```
1-- Supplier Table and Sample Data
CREATE TABLE Supplier (
  Sup_No VARCHAR(10),
  Sup_Name VARCHAR(50),
  Item_Supplied VARCHAR(50),
  Item_Price DECIMAL(10,2),
  City VARCHAR(50)
);

INSERT INTO Supplier VALUES
('s1','Ravi','Processor',7500,'Delhi'),
('s2','Raj','Keyboard',1500,'Mumbai'),
('s3','Ramesh','Mouse',800,'Delhi'),
('s4','Rohit','Monitor',5200,'Chennai');

-- 1. All records
SELECT * FROM Supplier;

-- 2. Count of suppliers
SELECT COUNT(*) AS TotalSuppliers FROM Supplier;

-- 3. Highest price
SELECT MAX(Item_Price) AS HighestPrice FROM Supplier;

-- 4. Price > 5000
SELECT * FROM Supplier WHERE Item_Price > 5000;

-- 5. Update city
UPDATE Supplier SET City='NewCity' WHERE Sup_No='s1'; SELECT * FROM Supplier;

-- 6. Delete supplier s2
```

```
DELETE FROM Supplier WHERE Sup_No='s2'; SELECT * FROM Supplier;

-- 7. Names starting with 'R'
SELECT Sup_No,Sup_Name FROM Supplier WHERE Sup_Name LIKE 'R%';

-- 8. Suppliers with Processor & city Delhi
SELECT Sup_Name FROM Supplier WHERE Item_Supplied='Processor' AND City='Delhi';

-- 9. Increase price of Keyboard by 200
UPDATE Supplier SET Item_Price = Item_Price + 200 WHERE Item_Supplied='Keyboard';

-- 10. Delhi suppliers sorted by price
SELECT Sup_No,Sup_Name,Item_Price FROM Supplier WHERE City='Delhi' ORDER BY Item_Price;

-- 11. Add new column CONTACTNO
ALTER TABLE Supplier ADD CONTACTNO VARCHAR(20); SELECT * FROM Supplier;

-- 12. Delete lowest price record
DELETE FROM Supplier WHERE Item_Price = (SELECT MIN(Item_Price) FROM Supplier); SELECT *
FROM Supplier;

-- 13. Descending by Item_Price for each Item
SELECT * FROM Supplier ORDER BY Item_Supplied, Item_Price DESC;

-- 14. Items other than Processor/Keyboard
SELECT * FROM Supplier WHERE Item_Supplied NOT IN ('Processor','Keyboard');
```

2-- Short SQL Script for Bank Database (Branch, Customer, Account, Loan, Borrower, Depositor)

```
-- Create Tables
CREATE TABLE Branch(
  b_name VARCHAR(50) PRIMARY KEY,
  b_city VARCHAR(50),
  Assets DECIMAL(15,2)
);

CREATE TABLE Customer(
  C_name VARCHAR(50) PRIMARY KEY,
  C_street VARCHAR(100),
  C_city VARCHAR(50)
);

CREATE TABLE Account(
  Ac_no INT PRIMARY KEY,
  Balance DECIMAL(10,2),
  b_name VARCHAR(50),
  FOREIGN KEY (b_name) REFERENCES Branch(b_name)
);

CREATE TABLE Loan(
  L_no INT PRIMARY KEY,
  Amt DECIMAL(10,2),
  b_name VARCHAR(50),
  FOREIGN KEY (b_name) REFERENCES Branch(b_name)
);

CREATE TABLE Borrower(
  C_name VARCHAR(50),
```

```
  L_no INT,
  FOREIGN KEY (C_name) REFERENCES Customer(C_name),
  FOREIGN KEY (L_no) REFERENCES Loan(L_no)
);

CREATE TABLE Depositor(
  C_name VARCHAR(50),
  Ac_no INT,
  FOREIGN KEY (C_name) REFERENCES Customer(C_name),
  FOREIGN KEY (Ac_no) REFERENCES Account(Ac_no)
);

-- Insert Data
INSERT INTO Branch VALUES ('Branch3','City3',800000),('Branch4','City4',900000);
INSERT INTO Customer VALUES
('Customer7','Street7','City5'),('Customer8','Street8','City6'),('Customer9','Street9','City5');
INSERT INTO Account VALUES (7,7000,'Branch3'),(8,8000,'Branch4'),(9,9000,'Branch4');
INSERT INTO Loan VALUES (107,50000,'Branch3'),(108,60000,'Branch4'),(109,70000,'Branch3');
INSERT INTO Borrower VALUES ('Customer7',107),('Customer8',108),('Customer9',109);
INSERT INTO Depositor VALUES ('Customer7',7),('Customer8',8),('Customer9',9);

-- Display Tables
SELECT * FROM Branch;
SELECT * FROM Customer;
SELECT * FROM Account;
SELECT * FROM Loan;
SELECT * FROM Borrower;
SELECT * FROM Depositor;

-- Queries
SELECT AVG(Balance) AS AverageBalance FROM Account;
```

```
SELECT MAX(Amt) AS MaximumLoanAmount FROM Loan;
SELECT C_name FROM Customer WHERE C_city='City5';
SELECT Ac_no, Balance FROM Account WHERE Balance>5000;
ALTER TABLE Customer ADD Email VARCHAR(100);
ALTER TABLE Branch RENAME COLUMN b_city TO BranchCity;
```

3) -- Short SQL Script for STUDENTS Table

```
-- Create Table
CREATE TABLE Students (
  Sid INT PRIMARY KEY,
  Sname VARCHAR(50),
  Course VARCHAR(30),
  Gender CHAR(1),
  Category VARCHAR(20),
  State VARCHAR(30),
  DOB DATE
);

-- Insert Sample Data
INSERT INTO Students VALUES
(1001,'Neha','Comp','F','OBC','Telangana','2001-05-10'),
(1002,'Arun','Comp','M','GEN','AndhraPradesh','2000-03-12'),
(1003,'Payal','Comp','F','OBC','Karnataka','2002-07-22'),
(1004,'Amritha','IT','F','SC','Kerala','2003-02-05'),
(1005,'Pavan','Comp','M','GEN','Maharashtra','2000-12-15');

-- 1. Students not from Telangana or Andhra Pradesh
SELECT * FROM Students WHERE State NOT IN ('Telangana','AndhraPradesh');

-- 2. Female students in Comp course & OBC
```

```
SELECT * FROM Students WHERE Course='Comp' AND Gender='F' AND Category='OBC';

-- 3. Student ID, name, and present age
SELECT Sid, Sname, EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM DOB) AS Age FROM Students;

-- 4. Students ordered by name for each course
SELECT Sid, Sname, Course FROM Students ORDER BY Course, Sname ASC;

-- 5. Add columns for contact and email
ALTER TABLE Students ADD (ContactNo VARCHAR(20), Email VARCHAR(100));

-- 6. Update contact and email
UPDATE Students SET ContactNo='1234567890', Email='neha@gmail.com' WHERE Sid=1001;
UPDATE Students SET ContactNo='9049315885', Email='arun@gmail.com' WHERE Sid=1002;
UPDATE Students SET ContactNo='8855994426', Email='payal@gmail.com' WHERE Sid=1003;
UPDATE Students SET ContactNo='7958647712', Email='amritha@gmail.com' WHERE Sid=1004;
UPDATE Students SET ContactNo='6958224564', Email='pavan@gmail.com' WHERE Sid=1005;

-- 7. Student names with 5-character length
SELECT Sname FROM Students WHERE LENGTH(Sname)=5;

-- 8. Delete students in Comp course born after 2002
DELETE FROM Students WHERE Course='Comp' AND EXTRACT(YEAR FROM DOB)>2002;

-- 9. Display names prefixed with Mr./Ms. based on gender
SELECT CASE UPPER(Gender)
        WHEN 'M' THEN 'Mr. '||Sname
        WHEN 'F' THEN 'Ms. '||Sname
        ELSE Sname
      END AS PrefixedName
```

```
FROM Students;
```

4) -- Short SQL Script: Employee and Department Tables

```
-- Create Tables
CREATE TABLE Department (
  Deptid VARCHAR(10) PRIMARY KEY,
  Dname VARCHAR(50) NOT NULL
);

CREATE TABLE Employee (
  Eid INT PRIMARY KEY,
  Ename VARCHAR(50),
  Deptid VARCHAR(10),
  Designation VARCHAR(30),
  Salary DECIMAL(10,2) CHECK (Salary >= 10000),
  DOJ DATE,
  FOREIGN KEY (Deptid) REFERENCES Department(Deptid)
);

-- Insert Sample Data
INSERT INTO Department VALUES ('D1','HR'),('D2','Finance'),('D3','IT');

INSERT INTO Employee VALUES
(101,'Ravi','D1','Manager',50000,'2011-02-15'),
(102,'Sita','D2','Clerk',28000,'2012-02-10'),
(103,'Arjun','D2','Clerk',32000,'2011-05-20'),
(104,'Kiran','D3','Analyst',45000,'2015-03-10'),
(105,'Meera','D1','HR Executive',38000,'2011-08-25');

-- 1. Employees earning above average salary
```

```
SELECT * FROM Employee WHERE Salary > (SELECT AVG(Salary) FROM Employee);

-- 2. Eid, Ename, and Department Name
SELECT e.Eid, e.Ename, d.Dname
FROM Employee e JOIN Department d ON e.Deptid = d.Deptid;

-- 3. Sort employees by salary descending
SELECT * FROM Employee ORDER BY Salary DESC;

-- 4. Distinct job designations
SELECT DISTINCT Designation FROM Employee;

-- 5. Employee details by department (asc salary)
SELECT * FROM Employee ORDER BY Deptid ASC, Salary ASC;

-- 6. All Clerks in Dept D2
SELECT * FROM Employee WHERE Designation='Clerk' AND Deptid='D2';

-- 7. Employees who joined in 2011
SELECT * FROM Employee WHERE EXTRACT(YEAR FROM DOJ)=2011;

-- 8. Employees who joined in February
SELECT * FROM Employee WHERE EXTRACT(MONTH FROM DOJ)=2;

-- 9. Employees with salary between 30000 and 45000
SELECT * FROM Employee WHERE Salary BETWEEN 30000 AND 45000;

-- 10. Employees with work experience till date
SELECT Eid, Ename, Deptid, Designation, Salary, DOJ,
    (EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM DOJ)) AS Experience
FROM Employee;
```

5) -- Short SQL Script: Library Information System

```
-- Create Table
CREATE TABLE BookDetails (
  BookId INT PRIMARY KEY,
  BookName VARCHAR(255) NOT NULL,
  Author VARCHAR(255) NOT NULL,
  DatePurchased DATE NOT NULL,
  Publisher VARCHAR(255) NOT NULL,
  Price INT NOT NULL
);

-- Insert Data
INSERT INTO BookDetails VALUES
(1,'Cost Accounting','Jain Narang','2013-02-11','Kalyani',800),
(2,'Business Statistics','OP Aggarwal','2011-12-22','Himalaya',750),
(3,'RDBMS','CJ Date','2015-03-02','TMH',900),
(4,'Mgmt. Accounting','RK Sharma','2016-04-19','Kalyani',450),
(5,'Operating Systems','Galvin','2013-11-25','PHI',750),
(6,'Advanced Accounting','SC Gupta','2018-04-16','Himalaya',600);

-- 1. Authors from Himalaya publications
SELECT Author FROM BookDetails WHERE Publisher='Himalaya';

-- 2. Total cost of books purchased (Publisher wise)
SELECT Publisher, SUM(Price) AS TotalCost FROM BookDetails GROUP BY Publisher;

-- 3. Count of books under Kalyani publications
SELECT COUNT(*) AS TotalBooks FROM BookDetails WHERE Publisher='Kalyani';

-- 4. Books in ascending order of purchase date
```

```
SELECT BookName, Author, DatePurchased, Publisher, Price
FROM BookDetails ORDER BY DatePurchased ASC;

-- 5. Create index on BookName and Author
CREATE INDEX idx_book_name_author ON BookDetails(BookName, Author);

-- 6. Books with price between 500 and 700
SELECT BookName, Author, Price FROM BookDetails WHERE Price BETWEEN 500 AND 700;

-- 7. Increase price by 200 for publishers other than Himalaya or Kalyani
UPDATE BookDetails SET Price=Price+200 WHERE Publisher NOT IN ('Himalaya','Kalyani');
SELECT * FROM BookDetails;

-- 8. Books where author contains 'Sharma'
SELECT BookName, Author, Publisher, Price FROM BookDetails WHERE Author LIKE '%Sharma%';

-- 9. Create a view of books published by Himalaya
CREATE VIEW BookDetailsByHimalaya AS
SELECT BookId, BookName FROM BookDetails WHERE Publisher='Himalaya';
SELECT * FROM BookDetailsByHimalaya;
```

6) -- Short SQL Script: Users, Products & Orders Tables

```
-- Create Tables
CREATE TABLE Users (
  UserID INT PRIMARY KEY,
  FirstName VARCHAR(50),
  LastName VARCHAR(50),
  Email VARCHAR(100) UNIQUE,
  BirthDate DATE,
  RegistrationDate DATETIME DEFAULT CURRENT_TIMESTAMP
```

```
);

CREATE TABLE Products (
  ProductID INT PRIMARY KEY,
  ProductName VARCHAR(100),
  Category VARCHAR(50),
  Price DECIMAL(10,2),
  StockQuantity INT
);

CREATE TABLE Orders (
  OrderID INT PRIMARY KEY,
  UserID INT,
  OrderDate DATETIME,
  TotalAmount DECIMAL(10,2),
  FOREIGN KEY (UserID) REFERENCES Users(UserID)
);

-- Insert Data
INSERT INTO Users VALUES
(101,'John','Doe','john@example.com','1990-05-15','2023-07-01'),
(102,'Jane','Smith','jane@example.com','1985-09-22','2023-07-02'),
(103,'Michael','Johnson','michael@example.com','1992-02-10','2023-07-03'),
(104,'Emily','Brown','emily@example.com','1998-07-01','2023-07-04'),
(105,'David','Wilson','david@example.com','1988-11-28','2023-07-05');

INSERT INTO Products VALUES
(1,'Laptop','Electronics',999.99,50),
(2,'Smartphone','Electronics',599.99,100),
(3,'T-Shirt','Clothing',19.99,200),
(4,'Coffee Maker','Appliances',79.99,30),
```

```
(5,'Running Shoes','Footwear',89.99,75);

INSERT INTO Orders VALUES
(1,101,'2023-08-10 09:15',159.98),
(2,102,'2023-08-11 14:30',219.98),
(3,103,'2023-08-12 11:45',89.99),
(4,104,'2023-08-12 15:20',329.97),
(5,101,'2023-08-13 08:00',599.99);

-- 1. Retrieve first and last names of all users
SELECT FirstName, LastName FROM Users;

-- 2. Total number of orders
SELECT COUNT(*) AS TotalOrders FROM Orders;

-- 3. Product names in Electronics category
SELECT ProductName FROM Products WHERE Category='Electronics';

-- 4. Total price of all orders
SELECT SUM(TotalAmount) AS TotalPrice FROM Orders;

-- 5. Top 5 most expensive products
SELECT ProductName, Price FROM Products ORDER BY Price DESC LIMIT 5;

-- 6. Orders placed by UserID 101
SELECT OrderID, OrderDate, TotalAmount FROM Orders WHERE UserID=101;

-- 7. Average stock quantity per category
SELECT Category, AVG(StockQuantity) AS AvgStockQuantity FROM Products GROUP BY Category;

-- 8. Number of orders per user (highest first)
```

```
SELECT UserID, COUNT(*) AS OrderCount FROM Orders GROUP BY UserID ORDER BY OrderCount
DESC;

-- 9. Users who never placed an order
SELECT u.UserID, u.FirstName, u.LastName, u.RegistrationDate
FROM Users u
LEFT JOIN Orders o ON u.UserID=o.UserID
WHERE o.OrderID IS NULL;

-- 10. Days difference between registration & order date
SELECT o.OrderID, u.FirstName, u.LastName,
    DATEDIFF(o.OrderDate, u.RegistrationDate) AS DaysDifference
FROM Orders o
JOIN Users u ON o.UserID=u.UserID;
```

PL/SQL

```
-- 1. Addition of Two Numbers
DECLARE a NUMBER:=43; b NUMBER:=5;
BEGIN DBMS_OUTPUT.PUT_LINE('Sum = '||(a+b)); END;
/

-- 2. Factorial of a Number
DECLARE n NUMBER:=5; f NUMBER:=1;
BEGIN FOR i IN 1..n LOOP f:=f*i; END LOOP;
DBMS_OUTPUT.PUT_LINE('Factorial='||f); END;
/

-- 3. For Loop Demo
BEGIN FOR i IN 1..10 LOOP DBMS_OUTPUT.PUT_LINE('i='||i); END LOOP; END;
/
```

```
-- 4. CASE Structure
DECLARE a NUMBER:=3; b NUMBER:=4; op CHAR:='+'; r NUMBER;
BEGIN CASE op WHEN '+' THEN r:=a+b WHEN '-' THEN r:=a-b WHEN '*' THEN r:=a*b ELSE r:=0 END CASE;
DBMS_OUTPUT.PUT_LINE(a||op||b||'='||r); END;
/

-- 5. Simple Loop
DECLARE i NUMBER:=1; BEGIN LOOP DBMS_OUTPUT.PUT_LINE('i='||i); EXIT WHEN i=5; i:=i+1; END LOOP; END;
/

-- 6. Increase Value by 10
DECLARE n NUMBER:=5; BEGIN n:=n+10; DBMS_OUTPUT.PUT_LINE('Value='||n); END;
/

-- 7. Arithmetic Operations
DECLARE a NUMBER:=8; b NUMBER:=4;
BEGIN DBMS_OUTPUT.PUT_LINE('Sum='||(a+b)||' Sub='||(a-b)||' Mul='||(a*b)||' Div='||(a/b)); END;
/

-- 8. Square, Cube, Double
DECLARE a NUMBER:=5;
BEGIN DBMS_OUTPUT.PUT_LINE('Sq='||(a*a)||' Cube='||(a*a*a)||' Double='||(a*2)); END;
/

-- 9. Swap Two Numbers
DECLARE a NUMBER:=4; b NUMBER:=5; t NUMBER;
BEGIN t:=a; a:=b; b:=t; DBMS_OUTPUT.PUT_LINE('a='||a||' b='||b); END;
/

-- 10. Multiplication Table
DECLARE n NUMBER:=5;
BEGIN FOR i IN 1..10 LOOP DBMS_OUTPUT.PUT_LINE(n||'*'||i||'='||(n*i)); END LOOP; END;
```

```
/

-- 11. Leap Year Check
DECLARE y NUMBER:=2024;
BEGIN IF MOD(y,400)=0 OR (MOD(y,4)=0 AND MOD(y,100)<>0)
THEN DBMS_OUTPUT.PUT_LINE(y||' Leap'); ELSE DBMS_OUTPUT.PUT_LINE(y||' Not Leap'); END IF; END;
/

-- 12. Delete Item with Number=4
BEGIN DELETE FROM item WHERE itemnum=4;
IF SQL%ROWCOUNT>0 THEN DBMS_OUTPUT.PUT_LINE('Deleted'); ELSE DBMS_OUTPUT.PUT_LINE('No Rows');
END IF; END;
/

-- 13. Reverse a Number
DECLARE n NUMBER:=12345; r NUMBER:=0; d NUMBER;
BEGIN WHILE n>0 LOOP d:=MOD(n,10); r:=r*10+d; n:=(n-d)/10; END LOOP;
DBMS_OUTPUT.PUT_LINE('Reverse='||r); END;
/

-- 14. Area of Circle (r=3..7)
BEGIN FOR r IN 3..7 LOOP DBMS_OUTPUT.PUT_LINE('r='||r||' area='||(3.14*r*r)); END LOOP; END;
/

-- 15. Display Emp Details by Empno
DECLARE e NUMBER:=7839;
BEGIN FOR x IN (SELECT empno,ename,sal FROM emp WHERE empno=e)
LOOP DBMS_OUTPUT.PUT_LINE(x.empno||' '||x.ename||' '||x.sal); END LOOP; END;
/

-- 16. Increase Salary by 100 if >1000
DECLARE e NUMBER:=101;
BEGIN UPDATE emp SET sal=sal+100 WHERE empno=e AND sal>1000;
```

```
DBMS_OUTPUT.PUT_LINE('Salary Updated'); END;
/

-- 17. Cursor for emp with sal+comm>2000
DECLARE CURSOR c IS SELECT empno,ename,sal+NVL(comm,0) s FROM emp WHERE sal+NVL(comm,0)>2000;
r c%ROWTYPE;
BEGIN DBMS_OUTPUT.PUT_LINE('EmpNo Name NetSal');
OPEN c; LOOP FETCH c INTO r; EXIT WHEN c%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(r.empno||' '||r.ename||' '||r.s); END LOOP; CLOSE c; END;
/

-- 18. Procedure to Update Salary
CREATE OR REPLACE PROCEDURE empupdate IS
BEGIN UPDATE emp SET sal=sal+1000 WHERE empno=7788; COMMIT;
DBMS_OUTPUT.PUT_LINE('Updated'); END;
/

-- 19. Empno Check
DECLARE e emp.empno%TYPE:=7698; v emp.empno%TYPE;
BEGIN SELECT empno INTO v FROM emp WHERE empno=e;
DBMS_OUTPUT.PUT_LINE(v||' Found');
EXCEPTION WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('Not Found'); END;
/

-- 20. Trigger to Lowercase Name & Job
CREATE OR REPLACE TRIGGER lowername
BEFORE INSERT OR UPDATE ON emp FOR EACH ROW
BEGIN :NEW.ename:=LOWER(:NEW.ename); :NEW.job:=LOWER(:NEW.job); END;
/

-- 21. IN OUT Procedure Example
DECLARE a NUMBER:=23;
PROCEDURE sq(x IN OUT NUMBER) IS BEGIN x:=x*x; END;
```

```
BEGIN sq(a); DBMS_OUTPUT.PUT_LINE('Square='||a); END;
/

-- 22. Function to Count Employees
CREATE OR REPLACE FUNCTION totalemp RETURN NUMBER IS t NUMBER;
BEGIN SELECT COUNT(*) INTO t FROM emp; RETURN t; END;
/

-- 23. ZERO_DIVIDE Exception
DECLARE x NUMBER:=5; y NUMBER:=0;
BEGIN DBMS_OUTPUT.PUT_LINE('Div='||(x/y));
EXCEPTION WHEN ZERO_DIVIDE THEN DBMS_OUTPUT.PUT_LINE('Check denominator'); END;
/
```