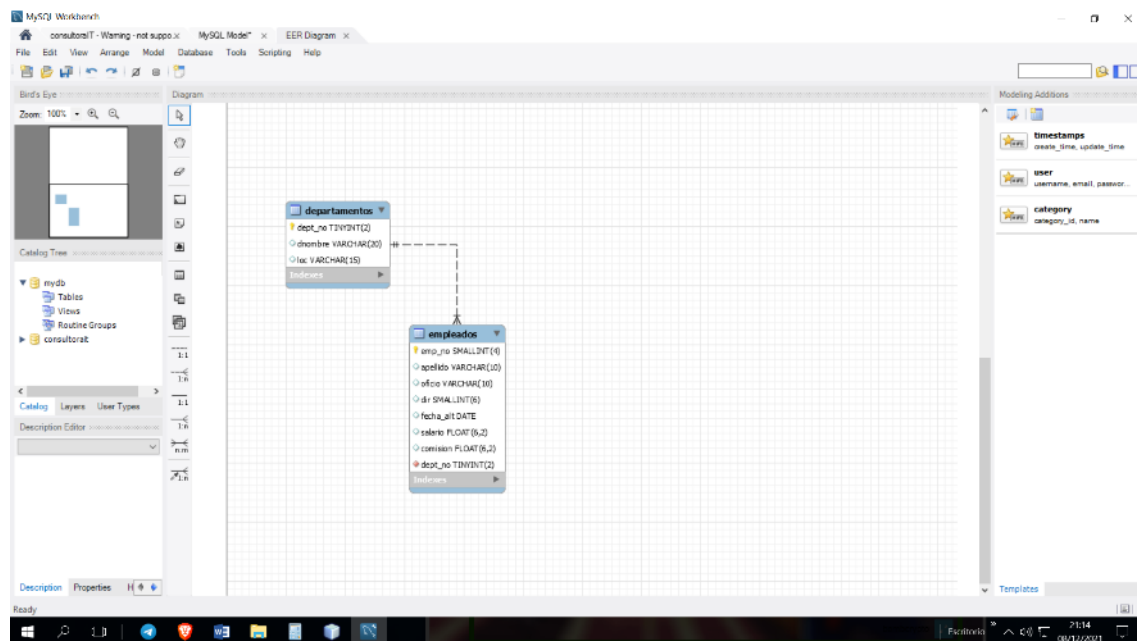
1.1 Ejecución del script SQL.



Para este ejercicio he creado una consulta en **MySQL Workbench** donde he ejecutado el script SQL facilitado en la tarea para la creación del esquema de la bdd consultoraIT.



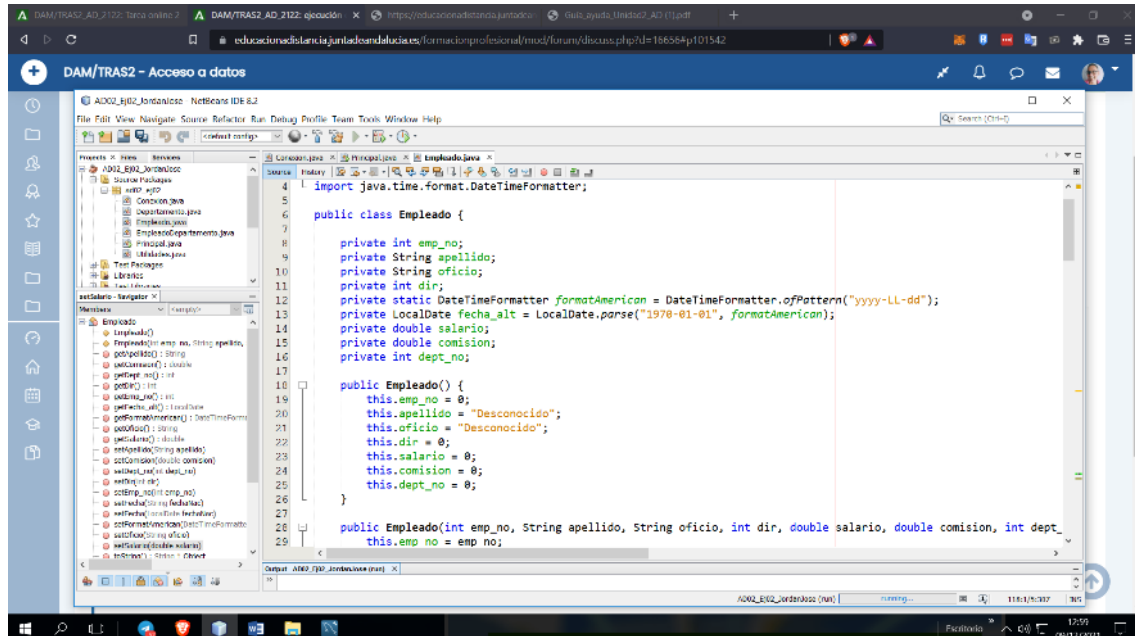
Una vez ejecutado como podemos comprobar se han creado correctamente las tablas y columnas correspondientes.



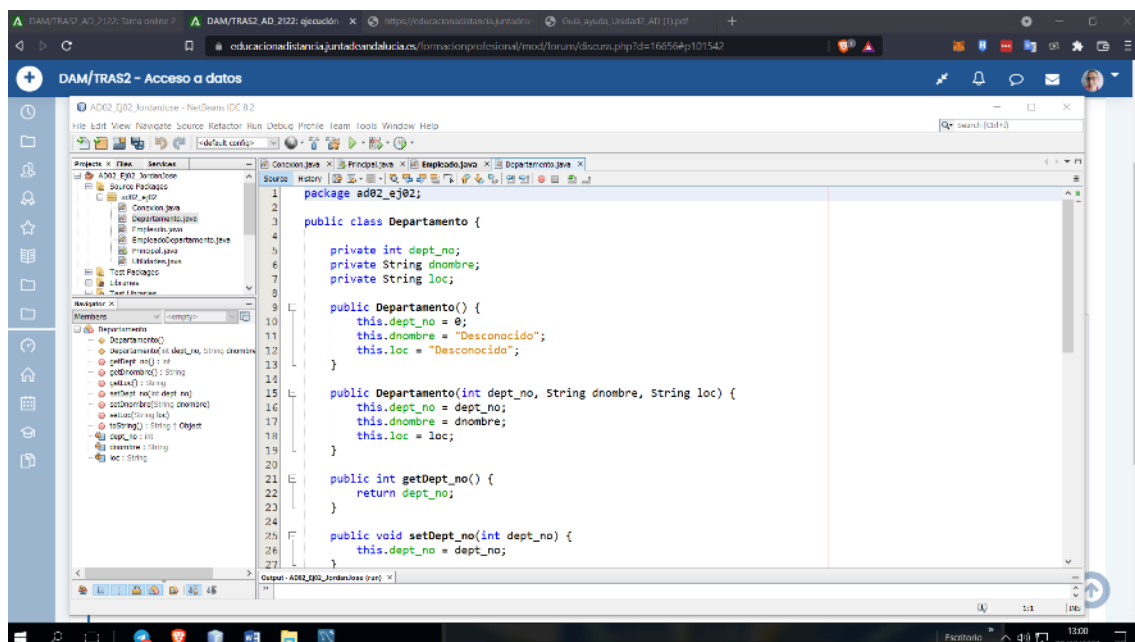
También se ha comprobado a través de la opción “**Ingeniería inversa**” de **MySQL Workbench** que el diagrama EER corresponde con el suministrado en la captura de pantalla del ejercicio.

Ejercicio 2:

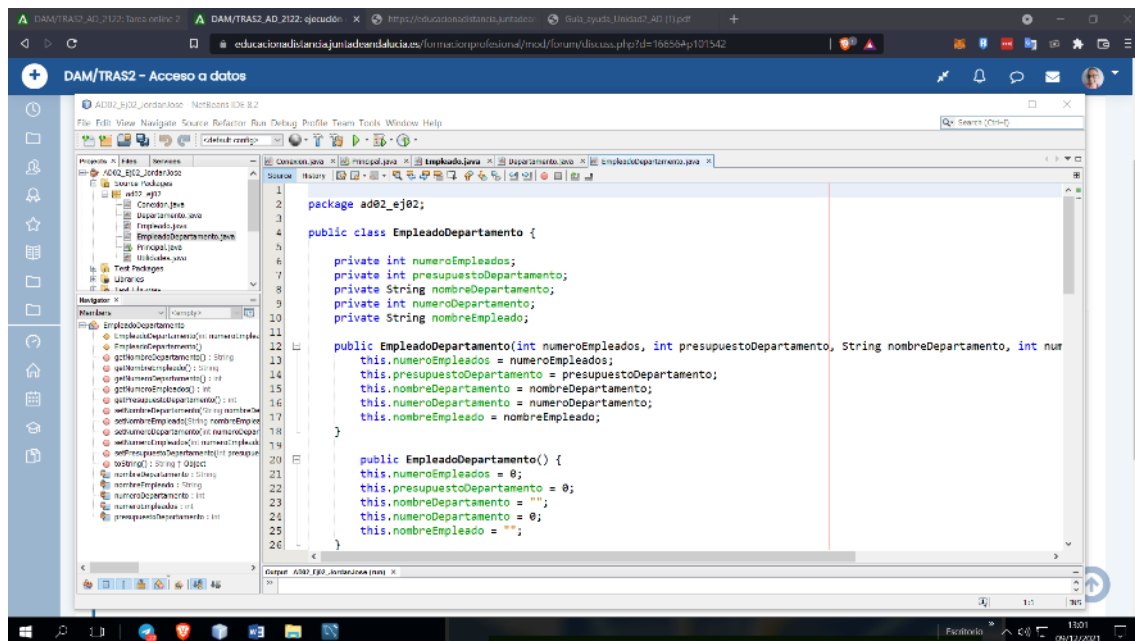
Para este proyecto he creado varias clases. Conexión, Departamento, Empleado, EmpleadoDepartamento, Utilidades y Principal.



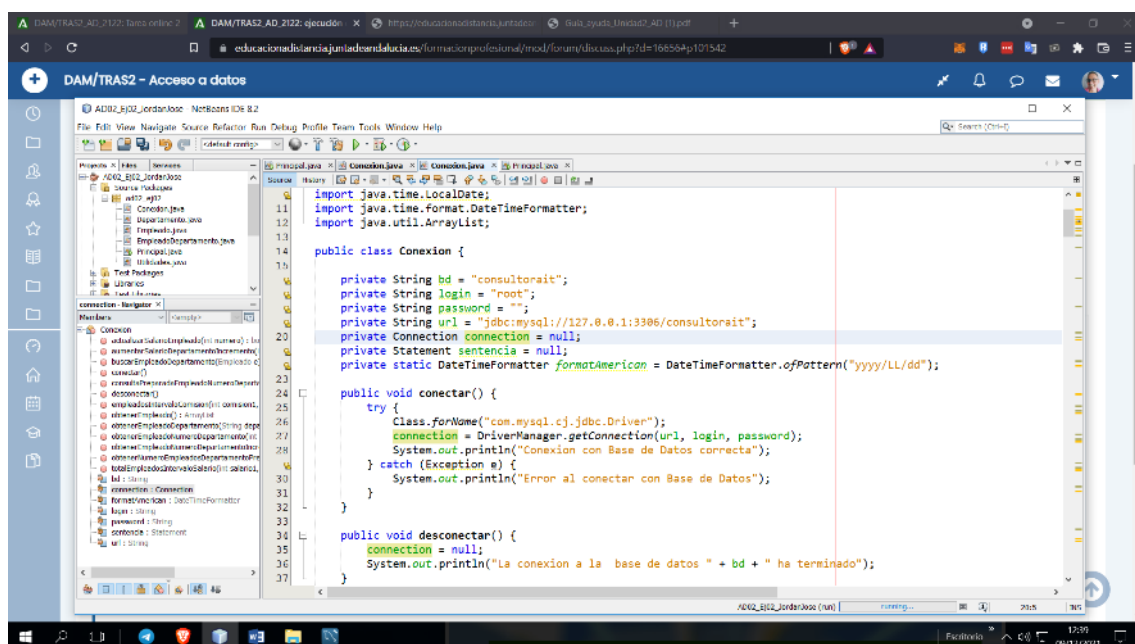
La clase Empleado con la que se gestionará la información que obtendremos de la tabla empleado.



Con la clase departamento gestionaremos la información obtenida de la tabla departamento.



Con la clase **EmpleadoDepartamento** podremos gestionar consultas relacionadas con ambas tablas.

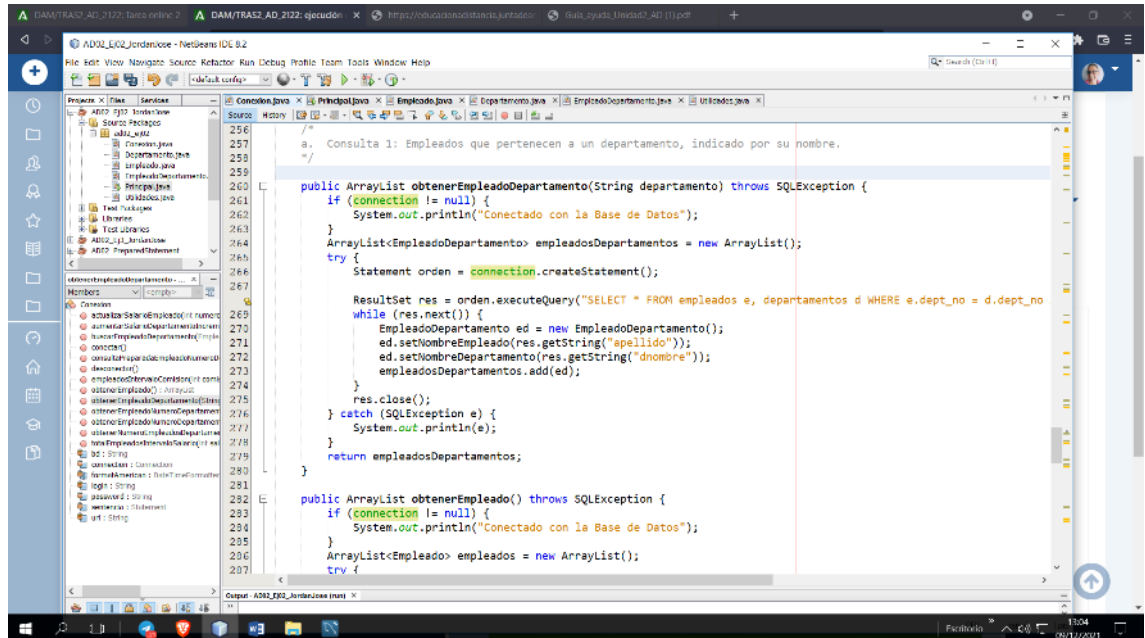


La clase **Conexion** se va a gestionar la conexión con la bbdd y en sus atributos contiene la información de la bbdd. Nombre de la bbdd, usuario, contraseña, url, connection y statement.

En los métodos **conectar()** y **desconectar()** se basarán las funciones de conectar y desconectar con la bbdd. Ambos métodos informarán con mensajes de su conexión/desconexión a la bbdd.

2.1 Consulta 1:

- a. **Consulta 1:** Empleados que pertenecen a un departamento, indicado por su nombre.



Para esta consulta primeramente en la clase Conexion he creado el método **obtenerEmpleadoDepartamento()** , al cual se le pasa por parámetro el nombre de un departamento.

Comprobaremos si el connection no es nulo y se informa que se ha conectado con la bbdd correctamente.

Se crea un arraylist de Empleado Departamento para guardar los resultados. En un try & catch se crea:

Un objeto Statement orden y un objeto ResultSet res al cual le pasaremos orden.executeQuery con la consulta que queremos realizar a la bbdd. En este caso le pasamos el parámetro departamento:

Consulta realizada:

```
"SELECT * FROM empleados e, departamentos d WHERE e.dept_no =  
d.dept_no AND d.dnombre=" + "'" + departamento + "'"
```

```
while (res.next()) {  
    EmpleadoDepartamento ed = new EmpleadoDepartamento();  
    ed.setNombreEmpleado(res.getString("apellido"));  
    ed.setNombreDepartamento(res.getString("dnombre"));  
    empleadosDepartamentos.add(ed);  
}  
res.close();  
} catch (SQLException e) {  
    System.out.println(e);  
}  
return empleadosDepartamentos;
```

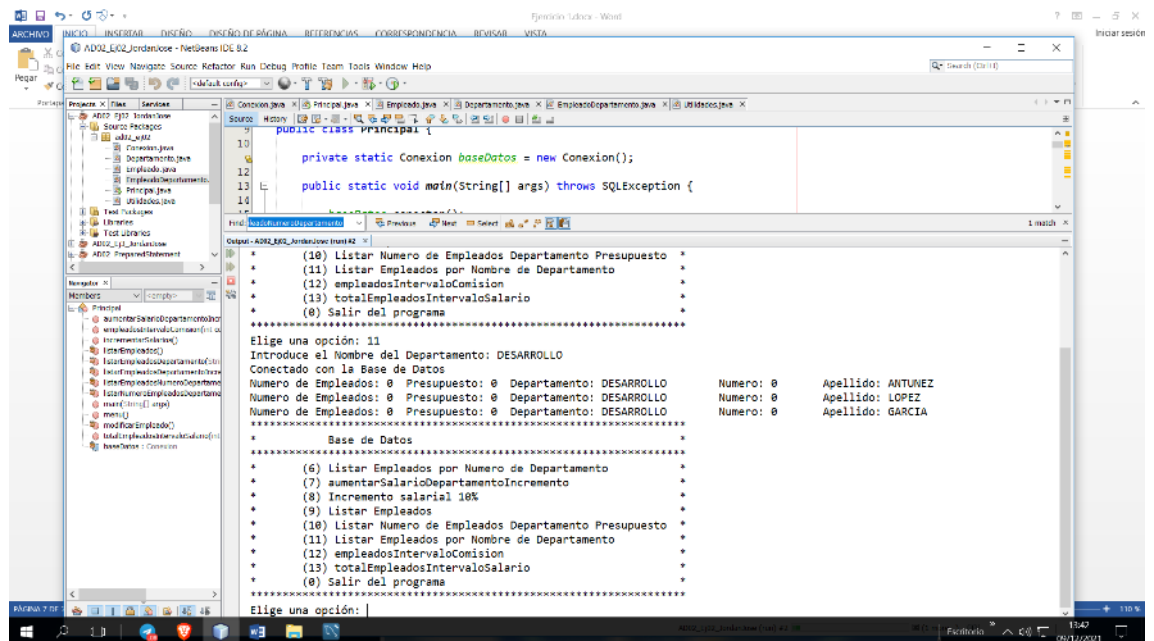
Luego con un bucle while recorreremos el Resultset y cuando finalizamos añadimos los datos del objeto de la clase EmpleadoDepartamento al ArrayList que hemos creado.

Cerramos resultset y retornamos el arraylist de empleadosDepartamento.

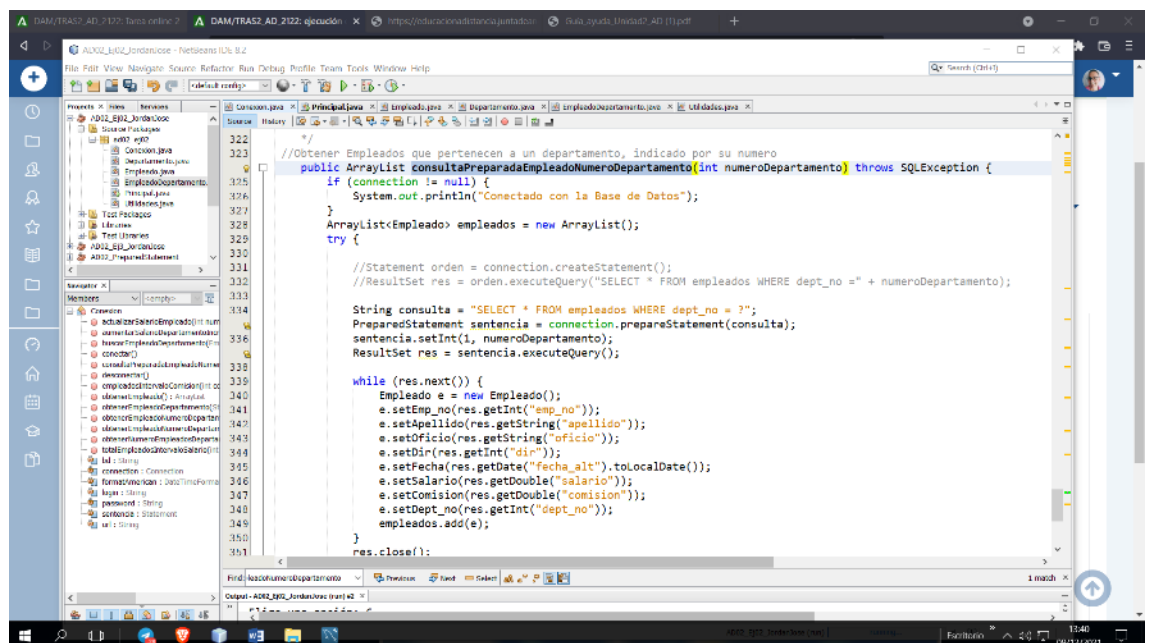
```
private static void listarEmpleadosDepartamento(String departamento) throws SQLException {  
    ArrayList<EmpleadoDepartamento> empleadosDepartamentos = new ArrayList();  
    empleadosDepartamentos = baseDatos.obtenerEmpleadoDepartamento(departamento);  
    for (EmpleadoDepartamento e : empleadosDepartamentos) {  
        System.out.println(e);  
    }  
}
```

En la clase principal se ha creado un método estático llamado **listarEmpleadosDepartamento()**. Al cual le pasamos como parámetro un departamento.

Creamos un ArrayList EmpleadoDepartamento al cual le pasaremos el ArrayList que obtenemos de la bbdd con el método **obtenerEmpleadosDepartamento()**.



Posteriormente imprimiremos el arraylist con los empleados de un determinado departamento en pantalla. En esta consulta imprime el departamento y el apellido del empleado. No imprime el número de empleados ni el presupuesto ya que esto está preparado para otra consulta que se hará más adelante .

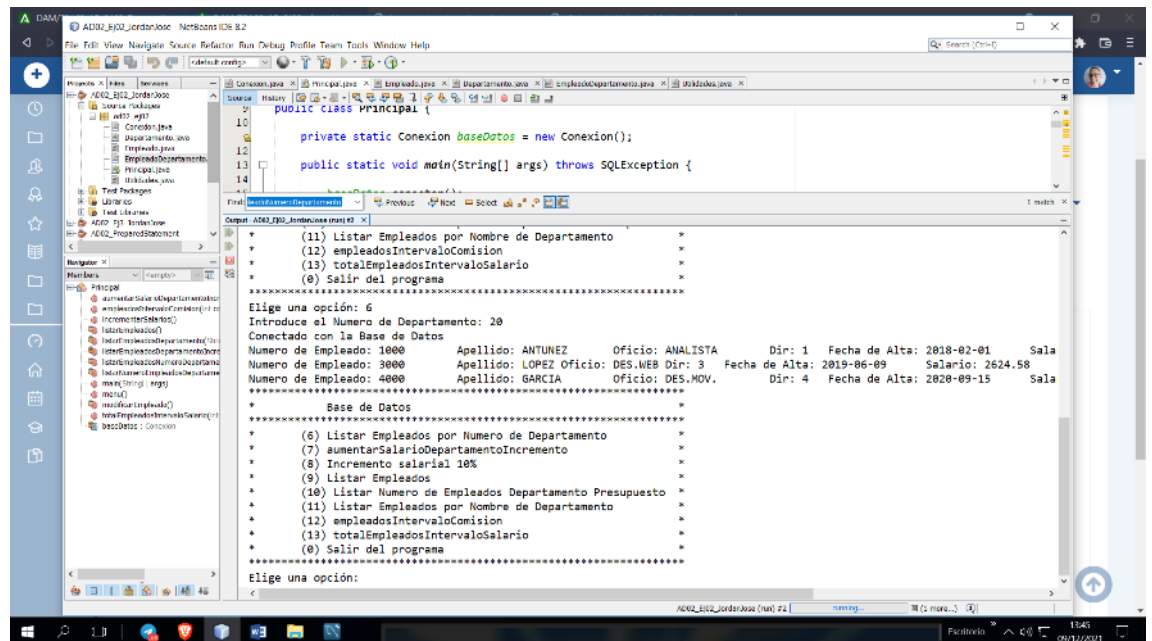


Adicionalmente he creado otro método con **PreparedStatement** al cual se le pasa un número de departamento por parámetro.


```
// listar empleados por numero de departamento con prepared statement

private static void ListarEmpleadosNumeroDepartamento(int numeroDepartamento) throws SQLException {
    ArrayList<Empleado> empleados = new ArrayList();
    empleados = baseDatos.consultaPreparadaEmpleadoNumeroDepartamento(numeroDepartamento);
    for (Empleado e : empleados) {
        System.out.println(e);
    }
}
```

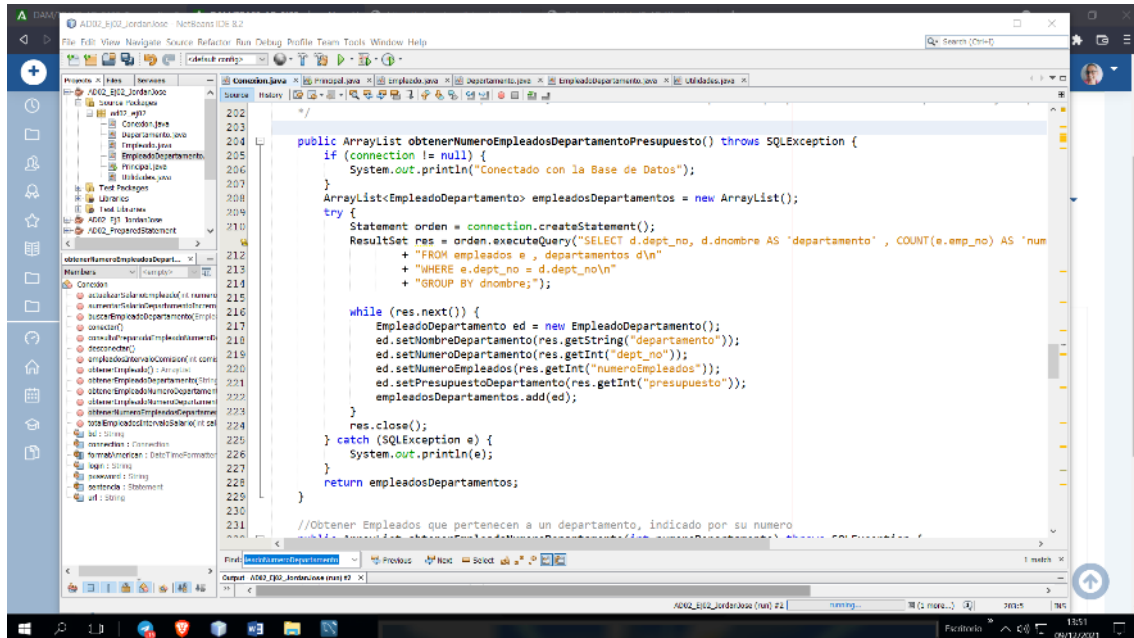
Para el cual también utilizo un método static para listar el número de Empleados pasándole un número de departamento.



En el método principal me pide el número de departamento y una vez introducido imprimirá en pantalla.

2.2 Consulta 2:

- b. **Consulta 2:** Para cada departamento, obtener el número de empleados que pertenecen a dicho departamento y el presupuesto (cantidad de dinero) destinado a ese departamento.



Para esta consulta he creado el método **obtenerNumeroEmpleadoDepartamentoPresupuesto()**:

Al igual que los métodos anteriores creo objetos `Statement` y `ResultSet`.

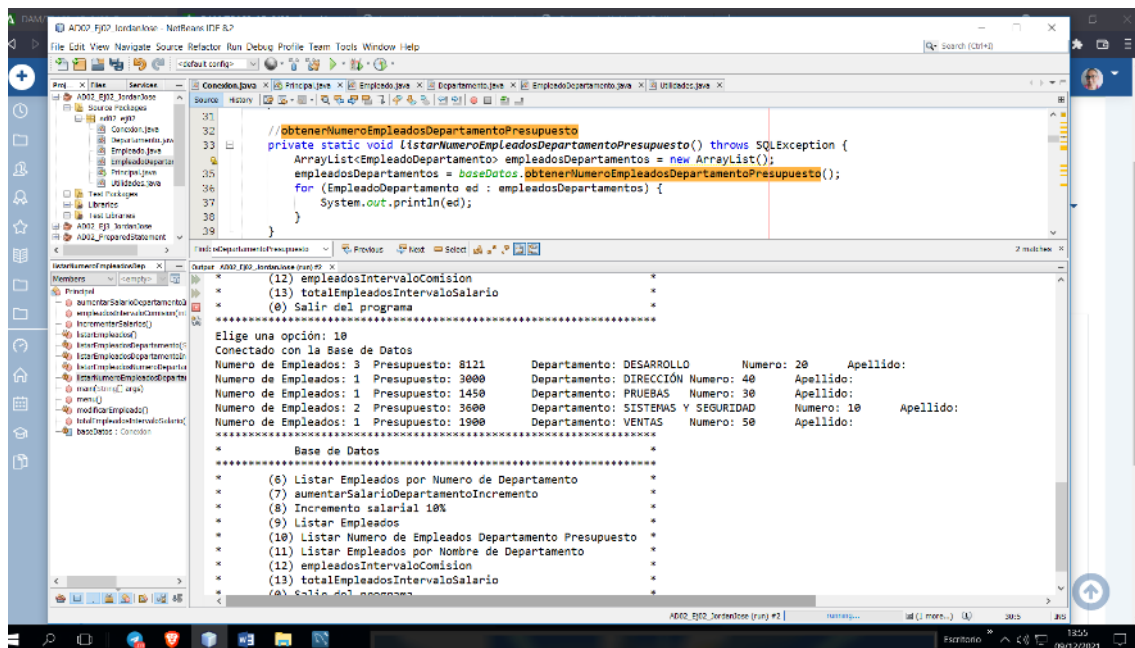
La consulta a ejecutar es la siguiente:

```
"SELECT d.dept_no, d.dnombre AS 'departamento' , COUNT(e.emp_no) AS  
'numeroEmpleados' , SUM(s.salario) AS 'presupuesto'\n"
```

```
+ "FROM empleados e , departamentos d\n"
```

```
+ "WHERE e.dept_no = d.dept_no\n"
```

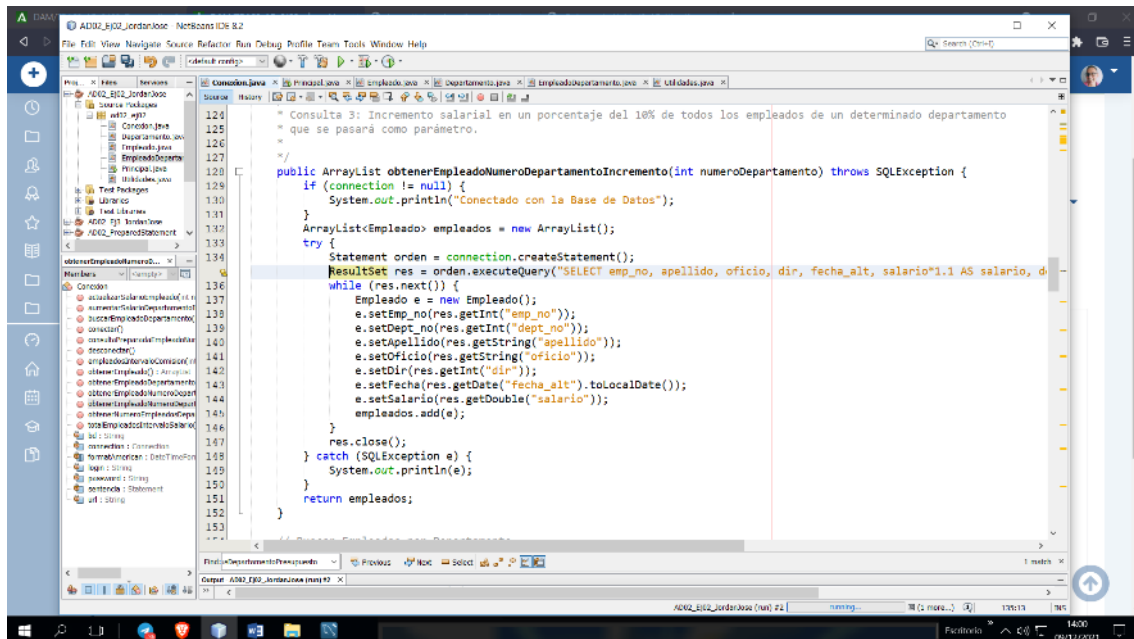
```
+ "GROUP BY dnombre;"
```



En el main tenemos un método static **listarNumeroEmpleadosDepartamentoPresupuesto()**. Que lista el número de empleados y el presupuesto.

2.3 Consulta 3:

- c. **Consulta 3:** Incremento salarial en un porcentaje del 10% de todos los empleados de un determinado departamento que se pasará como parámetro.



Para ésta consulta he creado un método en la clase Conexión.

obtenerEmpleadoNumeroDepartamentoIncremento() de tipo `ArrayList` al cual se le pasa un parámetro de tipo `int` que será el número de departamento. Al igual que los otros métodos se crean objetos `Statement` y `ResultSet` con los cuales se ejecutará la consulta siguiente:

"SELECT emp_no, apellido, oficio, dir, fecha_alt, salario*1.1 AS salario, dept_no FROM empleados WHERE dept_no =" + numeroDepartamento"

De esta forma se incrementa el salario en un 10% de los empleados cuyo número de departamento sea el que le pasamos como parámetro.

Los resultados que obtenemos se añaden al `ArrayList` que será devuelto.

```
private static void listarEmpleadosDepartamentoIncremento(int numDepartamento) throws SQLException {
    ArrayList<Empleado> empleados = new ArrayList();
    empleados = baseDatos.obtenerEmpleadoNumeroDepartamentoIncremento(numDepartamento);
    for (Empleado e : empleados) {
        System.out.println(e);
    }
}
```

En la clase principal tenemos un método estático **listarEmpleadosDepartamentoIncremento()** al cual le pasamos un parámetro de tipo `int` que será el número de departamento.

Utilizaremos el arraylist para almacenar e imprimir el arraylist que nos devuelve el método **obtenerEmpleadoNumeroDepartamentoIncremento()**.

```

43 empleados = baseDatos.obtenerEmpleado();
44 for (Empleado e : empleados) {
45     System.out.println(e);
46 }
47
48
49 private static void listarEmpleadosDepartamento(String departamento) throws SQLException {
50     ArrayList<EmpleadoDepartamento> empleadosDepartamentos = new ArrayList();
51     empleadosDepartamentos = baseDatos.obtenerEmpleadoDepartamento(departamento);
52     for (EmpleadoDepartamento e : empleadosDepartamentos) {
53         System.out.println(e);
54     }
55 }

```

Output:

```

o: ANTUNEZ Oficio: ANALISTA Dir: 1 Fecha de Alta: 2018-02-01 Salario: 2837.16 Comision: 10.0 Departamento: 20
o: CASINELLO Oficio: ADMIN.SIS. Dir: 2 Fecha de Alta: 2019-05-10 Salario: 1750.0 Comision: 0.0 Departamento: 10
o: LOPEZ Oficio: DES.WEB Dir: 3 Fecha de Alta: 2019-06-09 Salario: 2624.58 Comision: 10.0 Departamento: 20
o: GARCIA Oficio: DES.NOV. Dir: 4 Fecha de Alta: 2020-09-15 Salario: 2660.01 Comision: 10.0 Departamento: 20
o: FERNANDEZ Oficio: TESTER Dir: 5 Fecha de Alta: 2020-08-31 Salario: 1450.0 Comision: 5.0 Departamento: 30
o: DE FRUTOS Oficio: GERENTE Dir: 6 Fecha de Alta: 2017-12-10 Salario: 3000.0 Comision: 20.0 Departamento: 40
o: GONZALEZ Oficio: COMERCIAL Dir: 7 Fecha de Alta: 2018-03-01 Salario: 1900.0 Comision: 10.0 Departamento: 50
o: BELZUNCES Oficio: ADMIN.SEG. Dir: 8 Fecha de Alta: 2018-01-03 Salario: 1850.0 Comision: 0.0 Departamento: 10

```

Si listamos los empleados vemos sus salarios. Por ejemplo ANTUNEZ su salario es de 2837.16 Ahora vamos a incrementar el salario del departamento 20.

```

43 empleados = baseDatos.obtenerEmpleado();
44 for (Empleado e : empleados) {
45     System.out.println(e);
46 }
47
48
49 private static void listarEmpleadosDepartamento(String departamento) throws SQLException {
50     ArrayList<EmpleadoDepartamento> empleadosDepartamentos = new ArrayList();
51     empleadosDepartamentos = baseDatos.obtenerEmpleadoDepartamento(departamento);
52     for (EmpleadoDepartamento e : empleadosDepartamentos) {
53         System.out.println(e);
54     }
55 }

```

Output:

```

acción: 8
l Numero de Departamento: 20
n la Base de Datos
Empleado: 1000 Apellido: ANTUNEZ Oficio: ANALISTA Dir: 1 Fecha de Alta: 2018-02-01 Salario: 3120.88 Comis
Empleado: 3000 Apellido: LOPEZ Oficio: DES.WEB Dir: 3 Fecha de Alta: 2019-06-09 Salario: 2887.04 Comision: 0.0 Depar
Empleado: 4000 Apellido: GARCIA Oficio: DES.NOV. Dir: 4 Fecha de Alta: 2020-09-15 Salario: 2926.01 Comis
Base de Datos
Listar Empleados por Numero de Departamento
aumentarSalarioDepartamentoIncremento
Incremento salarial 10%
Listar Empleados
Listar Numero de Empleados Departamento Presupuesto
Listar Empleados por Nombre de Departamento
empleadosIntervaloComision
totalEmpleadosIntervaloSalario

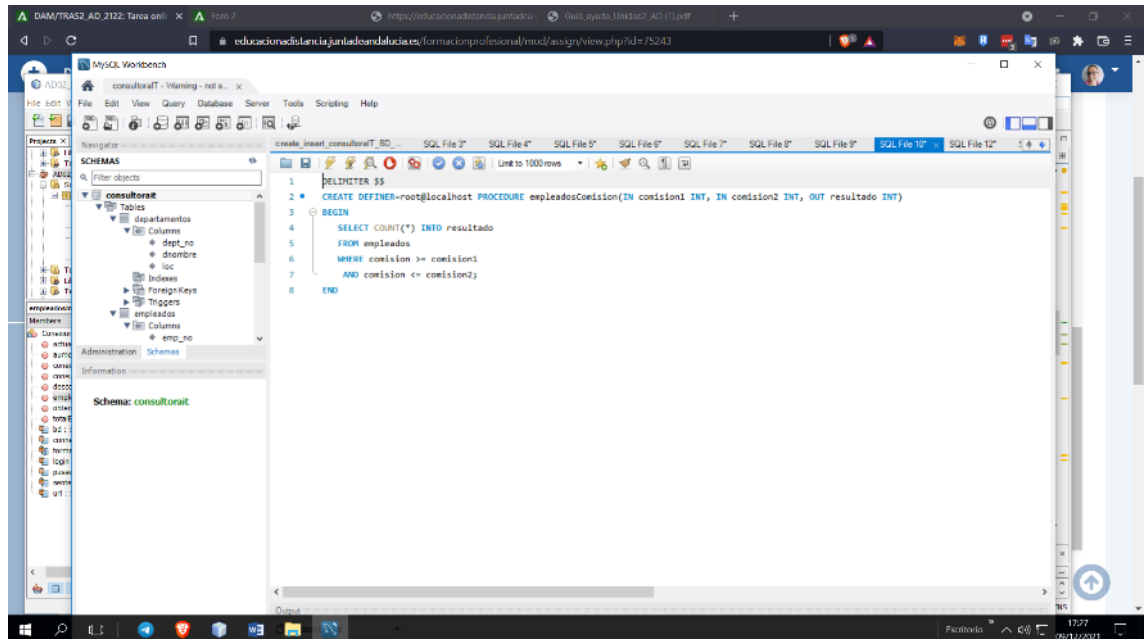
```

Podemos observar que el salario de ANTUNEZ ahora es de 3120.88.

Ejercicio 3:

3.1 Procedimiento Almacenado 1:

Procedimiento almacenado que obtenga en un parámetro de salida, el número total de empleados cuya comisión pertenece a un determinado intervalo. Se deben pasar como parámetros al procedimiento, tanto el valor inicial del intervalo como el valor final.



Crearemos previamente el procedimiento almacenado en MySQL Workbench , ejecutando lo siguiente:

```
DELIMITER $$
```

```
CREATE DEFINER=root@localhost PROCEDURE empleadosComision(IN comision1  
INT, IN comision2 INT, OUT resultado INT)
```

```
BEGIN
```

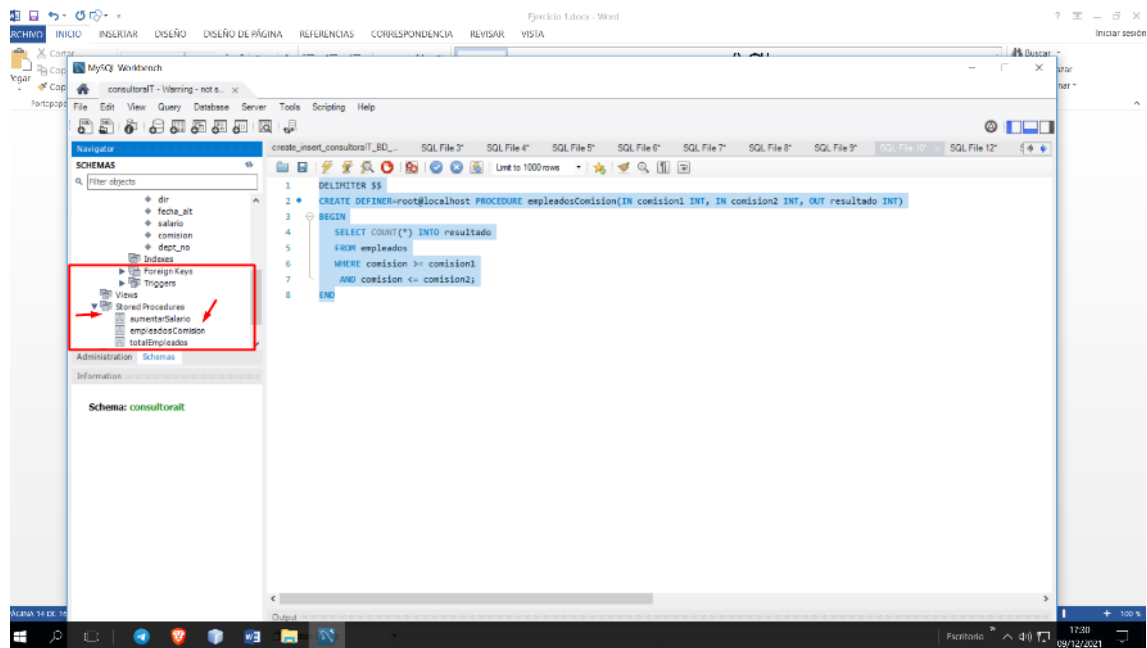
```
    SELECT COUNT(*) INTO resultado
```

```
    FROM empleados
```

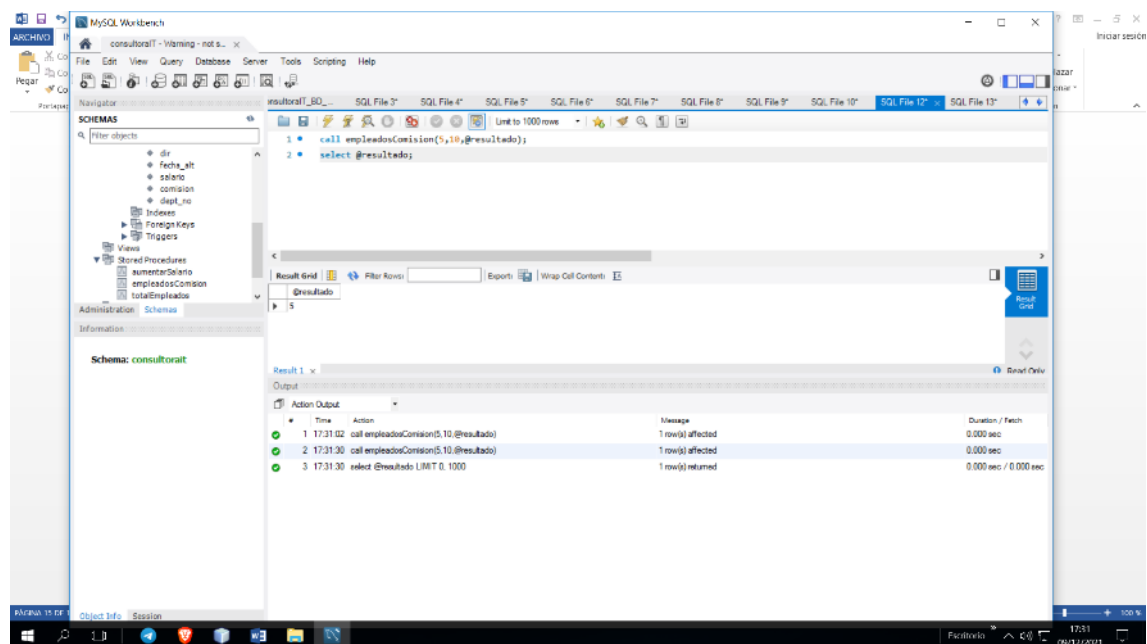
```
    WHERE comision >= comision1
```

```
    AND comision <= comision2;
```

```
END
```



Comprobamos en “Stored Procedures” que tenemos los PA .

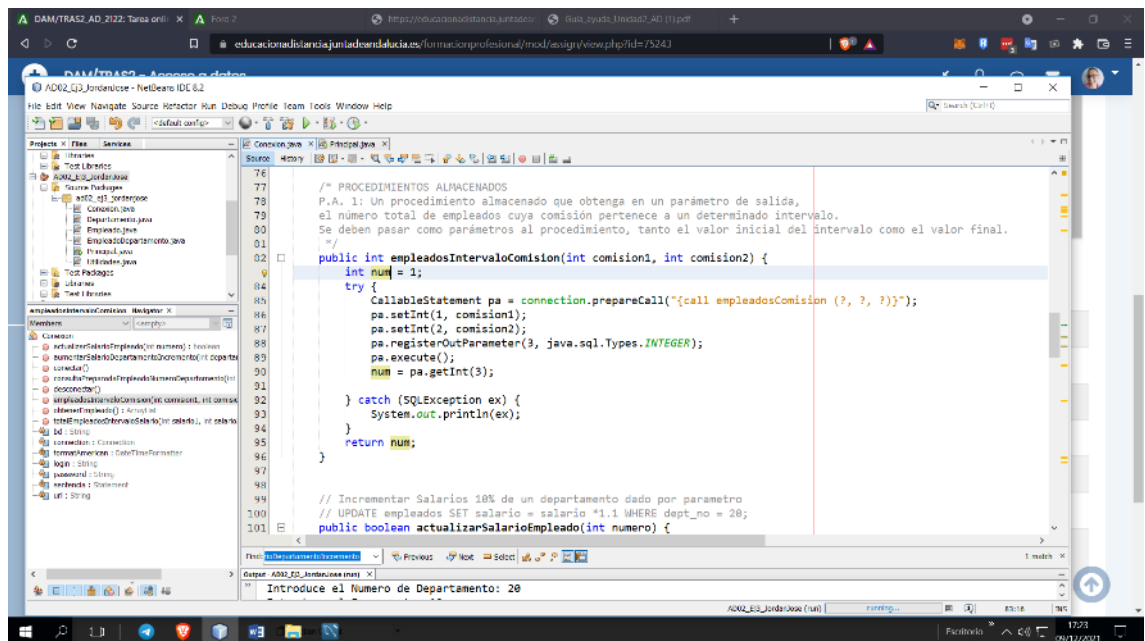


Ejecutamos y comprobamos su funcionamiento:

call empleadosComision(5,10,@resultado);

select @resultado;

El resultado es 5.



De nuevo en Netbeans y con la estructura del ejercicio anterior creamos un nuevo proyecto donde crearemos las clases para manejar los datos.

En la clase Conexión tenemos el método **empleadosIntervaloComision()**.

Creamos un objeto **CallableStatement** que será donde guardaremos nuestro PA.

```

/**
 *
 */
public int empleadosIntervaloComision(int comision1, int comision2) {
    int num = 1;
    try {
        CallableStatement pa = connection.prepareCall("{call empleadosComision (?, ?, ?)}");
        pa.setInt(1, comision1);
        pa.setInt(2, comision2);
        pa.registerOutParameter(3, java.sql.Types.INTEGER);
        pa.execute();
        num = pa.getInt(3);

    } catch (SQLException ex) {
        System.out.println(ex);
    }
    return num;
}

```

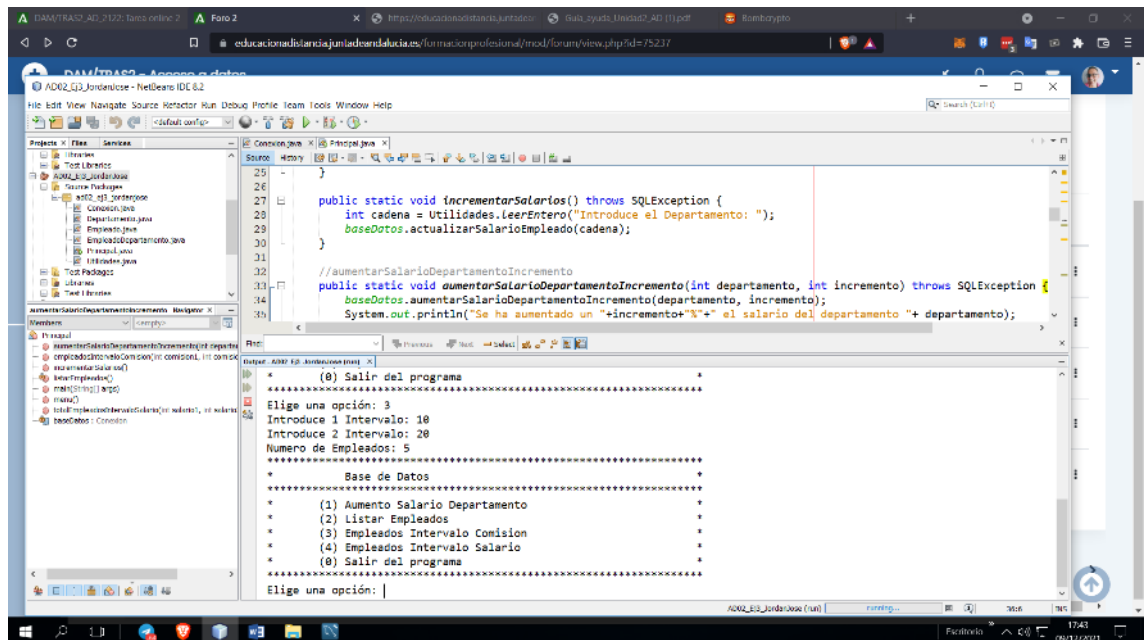
Podemos observar que al objeto **CallableStatement** se le pasa **conconnection.prepareCall** con la llamada al PA almacenado en la bbdd. En este caso se le pasan dos parámetros **comisión1** y **comisión2** cuando se ejecuta nos tiene que devolver el resultado que almacenaremos en una variable para mostrarla posteriormente.

```

public static void empleadosIntervaloComision(int comision1, int comision2) {
    int numero = baseDatos.empleadosIntervaloComision(comision1, comision2);
    System.out.println("Numero de Empleados: "+numero);
}

```

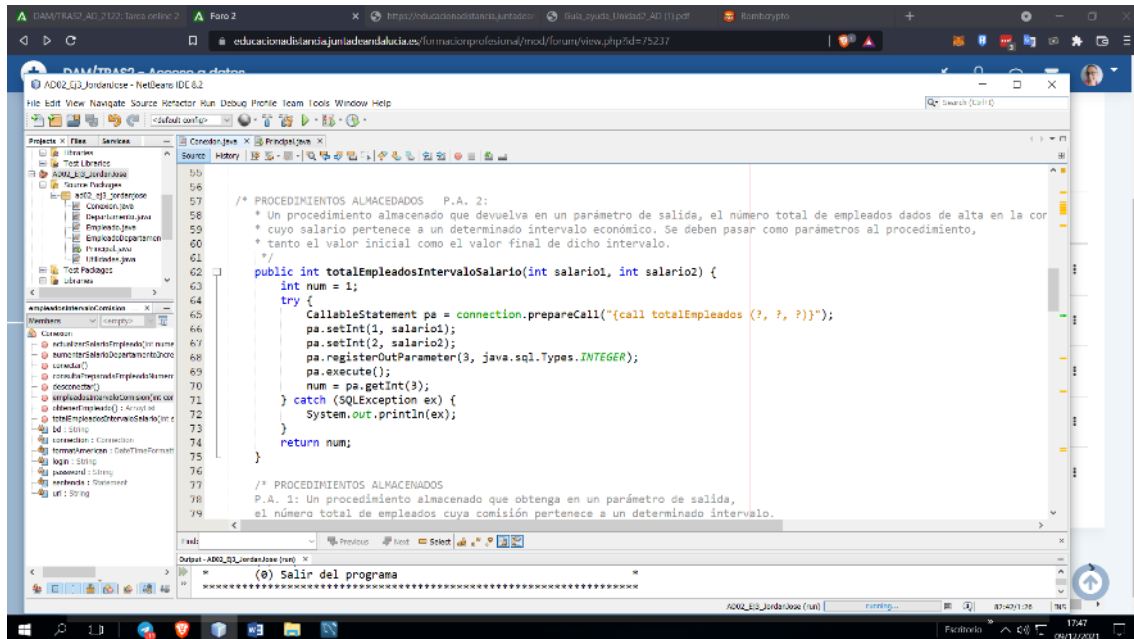
En la clase principal tenemos el método estático **empleadosIntervaloComision** que se encargará de mostrarnos el número de empleados tal y como hemos solicitado.



Intervalos de comisión de 10 y 20 Resultado 5.

3.2 Procedimiento Almacenado 2:

Procedimiento almacenado que devuelva en un parámetro de salida, el número total de empleados dados de alta en la consultora, cuyo salario pertenece a un determinado intervalo económico. Se deben pasar como parámetros al procedimiento, tanto el valor inicial como el valor final de dicho intervalo.

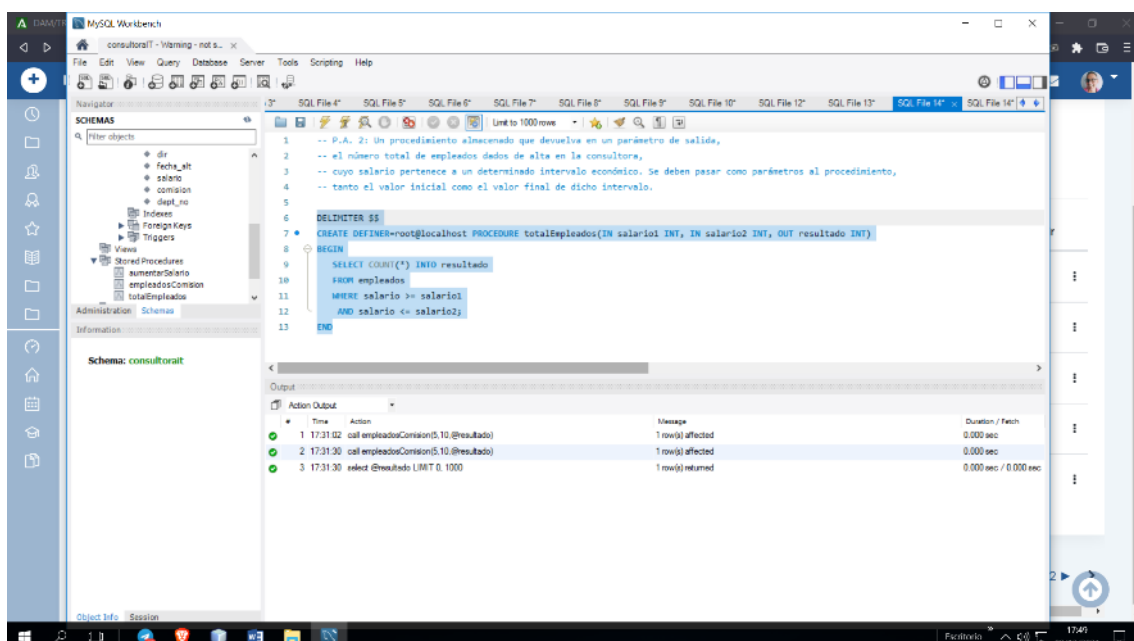


```
/* PROCEDIMIENTOS ALMACENADOS P.A. 2:
 * Un procedimiento almacenado que devuelva en un parámetro de salida, el número total de empleados dados de alta en la
 * consultora, cuyo salario pertenece a un determinado intervalo económico. Se deben pasar como parámetros al procedimiento,
 * tanto el valor inicial como el valor final de dicho intervalo.
 */
public int totalEmpleadosIntervaloSalario(int salario1, int salario2) {
    int num = 1;
    try {
        CallableStatement pa = connection.prepareCall("{call totalEmpleados(?, ?, ?)}");
        pa.setInt(1, salario1);
        pa.setInt(2, salario2);
        pa.registerOutParameter(3, java.sql.Types.INTEGER);
        pa.execute();
        num = pa.getInt(3);
    } catch (SQLException ex) {
        System.out.println(ex);
    }
    return num;
}

/* PROCEDIMIENTOS ALMACENADOS
 * P.A. 1: Un procedimiento almacenado que obtenga en un parámetro de salida,
 * el número total de empleados cuya comisión pertenece a un determinado intervalo.
 */
```

Se crea el método **totalEmpleadosComision**, al cual se le pasan como parámetros salario1 y salario2.

Al igual que el método anterior cuando se ejecuta nos tiene que devolver el resultado que almacenaremos en una variable para mostrarla posteriormente.



```
DELIMITER $$
CREATE DEFINER=root@localhost PROCEDURE totalEmpleados(IN salario1 INT, IN salario2 INT, OUT resultado INT)
BEGIN
    SELECT COUNT(*) INTO resultado
    FROM empleados
    WHERE salario >= salario1
    AND salario <= salario2;
END
```

#	Time	Action	Message	Duration / Fetch
1	17:31:02	call empleadosComision(5,10,@resultado)	1 row(s) affected	0.000 sec
2	17:31:30	call empleadosComision(5,10,@resultado)	1 row(s) affected	0.000 sec
3	17:31:30	select @resultado LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

Consulta:

DELIMITER \$\$

```
CREATE DEFINER=root@localhost PROCEDURE totalEmpleados(IN salario1  
INT, IN salario2 INT, OUT resultado INT)
```

BEGIN

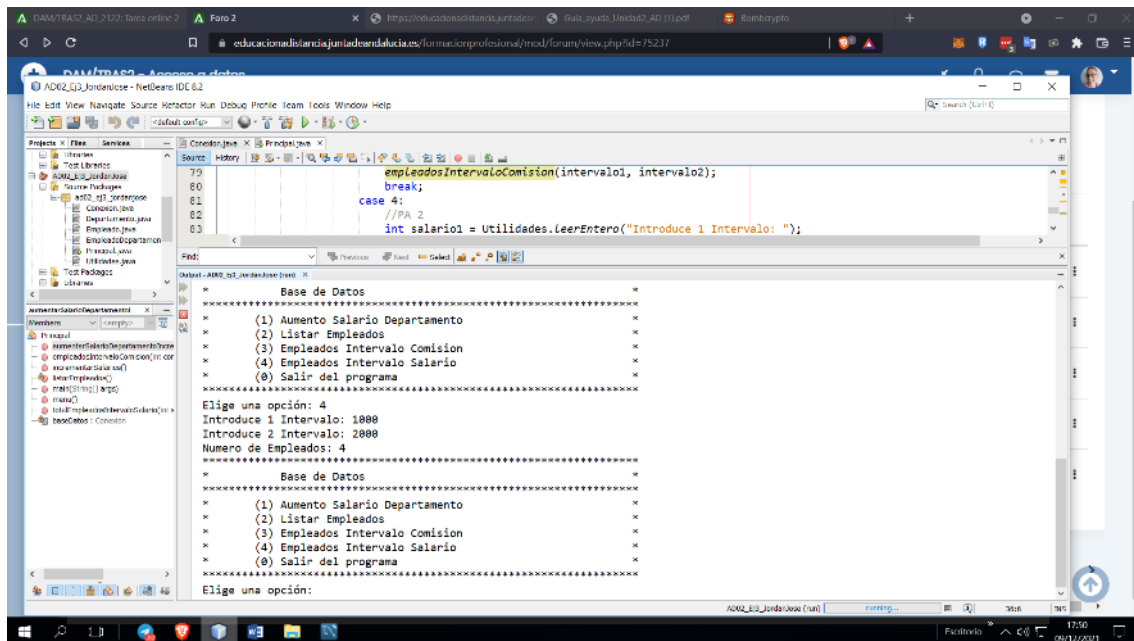
```
    SELECT COUNT(*) INTO resultado
```

```
    FROM empleados
```

```
    WHERE salario >= salario1
```

```
    AND salario <= salario2;
```

END

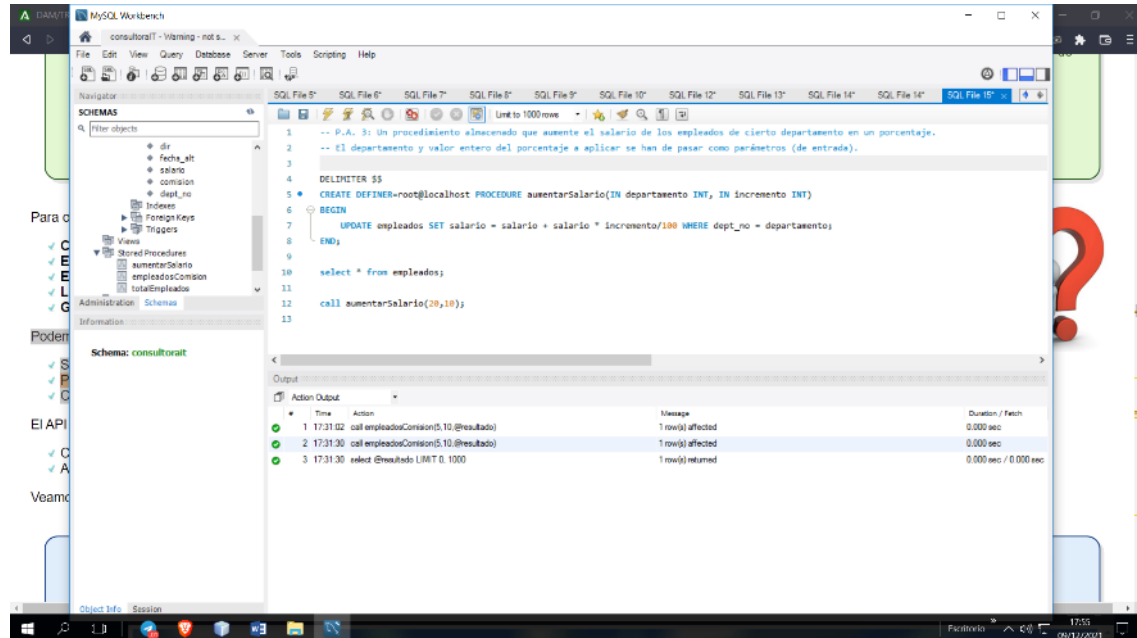


Programa en ejecución y salida mostrada.

SE pide que muestre el nuero de empleados con un salario de intervalo de 1000 / 2000 . Resultado 4 .

3.3 Procedimiento Almacenado 3:

Procedimiento almacenado que aumente el salario de los empleados de cierto departamento en un porcentaje. El departamento y valor entero del porcentaje a aplicar se han de pasar como parámetros (de entrada).



De nuevo en MySQL Workbench creamos el PA.

```
DELIMITER $$
```

```
CREATE DEFINER=root@localhost PROCEDURE aumentarSalario(IN  
departamento INT, IN incremento INT)
```

```
BEGIN
```

```
UPDATE empleados SET salario = salario + salario *  
incremento/100 WHERE dept_no = departamento;
```

```
END;
```

Probamos su funcionamiento

```
select * from empleados;
```

```
call aumentarSalario(20,10);
```

```

/* PROCEDIMIENTOS ALMACENADOS P.A. 3:
 * P.A. 3: Un procedimiento almacenado que aumente el salario de los empleados de cierto departamento en un porcentaje.
 * departamento y valor entero del porcentaje a aplicar se han de pasar como parámetros (de entrada).
 */

public void aumentarSalarioDepartamentoIncremento(int departamento, int incremento) {
    try {
        CallableStatement pa = connection.prepareCall("{call aumentarSalario (?, ?)}");
        pa.setInt(1, departamento);
        pa.setInt(2, incremento);
        pa.execute();
    } catch (SQLException ex) {
        System.out.println(ex);
    }
}

```

Creamos nuevamente un método **aumentarSalarioDepartamentoIncremento**

Al cual le pasaremos como parámetros el numero de departamento y el incremento. La estructura del método igual que la de los anteriores creamos un objeto callablestatement al cual le pasaremos los datos que recibimos por parametro . En este caso no muestra ninguna salida

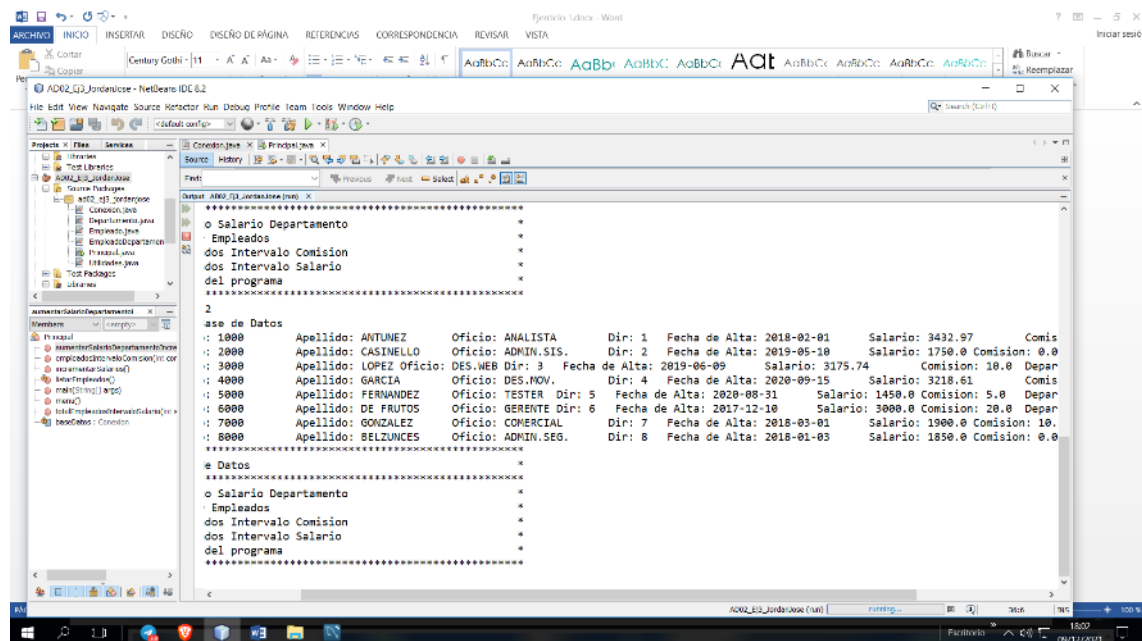
```

//aumentarSalarioDepartamentoIncremento
public static void aumentarSalarioDepartamentoIncremento(int departamento, int incremento) throws SQLException {
    baseDatos.aumentarSalarioDepartamentoIncremento(departamento, incremento);
    System.out.println("Se ha aumentado un "+incremento+" el salario del departamento "+ departamento);
}

public static void empleadosTotalIntervaloComision(int comision, int comision2) {

```

Creamos un método en la clase principal que nos ejecutará la consulta anterior.



Ejecutamos y vemos el salario de todos los empleados. Ahora ejecutaremos un aumento del 10% para el departamento 20.

```

Output: AD02_EJ3_JordanLopez - NetBeans IDE 8.2
=====
(1) Aumento Salario Departamento *
(2) Listar Empleados *
(3) Empleados Intervalo Comision *
(4) Empleados Intervalo Salario *
(0) Salir del programa *
=====
: una opción: 2
tado con la Base de Datos
o de Empleado: 1000 Apellido: ANTUNEZ Oficio: ANALISTA Dir: 1 Fecha de Alta: 2018-02-01 Salario: 3776.27
o de Empleado: 2000 Apellido: CASINELLO Oficio: ADMIN.SIS. Dir: 2 Fecha de Alta: 2019-05-10 Salario: 1750.0
o de Empleado: 3000 Apellido: LOPEZ Oficio: DES.WEB Dir: 3 Fecha de Alta: 2019-06-09 Salario: 3493.31 Comision
o de Empleado: 4000 Apellido: GARCIA Oficio: DES.MOV. Dir: 4 Fecha de Alta: 2020-09-15 Salario: 3540.47
o de Empleado: 5000 Apellido: FERNANDEZ Oficio: TESTER Dir: 5 Fecha de Alta: 2020-08-31 Salario: 1450.0 Comision
o de Empleado: 6000 Apellido: DE FRUTOS Oficio: GERENTE Dir: 6 Fecha de Alta: 2017-12-10 Salario: 3000.0 Comision
o de Empleado: 7000 Apellido: GONZALEZ Oficio: COMERCIAL Dir: 7 Fecha de Alta: 2018-03-01 Salario: 1900.0
o de Empleado: 8000 Apellido: BELZUNCES Oficio: ADMIN.SEG. Dir: 8 Fecha de Alta: 2018-01-03 Salario: 1850.0

Base de Datos
=====
(1) Aumento Salario Departamento *
(2) Listar Empleados *
(3) Empleados Intervalo Comision *
(4) Empleados Intervalo Salario *
(0) Salir del programa *
=====
: una opción:

```

Posteriormente de la ejecución del aumento vemos que el salario del departamento 20 ha aumentado.

