

Evaluación Unidad I

INTRODUCCIÓN A PL/SQL

ASIGNATURA:

Programación de Base de Datos

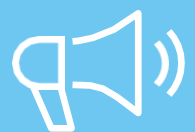
APRENDIZAJE ESPERADO

- En esta actividad lograrás construir componentes PL/SQL con flujos y resultados controlados por excepciones de acuerdo con los requerimientos de la organización.
- Verificar los componentes PL/SQL con flujos y resultados controlados por excepciones de acuerdo con los requerimientos de la organización.
- Validar los componentes PL/SQL con flujos y resultados controlados por excepciones de acuerdo con los requerimientos.

INSTRUCCIONES GENERALES

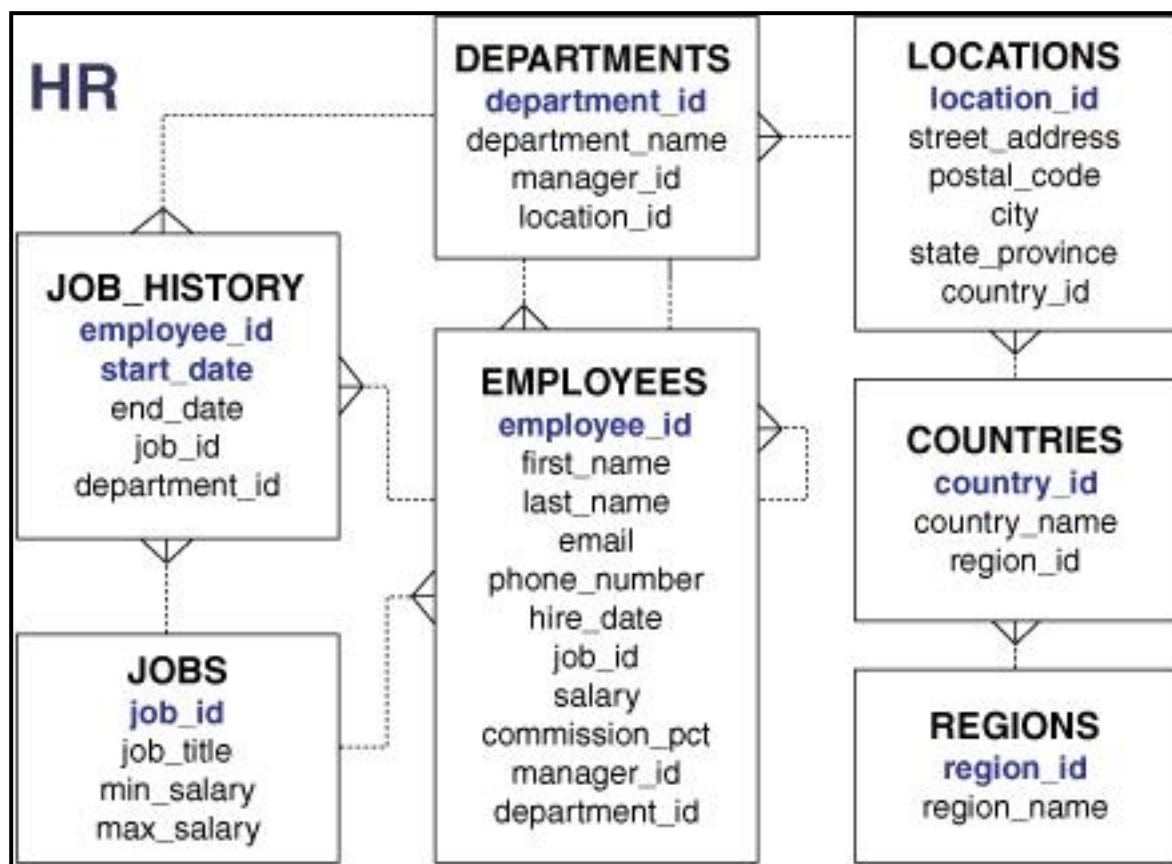
Estimado(a) estudiante:

- A partir de los siguientes requerimientos, se solicita desarrolle las sentencias PL/SQL que permitan dar solución a estos.
- Se pide leer atentamente los requerimientos y desarrollar las sentencias PL/SQL para dar solución a estos. Se han agregado imágenes para que pueda establecer una base de lo que se espera como resultado, una vez haya ejecutado su sentencia. Las imágenes son referenciales, puesto en ocasiones solo muestran una parcialidad de los registros (por temas de espacio dentro del documento).
- Para el desarrollo de la evaluación N°1 debe utilizar el software Oracle Developer asociado a su base datos Oracle Cloud con la conexión del usuario HR. Guarde la evaluación con el formato: NOMBRE_APELLIDO. sql o .txt.
- Como respaldo, también se requiere que envíe un Word con imágenes (pantallazos o screenshots) de lo desarrollado por usted, además de los pasos que describan el proceso de desarrollo de su evaluación.



Esta actividad se complementa con una pauta de autoevaluación al finalizar, para que verifiques el logro de tus aprendizajes

Modelo HR



Utilizando el esquema HR realice los siguientes requerimientos.

Crear Usuario

Usuario:

Contraseña Nueva:

Confirmar Contraseña:

☐ La contraseña ha vencido (el usuario debe cambiarla en la siguiente conexión)

☐ Usuario del Sistema Operativo

☐ La cuenta está bloqueada

☐ Edición Activada

Tablespace por defecto:

Tablespace Temporal:

Correcto

El comando SQL se ha procesado correctamente

Aceptar

Ayuda Aplicar Cerrar

NOTA: Los ejercicios deben ser desarrollados utilizando las tablas del esquema HR de la Base de Datos.

Requerimiento 1

- 1. El departamento de Finanzas de la empresa desea conocer cuál es la realidad de la empresa con relación a los sueldos que hasta la fecha perciben los empleados. Para ello, se desea que Ud. a través de un bloque PL/SQL genere un informe que muestre el salario mínimo, salario máximo, valor total de los salarios, salario promedio entre los empleados que perciben un sueldo superior a \$8.000 y entre cuántos empleados se efectuaron estos cálculos. El informe además debe mostrar la fecha en que se ejecutó el bloque. La información se debe mostrar en el formato del ejemplo y con los valores redondeados:

| | | | | |
|--|------------------|----------------|---------------|--|
| INFORME DE LA EMPRESA | | 29/08/23 | | |
| ----- | | | | |
| Salario Máximo | Salario Promedio | Salario Mínimo | Salario Total | |
| ----- | | | | |
| \$24,000 | \$11,049 | \$8,200 | \$364,616 | |
| ----- | | | | |
| Los valores calculados están efectuados sobre 33 empleados | | | | |

HR.sql

Hoja de Trabajo de SQL | Historial

Hoja de Trabajo

Generador de Consultas

--DESARROLLO REQUERIMIENTO 1

--DECLARACION DE VARIABLES

DECLARE

v_sal_max employees.salary%TYPE;

v_sal_prom employees.salary%TYPE;

v_sal_min employees.salary%TYPE;

v_sal_total employees.salary%TYPE;

v_cant_emp NUMBER(3) := 0;

--INICIO DE EJECUTABLE DEL BLOQUE

BEGIN

SELECT MAX(salary), AVG(salary), MIN(salary), SUM(salary), COUNT(employee_id)

INTO v_sal_max, v_sal_prom, v_sal_min, v_sal_total, v_cant_emp

FROM employees

WHERE salary > 8000;

--IMPRESION DE RESULTADOS

DBMS_OUTPUT.PUT_LINE('INFORME DE LA EMPRESA ' || TO_CHAR(SYSDATE, 'DD/MM/YY'));

DBMS_OUTPUT.PUT_LINE('-----');

DBMS_OUTPUT.PUT_LINE('Salario Máximo Salario Promedio Salario Mínimo Salario Total');

DBMS_OUTPUT.PUT_LINE(

TO_CHAR(v_sal_max, '\$999,999') ||

TO_CHAR(v_sal_prom, '\$999,999') ||

TO_CHAR(v_sal_min, '\$999,999') ||

TO_CHAR(v_sal_total, '\$999,999')

 || ' ' || ' ' || ' ' || ' ');

DBMS_OUTPUT.PUT_LINE('-----');

DBMS_OUTPUT.PUT_LINE('Los valores calculados están efectuados sobre ' || v_cant_emp || ' empleados');

--FINALIZACION DEL BLOQUE

END;

Salida de Script

Tarea terminada en 0,073 segundos

INFORME DE LA EMPRESA

21/12/24

Salario Máximo Salario Promedio Salario Mínimo Salario Total

\$24,000 \$11,049 \$8,200 \$364,616

Los valores calculados están efectuados sobre 33 empleados

Procedimiento PL/SQL terminado correctamente.

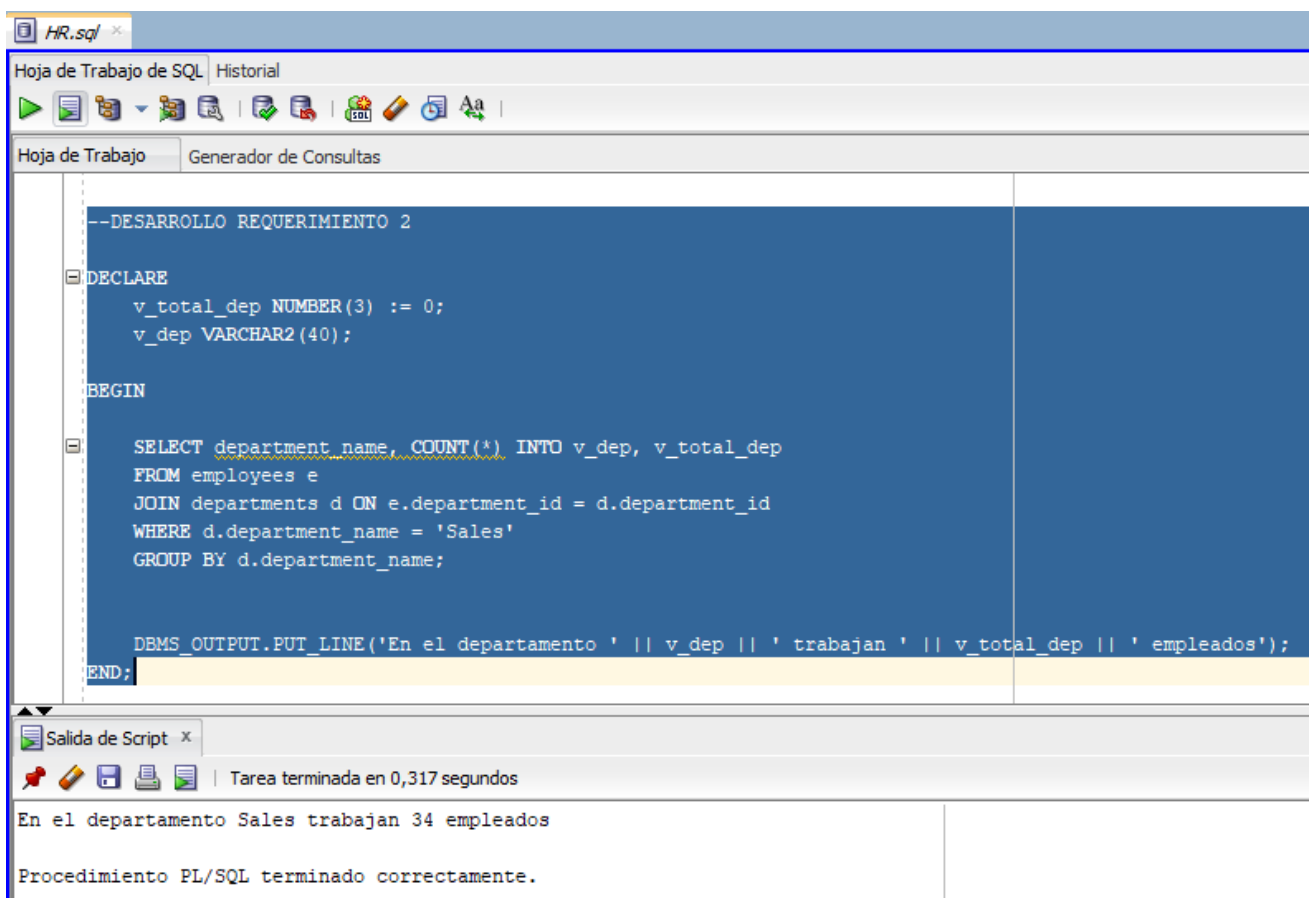
```
1  --DESARROLLO REQUERIMIENTO 1
2
3  --DECLARACION DE VARIABLES
4  DECLARE
5
6      v_sal_max employees.salary%TYPE;
7      v_sal_prom employees.salary%TYPE;
8      v_sal_min employees.salary%TYPE;
9      v_sal_total employees.salary%TYPE;
10     v_cant_emp NUMBER(3) := 0;
11
12 --INICIO DE EJECUTABLE DEL BLOQUE
13 BEGIN
14
15     SELECT MAX(salary), AVG(salary), MIN(salary), SUM(salary), COUNT(employee_id)
16     INTO v_sal_max, v_sal_prom, v_sal_min, v_sal_total, v_cant_emp
17     FROM employees
18     WHERE salary > 8000;
19
20 --IMPRESION DE RESULTADOS
21     DBMS_OUTPUT.PUT_LINE('INFORME DE LA EMPRESA ' || TO_CHAR(SYSDATE, 'DD/MM/YY'));
22     DBMS_OUTPUT.PUT_LINE('-----');
23     DBMS_OUTPUT.PUT_LINE('Salario M ximo          Salario Promedio          Salario M nimo          Salario Total');
24     DBMS_OUTPUT.PUT_LINE(
25         TO_CHAR(v_sal_max, '$999,999') || '          ' ||
26         TO_CHAR(v_sal_prom, '$999,999') || '          ' ||
27         TO_CHAR(v_sal_min, '$999,999') || '          ' ||
28         TO_CHAR(v_sal_total, '$999,999')
29     );
30     DBMS_OUTPUT.PUT_LINE('-----');
31     DBMS_OUTPUT.PUT_LINE('Los valores calculados est n efectuados sobre ' || v_cant_emp || ' empleados');
32
33 --FINALIZACION DEL BLOQUE
34 END;
```

Requerimiento 2

Se requiere crear un bloque PL/SQL anónimo que pueda visualizar el total de empleados que trabajan en el departamento de ventas. Las especificaciones que se deben considerar para la creación del bloque son:

- Definir la variable `v_total_dep` con su tipo de dato numérica con un largo de 3 e inicializada en cero.
- Definir la variable `v_dep` con su tipo de dato de carácter con un largo de 40.
- Obtener el nombre del departamento y el total de empleados que trabajen en el departamento de ventas y almacenar los valores en las variables definidas.
- Se debe mostrar la información en el siguiente formato:

En el departamento Sales trabajan 34 empleados



The screenshot shows the SQL Developer interface. The main window displays a PL/SQL block titled "--DESARROLLO REQUERIMIENTO 2". The code defines two variables: `v_total_dep` of type `NUMBER(3)` initialized to 0, and `v_dep` of type `VARCHAR2(40)`. The `BEGIN` block contains a `SELECT` statement that joins `employees` and `departments` tables, filtering for the 'Sales' department and grouping by department name. The results are stored in `v_dep` and `v_total_dep`. The `END;` block uses `DBMS_OUTPUT.PUT_LINE` to display the results in the specified format. The bottom pane shows the output: "En el departamento Sales trabajan 34 empleados" and "Procedimiento PL/SQL terminado correctamente."

```
--DESARROLLO REQUERIMIENTO 2

DECLARE
    v_total_dep NUMBER(3) := 0;
    v_dep VARCHAR2(40);

BEGIN
    SELECT department_name, COUNT(*) INTO v_dep, v_total_dep
    FROM employees e
    JOIN departments d ON e.department_id = d.department_id
    WHERE d.department_name = 'Sales'
    GROUP BY d.department_name;

    DBMS_OUTPUT.PUT_LINE('En el departamento ' || v_dep || ' trabajan ' || v_total_dep || ' empleados');
END;
```

Salida de Script x | Tarea terminada en 0,317 segundos

En el departamento Sales trabajan 34 empleados

Procedimiento PL/SQL terminado correctamente.

```
1  --DESARROLLO REQUERIMIENTO 2
2
3  DECLARE
4      v_total_dep NUMBER(3) := 0;
5      v_dep VARCHAR2(40);
6
7  BEGIN
8
9      SELECT department_name, COUNT(*) INTO v_dep, v_total_dep
10     FROM employees e
11     JOIN departments d ON e.department_id = d.department_id
12     WHERE d.department_name = 'Sales'
13     GROUP BY d.department_name;
14
15
16     DBMS_OUTPUT.PUT_LINE('En el departamento ' || v_dep || ' trabajan ' || v_total_dep || ' empleados');
17 END;
```

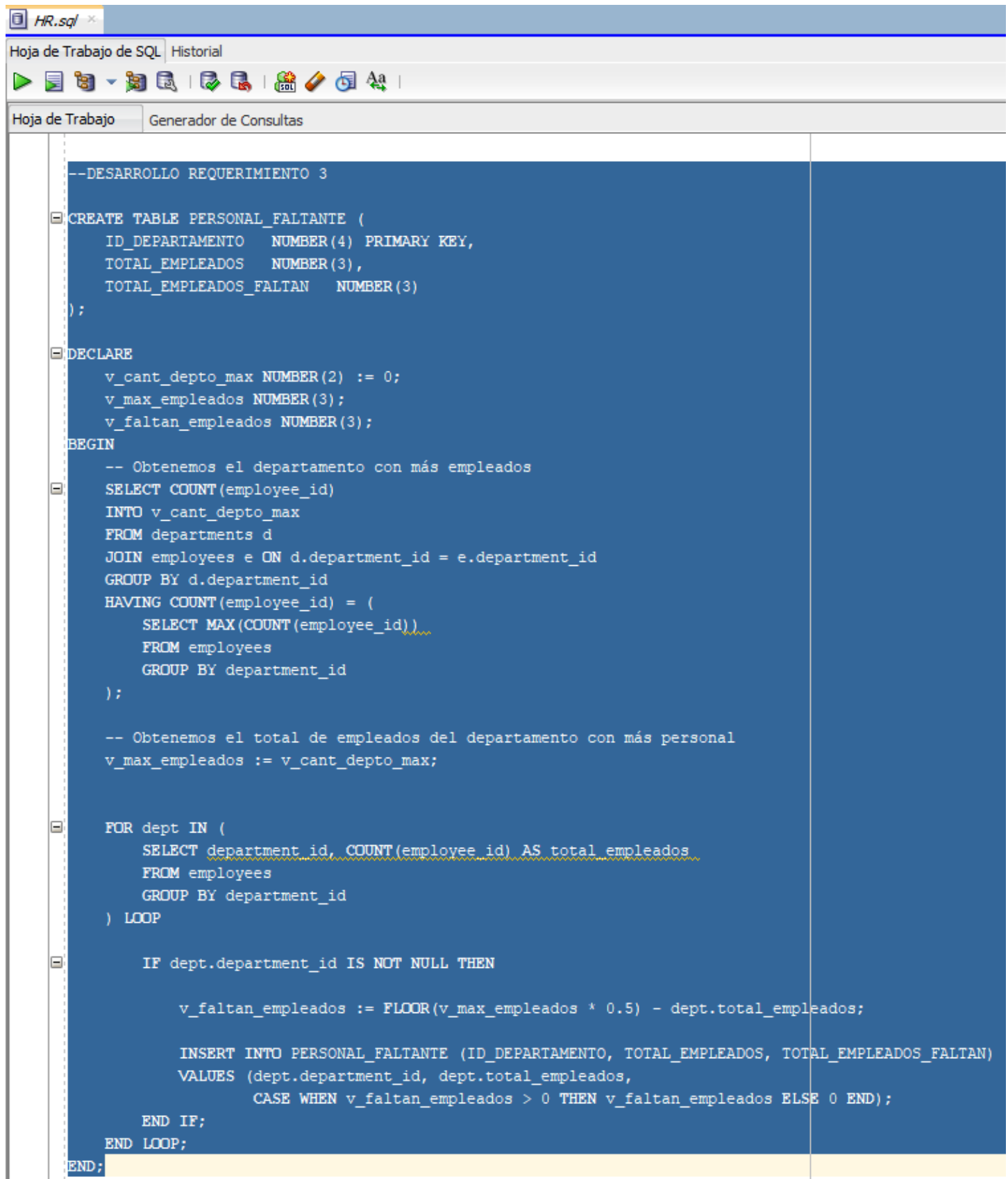
Requerimiento 3

Como una forma de resolver la diferencia de distribución del personal entre los diferentes departamentos, se desea saber los departamentos en los que se deberían aumentar los empleados para lograr contar con el 50% del total de empleados que posee el departamento con mayor cantidad de personal en la empresa. La información debe quedar almacenada en la tabla **PERSONAL_FALTANTE** la que debe ser creada con las siguientes columnas:

| NOMBRE COLUMNA | TIPO DE DATO | VALOR QUE ALMACENARÁ |
|------------------------|---------------------|---|
| ID_DEPARTAMENTO | Numérico de largo 4 | Identificación del departamento y clave primaria de la tabla. |
| TOTAL_EMPLEADOS | Numérico de largo 3 | Total de empleados que posee el departamento. |
| TOTAL_EMPLEADOS_FALTAN | Numérico de largo 3 | Total de empleados que faltan en el departamento para lograr contar con el 20% del total de empleados que posee el departamento con más personal. |

Posteriormente desarrolle el bloque PL/SQL que inserte en la tabla creada la información solicitada. Al ejecutar el bloque, la tabla **PERSONAL_FALTANTE** debería quedar con la información que se muestra en el ejemplo:

| | ID_DEPARTAMENTO | TOTAL_EMPLEADOS | TOTAL_EMPLEADOS_FALTAN |
|---|-----------------|-----------------|------------------------|
| 1 | 10 | 1 | 22 |
| 2 | 20 | 2 | 21 |
| 3 | 30 | 6 | 17 |
| 4 | 40 | 1 | 22 |
| 5 | 60 | 5 | 18 |
| 6 | 70 | 1 | 22 |
| 7 | 90 | 3 | 20 |
| 8 | 100 | 6 | 17 |
| 9 | 110 | 2 | 21 |



```
--DESARROLLO REQUERIMIENTO 3

CREATE TABLE PERSONAL_FALTANTE (
    ID_DEPARTAMENTO    NUMBER(4) PRIMARY KEY,
    TOTAL_EMPLEADOS    NUMBER(3),
    TOTAL_EMPLEADOS_FALTAN    NUMBER(3)
);

DECLARE
    v_cant_depto_max NUMBER(2) := 0;
    v_max_empleados NUMBER(3);
    v_faltan_empleados NUMBER(3);
BEGIN
    -- Obtenemos el departamento con más empleados
    SELECT COUNT(employee_id)
    INTO v_cant_depto_max
    FROM departments d
    JOIN employees e ON d.department_id = e.department_id
    GROUP BY d.department_id
    HAVING COUNT(employee_id) = (
        SELECT MAX(COUNT(employee_id))
        FROM employees
        GROUP BY department_id
    );

    -- Obtenemos el total de empleados del departamento con más personal
    v_max_empleados := v_cant_depto_max;

    FOR dept IN (
        SELECT department_id, COUNT(employee_id) AS total_empleados
        FROM employees
        GROUP BY department_id
    ) LOOP

        IF dept.department_id IS NOT NULL THEN

            v_faltan_empleados := FLOOR(v_max_empleados * 0.5) - dept.total_empleados;

            INSERT INTO PERSONAL_FALTANTE (ID_DEPARTAMENTO, TOTAL_EMPLEADOS, TOTAL_EMPLEADOS_FALTAN)
            VALUES (dept.department_id, dept.total_empleados,
                CASE WHEN v_faltan_empleados > 0 THEN v_faltan_empleados ELSE 0 END);

        END IF;
    END LOOP;
END;
```

| HR.sql | | PERSONAL_FALTANTE | |
|---|------------------------|-------------------|----------------|
| Columnas | Datos | Model | Restricciones |
| <div> <div> <div>Ordenar...</div> <div>Filtrar:</div> </div> </div> | | | |
| 1 | ID_DEPARTAM... | 2 | TOTAL_EMPLE... |
| 3 | TOTAL_EMPLEADOS_FALTAN | | |
| 1 | 10 | 1 | 21 |
| 2 | 20 | 2 | 20 |
| 3 | 30 | 6 | 16 |
| 4 | 40 | 1 | 21 |
| 5 | 50 | 45 | 0 |
| 6 | 60 | 5 | 17 |
| 7 | 70 | 1 | 21 |
| 8 | 80 | 34 | 0 |
| 9 | 90 | 3 | 19 |
| 10 | 100 | 6 | 16 |
| 11 | 110 | 2 | 20 |

```
1  --DESARROLLO REQUERIMIENTO 3
2
3  CREATE TABLE PERSONAL_FALTANTE (
4      ID_DEPARTAMENTO    NUMBER(4) PRIMARY KEY,
5      TOTAL_EMPLEADOS    NUMBER(3),
6      TOTAL_EMPLEADOS_FALTAN    NUMBER(3)
7  );
8
9  DECLARE
10     v_cant_depto_max    NUMBER(2) := 0;
11     v_max_empleados    NUMBER(3);
12     v_faltan_empleados    NUMBER(3);
13 BEGIN
14     -- Obtenemos el departamento con más empleados
15     SELECT COUNT(employee_id)
16     INTO v_cant_depto_max
17     FROM departments d
18     JOIN employees e ON d.department_id = e.department_id
19     GROUP BY d.department_id
20     HAVING COUNT(employee_id) = (
21         SELECT MAX(COUNT(employee_id))
22         FROM employees
23         GROUP BY department_id
24     );
25
26     -- Obtenemos el total de empleados del departamento con más personal
27     v_max_empleados := v_cant_depto_max;
28
29
30     FOR dept IN (
31         SELECT department_id, COUNT(employee_id) AS total_empleados
32         FROM employees
33         GROUP BY department_id
34     ) LOOP
35
36         IF dept.department_id IS NOT NULL THEN
37
38             v_faltan_empleados := FLOOR(v_max_empleados * 0.5) - dept.total_empleados;
39
40             INSERT INTO PERSONAL_FALTANTE (ID_DEPARTAMENTO, TOTAL_EMPLEADOS, TOTAL_EMPLEADOS_FALTAN)
41             VALUES (dept.department_id, dept.total_empleados,
42                 CASE WHEN v_faltan_empleados > 0 THEN v_faltan_empleados ELSE 0 END);
43         END IF;
44     END LOOP;
45 END;
46
47 SELECT * FROM PERSONAL_FALTANTE;
```

Requerimiento 4

En el nuevo sistema que se desarrollará, el departamento de personal requiere poder contar con una opción para efectuar el ingreso de los nuevos departamentos a la empresa la cual se implementará a través de un procedimiento almacenado. En esta primera etapa y para efectos de pruebas, se requiere que Ud. construya un bloque PL/SQL Anónimo que permita efectuar ingreso de nuevos departamentos de acuerdo con las siguientes especificaciones:

- Controlar las excepciones de:
 - Inserción de clave primaria duplicada.
 - Inserción de valor nulo en una columna definida como obligatoria.
- Los errores se deben grabar en una tabla con la siguiente estructura:

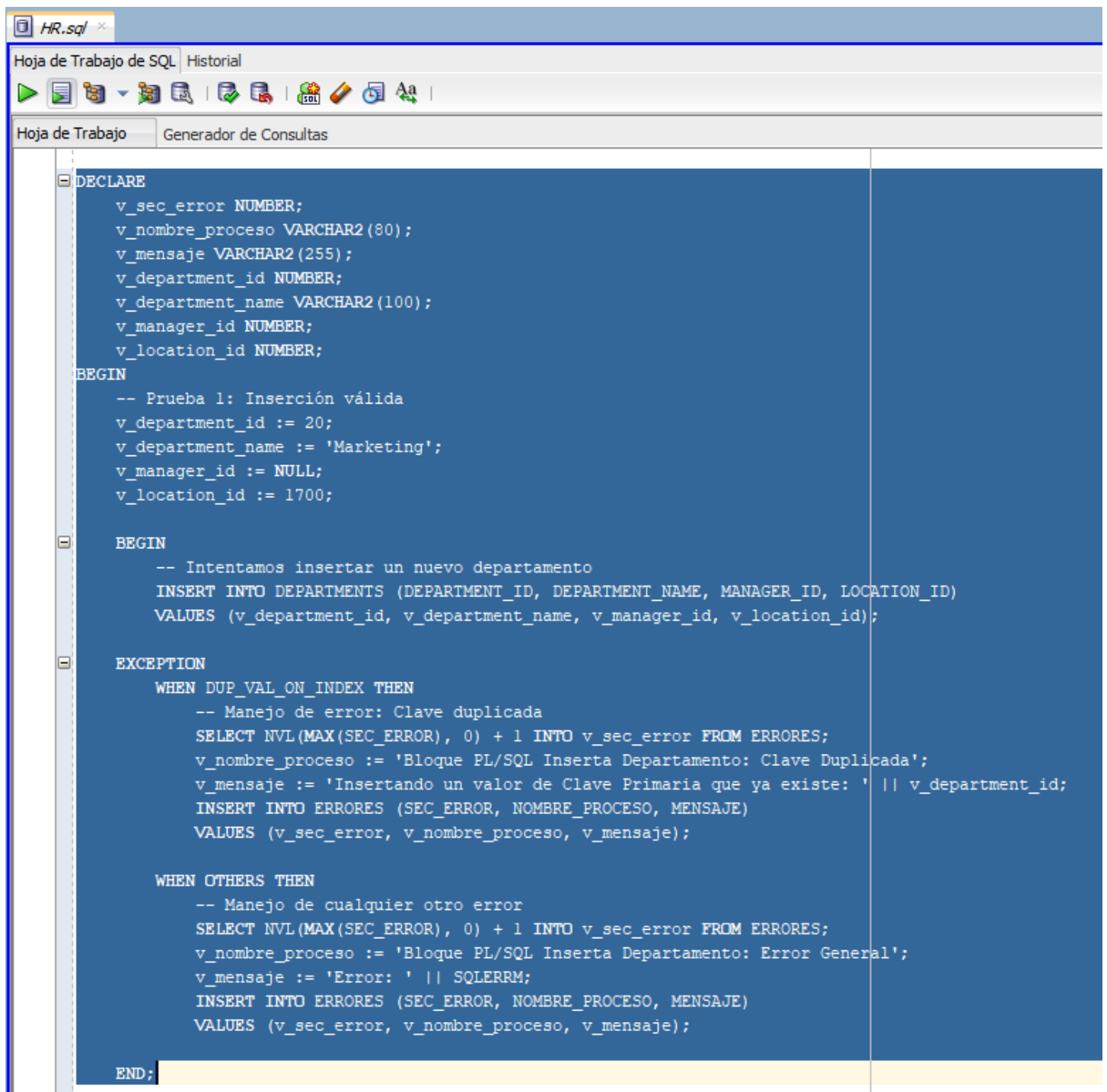
| NOMBRE DE COLUMNA | TIPO DE DATO | INFORMACIÓN QUE ALMACENARÁ |
|-------------------|---|--|
| SEC_ERROR | N Numérico de largo 5 (obligatorio) | Clave primaria de la tabla que corresponde a un número correlativo asignado al momento de grabar una fila. |
| NOMBRE_PROCESO | C Caracter de largo variable de largo 80 (obligatorio) | Proceso: Bloque PL/SQL Inserta Departamento: (Motivo por el cual se produjo la excepción). |
| MENSAJE | C Caracter de largo variable de largo 255 (obligatorio) | Mensaje que detalle el error producido al insertar el nuevo empleado. |

- Efectuar las siguientes pruebas de su bloque con los datos indicados:
 - **PRUEBA 1:** 20, 'Marketing', 300, 1700
 - **PRUEBA 2:** 280, NULL, NULL, NULL

Después de efectuar todas las pruebas, la tabla que construyó para almacenar los errores que el bloque debe controlar debería tener los siguientes datos:

| SEC_ERROR | NOMBRE_PROCESO | MENSAJE |
|-----------|--|---|
| 1 | 1Bloque PL/SQL Inserta Departamento: Clave Duplicada | Insertando un valor de Clave Primaria que ya existe |
| 2 | 2Bloque PL/SQL Inserta Departamento: Departamento Nulo. Error: -1400 | ORA-01400: cannot insert NULL into ("HR"."DEPARTMENTS"."DEPARTM |

```
--DESARROLLO REQUERIMIENTO 4
CREATE TABLE ERRORES (
  SEC_ERROR NUMBER(5) PRIMARY KEY,
  NOMBRE_PROCESO VARCHAR2(80) NOT NULL,
  MENSAJE VARCHAR2(255) NOT NULL
);
```



```
HR.sql x
Hoja de Trabajo de SQL Historial
[Icons]
Hoja de Trabajo Generador de Consultas

DECLARE
    v_sec_error NUMBER;
    v_nombre_proceso VARCHAR2(80);
    v_mensaje VARCHAR2(255);
    v_department_id NUMBER;
    v_department_name VARCHAR2(100);
    v_manager_id NUMBER;
    v_location_id NUMBER;
BEGIN
    -- Prueba 1: Inserción válida
    v_department_id := 20;
    v_department_name := 'Marketing';
    v_manager_id := NULL;
    v_location_id := 1700;

    BEGIN
        -- Intentamos insertar un nuevo departamento
        INSERT INTO DEPARTMENTS (DEPARTMENT_ID, DEPARTMENT_NAME, MANAGER_ID, LOCATION_ID)
        VALUES (v_department_id, v_department_name, v_manager_id, v_location_id);

    EXCEPTION
        WHEN DUP_VAL_ON_INDEX THEN
            -- Manejo de error: Clave duplicada
            SELECT NVL(MAX(SEC_ERROR), 0) + 1 INTO v_sec_error FROM ERRORES;
            v_nombre_proceso := 'Bloque PL/SQL Inserta Departamento: Clave Duplicada';
            v_mensaje := 'Insertando un valor de Clave Primaria que ya existe: ' || v_department_id;
            INSERT INTO ERRORES (SEC_ERROR, NOMBRE_PROCESO, MENSAJE)
            VALUES (v_sec_error, v_nombre_proceso, v_mensaje);

        WHEN OTHERS THEN
            -- Manejo de cualquier otro error
            SELECT NVL(MAX(SEC_ERROR), 0) + 1 INTO v_sec_error FROM ERRORES;
            v_nombre_proceso := 'Bloque PL/SQL Inserta Departamento: Error General';
            v_mensaje := 'Error: ' || SQLERRM;
            INSERT INTO ERRORES (SEC_ERROR, NOMBRE_PROCESO, MENSAJE)
            VALUES (v_sec_error, v_nombre_proceso, v_mensaje);

    END;
```



```

1  --DESARROLLO REQUERIMIENTO 4
2  CREATE TABLE ERRORES (
3  SEC_ERROR NUMBER(5) PRIMARY KEY,
4  NOMBRE_PROCESO VARCHAR2(80) NOT NULL,
5  MENSAJE VARCHAR2(255) NOT NULL
6  );
7
8  DECLARE
9      v_sec_error NUMBER;
10     v_nombre_proceso VARCHAR2(80);
11     v_mensaje VARCHAR2(255);
12     v_department_id NUMBER;
13     v_department_name VARCHAR2(100);
14     v_manager_id NUMBER;
15     v_location_id NUMBER;
16 BEGIN
17     -- Prueba 1: Inserción válida
18     v_department_id := 20;
19     v_department_name := 'Marketing';
20     v_manager_id := NULL;
21     v_location_id := 1700;
22
23     BEGIN
24         -- Intentamos insertar un nuevo departamento
25         INSERT INTO DEPARTMENTS (DEPARTMENT_ID, DEPARTMENT_NAME, MANAGER_ID, LOCATION_ID)
26         VALUES (v_department_id, v_department_name, v_manager_id, v_location_id);
27
28     EXCEPTION
29         WHEN DUP_VAL_ON_INDEX THEN
30             -- Manejo de error: Clave duplicada
31             SELECT NVL(MAX(SEC_ERROR), 0) + 1 INTO v_sec_error FROM ERRORES;
32             v_nombre_proceso := 'Bloque PL/SQL Inserta Departamento: Clave Duplicada';
33             v_mensaje := 'Insertando un valor de Clave Primaria que ya existe: ' || v_department_id;
34             INSERT INTO ERRORES (SEC_ERROR, NOMBRE_PROCESO, MENSAJE)
35             VALUES (v_sec_error, v_nombre_proceso, v_mensaje);
36
37         WHEN OTHERS THEN
38             -- Manejo de cualquier otro error
39             SELECT NVL(MAX(SEC_ERROR), 0) + 1 INTO v_sec_error FROM ERRORES;
40             v_nombre_proceso := 'Bloque PL/SQL Inserta Departamento: Error General';
41             v_mensaje := 'Error: ' || SQLERRM;
42             INSERT INTO ERRORES (SEC_ERROR, NOMBRE_PROCESO, MENSAJE)
43             VALUES (v_sec_error, v_nombre_proceso, v_mensaje);
44
45     END;
46
47     -- Prueba 2: Inserción con valores nulos
48     v_department_id := 280;
49     v_department_name := NULL;
50     v_manager_id := NULL;
51     v_location_id := NULL;
52
53     BEGIN
54         -- Intentamos insertar un nuevo departamento con valores nulos
55         INSERT INTO DEPARTMENTS (DEPARTMENT_ID, DEPARTMENT_NAME, MANAGER_ID, LOCATION_ID)
56         VALUES (v_department_id, v_department_name, v_manager_id, v_location_id);
57
58     EXCEPTION
59         WHEN DUP_VAL_ON_INDEX THEN
60             -- Manejo de error: Clave duplicada
61             SELECT NVL(MAX(SEC_ERROR), 0) + 1 INTO v_sec_error FROM ERRORES;
62             v_nombre_proceso := 'Bloque PL/SQL Inserta Departamento: Clave Duplicada';
63             v_mensaje := 'Insertando un valor de Clave Primaria que ya existe: ' || v_department_id;
64             INSERT INTO ERRORES (SEC_ERROR, NOMBRE_PROCESO, MENSAJE)
65             VALUES (v_sec_error, v_nombre_proceso, v_mensaje);
66
67         WHEN OTHERS THEN
68             -- Detectamos si el error es por valores nulos en campos obligatorios
69             SELECT NVL(MAX(SEC_ERROR), 0) + 1 INTO v_sec_error FROM ERRORES;
70             v_nombre_proceso := 'Bloque PL/SQL Inserta Departamento: Error al insertar.';
71             IF v_department_name IS NULL THEN
72                 v_mensaje := 'ORA-01400: cannot insert NULL into ("HR"."DEPARTMENTS"."DEPARTMENT_NAME")';
73             ELSE
74                 v_mensaje := 'Error desconocido: ' || SQLERRM;
75             END IF;
76             INSERT INTO ERRORES (SEC_ERROR, NOMBRE_PROCESO, MENSAJE)
77             VALUES (v_sec_error, v_nombre_proceso, v_mensaje);
78     END;
79
80     -- Realizamos un commit después de toda la transacción
81     COMMIT;
82
83 END;
84
85
86 SELECT * FROM ERRORES;
```


Requerimiento 5

Se deben categorizar a los empleados con identificación 145 al 179, de acuerdo con su porcentaje de comisión. Para ello es necesario crear un bloque PL/SQL que muestre la identificación del empleado, su porcentaje de comisión, seguido del mensaje que corresponda, según el valor del porcentaje de comisión que posee:

- Si el porcentaje de comisión es mayor a 0.3, el mensaje debe ser: “Es un buen porcentaje de comisión, no debe aumentar”.
- Si el porcentaje de comisión está entre 0.2 y 0.3 el mensaje debe ser: “Es un porcentaje de comisión normal. Se debe evaluar un reajuste”.
- Si el porcentaje de comisión es menor a 0.2 el mensaje debe ser: “El porcentaje de comisión es bajo. Debe aumentar en un 10%”.

Se debe mostrar la información en el siguiente formato:

| |
|---|
| La comisión actual del empleado 145 es de ,4. Es un buen porcentaje de comisión, no debe aumentar |
| La comisión actual del empleado 146 es de ,3. Es un porcentaje de comisión normal. Se debe evaluar un reajuste |
| La comisión actual del empleado 147 es de ,3. Es un porcentaje de comisión normal. Se debe evaluar un reajuste |
| La comisión actual del empleado 148 es de ,3. Es un porcentaje de comisión normal. Se debe evaluar un reajuste |
| La comisión actual del empleado 149 es de ,2. Es un porcentaje de comisión normal. Se debe evaluar un reajuste |
| La comisión actual del empleado 150 es de ,3. Es un porcentaje de comisión normal. Se debe evaluar un reajuste |
| La comisión actual del empleado 151 es de ,25. Es un porcentaje de comisión normal. Se debe evaluar un reajuste |
| La comisión actual del empleado 152 es de ,25. Es un porcentaje de comisión normal. Se debe evaluar un reajuste |
| La comisión actual del empleado 153 es de ,2. Es un porcentaje de comisión normal. Se debe evaluar un reajuste |
| La comisión actual del empleado 154 es de ,2. Es un porcentaje de comisión normal. Se debe evaluar un reajuste |
| La comisión actual del empleado 155 es de ,15. El porcentaje de comisión es bajo. Debe aumentar en un 10% |
| La comisión actual del empleado 156 es de ,35. Es un buen porcentaje de comisión, no debe aumentar |
| La comisión actual del empleado 157 es de ,35. Es un buen porcentaje de comisión, no debe aumentar |
| La comisión actual del empleado 158 es de ,35. Es un buen porcentaje de comisión, no debe aumentar |

**** Imagen referencial**

HR.sql

Hoja de Trabajo de SQL | Historial

Hoja de Trabajo | Generador de Consultas

```
--DESARROLLO REQUERIMIENTO 5
DECLARE
    v_emp_id employees.employee_id%TYPE;
    v_comision employees.commission_pct%TYPE;

BEGIN
    FOR i IN 145 .. 179 LOOP

        SELECT employee_id, commission_pct
        INTO v_emp_id, v_comision
        FROM employees
        WHERE employee_id = i;

        IF v_comision > 0.3 THEN
            DBMS_OUTPUT.PUT_LINE('La comisión actual del empleado ' || v_emp_id || ' es de ' || v_comision || '. Es un buen porcentaje de comisión, no debe aumentar');
        ELSIF v_comision >= 0.2 AND v_comision <= 0.3 THEN
            DBMS_OUTPUT.PUT_LINE('La comisión actual del empleado ' || v_emp_id || ' es de ' || v_comision || '. Es un porcentaje de comisión normal. Se debe evaluar un reajuste');
        ELSE
            DBMS_OUTPUT.PUT_LINE('La comisión actual del empleado ' || v_emp_id || ' es de ' || v_comision || '. El porcentaje de comisión es bajo. Se debe evaluar un reajuste del 10%');
        END IF;
    END LOOP;
END;
```

Salida de Script x

Tarea terminada en 0,496 segundos

```
La comisión actual del empleado 145 es de ,4. Es un buen porcentaje de comisión, no debe aumentar
La comisión actual del empleado 146 es de ,3. Es un porcentaje de comisión normal. Se debe evaluar un reajuste
La comisión actual del empleado 147 es de ,3. Es un porcentaje de comisión normal. Se debe evaluar un reajuste
La comisión actual del empleado 148 es de ,3. Es un porcentaje de comisión normal. Se debe evaluar un reajuste
La comisión actual del empleado 149 es de ,2. Es un porcentaje de comisión normal. Se debe evaluar un reajuste
La comisión actual del empleado 150 es de ,3. Es un porcentaje de comisión normal. Se debe evaluar un reajuste
La comisión actual del empleado 151 es de ,25. Es un porcentaje de comisión normal. Se debe evaluar un reajuste
La comisión actual del empleado 152 es de ,25. Es un porcentaje de comisión normal. Se debe evaluar un reajuste
La comisión actual del empleado 153 es de ,2. Es un porcentaje de comisión normal. Se debe evaluar un reajuste
La comisión actual del empleado 154 es de ,2. Es un porcentaje de comisión normal. Se debe evaluar un reajuste
La comisión actual del empleado 155 es de ,15. El porcentaje de comisión es bajo. Se debe evaluar un reajuste del 10%
La comisión actual del empleado 156 es de ,35. Es un buen porcentaje de comisión, no debe aumentar
La comisión actual del empleado 157 es de ,35. Es un buen porcentaje de comisión, no debe aumentar
La comisión actual del empleado 158 es de ,35. Es un buen porcentaje de comisión, no debe aumentar
La comisión actual del empleado 159 es de ,3. Es un porcentaje de comisión normal. Se debe evaluar un reajuste
La comisión actual del empleado 160 es de ,3. Es un porcentaje de comisión normal. Se debe evaluar un reajuste
La comisión actual del empleado 161 es de ,25. Es un porcentaje de comisión normal. Se debe evaluar un reajuste
La comisión actual del empleado 162 es de ,25. Es un porcentaje de comisión normal. Se debe evaluar un reajuste
La comisión actual del empleado 163 es de ,15. El porcentaje de comisión es bajo. Se debe evaluar un reajuste del 10%
```

```
1  --DESARROLLO REQUERIMIENTO 5
2  DECLARE
3      v_emp_id employees.employee_id%TYPE;
4      v_comision employees.commission_pct%TYPE;
5
6  BEGIN
7
8      FOR i IN 145 .. 179 LOOP
9
10         SELECT employee_id, commission_pct
11         INTO v_emp_id, v_comision
12         FROM employees
13         WHERE employee_id = i;
14
15         IF v_comision > 0.3 THEN
16             DBMS_OUTPUT.PUT_LINE('La comisión actual del empleado ' || v_emp_id || ' es de ' || v_comision || '. Es un buen porcentaje de comisión, no debe aumentar');
17         ELSIF v_comision >= 0.2 AND v_comision <= 0.3 THEN
18             DBMS_OUTPUT.PUT_LINE('La comisión actual del empleado ' || v_emp_id || ' es de ' || v_comision || '. Es un porcentaje de comisión normal. Se debe evaluar un reajuste');
19         ELSE
20             DBMS_OUTPUT.PUT_LINE('La comisión actual del empleado ' || v_emp_id || ' es de ' || v_comision || '. El porcentaje de comisión es bajo. Se debe evaluar un reajuste del 10%');
21         END IF;
22     END LOOP;
23
24
25 END;
```

PAUTA DE EVALUACIÓN

| Criterios de Evaluación | Indicadores de Logro | Puntaje |
|--|---|---------|
| 1. Evalúa la lógica de negocio considerando restricciones del lenguaje, requisitos de la lógica de negocios, requisitos de información y sistema de gestión de base de datos para solucionar los requerimientos de información planteados. 2. Utiliza los componentes básicos de un bloque PL/SQL para solucionar los requerimientos de información planteados. 3. Utiliza variables de tipo escalar que permitan almacenar y manipular datos para solucionar los requerimientos de información planteados. 4. Utiliza sentencias y Funciones SQL para solucionar los requerimientos de información planteados. 5. Utiliza operadores PL/SQL lógicos, de comparación, matemáticos, concatenación, de control de orden de las operaciones y exponenciales para solucionar los requerimientos de información planteados. | Declara correctamente las variables a utilizar en los distintos bloques PL/SQL. | 20 |
| | Aplica y construye adecuadamente la estructura de control iterativa en PL/SQL, de acuerdo con los requerimientos. | 20 |
| | Aplica y construye adecuadamente la estructura de control condicional en PL/SQL, de acuerdo con los requerimientos. | 20 |
| | Maneja adecuadamente las excepciones solicitadas en el requerimiento. | 20 |
| | Imprime los resultados con la misma estructura de ejemplo en los requerimientos. | 20 |
| Puntaje Total | | 100 |

PAUTA DE AUTOEVALUACIÓN:

Estimado estudiante:

La autoevaluación es la valoración y/o reflexión que usted realiza sobre su proceso de aprendizaje. Este ejercicio es fundamental para identificar tanto debilidades como fortalezas de la evaluación realizada.

Para realizar la autoevaluación usted deberá contestar las siguientes preguntas, las cuales se encontrarán en la pestaña AUTOEVALUACIÓN, donde deberá completar cada una de ellas.

Conteste con toda la sinceridad posible, ya que será un insumo para la retroalimentación que realizará posteriormente el docente.

- ¿Qué hice bien? – Generar el desarrollo de lo solicitado.
- ¿En qué necesito ayuda? – En el desarrollo de la lógica
- ¿Sobre qué quiero saber más? – En la estructura adecuada de las BD
- ¿Qué aprendí en esta evaluación? – Aplicación de: bloques, excepciones, impresión de estructuras, controles itinerarios.
- ¿Qué debo mejorar? – En el ordenamiento de la estructura de la BD
- ¿Qué acciones realizaré para mejorar? – Lectura y material relacionado a elaboración de bases.



4 INSTITUCION
ACREDITADA
NIVEL AVANZADO
AÑOS Hasta octubre 2025



GESTIÓN INSTITUCIONAL Y DOCENCIA DE PREGRADO