# PL/SQL Homework Solutions

## 1. Basic & Variables

### 1. Calculate Annual Salary

DECLARE  
 v\_basic\_salary NUMBER := 5000; -- Example value  
 v\_bonus NUMBER := 1000; -- Example value  
 v\_annual\_salary NUMBER;  
BEGIN  
 v\_annual\_salary := (v\_basic\_salary \* 12) + v\_bonus;  
 DBMS\_OUTPUT.PUT\_LINE('Annual Salary: ' || v\_annual\_salary);  
END;  
/

### 2. Calculate Average Marks

DECLARE  
 v\_marks1 NUMBER := 85; -- Example values  
 v\_marks2 NUMBER := 90;  
 v\_marks3 NUMBER := 78;  
 v\_average NUMBER;  
BEGIN  
 v\_average := (v\_marks1 + v\_marks2 + v\_marks3) / 3;  
 DBMS\_OUTPUT.PUT\_LINE('Average Marks: ' || ROUND(v\_average, 2));  
END;  
/

## 2. Conditional Statements

### 3. Bank Balance Check (IF-ELSIF)

DECLARE  
 v\_balance NUMBER := 3500; -- Example value  
BEGIN  
 IF v\_balance < 1000 THEN  
 DBMS\_OUTPUT.PUT\_LINE('Low Balance');  
 ELSIF v\_balance BETWEEN 1000 AND 5000 THEN  
 DBMS\_OUTPUT.PUT\_LINE('Sufficient Balance');  
 ELSE  
 DBMS\_OUTPUT.PUT\_LINE('High Balance');  
 END IF;  
END;  
/

### 4. Grading System (CASE)

DECLARE  
 v\_percentage NUMBER := 82; -- Example value  
 v\_grade VARCHAR2(10);  
BEGIN  
 v\_grade := CASE  
 WHEN v\_percentage BETWEEN 90 AND 100 THEN 'A Grade'  
 WHEN v\_percentage BETWEEN 75 AND 89 THEN 'B Grade'  
 WHEN v\_percentage BETWEEN 50 AND 74 THEN 'C Grade'  
 ELSE 'Fail'  
 END;  
 DBMS\_OUTPUT.PUT\_LINE('Grade: ' || v\_grade);  
END;  
/

### 5. Shopping Discount

DECLARE  
 v\_bill\_amount NUMBER := 3000; -- Example value  
 v\_discount NUMBER;  
 v\_final\_bill NUMBER;  
BEGIN  
 IF v\_bill\_amount > 5000 THEN  
 v\_discount := v\_bill\_amount \* 0.20;  
 ELSIF v\_bill\_amount BETWEEN 2000 AND 5000 THEN  
 v\_discount := v\_bill\_amount \* 0.10;  
 ELSE  
 v\_discount := 0;  
 END IF;  
   
 v\_final\_bill := v\_bill\_amount - v\_discount;  
 DBMS\_OUTPUT.PUT\_LINE('Original Bill: ' || v\_bill\_amount);  
 DBMS\_OUTPUT.PUT\_LINE('Discount: ' || v\_discount);  
 DBMS\_OUTPUT.PUT\_LINE('Final Bill: ' || v\_final\_bill);  
END;  
/

## 3. Looping

### 6. Multiplication Table

DECLARE  
 v\_number NUMBER := 7; -- Example number  
 v\_result NUMBER;  
BEGIN  
 FOR i IN 1..10 LOOP  
 v\_result := v\_number \* i;  
 DBMS\_OUTPUT.PUT\_LINE(v\_number || ' x ' || i || ' = ' || v\_result);  
 END LOOP;  
END;  
/

### 7. Print Employee IDs (FOR LOOP)

BEGIN  
 FOR emp\_id IN 100..120 LOOP  
 DBMS\_OUTPUT.PUT\_LINE('Employee ID: ' || emp\_id);  
 END LOOP;  
END;  
/

### 8. Factorial (WHILE LOOP)

DECLARE  
 v\_number NUMBER := 5; -- Example number  
 v\_factorial NUMBER := 1;  
 v\_counter NUMBER := 1;  
BEGIN  
 WHILE v\_counter <= v\_number LOOP  
 v\_factorial := v\_factorial \* v\_counter;  
 v\_counter := v\_counter + 1;  
 END LOOP;  
 DBMS\_OUTPUT.PUT\_LINE('Factorial of ' || v\_number || ' is ' || v\_factorial);  
END;  
/

### 9. Countdown Timer (REVERSE FOR)

BEGIN  
 FOR i IN REVERSE 1..10 LOOP  
 DBMS\_OUTPUT.PUT\_LINE('Countdown: ' || i);  
 END LOOP;  
END;  
/

## 4. Table-Based Scenarios

### 10. IT Department Employees

DECLARE  
 CURSOR c\_emp IS  
 SELECT emp\_name   
 FROM employees   
 WHERE dept\_id = 'IT';  
BEGIN  
 FOR emp\_rec IN c\_emp LOOP  
 DBMS\_OUTPUT.PUT\_LINE('IT Employee: ' || emp\_rec.emp\_name);  
 END LOOP;  
END;  
/

### 11. Salary Increase

DECLARE  
 CURSOR c\_emp IS  
 SELECT emp\_id, salary   
 FROM employees   
 WHERE salary < 3000  
 FOR UPDATE;  
BEGIN  
 FOR emp\_rec IN c\_emp LOOP  
 UPDATE employees   
 SET salary = salary \* 1.10   
 WHERE emp\_id = emp\_rec.emp\_id;  
 END LOOP;  
 COMMIT;  
 DBMS\_OUTPUT.PUT\_LINE('Salaries updated successfully');  
END;  
/

### 12. Above Average Salary Employees

DECLARE  
 v\_avg\_salary NUMBER;  
 CURSOR c\_emp IS  
 SELECT emp\_name, salary   
 FROM employees   
 WHERE salary > v\_avg\_salary;  
BEGIN  
 SELECT AVG(salary) INTO v\_avg\_salary FROM employees;  
   
 FOR emp\_rec IN c\_emp LOOP  
 DBMS\_OUTPUT.PUT\_LINE(emp\_rec.emp\_name || ' - ' || emp\_rec.salary);  
 END LOOP;  
END;  
/

### 13. Employee Earnings Category

DECLARE  
 CURSOR c\_emp IS  
 SELECT emp\_name, salary   
 FROM employees;  
BEGIN  
 FOR emp\_rec IN c\_emp LOOP  
 IF emp\_rec.salary > 8000 THEN  
 DBMS\_OUTPUT.PUT\_LINE(emp\_rec.emp\_name || ' - High Earner');  
 ELSIF emp\_rec.salary BETWEEN 4000 AND 8000 THEN  
 DBMS\_OUTPUT.PUT\_LINE(emp\_rec.emp\_name || ' - Mid Earner');  
 ELSE  
 DBMS\_OUTPUT.PUT\_LINE(emp\_rec.emp\_name || ' - Low Earner');  
 END IF;  
 END LOOP;  
END;  
/

### 14. Department Total Salary

DECLARE  
 CURSOR c\_dept IS  
 SELECT dept\_id, SUM(salary) as total\_salary  
 FROM employees  
 GROUP BY dept\_id;  
BEGIN  
 FOR dept\_rec IN c\_dept LOOP  
 DBMS\_OUTPUT.PUT\_LINE('Dept ' || dept\_rec.dept\_id ||   
 ' - Total Salary: ' || dept\_rec.total\_salary);  
 END LOOP;  
END;  
/

## 5. Challenge Level

### 15. Fibonacci Sequence

DECLARE  
 n NUMBER := 10; -- Example: first 10 terms  
 a NUMBER := 0;  
 b NUMBER := 1;  
 temp NUMBER;  
 i NUMBER := 1;  
BEGIN  
 DBMS\_OUTPUT.PUT\_LINE('Fibonacci Sequence:');  
   
 WHILE i <= n LOOP  
 IF i = 1 THEN  
 DBMS\_OUTPUT.PUT(a || ' ');  
 ELSIF i = 2 THEN  
 DBMS\_OUTPUT.PUT(b || ' ');  
 ELSE  
 temp := a + b;  
 a := b;  
 b := temp;  
 DBMS\_OUTPUT.PUT(b || ' ');  
 END IF;  
 i := i + 1;  
 END LOOP;  
 DBMS\_OUTPUT.NEW\_LINE;  
END;  
/

### 16. Bank Transaction Processing

DECLARE  
 v\_balance NUMBER := 10000; -- Initial balance  
 CURSOR c\_txn IS  
 SELECT amount, type   
 FROM transactions   
 ORDER BY txn\_id;  
BEGIN  
 FOR txn\_rec IN c\_txn LOOP  
 IF txn\_rec.type = 'CREDIT' THEN  
 v\_balance := v\_balance + txn\_rec.amount;  
 ELSIF txn\_rec.type = 'DEBIT' THEN  
 v\_balance := v\_balance - txn\_rec.amount;  
 END IF;  
 END LOOP;  
   
 DBMS\_OUTPUT.PUT\_LINE('Final Account Balance: ' || v\_balance);  
END;  
/

### 17. Employee Details Procedure

CREATE OR REPLACE PROCEDURE get\_employee\_details(  
 p\_emp\_id IN employees.emp\_id%TYPE  
)  
IS  
 v\_emp\_name employees.emp\_name%TYPE;  
 v\_dept\_name VARCHAR2(50);  
 v\_salary employees.salary%TYPE;  
BEGIN  
 SELECT e.emp\_name, d.dept\_name, e.salary  
 INTO v\_emp\_name, v\_dept\_name, v\_salary  
 FROM employees e  
 JOIN departments d ON e.dept\_id = d.dept\_id  
 WHERE e.emp\_id = p\_emp\_id;  
   
 DBMS\_OUTPUT.PUT\_LINE('Employee Name: ' || v\_emp\_name);  
 DBMS\_OUTPUT.PUT\_LINE('Department: ' || v\_dept\_name);  
 DBMS\_OUTPUT.PUT\_LINE('Salary: ' || v\_salary);  
   
EXCEPTION  
 WHEN NO\_DATA\_FOUND THEN  
 DBMS\_OUTPUT.PUT\_LINE('Employee not found');  
 WHEN OTHERS THEN  
 DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);  
END;  
/  
  
-- To execute the procedure:  
-- BEGIN  
-- get\_employee\_details(100);  
-- END;  
/