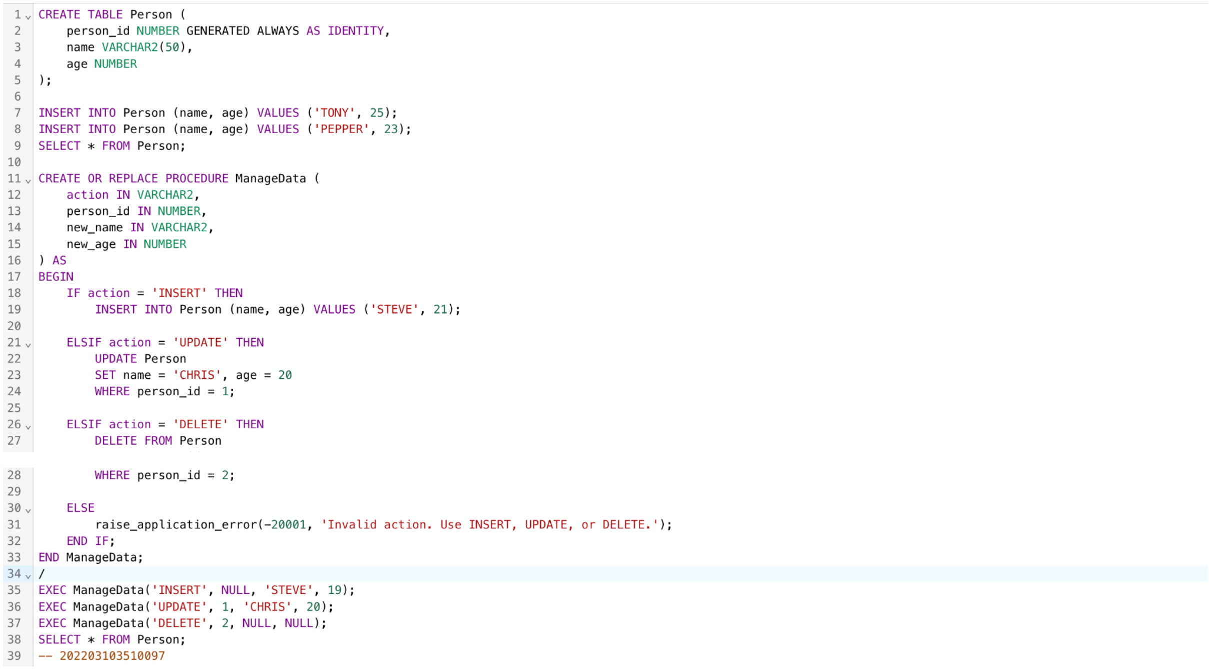
Practical No. 14

Aim: Programs development using creation of procedures, passing parameters IN and OUT of PROCEDURES

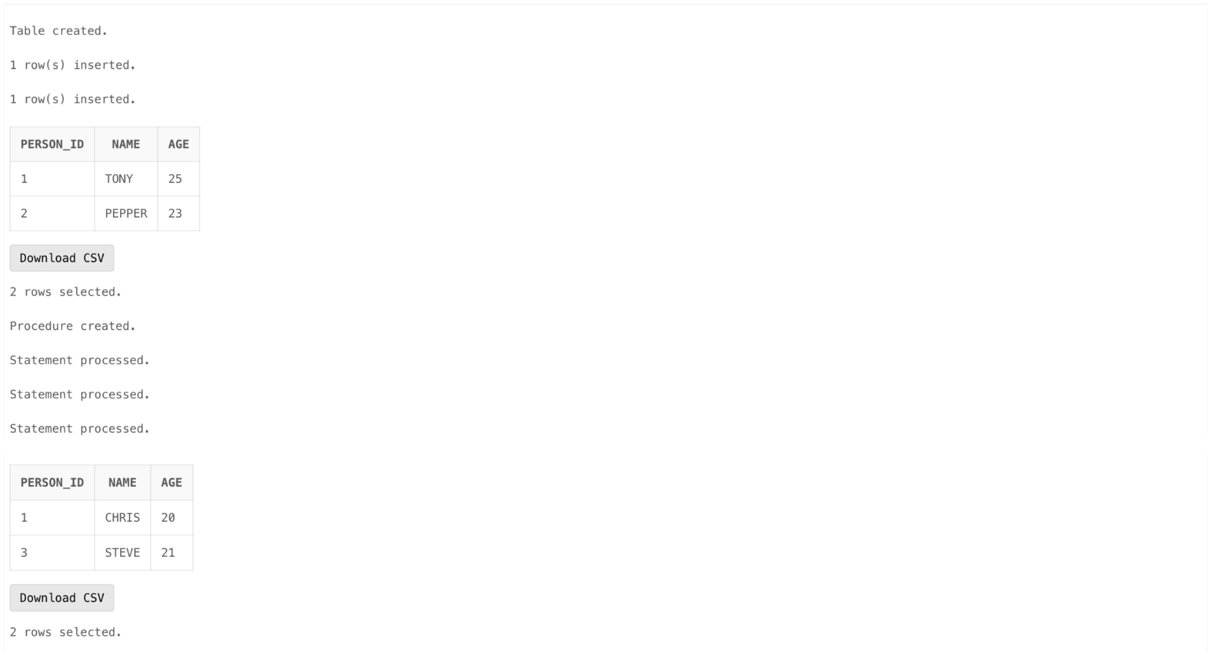
Theory:
Procedures are essential for creating modular and reusable code. They encapsulate a series of statements into a single unit, which can be called and executed independently. Procedures often take parameters, both as inputs (IN) and outputs (OUT), to allow data to be passed in and out of the procedure.

Queries:

1) Create one Stored Procedure to Insert, Delete and Update into one table.

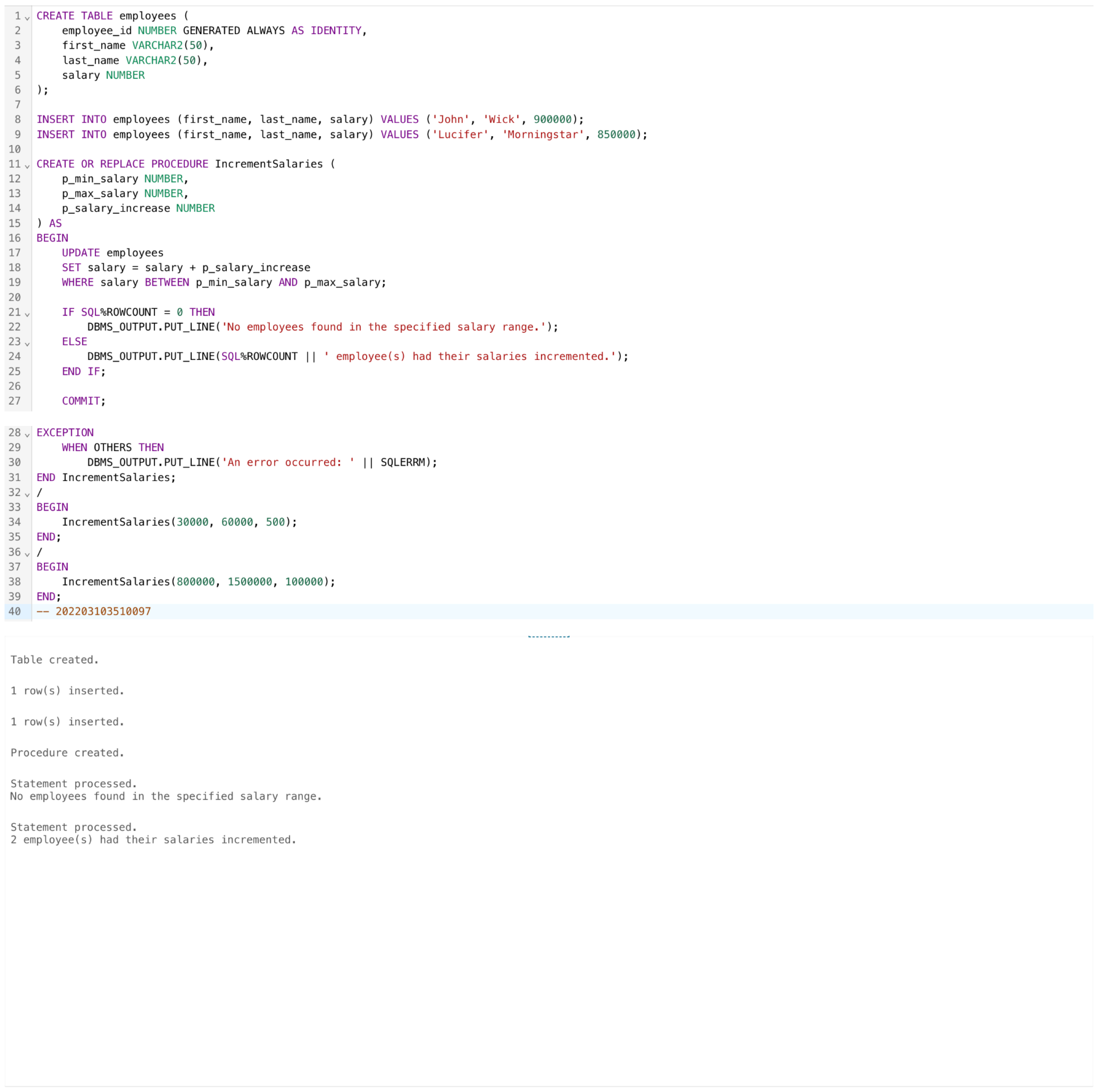


```sql
CREATE TABLE Person (
    person_id NUMBER GENERATED ALWAYS AS IDENTITY,
    name VARCHAR2(50),
    age NUMBER
);

INSERT INTO Person (name, age) VALUES ('TONY', 25);
INSERT INTO Person (name, age) VALUES ('PEPPER', 23);
SELECT * FROM Person;

CREATE OR REPLACE PROCEDURE ManageData (
    action IN VARCHAR2,
    person_id IN NUMBER,
    new_name IN VARCHAR2,
    new_age IN NUMBER
) AS
BEGIN
    IF action = 'INSERT' THEN
        INSERT INTO Person (name, age) VALUES ('STEVE', 21);

    ELSIF action = 'UPDATE' THEN
        UPDATE Person
        SET name = 'CHRIS', age = 20
        WHERE person_id = 1;

    ELSIF action = 'DELETE' THEN
        DELETE FROM Person
        WHERE person_id = 2;

    ELSE
        raise_application_error(-20001, 'Invalid action. Use INSERT, UPDATE, or DELETE.');
    END IF;
END ManageData;
/
EXEC ManageData('INSERT', NULL, 'STEVE', 19);
EXEC ManageData('UPDATE', 1, 'CHRIS', 20);
EXEC ManageData('DELETE', 2, NULL, NULL);
SELECT * FROM Person;
-- 202203103510097
```



```
Table created.

1 row(s) inserted.

1 row(s) inserted.

PERSON_ID   NAME     AGE
1           TONY     25
2           PEPPER   23

Download CSV

2 rows selected.

Procedure created.

Statement processed.

Statement processed.

Statement processed.

PERSON_ID   NAME    AGE
1           CHRIS   20
3           STEVE   21

Download CSV

2 rows selected.
```

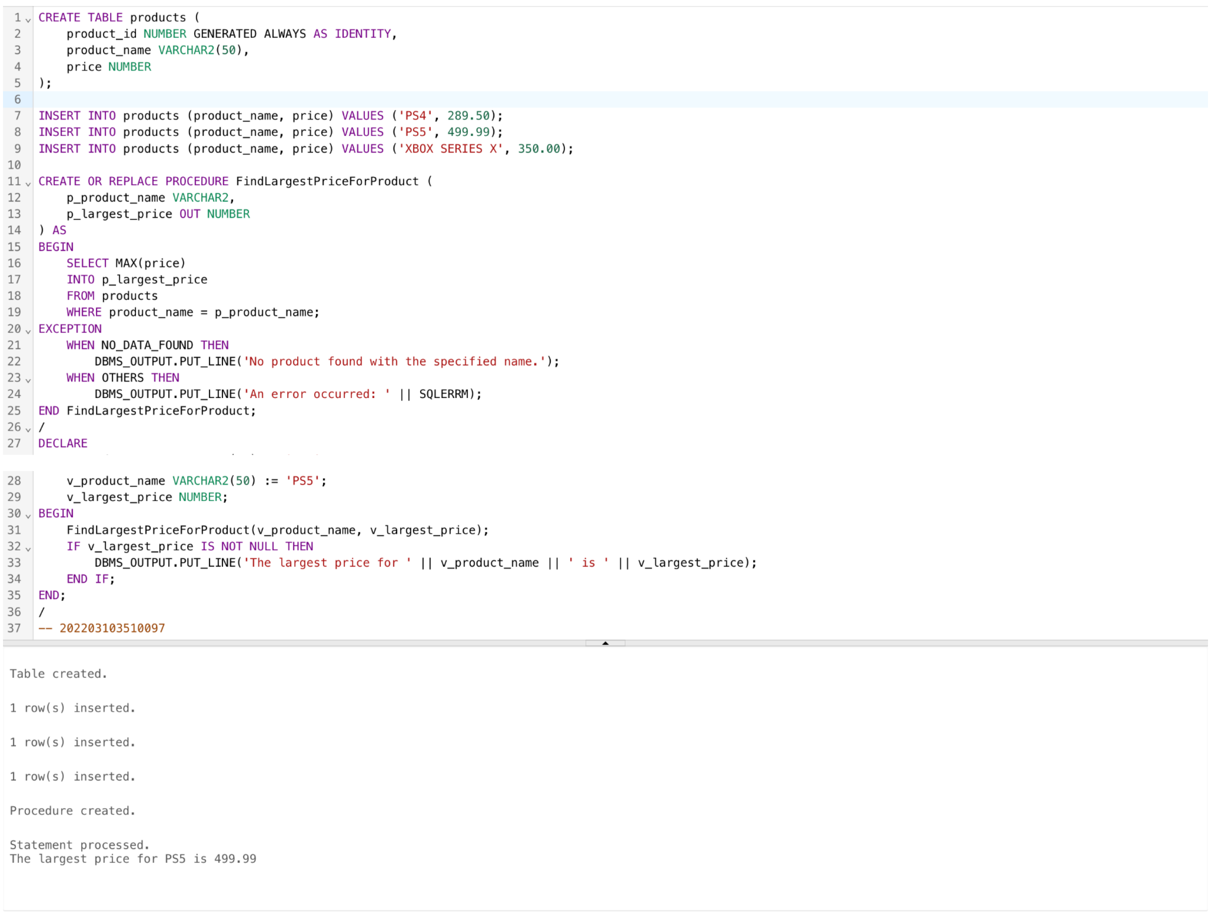
2) Create one Stored Procedure to increment salary with salary range limit.



```
CREATE TABLE employees (
    employee_id NUMBER GENERATED ALWAYS AS IDENTITY,
    first_name VARCHAR2(50),
    last_name VARCHAR2(50),
    salary NUMBER
);

INSERT INTO employees (first_name, last_name, salary) VALUES ('John', 'Wick', 900000);
INSERT INTO employees (first_name, last_name, salary) VALUES ('Lucifer', 'Morningstar', 850000);

CREATE OR REPLACE PROCEDURE IncrementSalaries (
    p_min_salary NUMBER,
    p_max_salary NUMBER,
    p_salary_increase NUMBER
) AS
BEGIN
    UPDATE employees
    SET salary = salary + p_salary_increase
    WHERE salary BETWEEN p_min_salary AND p_max_salary;

    IF SQL%ROWCOUNT = 0 THEN
        DBMS_OUTPUT.PUT_LINE('No employees found in the specified salary range.');
    ELSE
        DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT || ' employee(s) had their salaries incremented.');
    END IF;

    COMMIT;

EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END IncrementSalaries;
/
BEGIN
    IncrementSalaries(30000, 60000, 500);
END;
/
BEGIN
    IncrementSalaries(800000, 1500000, 100000);
END;
-- 202203103510097
```

```
Table created.

1 row(s) inserted.

1 row(s) inserted.

Procedure created.

Statement processed.
No employees found in the specified salary range.

Statement processed.
2 employee(s) had their salaries incremented.
```

3) Create one Stored Procedure to find largest price with that product name.



```
CREATE TABLE products (
    product_id NUMBER GENERATED ALWAYS AS IDENTITY,
    product_name VARCHAR2(50),
    price NUMBER
);

INSERT INTO products (product_name, price) VALUES ('PS4', 289.50);
INSERT INTO products (product_name, price) VALUES ('PS5', 499.99);
INSERT INTO products (product_name, price) VALUES ('XBOX SERIES X', 350.00);

CREATE OR REPLACE PROCEDURE FindLargestPriceForProduct (
    p_product_name VARCHAR2,
    p_largest_price OUT NUMBER
) AS
BEGIN
    SELECT MAX(price)
    INTO p_largest_price
    FROM products
    WHERE product_name = p_product_name;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No product found with the specified name.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END FindLargestPriceForProduct;
/
DECLARE
    v_product_name VARCHAR2(50) := 'PS5';
    v_largest_price NUMBER;
BEGIN
    FindLargestPriceForProduct(v_product_name, v_largest_price);
    IF v_largest_price IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE('The largest price for ' || v_product_name || ' is ' || v_largest_price);
    END IF;
END;
/
-- 202203103510097
```

```
Table created.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

Procedure created.

Statement processed.
The largest price for PS5 is 499.99
```

Conclusion: The use of procedures with IN and OUT parameters is a fundamental practice in program development, allowing for modularity, code reusability, and improved maintainability. These procedures are widely used in database systems and general-purpose programming to encapsulate and execute specific tasks. Proper parameter design and documentation are key factors in creating effective procedures.