Here are the SQL solutions and PL/SQL programs based on your request:

1. SQL Operations on Customer, Sailor, and Reserves Tables

Customer Table:

```
CREATE TABLE Customer (
   cust_id INT PRIMARY KEY,
   cust_name VARCHAR(50),
   cust_city VARCHAR(50)
);

-- Insert records
INSERT INTO Customer (cust_id, cust_name, cust_city) VALUES (1, 'John Doe', 'Mumbai');
INSERT INTO Customer (cust_id, cust_name, cust_city) VALUES (2, 'Amit Sharma', 'Pune');
INSERT INTO Customer (cust_id, cust_name, cust_city) VALUES (3, 'Sara Khan', 'Hyd');
INSERT INTO Customer (cust_id, cust_name, cust_city) VALUES (4, 'Priya Mehta', 'Delhi');

-- Add salary column
ALTER TABLE Customer ADD salary DECIMAL(10, 2);

-- Alter column domain (increase salary size)
ALTER TABLE Customer MODIFY salary DECIMAL(12, 2);

-- Drop salary column
ALTER TABLE Customer DROP COLUMN salary;

-- Delete rows where cust_city is 'hyd'
DELETE FROM Customer WHERE LOWER(cust_city) = 'hyd';
```

Sailor Table:

```
CREATE TABLE Sailor (
   Sid NUMBER PRIMARY KEY,
```

```
    Sname VARCHAR2(20),
    Rating VARCHAR2(20)
);

-- Add age column
ALTER TABLE Sailor ADD age NUMBER;

-- Insert values
INSERT INTO Sailor (Sid, Sname, Rating, age) VALUES (1, 'Rajesh Kumar', '7', 25);
INSERT INTO Sailor (Sid, Sname, Rating, age) VALUES (2, 'Anil Singh', '9', 30);
INSERT INTO Sailor (Sid, Sname, Rating, age) VALUES (3, 'Priya Reddy', '6', 22);
INSERT INTO Sailor (Sid, Sname, Rating, age) VALUES (4, 'John Smith', '8', 28);

-- Delete row with rating > 8
DELETE FROM Sailor WHERE TO_NUMBER(Rating) > 8;

-- Update sailor details
UPDATE Sailor SET age = 29 WHERE Sid = 1;
UPDATE Sailor SET Sname = 'Raj Kumar', Rating = '8' WHERE Sid = 1;

-- Insert NULL values
INSERT INTO Sailor (Sid, Sname, Rating, age) VALUES (5, 'Unknown Sailor', NULL, NULL);
```

Reserves Table:

```
CREATE TABLE reserves (
    Boat_id INTEGER,
    sid INTEGER,
    day INTEGER
);
```

```
-- Insert records
INSERT INTO reserves (Boat_id, sid, day) VALUES (101, 1, 15);
INSERT INTO reserves (Boat_id, sid, day) VALUES (102, 2, 20);
INSERT INTO reserves (Boat_id, sid, day) VALUES (103, 3, 25);

-- Add time column
ALTER TABLE reserves ADD time VARCHAR2(10);

-- Alter the day column to date type
ALTER TABLE reserves MODIFY day DATE;

-- Drop time column
ALTER TABLE reserves DROP COLUMN time;

-- Delete row with Boat_id = 102
DELETE FROM reserves WHERE Boat_id = 102;
```

2. SQL Operations on Department and Employee Tables

Department Table:

```
-- Renaming the table
ALTER TABLE DEPT RENAME TO DEPARTMENT;

-- Adding a column with a NOT NULL constraint
ALTER TABLE DEPARTMENT ADD PINCODE NUMBER(6) NOT NULL;

-- Renaming a column
ALTER TABLE DEPARTMENT RENAME COLUMN DNAME TO DEPT_NAME;

-- Modifying column data type
ALTER TABLE DEPARTMENT MODIFY LOC CHAR(10);
```

-- Dropping the table

DROP TABLE DEPARTMENT;

Employee Table Operations:

-- Display all fields from employee table

SELECT * FROM employee;

-- Retrieve employee number and their salary

SELECT empno, salary FROM employee;

-- Retrieve average salary of all employees

SELECT AVG(salary) FROM employee;

-- Retrieve number of employees

SELECT COUNT(*) FROM employee;

-- Retrieve distinct employee names

SELECT COUNT(DISTINCT emp_name) FROM employee;

-- Retrieve total salary of employees grouped by name and count similar names

SELECT EMP_NAME, SUM(SALARY), COUNT(*) FROM EMPLOYEE GROUP BY EMP_NAME;

-- Retrieve total salary for employees with a salary greater than 120000

SELECT EMP_NAME, SUM(SALARY) FROM EMPLOYEE GROUP BY EMP_NAME HAVING SUM(SALARY) > 120000;

-- Display employee names in descending order

SELECT emp_name FROM employee ORDER BY emp_name DESC;

-- Display details of employee whose name is 'Amit' and salary greater than 50000

```
SELECT * FROM employee WHERE emp_name = 'Amit' AND salary > 50000;
```

3. PL/SQL Code

PL/SQL Program to Print Integers from 1 to 10:

```
DECLARE
   n_times NUMBER := 10;
BEGIN
   FOR n_i IN 1..n_times LOOP
      DBMS_OUTPUT.PUT_LINE(n_i);
   END LOOP;
END;
```

PL/SQL Procedure to Insert Tuple (i, 'xxx') into Relation:

```
CREATE OR REPLACE PROCEDURE addtuple1(x IN NUMBER)
AS
BEGIN
   INSERT INTO T2 VALUES(x, 'xxx');
END addtuple1;
```

PL/SQL Function to Compute Factorial:

```
CREATE FUNCTION fact(x NUMBER) RETURN NUMBER IS
   f NUMBER;
BEGIN
   IF x = 0 THEN
      f := 1;
   ELSE
      f := x * fact(x - 1);
   END IF;
   RETURN f;
END;


-- Calling the function
```

```
DECLARE
   num NUMBER := 6;
   factorial NUMBER;
BEGIN
   factorial := fact(num);
   DBMS_OUTPUT.PUT_LINE('Factorial of ' || num || ' is ' || factorial);
END;
```

PL/SQL Procedure to Compute Square of a Value:

```
DECLARE
   a NUMBER := 23;

   PROCEDURE squareNum(x IN OUT NUMBER) IS
   BEGIN
      x := x * x;
   END;

BEGIN
   squareNum(a);
   DBMS_OUTPUT.PUT_LINE('Square of ' || a || ' is: ' || a);
END;
```

4. SQL Subquery to Delete Rows

Subquery to Delete from Student Table:

```
DELETE FROM student
WHERE student_id IN (
   SELECT student_id FROM students WHERE grade = 'F'
);
```

5. Join Operations

Inner Join Examples:

Equijoin:

```
SELECT e.E_id, e.E_name, e.Age, d.Dept_name
FROM emp e
JOIN dept d ON e.Dept_id = d.Dept_id;
```

Theta/Conditional Join:

```
SELECT e.E_id, e.E_name, e.Age, d.Dept_name
FROM emp e
JOIN dept d ON e.Dept_id != d.Dept_id;
```

Natural Join:

```
SELECT e.E_id, e.E_name, e.Age, d.Dept_name
FROM emp e
NATURAL JOIN dept d;
```

Cross Join:

```
SELECT e.E_id, e.E_name, d.Dept_name
FROM emp e
CROSS JOIN dept d;
```

Outer Join Examples:

Left Outer Join:

```
SELECT e.E_id, e.E_name, e.Age, d.Dept_name
FROM emp e
LEFT OUTER JOIN dept d ON e.Dept_id = d.Dept_id;
```

Right Outer Join:

```
SELECT e.E_id, e.E_name, e.Age, d.Dept_name
FROM emp e
RIGHT OUTER JOIN dept d ON e.Dept_id = d.Dept_id;
```

Full Outer Join:

```
SELECT e.E_id, e.E_name, e.Age, d.Dept_name
FROM emp e
FULL OUTER JOIN dept d ON e.Dept_id = d.Dept_id;
```

6. Creating Functions, Triggers, and Procedures

Function to Compute Maximum of Two Values:

```
CREATE OR REPLACE FUNCTION GetMaxValue(val1 INT, val2 INT)
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE max_val INT;

    IF val1 > val2 THEN
        SET max_val = val1;
    ELSE
        SET max_val = val2;
    END IF;

    RETURN max_val;
END;
```

Trigger to Display Message on Update Event:

```
CREATE OR REPLACE TRIGGER record_updated_trigger
AFTER UPDATE ON employee
FOR EACH ROW
BEGIN
    DBMS_OUTPUT.PUT_LINE('Record updated');
END;
```

These SQL and PL/SQL solutions should help you perform the various database operations you're looking for.