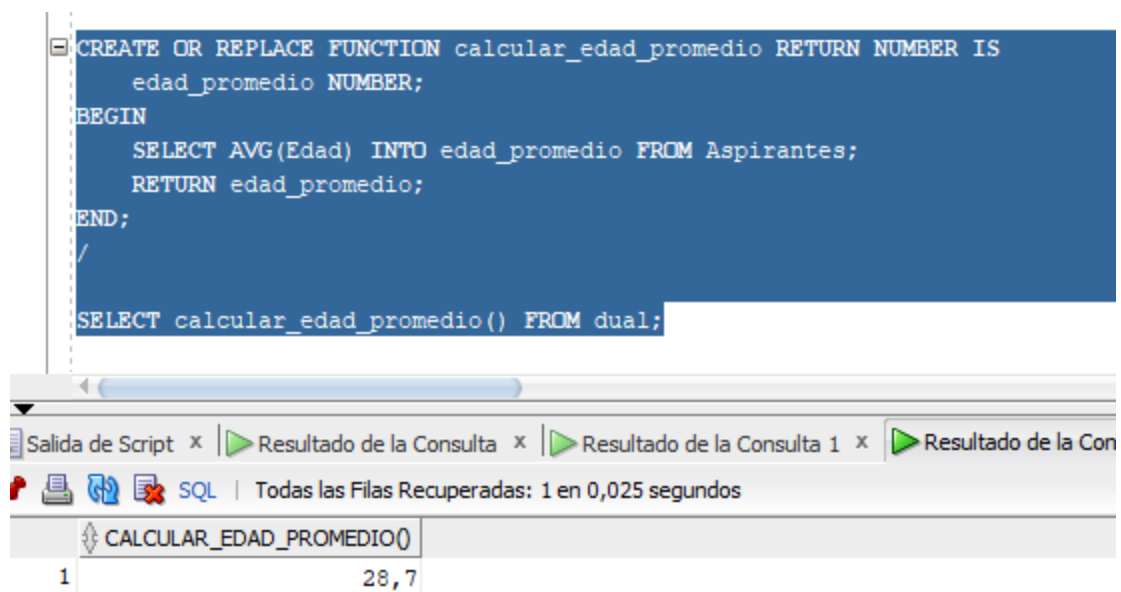


Función: Crea una función en PL/SQL que calcule y devuelva la edad promedio de los usuarios en la tabla "Usuarios"

```
CREATE OR REPLACE FUNCTION calcular_edad_promedio RETURN NUMBER IS
    edad_promedio NUMBER;
BEGIN
    SELECT AVG(Edad) INTO edad_promedio FROM Aspirantes;
    RETURN edad_promedio;
END;
/

SELECT calcular_edad_promedio() FROM dual;
```



En esta función, se utiliza la cláusula **SELECT AVG(Edad) INTO edad_promedio FROM Aspirantes** para calcular la edad promedio de los aspirantes en la tabla "Aspirantes" y almacenar el resultado en la variable **edad_promedio**. Luego, se utiliza la instrucción **RETURN edad_promedio** para devolver el valor de la edad promedio.

2. Procedimiento almacenado: Escribe un procedimiento almacenado en PL/SQL que reciba el ID de un usuario y muestre su nombre y correo electrónico.

```

CREATE OR REPLACE PROCEDURE obtener_nombre_y_correo (p_id_aspirante IN NUMBER) AS
    v_nombre VARCHAR2(50);
    v_correo VARCHAR2(100);
BEGIN
    SELECT Nombre, Email INTO v_nombre, v_correo FROM Aspirantes WHERE IDAspirante =
p_id_aspirante;

    DBMS_OUTPUT.PUT_LINE('Nombre: ' || v_nombre);
    DBMS_OUTPUT.PUT_LINE('Correo: ' || v_correo);
END;
/
BEGIN
    obtener_nombre_y_correo(3); -- Pasa el ID del aspirante que deseas consultar
END;

```

The screenshot shows a SQL IDE with a script editor and a results pane. The script editor contains the following code:

```

CREATE OR REPLACE PROCEDURE obtener_nombre_y_correo (p_id_aspirante IN NUMBER) AS
    v_nombre VARCHAR2(50);
    v_correo VARCHAR2(100);
BEGIN
    SELECT Nombre, Email INTO v_nombre, v_correo FROM Aspirantes WHERE IDAspirante = p_id_aspirante;
    DBMS_OUTPUT.PUT_LINE('Nombre: ' || v_nombre);
    DBMS_OUTPUT.PUT_LINE('Correo: ' || v_correo);
END;
/
BEGIN
    obtener_nombre_y_correo(3); -- Pasa el ID del aspirante que deseas consultar
END;

```

The results pane shows the output of the script:

```

Nombre: Carlos
Correo: carlos@example.com

Procedimiento PL/SQL terminado correctamente.

```

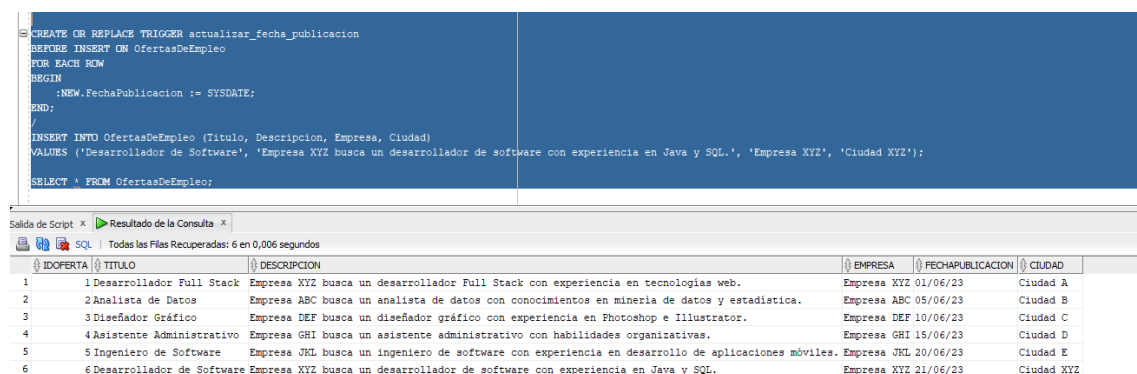
En este procedimiento almacenado, se utiliza la cláusula **SELECT Nombre, Email INTO v_nombre, v_correo FROM Aspirantes WHERE IDAspirante = p_id_aspirante** para obtener el nombre y correo electrónico del aspirante cuyo ID corresponde al parámetro **p_id_aspirante**. Luego, se utiliza **DBMS_OUTPUT.PUT_LINE** para mostrar el nombre y el correo electrónico en la salida.

3. Disparador: Crea un disparador en PL/SQL que se active cada vez que se inserte un nuevo registro en la tabla "Empleos" y actualice automáticamente la fecha de publicación a la fecha actual.

```
CREATE OR REPLACE TRIGGER actualizar_fecha_publicacion
BEFORE INSERT ON OfertasDeEmpleo
FOR EACH ROW
BEGIN
    :NEW.FechaPublicacion := SYSDATE;
END;
/

INSERT INTO OfertasDeEmpleo (Titulo, Descripcion, Empresa, Ciudad)
VALUES ('Desarrollador de Software', 'Empresa XYZ busca un desarrollador de software con
experiencia en Java y SQL.', 'Empresa XYZ', 'Ciudad XYZ');

SELECT * FROM OfertasDeEmpleo;
```



The screenshot shows a SQL IDE with a script editor and a results pane. The script editor contains the following PL/SQL code:

```
CREATE OR REPLACE TRIGGER actualizar_fecha_publicacion
BEFORE INSERT ON OfertasDeEmpleo
FOR EACH ROW
BEGIN
    :NEW.FechaPublicacion := SYSDATE;
END;
/

INSERT INTO OfertasDeEmpleo (Titulo, Descripcion, Empresa, Ciudad)
VALUES ('Desarrollador de Software', 'Empresa XYZ busca un desarrollador de software con experiencia en Java y SQL.', 'Empresa XYZ', 'Ciudad XYZ');

SELECT * FROM OfertasDeEmpleo;
```

The results pane shows the output of the SELECT statement, displaying 6 rows of data from the OfertasDeEmpleo table. The status bar indicates that 6 rows were recovered in 0.006 seconds.

IDOFERTA	TITULO	DESCRIPCION	EMPRESA	FECHAPUBLICACION	CIUDAD
1	1 Desarrollador Full Stack	Empresa XYZ busca un desarrollador Full Stack con experiencia en tecnologías web.	Empresa XYZ	01/06/23	Ciudad A
2	2 Analista de Datos	Empresa ABC busca un analista de datos con conocimientos en minería de datos y estadística.	Empresa ABC	05/06/23	Ciudad B
3	3 Diseñador Gráfico	Empresa DEF busca un diseñador gráfico con experiencia en Photoshop e Illustrator.	Empresa DEF	10/06/23	Ciudad C
4	4 Asistente Administrativo	Empresa GHI busca un asistente administrativo con habilidades organizativas.	Empresa GHI	15/06/23	Ciudad D
5	5 Ingeniero de Software	Empresa JKL busca un ingeniero de software con experiencia en desarrollo de aplicaciones móviles.	Empresa JKL	20/06/23	Ciudad E
6	6 Desarrollador de Software	Empresa XYZ busca un desarrollador de software con experiencia en Java y SQL.	Empresa XYZ	21/06/23	Ciudad XYZ

En este disparador, utilizamos la cláusula **BEFORE INSERT ON OfertasDeEmpleo** para indicar que se activará antes de que se inserte un nuevo registro en la tabla "OfertasDeEmpleo". La cláusula **FOR EACH ROW** especifica que el disparador se ejecutará una vez por cada fila afectada por la inserción.

En el bloque del disparador, asignamos la fecha actual a la columna **FechaPublicacion** del nuevo registro que se está insertando, utilizando **:NEW.FechaPublicacion := SYSDATE;**. **:NEW** hace referencia a la fila que se está insertando.

De esta manera, cada vez que se inserte un nuevo registro en la tabla "OfertasDeEmpleo", la columna **FechaPublicacion** se actualizará automáticamente con la fecha actual.

4. Procedimiento con cursor: Escribe un procedimiento almacenado en PL/SQL que utilice un cursor para mostrar todos los usuarios que tienen una edad mayor a 30 años.

```
CREATE OR REPLACE PROCEDURE mostrar_aspirantes_mayor_30AS
CURSOR c_aspirantes IS
    SELECT Nombre, Apellido, Edad
    FROM Aspirantes
    WHERE Edad > 30;
v_nombre Aspirantes.Nombre%TYPE;
v_apellido Aspirantes.Apellido%TYPE;
v_edad Aspirantes.Edad%TYPE;
BEGIN
    OPEN c_aspirantes;
    LOOP
        FETCH c_aspirantes INTO v_nombre, v_apellido, v_edad;
        EXIT WHEN c_aspirantes%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Nombre:' || v_nombre || ', Apellido:' || v_apellido || ', Edad:
' || v_edad);
    END LOOP;
    CLOSE c_aspirantes;
END;
/
BEGIN
    mostrar_aspirantes_mayor_30;
END;
/
```

```
CREATE OR REPLACE PROCEDURE mostrar_aspirantes_mayor_30 AS
CURSOR c_aspirantes IS
    SELECT Nombre, Apellido, Edad
    FROM Aspirantes
    WHERE Edad > 30;
    v_nombre Aspirantes.Nombre%TYPE;
    v_apellido Aspirantes.Apellido%TYPE;
    v_edad Aspirantes.Edad%TYPE;
BEGIN
    OPEN c_aspirantes;
    LOOP
        FETCH c_aspirantes INTO v_nombre, v_apellido, v_edad;
        EXIT WHEN c_aspirantes%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Nombre: ' || v_nombre || ', Apellido: ' || v_apellido || ', Edad: ' || v_edad);
    END LOOP;
    CLOSE c_aspirantes;
END;
```

Salida de Script x Resultado de la Consulta x

Tarea terminada en 0,027 segundos

Procedure MOSTRAR_ASPIRANTES_MAYOR_30 compilado

Procedimiento PL/SQL terminado correctamente.

Nombre: Laura, Apellido: Fernández, Edad: 35
Nombre: Luis, Apellido: García, Edad: 32
Nombre: Jorge, Apellido: Hernández, Edad: 31

Procedimiento PL/SQL terminado correctamente.

Nombre: Laura, Apellido: Fernández, Edad: 35
Nombre: Luis, Apellido: García, Edad: 32
Nombre: Jorge, Apellido: Hernández, Edad: 31

Procedimiento PL/SQL terminado correctamente.

En este procedimiento, definimos un cursor llamado **c_aspirantes** que selecciona los nombres, apellidos y edades de los aspirantes que tienen una edad mayor a 30 años.

Dentro del bloque del procedimiento, abrimos el cursor con la sentencia **OPEN c_aspirantes** y luego utilizamos un bucle **LOOP** para recorrer los resultados del cursor. En cada iteración del bucle, almacenamos los valores de nombre, apellido y edad en variables **v_nombre**, **v_apellido** y **v_edad**, respectivamente, utilizando la sentencia **FETCH c_aspirantes INTO v_nombre, v_apellido, v_edad**.

Luego, utilizamos **DBMS_OUTPUT.PUT_LINE** para mostrar en la salida estándar los datos de cada aspirante que cumple con la condición de edad mayor a 30 años.

Finalmente, cerramos el cursor con la sentencia **CLOSE c_aspirantes**.

5. Procedimiento con bucle: Define un procedimiento almacenado en PL/SQL que utilice un bucle para mostrar todos los empleos de la tabla "Empleos" junto con la cantidad de usuarios inscritos en cada uno.

```
CREATE OR REPLACE PROCEDURE mostrar_ofertas_con_inscritos AS
```

```

v_idoferta OfertasDeEmpleo.IDOferta%TYPE;
v_titulo OfertasDeEmpleo.Titulo%TYPE;
v_cantidad NUMBER;
BEGIN
  FOR oferta IN (SELECT IDOferta, Titulo FROM OfertasDeEmpleo) LOOP
    v_idoferta := oferta.IDOferta;
    v_titulo := oferta.Titulo;
    SELECT COUNT(*) INTO v_cantidad
    FROM Candidaturas
    WHERE IDOferta = v_idoferta;
    DBMS_OUTPUT.PUT_LINE('Oferta: ' || v_titulo || ', Inscritos: ' || v_cantidad);
  END LOOP;
END;
/
BEGIN
  mostrar_ofertas_con_inscritos;
END;
/

```

```
CREATE OR REPLACE PROCEDURE mostrar_ofertas_con_inscritos AS
  v_idoferta OfertasDeEmpleo.IDOferta%TYPE;
  v_titulo OfertasDeEmpleo.Titulo%TYPE;
  v_cantidad NUMBER;
BEGIN
  FOR oferta IN (SELECT IDOferta, Titulo FROM OfertasDeEmpleo) LOOP
    v_idoferta := oferta.IDOferta;
    v_titulo := oferta.Titulo;
    SELECT COUNT(*) INTO v_cantidad
    FROM Candidaturas
    WHERE IDOferta = v_idoferta;
    DBMS_OUTPUT.PUT_LINE('Oferta: ' || v_titulo || ', Inscritos: ' || v_cantidad);
  END LOOP;
END;
/
BEGIN
  mostrar_ofertas_con_inscritos;
END;
/
```

Salida de Script x Resultado de la Consulta x

Tarea terminada en 0,041 segundos

```
Procedure MOSTRAR_OFERTAS_CON_INSCRITOS compilado

Oferta: Desarrollador Full Stack, Inscritos: 3
Oferta: Analista de Datos, Inscritos: 4
Oferta: Diseñador Gráfico, Inscritos: 4
Oferta: Asistente Administrativo, Inscritos: 4
Oferta: Ingeniero de Software, Inscritos: 3
Oferta: Desarrollador de Software, Inscritos: 0

Procedimiento PL/SQL terminado correctamente.
```

En este procedimiento, utilizamos un bucle **FOR** para iterar a través de las filas de la tabla "OfertasDeEmpleo". En cada iteración del bucle, almacenamos el ID de la oferta y el título en las variables **v_idoferta** y **v_titulo**, respectivamente.

Luego, ejecutamos una consulta **SELECT COUNT(*)** para obtener la cantidad de usuarios inscritos en cada oferta. Utilizamos la cláusula **WHERE** para filtrar las candidaturas por el ID de la oferta actual en la iteración del bucle.

Guardamos el resultado de la consulta en la variable **v_cantidad** utilizando la sentencia **INTO**.

Finalmente, utilizamos **DBMS_OUTPUT.PUT_LINE** para mostrar en la salida estándar el título de la oferta y la cantidad de usuarios inscritos.

Se llamará al procedimiento **mostrar_ofertas_con_inscritos** y mostrará en la salida estándar el título de cada oferta de empleo y la cantidad de usuarios inscritos en cada una.