LABORATORIO CALIFICADO

Creación de Tablas

```
-- ========================================
-- CREACIÓN DE TABLAS BASE
-- ========================================

CREATE TABLE region (
  region_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  nombre VARCHAR2(50) NOT NULL
);

CREATE TABLE puesto (
  puesto_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  nombre VARCHAR2(50) NOT NULL,
  salario NUMBER(10,2)
);

CREATE TABLE employee (
  employee_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  nombre VARCHAR2(50) NOT NULL,
  apellido VARCHAR2(50) NOT NULL,
  fecha_ingreso DATE DEFAULT SYSDATE,
  puesto_id NUMBER,
  region_id NUMBER,
  salario NUMBER(10,2),
  CONSTRAINT fk_employee_puesto FOREIGN KEY (puesto_id) REFERENCES puesto(puesto_id),
  CONSTRAINT fk_employee_region FOREIGN KEY (region_id) REFERENCES region(region_id)
);

CREATE TABLE historial_puestos (
  historial_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  employee_id NUMBER NOT NULL,
  puesto_id NUMBER NOT NULL,
  fecha_cambio DATE DEFAULT SYSDATE,
  CONSTRAINT fk_historial_employee FOREIGN KEY (employee_id) REFERENCES
employee(employee_id),
  CONSTRAINT fk_historial_puesto FOREIGN KEY (puesto_id) REFERENCES puesto(puesto_id)
);
```

Inserción de valores de ejemplo

```
INSERT INTO region (nombre) VALUES ('Lima');
INSERT INTO region (nombre) VALUES ('Cusco');
INSERT INTO region (nombre) VALUES ('Arequipa');

INSERT INTO puesto (nombre, salario) VALUES ('Analista', 3500);
INSERT INTO puesto (nombre, salario) VALUES ('Programador', 4200);
INSERT INTO puesto (nombre, salario) VALUES ('Jefe de Proyecto', 6000);
INSERT INTO puesto (nombre, salario) VALUES ('Gerente', 8500);

INSERT INTO employee (nombre, apellido, fecha_ingreso, puesto_id, region_id, salario)
VALUES ('Renzo', 'Munayco', DATE '2020-02-15', 2, 1, 4200);
INSERT INTO employee (nombre, apellido, fecha_ingreso, puesto_id, region_id, salario)
VALUES ('María', 'Vargas', DATE '2018-10-03', 3, 2, 6000);
INSERT INTO employee (nombre, apellido, fecha_ingreso, puesto_id, region_id, salario)
VALUES ('Carlos', 'Rojas', DATE '2016-05-20', 1, 3, 3500);
INSERT INTO employee (nombre, apellido, fecha_ingreso, puesto_id, region_id, salario)
```

```
VALUES ('Lucía', 'Paredes', DATE '2017-01-10', 4, 1, 8500);

INSERT INTO historial_puestos (employee_id, puesto_id, fecha_cambio) VALUES (1, 1, DATE
'2020-05-01');
INSERT INTO historial_puestos (employee_id, puesto_id, fecha_cambio) VALUES (1, 2, DATE
'2021-01-10');
INSERT INTO historial_puestos (employee_id, puesto_id, fecha_cambio) VALUES (1, 3, DATE
'2023-04-22');
INSERT INTO historial_puestos (employee_id, puesto_id, fecha_cambio) VALUES (2, 3, DATE
'2019-07-15');
INSERT INTO historial_puestos (employee_id, puesto_id, fecha_cambio) VALUES (2, 4, DATE
'2022-02-18');
INSERT INTO historial_puestos (employee_id, puesto_id, fecha_cambio) VALUES (3, 1, DATE
'2016-05-20');
INSERT INTO historial_puestos (employee_id, puesto_id, fecha_cambio) VALUES (3, 2, DATE
'2018-12-01');
INSERT INTO historial_puestos (employee_id, puesto_id, fecha_cambio) VALUES (4, 4, DATE
'2017-01-10');
COMMIT;

CREATE TABLE horario (
  dia_semana VARCHAR2(15),
  turno VARCHAR2(20),
  hora_inicio DATE,
  hora_fin DATE
);

CREATE TABLE empleado_horario (
  dia_semana VARCHAR2(15),
  turno VARCHAR2(20),
  employee_id NUMBER,
  CONSTRAINT fk_emp_horario FOREIGN KEY (employee_id) REFERENCES
employee(employee_id)
);

CREATE TABLE asistencia_empleado (
  employee_id NUMBER,
  dia_semana VARCHAR2(15),
  fecha_real DATE,
  hora_inicio_real DATE,
  hora_fin_real DATE,
  CONSTRAINT fk_asistencia_emp FOREIGN KEY (employee_id) REFERENCES
employee(employee_id)
);

INSERT INTO horario VALUES ('Lunes', 'Mañana', TO_DATE('08:00', 'HH24:MI'),
TO_DATE('16:00', 'HH24:MI'));
INSERT INTO horario VALUES ('Martes', 'Mañana', TO_DATE('08:00', 'HH24:MI'),
TO_DATE('16:00', 'HH24:MI'));
INSERT INTO horario VALUES ('Miércoles', 'Mañana', TO_DATE('08:00', 'HH24:MI'),
TO_DATE('16:00', 'HH24:MI'));
INSERT INTO horario VALUES ('Jueves', 'Mañana', TO_DATE('08:00', 'HH24:MI'),
```

```
TO_DATE('16:00', 'HH24:MI'));
INSERT INTO horario VALUES ('Viernes', 'Mañana', TO_DATE('08:00', 'HH24:MI'),
TO_DATE('16:00', 'HH24:MI'));

INSERT INTO empleado_horario VALUES ('Lunes', 'Mañana', 1);
INSERT INTO empleado_horario VALUES ('Martes', 'Mañana', 1);
INSERT INTO empleado_horario VALUES ('Miércoles', 'Mañana', 1);
INSERT INTO empleado_horario VALUES ('Jueves', 'Mañana', 2);
INSERT INTO empleado_horario VALUES ('Viernes', 'Mañana', 2);

INSERT INTO asistencia_empleado VALUES (1, 'Lunes', TO_DATE('2024-10-07', 'YYYY-MM-
DD'), TO_DATE('08:05', 'HH24:MI'), TO_DATE('16:02', 'HH24:MI'));
INSERT INTO asistencia_empleado VALUES (1, 'Martes', TO_DATE('2024-10-08', 'YYYY-MM-
DD'), TO_DATE('08:03', 'HH24:MI'), TO_DATE('16:01', 'HH24:MI'));
INSERT INTO asistencia_empleado VALUES (2, 'Jueves', TO_DATE('2024-10-10', 'YYYY-MM-
DD'), TO_DATE('08:10', 'HH24:MI'), TO_DATE('16:15', 'HH24:MI'));
COMMIT;
```

3.1.1.

```
 PROCEDURE top_empleados_rotacion IS
 BEGIN
   DBMS_OUTPUT.PUT_LINE('CÓDIGO | APELLIDO | NOMBRE | PUESTO ACTUAL |
ROTACIONES');
   FOR r IN (
     SELECT e.employee_id, e.apellido, e.nombre, p.nombre AS puesto, COUNT(h.historial_id) AS
cambios
     FROM employee e
     JOIN puesto p ON e.puesto_id = p.puesto_id
     JOIN historial_puestos h ON e.employee_id = h.employee_id
     GROUP BY e.employee_id, e.apellido, e.nombre, p.nombre
     ORDER BY cambios DESC FETCH FIRST 4 ROWS ONLY
   ) LOOP
     DBMS_OUTPUT.PUT_LINE(r.employee_id || ' | ' || r.apellido || ' | ' || r.nombre || ' | ' || r.puesto || ' | ' ||
r.cambios);
   END LOOP;
 END;
```

```
CÓDIGO | APELLIDO | NOMBRE | PUESTO ACTUAL | ROTACIONES
1 | Perez | Juan | Programador | 18
2 | Vargas | María | Jefe de Proyecto | 12
3 | Rojas | Carlos | Analista | 12
4 | Paredes | Lucía | Gerente | 6
```

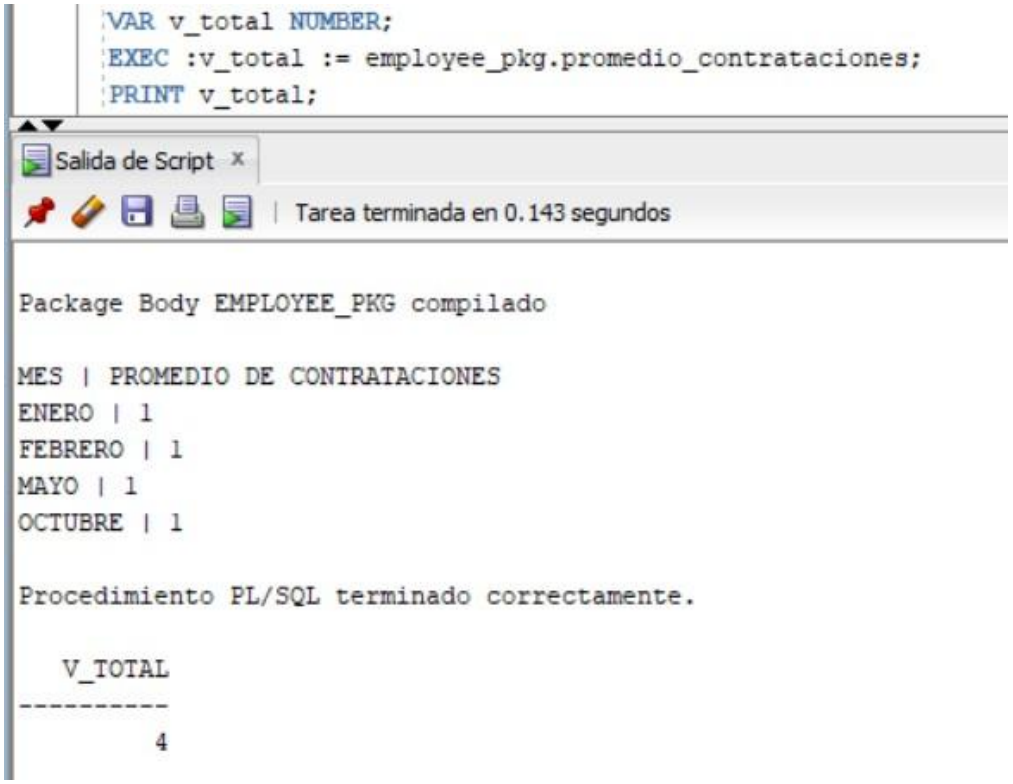
3.1.2.

```
FUNCTION promedio_contrataciones RETURN NUMBER IS
   v_total NUMBER := 0;
 BEGIN
   FOR r IN (
     SELECT TO_CHAR(fecha_ingreso, 'Month', 'NLS_DATE_LANGUAGE=SPANISH') AS mes,
```

```
        ROUND(COUNT(*) / COUNT(DISTINCT EXTRACT(YEAR FROM fecha_ingreso)), 2)
AS promedio
    FROM employee
    GROUP BY TO_CHAR(fecha_ingreso, 'Month', 'NLS_DATE_LANGUAGE=SPANISH'),
          TO_NUMBER(TO_CHAR(fecha_ingreso, 'MM'))
    ORDER BY TO_NUMBER(TO_CHAR(fecha_ingreso, 'MM'))
   ) LOOP
    DBMS_OUTPUT.PUT_LINE(RTRIM(r.mes) || ' | ' || r.promedio);
    v_total := v_total + 1;
   END LOOP;
   RETURN v_total;
 END;

3.1.3
 PROCEDURE estadistica_regional IS
 BEGIN
   DBMS_OUTPUT.PUT_LINE('REGIÓN | TOTAL SALARIOS | EMPLEADOS | MÁS
ANTIGUO');
   FOR r IN (
    SELECT reg.nombre AS region,
         SUM(emp.salario) AS total_salario,
         COUNT(emp.employee_id) AS empleados,
         MIN(emp.fecha_ingreso) AS antiguo
    FROM employee emp
    JOIN region reg ON emp.region_id = reg.region_id
    GROUP BY reg.nombre
   ) LOOP
    DBMS_OUTPUT.PUT_LINE(r.region || ' | ' || r.total_salario || ' | ' || r.empleados || ' | ' ||
TO_CHAR(r.antiguo, 'DD-MON-YYYY'));
   END LOOP;
 END;


 -- ========== 3.1.1 ==========

 PROCEDURE top_empleados_rotacion IS

 BEGIN

    DBMS_OUTPUT.PUT_LINE('CÓDIGO | APELLIDO | NOMBRE | PUESTO ACTUAL | N°
ROTACIONES');
```

```
  FOR r IN (
    SELECT e.employee_id AS codigo,
           e.apellido,
           e.nombre,
           p.nombre AS puesto_actual,
           COUNT(h.historial_id) AS rotaciones
    FROM employee e
    JOIN puesto p ON e.puesto_id = p.puesto_id
    JOIN historial_puestos h ON e.employee_id = h.employee_id
    GROUP BY e.employee_id, e.apellido, e.nombre, p.nombre
    ORDER BY rotaciones DESC
    FETCH FIRST 4 ROWS ONLY
  ) LOOP
     DBMS_OUTPUT.PUT_LINE(r.codigo || ' | ' || r.apellido || ' | ' || r.nombre || ' | ' || r.puesto_actual || '
| ' || r.rotaciones);
  END LOOP;
 END top_empleados_rotacion;

 -- ========== 3.1.2 ==========
 FUNCTION promedio_contrataciones RETURN NUMBER IS
  v_total_meses NUMBER := 0;
 BEGIN
  DBMS_OUTPUT.PUT_LINE('MES | PROMEDIO DE CONTRATACIONES');

  FOR r IN (
    SELECT
      TO_CHAR(fecha_ingreso, 'MONTH', 'NLS_DATE_LANGUAGE=SPANISH') AS mes,
       ROUND(COUNT(*) / COUNT(DISTINCT EXTRACT(YEAR FROM fecha_ingreso)), 2) AS
promedio
    FROM employee
    GROUP BY TO_CHAR(fecha_ingreso, 'MONTH', 'NLS_DATE_LANGUAGE=SPANISH'),
```

```
            TO_NUMBER(TO_CHAR(fecha_ingreso, 'MM'))
    ORDER BY TO_NUMBER(TO_CHAR(fecha_ingreso, 'MM'))
   ) LOOP
    DBMS_OUTPUT.PUT_LINE(RTRIM(r.mes) || ' | ' || r.promedio);
    v_total_meses := v_total_meses + 1;
   END LOOP;

   RETURN v_total_meses;
  END promedio_contrataciones;

END employee_pkg;
/

VAR v_total NUMBER;
EXEC :v_total := employee_pkg.promedio_contrataciones;
PRINT v_total;
```



¿

```
-- ========== 3.1.3 ==========

 PROCEDURE estadistica_regional IS
 BEGIN
   DBMS_OUTPUT.PUT_LINE('REGIÓN | TOTAL SALARIOS | EMPLEADOS | MÁS
ANTIGUO');
   FOR r IN (
     SELECT reg.nombre AS region,
          SUM(emp.salario) AS total_salario,
          COUNT(emp.employee_id) AS empleados,
          MIN(emp.fecha_ingreso) AS antiguo
     FROM employee emp
     JOIN region reg ON emp.region_id = reg.region_id
     GROUP BY reg.nombre
   ) LOOP
     DBMS_OUTPUT.PUT_LINE(r.region || ' | ' || r.total_salario || ' | ' || r.empleados || ' | ' ||
TO_CHAR(r.antiguo, 'DD-MON-YYYY'));
   END LOOP;
 END;
```

```
REGIÓN | TOTAL SALARIOS | EMPLEADOS | MÁS ANTIGUO
Lima | 93000 | 16 | 10-ENE-2017
Cusco | 48000 | 8 | 03-OCT-2018
Arequipa | 21000 | 6 | 20-MAY-2016

Procedimiento PL/SQL terminado correctamente.
```

```
  BEGIN
        DBMS_OUTPUT.PUT_LINE('REGIÓN | TOTAL SALARIOS | N° EMPLEADOS |
EMPLEADO MÁS ANTIGUO');

   FOR r IN (
    SELECT
     reg.nombre AS region,
     SUM(emp.salario) AS total_salario,
     COUNT(emp.employee_id) AS total_empleados,
    MIN(emp.fecha_ingreso) AS fecha_antigua
    FROM employee emp
    JOIN region reg ON emp.region_id = reg.region_id
    GROUP BY reg.nombre
    ORDER BY reg.nombre
   ) LOOP
    DBMS_OUTPUT.PUT_LINE(
     RPAD(r.region, 10) || ' | ' ||
     LPAD(TO_CHAR(r.total_salario, '999,999.99'), 12) || ' | ' ||
     LPAD(r.total_empleados, 5) || ' | ' ||
     TO_CHAR(r.fecha_antigua, 'DD-MON-YYYY')
    );
   END LOOP;
  END estadistica_regional;

END employee_pkg;
/

EXEC employee_pkg.estadistica_regional;
```

```
EXEC employee_pkg.estadistica_regional;
```


```
Package Body EMPLOYEE_PKG compilado

REGIÓN | TOTAL SALARIOS | N° EMPLEADOS | EMPLEADO MÁS ANTIGUO
Arequipa    |     3,500.00 |     1 | 20-MAY-2016
Cusco       |     6,000.00 |     1 | 03-OCT-2018
Lima        |    12,700.00 |     2 | 10-ENE-2017

Procedimiento PL/SQL terminado correctamente.
```

```
 -- ========== 3.1.4 ==========
 FUNCTION tiempo_servicio RETURN NUMBER IS
   v_total NUMBER := 0;
   v_meses NUMBER;
 BEGIN
   FOR r IN (
     SELECT employee_id, nombre, apellido, TRUNC(MONTHS_BETWEEN(SYSDATE,
fecha_ingreso)/12) AS anios
     FROM employee
   ) LOOP
     v_meses := r.anios;
     DBMS_OUTPUT.PUT_LINE(r.nombre || ' ' || r.apellido || ' | ' || r.anios || ' años | ' || v_meses || '
meses');
     v_total := v_total + v_meses;
   END LOOP;
   RETURN v_total;
 END;

 -- ========== 3.1.5 ==========

FUNCTION horas_trabajadas(p_employee_id NUMBER, p_mes NUMBER, p_anio NUMBER)
RETURN NUMBER IS
   v_total_horas NUMBER := 0;
 BEGIN
   DBMS_OUTPUT.PUT_LINE('FECHA | HORAS TRABAJADAS');
   FOR r IN (
     SELECT fecha_real,
          ROUND((hora_fin_real - hora_inicio_real) * 24, 2) AS horas
     FROM asistencia_empleado
     WHERE employee_id = p_employee_id
       AND EXTRACT(MONTH FROM fecha_real) = p_mes
       AND EXTRACT(YEAR FROM fecha_real) = p_anio
   ) LOOP
     DBMS_OUTPUT.PUT_LINE(TO_CHAR(r.fecha_real, 'DD-MON-YYYY') || ' | ' || r.horas);
     v_total_horas := v_total_horas + r.horas;
   END LOOP;
   RETURN v_total_horas;
 END;

END employee_pkg;
```

```
/

SET SERVEROUTPUT ON;
EXEC employee_pkg.top_empleados_rotacion;
VAR v_horas NUMBER;
EXEC :v_horas := employee_pkg.horas_trabajadas(1, 10, 2024);
PRINT v_horas;
```

```
Procedimiento PL/SQL terminado correctamente.
FECHA | HORAS TRABAJADAS
07-OCT-2024 | 7,95
08-OCT-2024 | 7,97
07-OCT-2024 | 7,95
08-OCT-2024 | 7,97
07-OCT-2024 | 7,95
08-OCT-2024 | 7,97
07-OCT-2024 | 7,95
08-OCT-2024 | 7,97
07-OCT-2024 | 7,95
08-OCT-2024 | 7,97
07-OCT-2024 | 7,95
08-OCT-2024 | 7,97

Procedimiento PL/SQL terminado correctamente.

   V_HORAS
----------
     95.52
```

```
-- --------------------------------
-- Ejercicio 3.1.6.
-- --------------------------------
CREATE OR REPLACE PACKAGE employee_pkg AS
  PROCEDURE top_empleados_rotacion;
  FUNCTION promedio_contrataciones RETURN NUMBER;
  PROCEDURE estadistica_regional;
  FUNCTION tiempo_servicio RETURN NUMBER;
  FUNCTION horas_trabajadas(p_employee_id NUMBER, p_mes NUMBER, p_anio NUMBER)
RETURN NUMBER;
  FUNCTION horas_faltadas(p_employee_id NUMBER, p_mes NUMBER, p_anio NUMBER)
RETURN NUMBER;
END employee_pkg;
/
CREATE OR REPLACE PACKAGE BODY employee_pkg AS

  -- 3.1.1
  PROCEDURE top_empleados_rotacion IS
  BEGIN
    DBMS_OUTPUT.PUT_LINE('CÓDIGO | APELLIDO | NOMBRE | PUESTO ACTUAL |
ROTACIONES');
    FOR r IN (
      SELECT e.employee_id, e.apellido, e.nombre, p.nombre AS puesto, COUNT(h.historial_id) AS
cambios
      FROM employee e
      JOIN puesto p ON e.puesto_id = p.puesto_id
      JOIN historial_puestos h ON e.employee_id = h.employee_id
      GROUP BY e.employee_id, e.apellido, e.nombre, p.nombre
```

```
    ORDER BY cambios DESC FETCH FIRST 4 ROWS ONLY
  ) LOOP
    DBMS_OUTPUT.PUT_LINE(r.employee_id || ' | ' || r.apellido || ' | ' || r.nombre || ' | ' || r.puesto || ' |
' || r.cambios);
  END LOOP;
 END;

 -- 3.1.2
 FUNCTION promedio_contrataciones RETURN NUMBER IS
  v_total NUMBER := 0;
 BEGIN
  FOR r IN (
    SELECT TO_CHAR(fecha_ingreso, 'Month', 'NLS_DATE_LANGUAGE=SPANISH') AS mes,
         ROUND(COUNT(*) / COUNT(DISTINCT EXTRACT(YEAR FROM fecha_ingreso)), 2)
AS promedio
    FROM employee
    GROUP BY TO_CHAR(fecha_ingreso, 'Month', 'NLS_DATE_LANGUAGE=SPANISH'),
          TO_NUMBER(TO_CHAR(fecha_ingreso, 'MM'))
    ORDER BY TO_NUMBER(TO_CHAR(fecha_ingreso, 'MM'))
  ) LOOP
    DBMS_OUTPUT.PUT_LINE(RTRIM(r.mes) || ' | ' || r.promedio);
    v_total := v_total + 1;
  END LOOP;
  RETURN v_total;
 END;

 -- 3.1.3
 PROCEDURE estadistica_regional IS
 BEGIN
  DBMS_OUTPUT.PUT_LINE('REGIÓN | TOTAL SALARIOS | EMPLEADOS | MÁS
ANTIGUO');
  FOR r IN (
    SELECT reg.nombre AS region,
         SUM(emp.salario) AS total_salario,
         COUNT(emp.employee_id) AS empleados,
         MIN(emp.fecha_ingreso) AS antiguo
    FROM employee emp
    JOIN region reg ON emp.region_id = reg.region_id
    GROUP BY reg.nombre
  ) LOOP
    DBMS_OUTPUT.PUT_LINE(r.region || ' | ' || r.total_salario || ' | ' || r.empleados || ' | ' ||
TO_CHAR(r.antiguo, 'DD-MON-YYYY'));
  END LOOP;
 END;

 -- 3.1.4
 FUNCTION tiempo_servicio RETURN NUMBER IS
  v_total NUMBER := 0;
  v_meses NUMBER;
 BEGIN
  FOR r IN (
    SELECT employee_id, nombre, apellido, TRUNC(MONTHS_BETWEEN(SYSDATE,
fecha_ingreso)/12) AS anios
    FROM employee
  ) LOOP
```

```
    v_meses := r.anios;
    DBMS_OUTPUT.PUT_LINE(r.nombre || ' ' || r.apellido || ' | ' || r.anios || ' años | ' || v_meses || '
meses');
    v_total := v_total + v_meses;
  END LOOP;
  RETURN v_total;
 END;
 -- 3.1.5
 FUNCTION horas_trabajadas(p_employee_id NUMBER, p_mes NUMBER, p_anio NUMBER)
RETURN NUMBER IS
  v_total_horas NUMBER := 0;
 BEGIN
  DBMS_OUTPUT.PUT_LINE('FECHA | HORAS TRABAJADAS');
  FOR r IN (
    SELECT fecha_real,
           ROUND((hora_fin_real - hora_inicio_real) * 24, 2) AS horas
    FROM asistencia_empleado
    WHERE employee_id = p_employee_id
      AND EXTRACT(MONTH FROM fecha_real) = p_mes
      AND EXTRACT(YEAR FROM fecha_real) = p_anio
  ) LOOP
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(r.fecha_real, 'DD-MON-YYYY') || ' | ' || r.horas);
    v_total_horas := v_total_horas + r.horas;
  END LOOP;
  RETURN v_total_horas;
 END;
```

```
FECHA | HORAS TRABAJADAS
07-OCT-2024 | 7,95
08-OCT-2024 | 7,97
07-OCT-2024 | 7,95
08-OCT-2024 | 7,97
07-OCT-2024 | 7,95
08-OCT-2024 | 7,97
07-OCT-2024 | 7,95
08-OCT-2024 | 7,97
07-OCT-2024 | 7,95
08-OCT-2024 | 7,97
07-OCT-2024 | 7,95
08-OCT-2024 | 7,97
Horas programadas: 864
Horas trabajadas:  95,52
Horas faltadas:    768,48
```

```
 -- 3.1.6
 FUNCTION horas_faltadas(p_employee_id NUMBER, p_mes NUMBER, p_anio NUMBER)
RETURN NUMBER IS
  v_horas_programadas NUMBER := 0;
  v_horas_trabajadas  NUMBER := 0;
  v_horas_faltadas    NUMBER := 0;
 BEGIN
  -- Total de horas programadas según horario
  SELECT SUM((h.hora_fin - h.hora_inicio) * 24)
  INTO v_horas_programadas
  FROM horario h
  JOIN empleado_horario eh ON h.dia_semana = eh.dia_semana AND h.turno = eh.turno
  WHERE eh.employee_id = p_employee_id;
```

```
    -- Total de horas efectivamente trabajadas (usando la función anterior)
    v_horas_trabajadas := horas_trabajadas(p_employee_id, p_mes, p_anio);

    -- Diferencia
    v_horas_faltadas := v_horas_programadas - v_horas_trabajadas;

    DBMS_OUTPUT.PUT_LINE('Horas programadas: ' || v_horas_programadas);
    DBMS_OUTPUT.PUT_LINE('Horas trabajadas:  ' || v_horas_trabajadas);
    DBMS_OUTPUT.PUT_LINE('Horas faltadas:    ' || v_horas_faltadas);

    RETURN v_horas_faltadas;
  END;

END employee_pkg;
/
SET SERVEROUTPUT ON;
VAR v_faltas NUMBER;
EXEC :v_faltas := employee_pkg.horas_faltadas(1, 10, 2024);
PRINT v_faltas;
```

```
Procedimiento PL/SQL terminado correctamente.

  V_FALTAS
----------
    768.48
```

```
-- --------------------------------
-- Ejercicio 3.1.7.
-- --------------------------------
CREATE OR REPLACE PACKAGE employee_pkg AS
  PROCEDURE top_empleados_rotacion;
  FUNCTION promedio_contrataciones RETURN NUMBER;
  PROCEDURE estadistica_regional;
  FUNCTION tiempo_servicio RETURN NUMBER;
  FUNCTION horas_trabajadas(p_employee_id NUMBER, p_mes NUMBER, p_anio NUMBER)
RETURN NUMBER;
  FUNCTION horas_faltadas(p_employee_id NUMBER, p_mes NUMBER, p_anio NUMBER)
RETURN NUMBER;
  PROCEDURE calcular_sueldo(p_mes NUMBER, p_anio NUMBER);
END employee_pkg;
/
CREATE OR REPLACE PACKAGE BODY employee_pkg AS

  -- 3.1.1
  PROCEDURE top_empleados_rotacion IS
  BEGIN
    NULL;
  END;

  -- 3.1.2
  FUNCTION promedio_contrataciones RETURN NUMBER IS
  BEGIN
    RETURN 0;
  END;

  -- 3.1.3
```

```
  PROCEDURE estadistica_regional IS
  BEGIN
    NULL;
  END;

  -- 3.1.4
  FUNCTION tiempo_servicio RETURN NUMBER IS
  BEGIN
    RETURN 0;
  END;

  -- 3.1.5
  FUNCTION horas_trabajadas(p_employee_id NUMBER, p_mes NUMBER, p_anio NUMBER)
RETURN NUMBER IS
    v_total_horas NUMBER := 0;
  BEGIN
    FOR r IN (
      SELECT fecha_real,
           ROUND((hora_fin_real - hora_inicio_real) * 24, 2) AS horas
      FROM asistencia_empleado
      WHERE employee_id = p_employee_id
        AND EXTRACT(MONTH FROM fecha_real) = p_mes
        AND EXTRACT(YEAR FROM fecha_real) = p_anio
    ) LOOP
      v_total_horas := v_total_horas + r.horas;
    END LOOP;
    RETURN v_total_horas;
  END;

  -- 3.1.6
  FUNCTION horas_faltadas(p_employee_id NUMBER, p_mes NUMBER, p_anio NUMBER)
RETURN NUMBER IS
    v_horas_programadas NUMBER := 0;
    v_horas_trabajadas  NUMBER := 0;
    v_horas_faltadas    NUMBER := 0;
  BEGIN
    SELECT SUM((h.hora_fin - h.hora_inicio) * 24)
    INTO v_horas_programadas
    FROM horario h
    JOIN empleado_horario eh ON h.dia_semana = eh.dia_semana AND h.turno = eh.turno
    WHERE eh.employee_id = p_employee_id;

    v_horas_trabajadas := horas_trabajadas(p_employee_id, p_mes, p_anio);
    v_horas_faltadas := v_horas_programadas - v_horas_trabajadas;

    RETURN v_horas_faltadas;
  END;

  -- 3.1.7
  PROCEDURE calcular_sueldo(p_mes NUMBER, p_anio NUMBER) IS
    v_horas_programadas NUMBER;
    v_horas_trabajadas  NUMBER;
    v_sueldo_proporcional NUMBER;
  BEGIN
    DBMS_OUTPUT.PUT_LINE('NOMBRE | APELLIDO | SUELDO MENSUAL AJUSTADO');
```

```
    FOR r IN (SELECT employee_id, nombre, apellido, salario FROM employee) LOOP

      -- Total de horas programadas
      SELECT SUM((h.hora_fin - h.hora_inicio) * 24)
      INTO v_horas_programadas
      FROM horario h
      JOIN empleado_horario eh ON h.dia_semana = eh.dia_semana AND h.turno = eh.turno
      WHERE eh.employee_id = r.employee_id;

      -- Total de horas trabajadas
      v_horas_trabajadas := horas_trabajadas(r.employee_id, p_mes, p_anio);

      -- Cálculo proporcional del sueldo
      IF v_horas_programadas > 0 THEN
        v_sueldo_proporcional := (v_horas_trabajadas / v_horas_programadas) * r.salario;
      ELSE
        v_sueldo_proporcional := 0;
      END IF;

      DBMS_OUTPUT.PUT_LINE(
        RPAD(r.nombre, 10) || ' | ' ||
        RPAD(r.apellido, 10) || ' | ' ||
        LPAD(TO_CHAR(v_sueldo_proporcional, '9999.99'), 10)
      );

    END LOOP;
  END;

END employee_pkg;
/
SET SERVEROUTPUT ON;
EXEC employee_pkg.calcular_sueldo(10, 2024);
```

```
Package Body EMPLOYEE_PKG compilado

NOMBRE | APELLIDO | SUELDO MENSUAL AJUSTADO
Juan       | Perez      |     464.33
María      | Vargas     |     505.00
Carlos     | Rojas      |        .00
Lucía      | Paredes    |        .00
Diego      | Morales    |        .00
Diego      | Morales    |        .00
María      | Vargas     |        .00
Carlos     | Rojas      |        .00
Lucía      | Paredes    |        .00
Diego      | Morales    |        .00
María      | Vargas     |        .00
Carlos     | Rojas      |        .00
Lucía      | Paredes    |        .00
Diego      | Morales    |        .00
Diego      | Morales    |        .00
María      | Vargas     |        .00
Carlos     | Rojas      |        .00
Lucía      | Paredes    |        .00
Diego      | Morales    |        .00
Miriam     | Cotito     |        .00
Carlos     | Rojas      |        .00
Lucía      | Paredes    |        .00
Diego      | Morales    |        .00
Diego      | Morales    |        .00
Miriam     | Cotito     |        .00
Miriam     | Cotito     |        .00
Diego      | Morales    |        .00
Miriam     | Cotito     |        .00
Carlos     | Rojas      |        .00
Lucía      | Paredes    |        .00
```

```
-- ========================================
-- Ejercicio 3.1.1.
-- ========================================
-- ========================================
-- CREACIÓN DE TABLAS DE CAPACITACIÓN
-- ========================================

CREATE TABLE capacitacion (
  capacitacion_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  nombre VARCHAR2(100) NOT NULL,
  horas NUMBER(5,2) NOT NULL,
```

```
  descripcion VARCHAR2(200)
);

CREATE TABLE empleado_capacitacion (
  employee_id NUMBER NOT NULL,
  capacitacion_id NUMBER NOT NULL,
  CONSTRAINT fk_emp_cap_emp FOREIGN KEY (employee_id) REFERENCES
employee(employee_id),
  CONSTRAINT fk_emp_cap_cap FOREIGN KEY (capacitacion_id) REFERENCES
capacitacion(capacitacion_id)
);

-- ========================================
-- INSERCIÓN DE DATOS DE PRUEBA
-- ========================================

-- Capacitaciones
INSERT INTO capacitacion (nombre, horas, descripcion) VALUES ('Gestión de Proyectos', 12,
'Taller sobre planificación y gestión ágil.');
INSERT INTO capacitacion (nombre, horas, descripcion) VALUES ('Seguridad Informática', 8,
'Buenas prácticas en ciberseguridad.');
INSERT INTO capacitacion (nombre, horas, descripcion) VALUES ('Liderazgo', 10, 'Formación en
habilidades de liderazgo.');
INSERT INTO capacitacion (nombre, horas, descripcion) VALUES ('Comunicación Efectiva', 6,
'Mejora de habilidades comunicativas.');
INSERT INTO capacitacion (nombre, horas, descripcion) VALUES ('Bases de Datos Avanzadas', 14,
'Optimización y diseño de bases de datos.');
INSERT INTO capacitacion (nombre, horas, descripcion) VALUES ('Desarrollo Web', 16, 'HTML,
CSS, JavaScript y frameworks modernos.');
INSERT INTO capacitacion (nombre, horas, descripcion) VALUES ('Machine Learning', 20,
'Fundamentos del aprendizaje automático.');
INSERT INTO capacitacion (nombre, horas, descripcion) VALUES ('Excel Avanzado', 8, 'Análisis
de datos con Excel.');
INSERT INTO capacitacion (nombre, horas, descripcion) VALUES ('Atención al Cliente', 5, 'Mejora
de la experiencia del cliente.');
INSERT INTO capacitacion (nombre, horas, descripcion) VALUES ('Trabajo en Equipo', 7,
'Dinámicas y metodologías colaborativas.');
```

```
1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.
```

```
-- Empleado-Capacitación (asignaciones)
INSERT INTO empleado_capacitacion VALUES (1, 1);
INSERT INTO empleado_capacitacion VALUES (1, 3);
INSERT INTO empleado_capacitacion VALUES (1, 5);
INSERT INTO empleado_capacitacion VALUES (2, 2);
INSERT INTO empleado_capacitacion VALUES (2, 6);
INSERT INTO empleado_capacitacion VALUES (3, 4);
INSERT INTO empleado_capacitacion VALUES (3, 7);
INSERT INTO empleado_capacitacion VALUES (4, 8);
INSERT INTO empleado_capacitacion VALUES (4, 9);
INSERT INTO empleado_capacitacion VALUES (4, 10);

COMMIT;

CREATE OR REPLACE PACKAGE capacitacion_pkg AS
  FUNCTION horas_capacitacion_total(p_employee_id NUMBER) RETURN NUMBER;
END capacitacion_pkg;
/

CREATE OR REPLACE PACKAGE BODY capacitacion_pkg AS

  FUNCTION horas_capacitacion_total(p_employee_id NUMBER) RETURN NUMBER IS
    v_total_horas NUMBER := 0;
  BEGIN
    SELECT NVL(SUM(c.horas), 0)
    INTO v_total_horas
    FROM capacitacion c
    JOIN empleado_capacitacion ec ON c.capacitacion_id = ec.capacitacion_id
    WHERE ec.employee_id = p_employee_id;

    DBMS_OUTPUT.PUT_LINE('Empleado ID: ' || p_employee_id || ' | Total horas de capacitación: '
```

```
|| v_total_horas);

   RETURN v_total_horas;
  END;

END capacitacion_pkg;
/

SET SERVEROUTPUT ON;

VAR v_horas_cap NUMBER;
EXEC :v_horas_cap := capacitacion_pkg.horas_capacitacion_total(1);
PRINT v_horas_cap;
```

```
Package CAPACITACION_PKG compilado


Package Body CAPACITACION_PKG compilado

Empleado ID: 1 | Total horas de capacitación: 36

Procedimiento PL/SQL terminado correctamente.

V_HORAS_CAP
-----------
         36
```

```
-- ========================================
-- Ejercicio 3.1.2.
-- ========================================
-- ========================================
-- 3.1.2 PROCEDIMIENTO: LISTAR CAPACITACIONES Y HORAS POR EMPLEADO
-- ========================================

CREATE OR REPLACE PACKAGE capacitacion_pkg AS
  PROCEDURE listar_capacitaciones_empleado;
END capacitacion_pkg;
/

CREATE OR REPLACE PACKAGE BODY capacitacion_pkg AS

  PROCEDURE listar_capacitaciones_empleado IS
  BEGIN
    DBMS_OUTPUT.PUT_LINE('NOMBRE EMPLEADO | APELLIDO | TOTAL HORAS |
CAPACITACIONES');
    DBMS_OUTPUT.PUT_LINE('------------------------------------------------------------');

    FOR r IN (
      SELECT
        e.nombre,
        e.apellido,
        SUM(c.horas) AS total_horas,
        LISTAGG(c.nombre, ', ') WITHIN GROUP (ORDER BY c.nombre) AS lista_capacitaciones
      FROM employee e
      JOIN empleado_capacitacion ec ON e.employee_id = ec.employee_id
      JOIN capacitacion c ON ec.capacitacion_id = c.capacitacion_id
```

```
    GROUP BY e.nombre, e.apellido
    ORDER BY total_horas DESC
  ) LOOP
    DBMS_OUTPUT.PUT_LINE(
      RPAD(r.nombre, 12) || ' | ' ||
      RPAD(r.apellido, 12) || ' | ' ||
      LPAD(r.total_horas, 5) || ' | ' ||
      r.lista_capacitaciones
    );
  END LOOP;
 END listar_capacitaciones_empleado;

END capacitacion_pkg;
/

SET SERVEROUTPUT ON;
EXEC capacitacion_pkg.listar_capacitaciones_empleado;
```

```
Package Body CAPACITACION_PKG compilado

NOMBRE EMPLEADO | APELLIDO | TOTAL HORAS | CAPACITACIONES
------------------------------------------------------------
Juan         | Perez        |    36 | Bases de Datos Avanzadas, Gestión de Proyectos, Liderazgo
Carlos       | Rojas        |    26 | Comunicación Efectiva, Machine Learning
María        | Vargas       |    24 | Desarrollo Web, Seguridad Informática
Lucía        | Paredes      |    20 | Atención al Cliente, Excel Avanzado, Trabajo en Equipo
```

```
-- ========================================
-- 3.2 TRIGGER: VALIDAR INSERCIÓN DE ASISTENCIA
-- ========================================

CREATE OR REPLACE TRIGGER trg_validar_asistencia
BEFORE INSERT ON asistencia_empleado
FOR EACH ROW
DECLARE
 v_dia_semana VARCHAR2(15);
 v_hora_inicio DATE;
 v_hora_fin DATE;
 v_turno VARCHAR2(20);
BEGIN
 = INITCAP(TO_CHAR(:NEW.fecha_real, 'DAY', 'NLS_DATE_LANGUAGE=SPANISH'));
RTRIM(v_dia_semana);
 IF UPPER(v_dia_semana) <> UPPER(:NEW.dia_semana) THEN
  RAISE_APPLICATION_ERROR(-20001, 'Error: El día de la semana no corresponde con la fecha
ingresada.');
 END IF;

 SELECT h.hora_inicio, h.hora_fin, h.turno
 INTO v_hora_inicio, v_hora_fin, v_turno
 FROM horario h
 JOIN empleado_horario eh
  ON h.dia_semana = eh.dia_semana AND h.turno = eh.turno
 WHERE eh.employee_id = :NEW.employee_id
  AND UPPER(h.dia_semana) = UPPER(:NEW.dia_semana);

 IF ABS((:NEW.hora_inicio_real - v_hora_inicio) * 24 * 60) > 15 THEN
```

```
   RAISE_APPLICATION_ERROR(-20002, 'Error: Hora de inicio real no corresponde al horario asignado.');
  END IF;

  IF ABS((:NEW.hora_fin_real - v_hora_fin) * 24 * 60) > 15 THEN
   RAISE_APPLICATION_ERROR(-20003, 'Error: Hora de término real no corresponde al horario asignado.');
  END IF;

EXCEPTION
  WHEN NO_DATA_FOUND THEN
   RAISE_APPLICATION_ERROR(-20004, 'Error: No existe un horario asignado para este empleado en ese día.');
END;
/

-- Pruebas de inserción
INSERT INTO asistencia_empleado
VALUES (1, 'Lunes', TO_DATE('2024-10-07', 'YYYY-MM-DD'), TO_DATE('08:02', 'HH24:MI'), TO_DATE('16:05', 'HH24:MI'));
```

```
-- Pruebas de inserción
INSERT INTO employee (nombre, apellido, fecha_ingreso, puesto_id, region_id, salario)
VALUES ('José', 'Ramírez', SYSDATE, 2, 1, 4500);  -- Programador (rango 4000-5000)
```

Salida de Script

Tarea terminada en 0.053 segundos

```
Confirmación terminada.

Trigger TRG_VALIDAR_SALARIO compilado


1 fila insertadas.
```

```
-- Pruebas de actualización
UPDATE employee SET salario = 4100 WHERE employee_id = 1;
```

Salida de Script

Tarea terminada en 0.056 segundos

```
1 fila insertadas.


1 fila actualizadas.
```

```
ALTER TABLE puesto ADD (salario_min NUMBER(10,2), salario_max NUMBER(10,2));

-- Actualizamos los rangos de ejemplo
UPDATE puesto SET salario_min = 3000, salario_max = 4000 WHERE nombre = 'Analista';
UPDATE puesto SET salario_min = 4000, salario_max = 5000 WHERE nombre = 'Programador';
UPDATE puesto SET salario_min = 5500, salario_max = 6500 WHERE nombre = 'Jefe de Proyecto';
UPDATE puesto SET salario_min = 8000, salario_max = 9000 WHERE nombre = 'Gerente';

COMMIT;

-- ========================================
-- 3.3 TRIGGER: VALIDAR RANGO DE SALARIO
```

```
-- ========================================

CREATE OR REPLACE TRIGGER trg_validar_salario
BEFORE INSERT OR UPDATE OF salario, puesto_id ON employee
FOR EACH ROW
DECLARE
  v_min NUMBER;
  v_max NUMBER;
  v_nombre_puesto VARCHAR2(50);
BEGIN
  -- Obtener los rangos del puesto correspondiente
  SELECT salario_min, salario_max, nombre
  INTO v_min, v_max, v_nombre_puesto
  FROM puesto
  WHERE puesto_id = :NEW.puesto_id;

  -- Validar que el salario esté dentro del rango
  IF :NEW.salario < v_min OR :NEW.salario > v_max THEN
    RAISE_APPLICATION_ERROR(
      -20010,
      'Error: El salario (' || :NEW.salario ||
      ') no está dentro del rango permitido para el puesto "' ||
      v_nombre_puesto || '" (' || v_min || ' - ' || v_max || ').'
    );
  END IF;
END;
/

-- Pruebas de inserción
INSERT INTO employee (nombre, apellido, fecha_ingreso, puesto_id, region_id, salario)
VALUES ('José', 'Ramírez', SYSDATE, 2, 1, 4500);  -- Programador (rango 4000–5000)
-- Pruebas de actualización
UPDATE employee SET salario = 4100 WHERE employee_id = 1;
-- ========================================
-- 3.4 TRIGGER: CONTROL DE INGRESO Y MARCA DE INASISTENCIA
-- ========================================

CREATE OR REPLACE TRIGGER trg_validar_ingreso
BEFORE INSERT ON asistencia_empleado
FOR EACH ROW
DECLARE
  v_hora_inicio horario.hora_inicio%TYPE;
  v_hora_fin horario.hora_fin%TYPE;
  v_turno VARCHAR2(20);
  v_dia VARCHAR2(15);
BEGIN
  -- Obtener día y turno correspondiente al empleado
  SELECT h.hora_inicio, h.hora_fin, h.turno, h.dia_semana
  INTO v_hora_inicio, v_hora_fin, v_turno, v_dia
  FROM horario h
  JOIN empleado_horario eh
    ON h.dia_semana = eh.dia_semana AND h.turno = eh.turno
  WHERE eh.employee_id = :NEW.employee_id
    AND eh.dia_semana = :NEW.dia_semana;
```

```
  -- Verificar tolerancia de media hora antes o después (± 30 minutos)
  IF :NEW.hora_inicio_real < (v_hora_inicio - (30/1440))
     OR :NEW.hora_inicio_real > (v_hora_inicio + (30/1440)) THEN

    -- Marcamos inasistencia silenciosa
    :NEW.hora_inicio_real := NULL;
    :NEW.hora_fin_real := NULL;

    -- Registrar en consola (solo el administrador lo ve)
    DBMS_OUTPUT.PUT_LINE('Aviso: Empleado ID ' || :NEW.employee_id ||
                         ' llegó fuera del rango permitido y fue marcado como inasistente.');
  END IF;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No existe horario asignado para el empleado ID ' ||
:NEW.employee_id ||
                         ' en el día ' || :NEW.dia_semana);
END;
/

-- Prueba de funcionamiento
ALTER TRIGGER trg_validar_asistencia DISABLE;
INSERT INTO asistencia_empleado
VALUES (1, 'Lunes', TO_DATE('2024-10-21', 'YYYY-MM-DD'),
        TO_DATE('07:45', 'HH24:MI'), TO_DATE('16:00', 'HH24:MI'));

-- Verificando
SELECT * FROM asistencia_empleado;
```

```
-- Verificando
SELECT * FROM asistencia_empleado;
```

Salida de Script | Tarea terminada en 0.097 segundos

```
*Action:   Check the trigger name.

1 fila insertadas.


EMPLOYEE_ID DIA_SEMANA      FECHA_RE HORA_INI HORA_FIN
----------- --------------- -------- -------- --------
          1 Lunes           07/10/24 01/12/25 01/12/25
          1 Martes          08/10/24 01/12/25 01/12/25
          2 Jueves          10/10/24 01/12/25 01/12/25
          1 Lunes           07/10/24 01/12/25 01/12/25
          1 Martes          08/10/24 01/12/25 01/12/25
          2 Jueves          10/10/24 01/12/25 01/12/25
          1 Lunes           07/10/24 01/12/25 01/12/25
          1 Martes          08/10/24 01/12/25 01/12/25
          2 Jueves          10/10/24 01/12/25 01/12/25
          1 Lunes           07/10/24 01/12/25 01/12/25
          1 Martes          08/10/24 01/12/25 01/12/25

EMPLOYEE_ID DIA_SEMANA      FECHA_RE HORA_INI HORA_FIN
----------- --------------- -------- -------- --------
          2 Jueves          10/10/24 01/12/25 01/12/25
          1 Lunes           07/10/24 01/12/25 01/12/25
          1 Martes          08/10/24 01/12/25 01/12/25
          2 Jueves          10/10/24 01/12/25 01/12/25
          1 Lunes           07/10/24 01/12/25 01/12/25
          1 Martes          08/10/24 01/12/25 01/12/25
          2 Jueves          10/10/24 01/12/25 01/12/25
          1 Lunes           07/10/24 01/12/25 01/12/25
          1 Lunes           21/10/24 01/12/25 01/12/25

20 filas seleccionadas.
```