-- 1. Write a PL/SQL block to declare two variables – Message and Date\_Written of the type Varchar and Date respectively.

DROP TABLE TEMP;

DELIMITER $

CREATE TABLE TEMP (Message VARCHAR2(100), Date\_Written DATE);

DECLARE

Message VARCHAR2(100) := 'My First PL/SQL program';

Date\_Written DATE := SYSDATE;

BEGIN

INSERT INTO TEMP (Message, Date\_Written) VALUES (Message, Date\_Written);

COMMIT;

END;

DELIMITER ;

-- 2. Write a block to accept values for NUM1 and NUM2 at run time.

DELIMITER $

DECLARE

NUM1 NUMBER;

NUM2 NUMBER;

RESULT NUMBER;

BEGIN

NUM1 := &NUM1;

NUM2 := &NUM2;

RESULT := MOD(NUM1, NUM2);

DBMS\_OUTPUT.PUT\_LINE('The remainder is: ' || RESULT);

END;

DELIMITER ;

-- 3. Build a block that computes the total compensation for one year.

DELIMITER $

DECLARE

annual\_salary NUMBER := &annual\_salary;

annual\_bonus\_percentage NUMBER := &annual\_bonus\_percentage;

total\_compensation NUMBER;

BEGIN

IF annual\_salary IS NULL THEN

annual\_salary := 0;

END IF;

total\_compensation := annual\_salary + (annual\_salary \* annual\_bonus\_percentage / 100);

DBMS\_OUTPUT.PUT\_LINE('Total Compensation: ' || total\_compensation);

END;

DELIMITER ;

-- 4. Create a block that selects the maximum department number from the department table.

DELIMITER $

DECLARE

max\_dept\_no NUMBER;

BEGIN

SELECT MAX(deptno) INTO max\_dept\_no FROM department;

DBMS\_OUTPUT.PUT\_LINE('Maximum Department Number: ' || max\_dept\_no);

END;

DELIMITER ;

-- 5. Create a block that updates the location for an existing department.

DELIMITER $

DROP TABLE department;

CREATE TABLE department (deptno NUMBER, location VARCHAR2(100));

DECLARE

v\_deptno NUMBER := &deptno;

v\_location VARCHAR2(100) := &location;

BEGIN

UPDATE department SET location = v\_location WHERE deptno = v\_deptno;

DBMS\_OUTPUT.PUT\_LINE('Updated Department: ' || v\_deptno);

COMMIT;

END;

DELIMITER ;

-- 6. Create a block that deletes the department created in the previous exercise.

DELIMITER $

DECLARE

v\_deptno NUMBER := &deptno;

v\_rows\_deleted NUMBER;

BEGIN

DELETE FROM department WHERE deptno = v\_deptno;

v\_rows\_deleted := SQL%ROWCOUNT;

DBMS\_OUTPUT.PUT\_LINE(v\_rows\_deleted || ' rows deleted.');

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Department number does not exist.');

END;

DELIMITER ;

-- 7. Create a block to insert numbers into the Temp table.

DELIMITER $

DROP TABLE TEMP;

CREATE TABLE TEMP (number\_column NUMBER);

BEGIN

FOR i IN 1..10 LOOP

IF i NOT IN (6, 8) THEN

INSERT INTO TEMP (number\_column) VALUES (i);

END IF;

END LOOP;

COMMIT;

END;

DELIMITER ;

-- 8. Create a block that computes the commission amount for a given employee.

DELIMITER $

DROP TABLE emp;

CREATE TABLE emp (empno NUMBER, salary NUMBER, commission NUMBER);

DECLARE

empno NUMBER := &empno;

salary NUMBER;

commission\_amount NUMBER;

BEGIN

INSERT INTO emp (empno, salary) VALUES (empno, NULL);

SELECT salary INTO salary FROM emp WHERE empno = empno;

IF salary IS NULL THEN

commission\_amount := 0;

ELSIF salary < 1000 THEN

commission\_amount := salary \* 0.1;

ELSIF salary BETWEEN 1000 AND 1500 THEN

commission\_amount := salary \* 0.15;

ELSE

commission\_amount := salary \* 0.2;

END IF;

DBMS\_OUTPUT.PUT\_LINE('Commission Amount: ' || commission\_amount);

UPDATE emp SET commission = commission\_amount WHERE empno = empno;

COMMIT;

END;

DELIMITER ;

-- 9. Create a block that determines the top employees with respect to salaries.

DELIMITER $

DROP TABLE TEMP;

CREATE TABLE TEMP (name VARCHAR2(100), salary NUMBER);

DECLARE

n NUMBER := &n;

BEGIN

INSERT INTO TEMP (name, salary)

SELECT name, salary

FROM emp

ORDER BY salary DESC

FETCH FIRST n ROWS ONLY;

COMMIT;

END;

DELIMITER ;

-- 10. Create a Procedure to insert, modify and delete a product from the PRODUCT table.

DELIMITER $

DROP PROCEDURE manage\_product;

CREATE PROCEDURE manage\_product (

p\_product\_no IN NUMBER,

p\_description IN VARCHAR2

) AS

BEGIN

INSERT INTO product (product\_no, description) VALUES (p\_product\_no, p\_description);

COMMIT;

END;

DELIMITER ;

-- 11. Create Procedure to query the EMP table.

DELIMITER $

DROP PROCEDURE get\_emp\_details;

CREATE PROCEDURE get\_emp\_details (emp\_id IN NUMBER, sal OUT NUMBER, job OUT VARCHAR2) AS

BEGIN

SELECT salary, job

INTO sal, job

FROM emp

WHERE empno = emp\_id;

END;

DELIMITER ;

-- 12. Create a function to return a product description to a variable.

DELIMITER $

DROP FUNCTION get\_product\_description;

CREATE FUNCTION get\_product\_description (p\_product\_no IN NUMBER) RETURN VARCHAR2 AS

v\_description VARCHAR2(100);

BEGIN

SELECT description INTO v\_description FROM product WHERE product\_no = p\_product\_no;

RETURN v\_description;

END;

DELIMITER ;

-- 13. Create a function Annual compensation to return the annual salary.

DELIMITER $

DROP FUNCTION annual\_compensation;

CREATE FUNCTION annual\_compensation (

p\_monthly\_salary IN NUMBER,

p\_commission IN NUMBER

) RETURN NUMBER AS

v\_annual\_salary NUMBER;

BEGIN

v\_annual\_salary := NVL(p\_monthly\_salary, 0) \* 12 + NVL(p\_commission, 0) \* 12;

RETURN v\_annual\_salary;

END;

DELIMITER ;

-- 14. Create a procedure to insert a new employee into the EMP table.

DELIMITER $

DROP PROCEDURE insert\_employee;

CREATE PROCEDURE insert\_employee (

empno IN NUMBER,

deptno IN NUMBER

) AS

BEGIN

IF NOT Valid\_Deptno(deptno) THEN

DBMS\_OUTPUT.PUT\_LINE('Invalid Department Number');

ELSE

INSERT INTO emp (empno, deptno) VALUES (empno, deptno);

COMMIT;

END IF;

END;

DELIMITER ;

-- Function to Validate Department Number

DELIMITER $

DROP FUNCTION Valid\_Deptno;

CREATE FUNCTION Valid\_Deptno (p\_deptno IN NUMBER) RETURN BOOLEAN AS

v\_count NUMBER;

BEGIN

SELECT COUNT(\*) INTO v\_count FROM dept WHERE deptno = p\_deptno;

RETURN v\_count > 0;

END;

DELIMITER ;

-- 15. Write a program to reverse the string.

DELIMITER $

DECLARE

original\_string VARCHAR2(100) := 'Hello World';

reversed\_string VARCHAR2(100) := '';

BEGIN

FOR i IN REVERSE 1..LENGTH(original\_string) LOOP

reversed\_string := reversed\_string || SUBSTR(original\_string, i, 1);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Reversed String: ' || reversed\_string);

END;

DELIMITER ;

-- 16. Write a program to convert a lowercase string into uppercase string.

DELIMITER $

DECLARE

lower\_string VARCHAR2(100) := 'hello world';

upper\_string VARCHAR2(100);

BEGIN

upper\_string := UPPER(lower\_string);

DBMS\_OUTPUT.PUT\_LINE('Uppercase String: ' || upper\_string);

END;

DELIMITER ;

-- 17. Write a program to find tables of a number entered by the user.

DELIMITER $

DECLARE

num NUMBER := &num;

BEGIN

FOR i IN 1..10 LOOP

DBMS\_OUTPUT.PUT\_LINE(num || ' x ' || i || ' = ' || num \* i);

END LOOP;

END;

DELIMITER ;

-- 18. Write a program to update the salary of an employee.

DELIMITER $

DECLARE

empno NUMBER := &empno;

new\_salary NUMBER := &new\_salary;

BEGIN

UPDATE emp SET salary = new\_salary WHERE empno = empno;

COMMIT;

END;

DELIMITER ;

-- 19. Write a PL/SQL block to satisfy specific salary conditions.

DELIMITER $

DECLARE

empno NUMBER := &empno;

current\_salary NUMBER;

BEGIN

SELECT salary INTO current\_salary FROM emp WHERE empno = empno;

IF current\_salary < 1500 THEN

UPDATE emp SET salary = current\_salary + 1000 WHERE empno = empno;

ELSIF current\_salary = 1500 THEN

UPDATE emp SET salary = current\_salary + 1500 WHERE empno = empno;

ELSIF current\_salary > 1500 THEN

UPDATE emp SET salary = current\_salary + 2500 WHERE empno = empno;

END IF;

COMMIT;

END;

DELIMITER ;