Aquí tienes un documento completo con varios ejemplos comentados y listos para copiar en Visual Studio Code o en Google Docs. Los ejemplos están diseñados de acuerdo con los temas de tu examen: función (BD Botiga), procedimiento y paquete (BD HR). Puedes ajustar nombres, comentarios o parámetros si es necesario.

CHEAT SHEET: PL/SQL - Funciones, Procedimientos y Paquetes

PARTE 1: FUNCIONES (BD Botiga)

Ejemplo 1.1: Contar Empleados de un Departamento

Esta función recibe un código de departamento y devuelve la cantidad de empleados en ese departamento.

```
-- Función: contar_empleados
CREATE OR REPLACE FUNCTION contar_empleados(p_dept_id IN NUMBER) RETURN
NUMBER IS
   v_total NUMBER;  -- Variable para guardar el número total de empleados
BEGIN
   SELECT COUNT(*)
     INTO v_total
     FROM employees
    WHERE department_id = p_dept_id;
   RETURN v_total;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RETURN 0;
   WHEN OTHERS THEN
      RETURN -1;  -- Se retorna -1 en caso de error
END;
```

Notas:

- Se utiliza `CREATE OR REPLACE FUNCTION` para definir la función.
- El parámetro `p_dept_id` es de solo entrada (IN).

- Se emplea `SELECT COUNT(*) INTO v_total` para obtener el resultado.
- Se captura la excepción para no bloquear la ejecución del código en caso de error.

Ejemplo 1.2: Calcular Bonus de Vendedor

Esta función calcula el bonus de un vendedor en función de su cargo, el año de venta, y el número total de ventas. Solo calcula si el cargo es "Sales Manager" o "Sales Representative" y el año está dentro de un rango válido.

```
-- Función: bonus_vendedor
CREATE OR REPLACE FUNCTION bonus_vendedor(
    p_employee_id NUMBER,  -- ID del vendedor
    p_year        NUMBER   -- Año de las ventas
) RETURN NUMBER IS
    v_job_title    VARCHAR2(100);  -- Almacena el cargo del vendedor
    v_total_sales  NUMBER;         -- Total de ventas realizadas
    v_current_year NUMBER := EXTRACT(YEAR FROM SYSDATE);  -- Año actual
    v_bonus        NUMBER;         -- Calcula el bonus
BEGIN
    -- 1. Obtener el título del trabajo del empleado
    SELECT job_title INTO v_job_title
      FROM employees
     WHERE employee_id = p_employee_id;

    -- 2. Verificar si el cargo es adecuado
    IF v_job_title NOT IN ('Sales Manager', 'Sales Representative') THEN
        RETURN NULL;
    END IF;

    -- 3. Validar que el año esté en el rango permitido
    IF p_year < 2000 OR p_year > v_current_year THEN
        RETURN NULL;
    END IF;

    -- 4. Contar las ventas en el año indicado
    SELECT COUNT(*)
      INTO v_total_sales
      FROM orders
     WHERE salesman_id = p_employee_id
       AND EXTRACT(YEAR FROM order_date) = p_year;

    -- 5. Calcular el bonus
    v_bonus := v_total_sales * 150;  -- 150 por cada venta
```

```
   RETURN v_bonus;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      RETURN NULL;
   WHEN OTHERS THEN
      RETURN NULL;
END;
```

Notas:

- Se comprueba el cargo del vendedor antes de proceder.
- Se verifica que el año se encuentre entre 2000 y el año actual.
- Se utiliza el multiplicador 150 para el cálculo del bonus.

PARTE 2: PROCEDIMIENTOS (BD HR)

Ejemplo 2.1: Actualizar Salario de un Empleado

Este procedimiento actualiza el salario de un empleado sumándole un incremento dado. También realiza un COMMIT para salvar la transacción.

```
-- Procedimiento: actualizar_salario
CREATE OR REPLACE PROCEDURE actualizar_salario(
   p_emp_id    IN NUMBER,  -- ID del empleado
   p_increment IN NUMBER   -- Monto a incrementar
) IS
BEGIN
   UPDATE employees
      SET salary = salary + p_increment
    WHERE employee_id = p_emp_id;

   COMMIT;  -- Confirmar la transacción
EXCEPTION
   WHEN OTHERS THEN
      ROLLBACK;  -- Si ocurre un error, se revierte la transacción
      DBMS_OUTPUT.PUT_LINE('Error al actualizar el salario del empleado ' || p_emp_id);
END;
```

Notas:

- Se usa `UPDATE` para modificar el salario.
- Se aplica `COMMIT` después de la actualización.
- Existe manejo básico de errores con `EXCEPTION`.

Ejemplo 2.2: Listar Empleados por Departamento

Este procedimiento muestra, usando un cursor, todos los empleados de un departamento. Se verifica que el departamento exista y se despliegan los datos mediante `DBMS_OUTPUT`.

```
-- Procedimiento: listar_empleados_departamento
CREATE OR REPLACE PROCEDURE listar_empleados_departamento(p_dept_id IN
NUMBER) IS
   v_dept_name departments.department_name%TYPE;
   v_emp_count NUMBER;

   CURSOR cur_empleados IS
      SELECT e.first_name, e.last_name,
           TO_CHAR(e.hire_date, 'DD-MM-YYYY') AS hire_date,
           j.job_title
       FROM employees e
       JOIN jobs j ON e.job_id = j.job_id
      WHERE e.department_id = p_dept_id;
BEGIN
   -- Verificar que el parámetro no sea NULL
   IF p_dept_id IS NULL THEN
      DBMS_OUTPUT.PUT_LINE('El departamento no puede ser NULL.');
      RETURN;
   END IF;

   -- Comprobar si existe el departamento
   BEGIN
      SELECT department_name INTO v_dept_name
       FROM departments
      WHERE department_id = p_dept_id;
   EXCEPTION
      WHEN NO_DATA_FOUND THEN
         DBMS_OUTPUT.PUT_LINE('No existe el departamento con ID ' || p_dept_id);
         RETURN;
   END;

   DBMS_OUTPUT.PUT_LINE('Departamento: ' || v_dept_name);
   DBMS_OUTPUT.PUT_LINE('--------------------------------');
```

```
    -- Contar número de empleados en el departamento
    SELECT COUNT(*)
      INTO v_emp_count
      FROM employees
     WHERE department_id = p_dept_id;

    IF v_emp_count = 0 THEN
        DBMS_OUTPUT.PUT_LINE('El departamento no tiene empleados.');
    ELSE
        -- Iterar y mostrar los empleados
        FOR emp IN cur_empleados LOOP
            DBMS_OUTPUT.PUT_LINE(emp.first_name || ' ' || emp.last_name || ' | ' ||
                                 emp.hire_date || ' | ' || emp.job_title);
        END LOOP;
    END IF;
END;
```

Notas:

- Se valida que el parámetro no sea `NULL`.
- Se utiliza un bloque anidado para obtener el nombre del departamento y capturar una excepción en caso de que no exista.
- El cursor `cur_empleados` recorre y muestra cada registro, formateado a la salida con `DBMS_OUTPUT.PUT_LINE`.

PARTE 3: PAQUETES (BD HR)

Agrupar funciones y procedimientos en un paquete facilita la organización del código.

Ejemplo 3.1: Paquete para Gestión de Empleados

Este paquete agrupa dos funcionalidades: incrementar el salario y obtener el salario actual de un empleado.

Especificación del Paquete

```
-- Especificación del paquete: gestio_empleats
CREATE OR REPLACE PACKAGE gestio_empleats AS
    -- Procedimiento para incrementar el salario
    PROCEDURE incrementar_salario(
        p_emp_id    IN NUMBER,
```

```
    p_increment IN NUMBER
  );

  -- Función para obtener el salario actual del empleado
  FUNCTION obtener_salario(
    p_emp_id IN NUMBER
  ) RETURN NUMBER;
END gestio_empleats;
/
```

Cuerpo del Paquete

```
-- Cuerpo del paquete: gestio_empleats
CREATE OR REPLACE PACKAGE BODY gestio_empleats AS
  PROCEDURE incrementar_salario(
    p_emp_id    IN NUMBER,
    p_increment IN NUMBER
  ) IS
  BEGIN
    UPDATE employees
      SET salary = salary + p_increment
     WHERE employee_id = p_emp_id;
    COMMIT;
  EXCEPTION
    WHEN OTHERS THEN
      ROLLBACK;
      DBMS_OUTPUT.PUT_LINE('Error incrementando salario para el empleado ' ||
p_emp_id);
  END incrementar_salario;

  FUNCTION obtener_salario(
    p_emp_id IN NUMBER
  ) RETURN NUMBER IS
    v_salary employees.salary%TYPE;
  BEGIN
    SELECT salary
      INTO v_salary
      FROM employees
     WHERE employee_id = p_emp_id;
    RETURN v_salary;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      RETURN NULL;
    WHEN OTHERS THEN
      RETURN -1;
  END obtener_salario;
END gestio_empleats;
/
```

Notas:

- En la especificación se definen las “firmas” públicas: el procedimiento `incrementar_salario` y la función `obtener_salario`.
- En el cuerpo se implementa la lógica:
    - El procedimiento actualiza y confirma la operación con `COMMIT`.
    - La función obtiene y retorna el salario, controlando excepciones.
- Asegúrate de respetar la firma declarada en la especificación.

Consejos para el Examen

- **Copia y pega el contenido:** Puedes copiar todo este documento a tu editor favorito (Visual Studio Code o Google Docs) para tenerlo a mano durante el examen.
- **Entiende cada línea:** No te limites a memorizar; trata de comprender la lógica para poder adaptar o reestructurar el código si te lo piden.
- **Revisa errores de sintaxis:** Asegúrate de que las barras `/` y puntos y coma (`;`) estén en los lugares correctos.
- **Prueba mentalmente el flujo:** Antes de copiar en el examen, revisa que cada bloque (función, procedimiento, paquete) tenga su parte declarativa, ejecutable y manejo de excepciones adecuado.

Copia este documento en tu entorno de consulta y utiliza los ejemplos como referencia para comprender y repasar la estructura y la lógica de cada ejercicio. ¡Mucho éxito en tu examen!