PL/SQL Cheat Sheet (No Joins)

SQL Basics (No JOINs)

```
-- Create Table
CREATE TABLE employees (
    emp_id NUMBER PRIMARY KEY,
    name VARCHAR2(100),
    salary NUMBER,
    dept_id NUMBER
);

-- Insert Data
INSERT INTO employees (emp_id, name, salary, dept_id)
VALUES (1, 'Alice', 5000, 10);

-- Update Data
UPDATE employees
SET salary = 5500
WHERE emp_id = 1;

-- Delete Data
DELETE FROM employees
WHERE emp_id = 1;

-- Select Data
SELECT * FROM employees WHERE dept_id = 10;

-- Order By
SELECT name, salary FROM employees ORDER BY salary DESC;

-- Aggregate Functions
SELECT COUNT(*), MAX(salary), MIN(salary), AVG(salary)
FROM employees;

-- WHERE + AND/OR
SELECT * FROM employees
WHERE dept_id = 10 AND salary > 4000;
```

PL/SQL Blocks

```
DECLARE
    v_salary NUMBER;
BEGIN
    SELECT salary INTO v_salary
    FROM employees
    WHERE emp_id = 1;

    DBMS_OUTPUT.PUT_LINE('Salary: ' || v_salary);
END;
```

PL/SQL Cheat Sheet (No Joins)

Functions

```
CREATE OR REPLACE FUNCTION get_bonus(p_salary NUMBER)
RETURN NUMBER
IS
BEGIN
    RETURN p_salary * 0.10;
END;
```

Using a Function

```
DECLARE
    v_bonus NUMBER;
BEGIN
    v_bonus := get_bonus(5000);
    DBMS_OUTPUT.PUT_LINE('Bonus: ' || v_bonus);
END;
```

Procedures

```
CREATE OR REPLACE PROCEDURE update_salary(
    p_emp_id IN NUMBER,
    p_new_salary IN NUMBER
)
IS
BEGIN
    UPDATE employees
    SET salary = p_new_salary
    WHERE emp_id = p_emp_id;

    COMMIT;
END;
```

Using a Procedure

```
BEGIN
    update_salary(1, 6000);
END;
```

Implicit Cursor

```
BEGIN
    UPDATE employees SET salary = salary + 100 WHERE dept_id = 10;

    DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT || ' rows updated.');
END;
```

PL/SQL Cheat Sheet (No Joins)

Explicit Cursor

```
DECLARE
    CURSOR emp_cursor IS
        SELECT emp_id, name FROM employees WHERE dept_id = 10;

    v_emp_id employees.emp_id%TYPE;
    v_name employees.name%TYPE;
BEGIN
    OPEN emp_cursor;
    LOOP
        FETCH emp_cursor INTO v_emp_id, v_name;
        EXIT WHEN emp_cursor%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE(v_emp_id || ': ' || v_name);
    END LOOP;
    CLOSE emp_cursor;
END;
```

Exception Handling

```
BEGIN
    -- Risky operation
    UPDATE employees SET salary = salary / 0 WHERE emp_id = 1;
EXCEPTION
    WHEN ZERO_DIVIDE THEN
        DBMS_OUTPUT.PUT_LINE('Division by zero error.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
```

Triggers

```
CREATE OR REPLACE TRIGGER trg_before_insert_emp
BEFORE INSERT ON employees
FOR EACH ROW
BEGIN
    IF :NEW.salary < 1000 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Salary too low!');
    END IF;
END;
```

Enable Output

```
-- To see output in SQL Developer or SQL*Plus:
SET SERVEROUTPUT ON;
```