

**NSHURO ARNAUD NELLIGAN**

**27960**

**PHASE VI**

**Objective:**

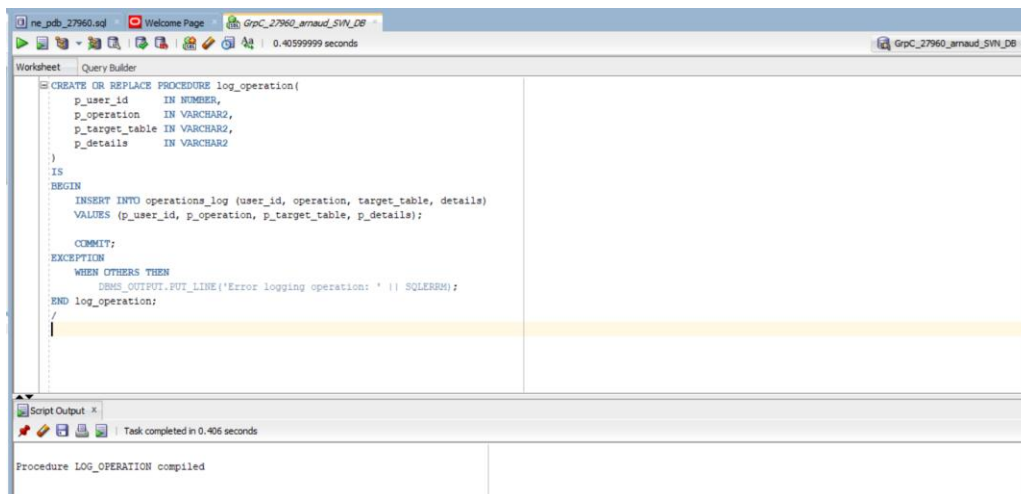
**Develop PL/SQL procedures, functions, packages, and cursors to handle:**

- **Logging database operations**
- **Detecting violations**
- **Sending alerts to administrators**
- **Exception handling**
- **Basic analytics preparation**

## **1. PL/SQL Procedures**

### **Procedure 1: Log Operation**

Logs any user operation in the database.



The screenshot shows the Oracle SQL Developer interface. The main window is the 'Query Builder' tab, displaying the following PL/SQL code:

```
CREATE OR REPLACE PROCEDURE log_operation(  
    p_user_id      IN NUMBER,  
    p_operation    IN VARCHAR2,  
    p_target_table IN VARCHAR2,  
    p_details      IN VARCHAR2  
)  
IS  
BEGIN  
    INSERT INTO operations_log (user_id, operation, target_table, details)  
    VALUES (p_user_id, p_operation, p_target_table, p_details);  
  
    COMMIT;  
  
    EXCEPTION  
    WHEN OTHERS THEN  
        DBMS_OUTPUT.PUT_LINE('Error logging operation: ' || SQLERRM);  
END log_operation;  
/
```

The bottom status bar indicates 'Task completed in 0.406 seconds' and 'Procedure LOG\_OPERATION compiled'.

### **Procedure 2: Notify Administrator**

Sends alerts to admins when a violation occurs.

The screenshot shows the SQL Developer interface with a query editor containing the following PL/SQL code:

```
CREATE OR REPLACE PROCEDURE notify_admin(
  p_violation_id IN NUMBER
)
IS
  v_user      VARCHAR2(50);
  v_details   VARCHAR2(400);
BEGIN
  SELECT u.username, o.details
  INTO v_user, v_details
  FROM violations v
  JOIN operations_log o ON v.log_id = o.log_id
  JOIN users u ON o.user_id = u.user_id
  WHERE v.violation_id = p_violation_id;

  DBMS_OUTPUT.PUT_LINE('ALERT: Violation detected! User: ' || v_user || ', Details: ' || v_details);

  -- Optional: email logic using UTL_MAIL package

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No violation found for ID ' || p_violation_id);
END notify_admin;
```

The Script Output pane at the bottom shows the message: "Task completed in 0.055 seconds" and "Procedure NOTIFY\_ADMIN compiled".

## Procedure 3: Resolve Violation

Marks a violation as resolved.

The screenshot shows the SQL Developer interface with a query editor containing the following PL/SQL code:

```
CREATE OR REPLACE PROCEDURE resolve_violation(
  p_violation_id IN NUMBER
)
IS
BEGIN
  UPDATE violations
  SET status = 'RESOLVED'
  WHERE violation_id = p_violation_id;

  COMMIT;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error resolving violation: ' || SQLERRM);
END resolve_violation;
```

The Script Output pane at the bottom shows the message: "Task completed in 0.037 seconds" and "Procedure RESOLVE\_VIOLATION compiled".

## 2. PL/SQL Functions

### Function 1: Check Violation Status

Returns current status of a violation.

The screenshot shows the SQL Developer interface with a query editor containing the following PL/SQL code:

```
CREATE OR REPLACE FUNCTION get_violation_status(
  p_violation_id IN NUMBER
) RETURN VARCHAR2
IS
  v_status VARCHAR2(20);
BEGIN
  SELECT status INTO v_status
  FROM violations
  WHERE violation_id = p_violation_id;

  RETURN v_status;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RETURN 'NOT FOUND';
END get_violation_status;
```

The Script Output pane at the bottom shows the message: "Task completed in 0.048 seconds" and "Function GET\_VIOLATION\_STATUS compiled".

### Function 2: Count Violations by User

The screenshot shows the SQL Developer interface with a query window titled 'GrpC\_27960\_arnaud\_SVN\_DB'. The 'Query Builder' tab is active, displaying the following PL/SQL code:

```
CREATE OR REPLACE FUNCTION count_user_violations(  
  p_user_id IN NUMBER  
)  
  RETURN NUMBER  
IS  
  v_count NUMBER;  
BEGIN  
  SELECT COUNT(*) INTO v_count  
  FROM violations v  
  JOIN operations_log o ON v.log_id = o.log_id  
  WHERE o.user_id = p_user_id;  
  RETURN v_count;  
END count_user_violations;  
/
```

The 'Script Output' tab at the bottom shows the message: 'Function COUNT\_USER\_VIOLATIONS compiled'.

### 3. Cursors

#### Explicit Cursor: List All Violations

The screenshot shows the SQL Developer interface with a query window titled 'GrpC\_27960\_arnaud\_SVN\_DB'. The 'Query Builder' tab is active, displaying the following PL/SQL code:

```
DECLARE  
  CURSOR c_violations IS  
    SELECT v.violation_id, u.username, v.severity, v.status  
    FROM violations v  
    JOIN operations_log o ON v.log_id = o.log_id  
    JOIN users u ON o.user_id = u.user_id;  
  r_violation c_violations%BOWTYPE;  
BEGIN  
  OPEN c_violations;  
  LOOP  
    FETCH c_violations INTO r_violation;  
    EXIT WHEN c_violations%NOTFOUND;  
    DBMS_OUTPUT.PUT_LINE('Violation ID: ' || r_violation.violation_id ||  
      ', User: ' || r_violation.username ||  
      ', Severity: ' || r_violation.severity ||  
      ', Status: ' || r_violation.status);  
  END LOOP;  
  CLOSE c_violations;  
END;  
/
```

The 'Script Output' tab at the bottom shows the message: 'PL/SQL procedure successfully completed.'

### 4. Package Example

#### Package Specification

The screenshot shows the SQL Developer interface with a query window titled 'GrpC\_27960\_arnaud\_SVN\_DB'. The 'Query Builder' tab is active, displaying the following PL/SQL code:

```
CREATE OR REPLACE PACKAGE pkg_security AS  
  PROCEDURE log_operation(p_user_id NUMBER, p_operation VARCHAR2, p_target_table VARCHAR2, p_details VARCHAR2);  
  PROCEDURE notify_admin(p_violation_id NUMBER);  
  PROCEDURE resolve_violation(p_violation_id NUMBER);  
  FUNCTION get_violation_status(p_violation_id NUMBER) RETURN VARCHAR2;  
  FUNCTION count_user_violations(p_user_id NUMBER) RETURN NUMBER;  
END pkg_security;  
/
```

The 'Script Output' tab at the bottom shows the message: 'Package PKG\_SECURITY compiled'.

#### Package Body

nepdb\_27960.scd Welcome Page GrpC\_27960\_amaud\_SVN\_DB 0.068 seconds

Worksheet Query Builder

```
CREATE OR REPLACE PACKAGE BODY pkg_security AS
PROCEDURE log_operation(p_user_id NUMBER, p_operation VARCHAR2, p_target_table VARCHAR2, p_details VARCHAR2) IS
BEGIN
INSERT INTO operations_log (user_id, operation, target_table, details)
VALUES (p_user_id, p_operation, p_target_table, p_details);
COMMIT;
END;

PROCEDURE notify_admin(p_violation_id NUMBER) IS
v_user VARCHAR2(50);
v_details VARCHAR2(400);
BEGIN
SELECT u.username, o.details INTO v_user, v_details
FROM violations v
JOIN operations_log o ON v.log_id = o.log_id
JOIN users u ON o.user_id = u.user_id
WHERE v.violation_id = p_violation_id;
DBMS_OUTPUT.PUT_LINE('ALERT: Violation detected! User: ' || v_user || ', Details: ' || v_details);
END;

PROCEDURE resolve_violation(p_violation_id NUMBER) IS
BEGIN
UPDATE violations
SET status = 'RESOLVED'
WHERE violation_id = p_violation_id;
COMMIT;
END;

FUNCTION get_violation_status(p_violation_id NUMBER) RETURN VARCHAR2 IS

```

Script Output x

Task completed in 0.068 seconds

Package Body PKG\_SECURITY compiled