**NAME: Abdullah Mohsin**

**QUERIES PRACTICE**

**ALL QUERIES CONTENT:**

[**TYPES OF DATA LANGUAGES:** 3](#_Toc186214282)

[**Figure:1.1(ENTITY RELATIONSHIP DIAGRAM)** 3](#_Toc186214283)

[**RELATIONAL TABLES:** 3](#_Toc186214284)

[**1.** **MART TABLE:** 3](#_Toc186214285)

[**2.** **BASKET TABLE:** 3](#_Toc186214286)

[**3.** **FRUIT TABLE:** 3](#_Toc186214287)

[ **TUTORIAL:** 4](#_Toc186214288)

[**Now come to Database languages:** 4](#_Toc186214289)

[**1.** **DDL (DATA DEFINITION LANGUAGE):** 4](#_Toc186214290)

[**COMMANDS:** 4](#_Toc186214291)

[**1.** **CREATE:** 4](#_Toc186214292)

[**2.** **ALTER:** 4](#_Toc186214293)

[**11.** **DROP:** 4](#_Toc186214294)

[**13.** **TRUNCATE:** 4](#_Toc186214295)

[**15.** **RENAME:** 4](#_Toc186214296)

[**16.** **COMMENT:** 5](#_Toc186214297)

[**17.** **USE:** 5](#_Toc186214298)

[**18.** **PARTITION:** 5](#_Toc186214299)

[**2.** **DML (Data Manipulation Language):** 5](#_Toc186214300)

[**COMMANDS:** 5](#_Toc186214301)

[**1.** **SELECT:** 5](#_Toc186214302)

[**2.** **INSERT:** 5](#_Toc186214303)

[**3.** **UPDATE:** 5](#_Toc186214304)

[**4.** **DELETE:** 5](#_Toc186214305)

[**5.** **MERGE:** 5](#_Toc186214306)

[**6.** **CALL:** 5](#_Toc186214307)

[**7.** **EXPLAIN PLAN:** 5](#_Toc186214308)

[**8.** **LOCK TABLE:** 5](#_Toc186214309)

[ **DQL(DATA QUERY LANGUAGE):** 5](#_Toc186214310)

[**COMMANDS:** 6](#_Toc186214311)

[**1.** **SELECT:** 6](#_Toc186214312)

[**2.** **FROM:** 6](#_Toc186214313)

[**3.** **WHERE:** 6](#_Toc186214314)

[**4.** **GROUP BY:** 6](#_Toc186214315)

[**5.** **HAVING:** 6](#_Toc186214316)

[**6.** **ORDER BY:** 6](#_Toc186214317)

[**7.** **JOIN:** 6](#_Toc186214318)

[**8.** **DISTINCT:** 6](#_Toc186214319)

[**9.** **LIMIT:** 6](#_Toc186214320)

[**10.** **OFFSET:** 6](#_Toc186214321)

[**11.** **TOP:** 6](#_Toc186214322)

[**3.** **DCL (Data Control Language):** 6](#_Toc186214323)

[**COMMANDS:** 6](#_Toc186214324)

[**1.** **GRANT:** 6](#_Toc186214325)

[**2.** **REVOKE:** 6](#_Toc186214326)

[**4.** **TCL (Transactional Control Language):** 6](#_Toc186214327)

[**COMMANDS:** 6](#_Toc186214328)

[**1.** **COMMIT:** 6](#_Toc186214329)

[**2.** **ROLLBACK:** 6](#_Toc186214330)

[**3.** **SAVEPOINT:** 6](#_Toc186214331)

[**4.** **SET TRANSACTION:** 6](#_Toc186214332)

[ **OBJECTS IN SQL:** 7](#_Toc186214333)

[**1.** **Tables:** 7](#_Toc186214334)

[**2.** **Views:** 7](#_Toc186214335)

[**3.** **Indexes:** 7](#_Toc186214336)

[**4.** **Schemas:** 7](#_Toc186214337)

[**5.** **Sequences:** 7](#_Toc186214338)

[**6.** **Triggers:** 7](#_Toc186214339)

[**7.** **Stored Procedures:** 7](#_Toc186214340)

[**8.** **Functions:** 7](#_Toc186214341)

[**9.** **Constraints (e.g., PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK, NOT NULL):** 7](#_Toc186214342)

[**10.** **Synonyms:** 7](#_Toc186214343)

[**11.** **Users and Roles (for database security management):** 7](#_Toc186214344)

[ **PL/SQL COMMANDS & OBJECTS IN SQL :** 7](#_Toc186214345)

[**1.** **Anonymous Blocks:** 7](#_Toc186214346)

[**2.** **Stored Procedures:** 7](#_Toc186214347)

[**3.** **Functions:** 8](#_Toc186214348)

[**4.** **Packages:** 8](#_Toc186214349)

[**5.** **Triggers:** 8](#_Toc186214350)

[**6.** **Views:** 8](#_Toc186214351)

[**7.** **Cursors (Implicit and Explicit):** 8](#_Toc186214352)

[**8.** **Records:** 8](#_Toc186214353)

[**9.** **Exceptions:** 8](#_Toc186214354)

[**10.** **Variables:** 9](#_Toc186214355)

[**11.** **Collections (Associative Arrays, Nested Tables, Varrays):** 9](#_Toc186214356)

# **TYPES OF DATA LANGUAGES:**

There are three types of data models:

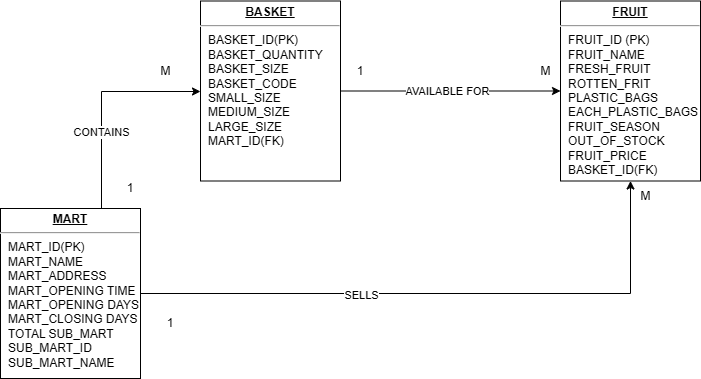
1.Data Definition Language

2.Data Manipulation Language

3.Data Control Language

4.Transactional Control Language

EXAMPLES: CONSIDER A ER-DIAGRAM with MART, FRUIT and BASKET:



# **Figure:1.1(ENTITY RELATIONSHIP DIAGRAM)**

# **RELATIONAL TABLES:**

# **MART TABLE:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| MART-ID(PK) | MART\_NAME | MART\_  ADDRESS | MART\_OPENING\_TIME | MART\_  OPENING DAYS | MART\_CLOSING DAYS | TOTAL\_  SUBMART | SUB\_  MART\_ID | SUB\_  MART\_NAME |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

# **BASKET TABLE:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| BASKET-ID(PK) | BASKET\_QUANTITY | BASKET\_SIZE | BASKET\_CODE | SMALL\_SIZE | MEDIUM\_SIZE | LARGE\_SIZE | MART\_ID(FK) |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

# **FRUIT TABLE:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| FRUIT-ID(PK) | FRUIT\_  NAME | FRESH\_  FRUIT | ROTTEN\_  FRUIT | PLASTIC\_BAGS | EACH\_PLASTIC\_BAGS | FRUIT\_  SEASON | OUT\_OF\_STOCK | FRUIT\_PRICE | BASKET\_ID  (FK) |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

# **TUTORIAL:**

**W**e are familiar with database, database is the method of storing and managing data in hardware. So, we must need to use a software, this software is known as database management system. Here in this document we are studying SQL, it is relational database management system.

There are three model in order to create a database system:

1.Conceptual Model

2.Representational Model

3.Physical model

You can see in figure 1.1, ER-Diagram has been shown and from that diagram we have created relation schema. We will use above information in order to implement it physically. You can use SQL by downloading it or running queries online.

**CLICK LINK:** [**https://livesql.oracle.com/ords/f?p=590:1000**](https://livesql.oracle.com/ords/f?p=590:1000)

In order to download or use oracle live without any hurdle login in SQL and enjoy writing query, if any doubt contact me :03363736231

# **Now come to Database languages:**

# **DDL (DATA DEFINITION LANGUAGE):**

# **COMMANDS:**

# **CREATE:**

1. --DATABASE LANGUAGE COMMANDS
2. --1.DDL(DATA DEFINITION LANGUAGE)
3. --ENTITIES MART,BASKET,FRUIT
4. CREATE TABLE MART(
5. MART\_ID INT  PRIMARY KEY,
6. MART\_NAME VARCHAR(10),
7. MART\_ADDRESS VARCHAR(30),
8. MART\_OPENING\_TIME TIMESTAMP,
9. MART\_OPENING\_DAYS VARCHAR(10),
10. MART\_CLOSING\_DAYS VARCHAR(10),
11. SUB\_MART\_ID INT UNIQUE,
12. SUB\_MART\_NAME VARCHAR(5)
13. );
14. CREATE TABLE BASKET(
15. BASKET\_ID INT PRIMARY KEY,
16. BASKET\_QUANTITY NUMBER(23,2),
17. BASKET\_SIZE INT,
18. SMALL\_SIZE INT,
19. MEDIUM\_SIZE INT,
20. LARGE\_SIZE INT,
21. MART\_ID INT,
22. CONSTRAINT FK\_MART FOREIGN KEY(MART\_ID)REFERENCES MART(MART\_ID),
23. CONSTRAINT UNIQUE\_BASKET\_SIZE UNIQUE(BASKET\_SIZE, SMALL\_SIZE, MEDIUM\_SIZE, LARGE\_SIZE)
24. );
25. CREATE TABLE FRUIT(
26. FRUIT\_ID INT PRIMARY KEY,
27. FRUIT\_NAME VARCHAR(10),
28. FRESH\_FRUIT INT,
29. ROTTEN\_FRUIT INT,
30. PLASTIC\_BAGS NUMBER(10),
31. EACH\_PLASTIC\_BAGS VARCHAR(1),
32. FRUIT\_SEASON VARCHAR(5),
33. OUT\_OF\_STOCK NUMBER(20),
34. FRUIT\_PRICE NUMBER(5,2),
35. BASKET\_ID INT,
36. MART\_ID INT,
37. CONSTRAINT FK\_BASKETS FOREIGN KEY(BASKET\_ID)REFERENCES BASKET(BASKET\_ID),
38. CONSTRAINT FK\_MARTS FOREIGN KEY(MART\_ID)REFERENCES MART(MART\_ID)
39. );

# **ALTER:**

1. -2.ALTER
2. ALTER TABLE MART ADD MANAGER\_NAME VARCHAR(5);--ADD COLUMN
3. ALTER TABLE MART MODIFY MART\_NAME VARCHAR(20);--MODIFY COLUMN
4. --CHANGEMENT IN DATATYPE
5. --RENAME TABLE
6. ALTER TABLE MART RENAME COLUMN MART\_ID TO MART\_NO;
7. --DROP COLUMN
8. ALTER TABLE MART DROP COLUMN MART\_NAME;

# **DROP:**

1. DROP TABLE BASKET;

# **TRUNCATE:**

1. TRUNCATE TABLE BASKET;

# **RENAME:**

15.ALTER TABLE BASKET RENAME TO SHOPPING\_BASKET;

# **COMMENT:**

16.COMMENT ON TABLE BASKET IS 'This table contains information about different baskets used in the store.';

# **USE:**

17.USE my\_database;

# **PARTITION:**

1. CREATE TABLE MART (
2. MART\_ID INT PRIMARY KEY,
3. MART\_NAME VARCHAR(10),
4. MART\_ADDRESS VARCHAR(30),
5. MART\_OPENING\_TIME TIMESTAMP,
6. MART\_OPENING\_DAYS VARCHAR(10),
7. MART\_CLOSING\_DAYS VARCHAR(10),
8. SUB\_MART\_ID INT UNIQUE,
9. SUB\_MART\_NAME VARCHAR(5)
10. )
11. PARTITION BY LIST (MART\_OPENING\_DAYS) (
12. PARTITION p\_mon VALUES ('Monday'),
13. PARTITION p\_tue VALUES ('Tuesday'),
14. PARTITION p\_wed VALUES ('Wednesday'),
15. PARTITION p\_thu VALUES ('Thursday'),
16. PARTITION p\_fri VALUES ('Friday'),
17. PARTITION p\_sat VALUES ('Saturday'),
18. PARTITION p\_sun VALUES ('Sunday')
19. );

# **DML (Data Manipulation Language):**

# **COMMANDS:**

# **SELECT:**

SELECT \* FROM MART;

# **INSERT:**

INSERT INTO MART (MART\_ID, MART\_NAME, MART\_ADDRESS, MART\_OPENING\_TIME, MART\_OPENING\_DAYS, MART\_CLOSING\_DAYS, SUB\_MART\_ID, SUB\_MART\_NAME)

VALUES (2, 'FruitMart', '456 Elm St', TO\_TIMESTAMP('2024-01-01 09:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'Monday,Thursday', 'Friday,Saturday', 102, 'SubMart2');

# **UPDATE:**

UPDATE BASKET

SET BASKET\_QUANTITY = 150

WHERE BASKET\_ID = 1;

# **DELETE:**

DELETE FROM FRUIT

WHERE FRUIT\_ID = 1;

# **MERGE:**

MERGE INTO FRUIT f

USING (SELECT 2 AS FRUIT\_ID, 'Grapes' AS FRUIT\_NAME FROM DUAL) new\_fruit

ON (f.FRUIT\_ID = new\_fruit.FRUIT\_ID)

WHEN MATCHED THEN

UPDATE SET f.FRUIT\_NAME = new\_fruit.FRUIT\_NAME

WHEN NOT MATCHED THEN

INSERT (FRUIT\_ID, FRUIT\_NAME) VALUES (new\_fruit.FRUIT\_ID, new\_fruit.FRUIT\_NAME);

# **CALL:**

CALL UpdateFruitStock(1, 100); -- Calls a stored procedure to update stock

# **EXPLAIN PLAN:**

EXPLAIN PLAN FOR

SELECT \* FROM MART WHERE MART\_ID = 1;

# **LOCK TABLE:**

LOCK TABLE MART IN EXCLUSIVE MODE;

# **DQL(DATA QUERY LANGUAGE):**

Here are the DQL (Data Query Language) commands and clauses:

# **COMMANDS:**

# **SELECT:**

SELECT MART\_NAME, MART\_ADDRESS FROM MART;

# **FROM:**

SELECT \* FROM BASKET;

# **WHERE:**

SELECT FRUIT\_NAME FROM FRUIT

WHERE FRUIT\_PRICE > 1.00;

# **GROUP BY:**

SELECT MART\_ID, COUNT(\*) AS FRUIT\_COUNT

FROM FRUIT

GROUP BY MART\_ID;

# **HAVING:**

SELECT MART\_ID, COUNT(\*) AS FRUIT\_COUNT

FROM FRUIT

GROUP BY MART\_ID

HAVING COUNT(\*) > 10;

# **ORDER BY:**

SELECT FRUIT\_NAME, FRUIT\_PRICE

FROM FRUIT

ORDER BY FRUIT\_PRICE DESC;

# **JOIN:**

SELECT FRUIT.FRUIT\_NAME, BASKET.BASKET\_SIZE

FROM FRUIT

INNER JOIN BASKET ON FRUIT.BASKET\_ID = BASKET.BASKET\_ID;

## **i.INNER JOIN:**

SELECT FRUIT.FRUIT\_NAME, BASKET.BASKET\_SIZE

FROM FRUIT

INNER JOIN BASKET ON FRUIT.BASKET\_ID = BASKET.BASKET\_ID;

**ii.LEFT JOIN:**

SELECT FRUIT.FRUIT\_NAME, BASKET.BASKET\_SIZE

FROM FRUIT

LEFT JOIN BASKET ON FRUIT.BASKET\_ID = BASKET.BASKET\_ID;

**iii.RIGHT JOIN:**

**SELECT FRUIT.FRUIT\_NAME, BASKET.BASKET\_SIZE**

**FROM FRUIT**

**RIGHT JOIN BASKET ON FRUIT.BASKET\_ID = BASKET.BASKET\_ID;**

**iv. FULL JOIN:**

**SELECT FRUIT.FRUIT\_NAME, BASKET.BASKET\_SIZE**

**FROM FRUIT**

**FULL JOIN BASKET ON FRUIT.BASKET\_ID = BASKET.BASKET\_ID;**

**v.CROSS JOIN:**

**SELECT FRUIT.FRUIT\_NAME, BASKET.BASKET\_SIZE**

**FROM FRUIT**

**CROSS JOIN BASKET;**

**Vi .INNER JOIN:**

**SELECT A.FRUIT\_NAME AS FRUIT1, B.FRUIT\_NAME AS FRUIT2**

**FROM FRUIT A**

**INNER JOIN FRUIT B ON A.BASKET\_ID = B.BASKET\_ID**

**WHERE A.FRUIT\_ID <> B.FRUIT\_ID;**

**Vii.NATURAL JOIN:**

**SELECT FRUIT\_NAME, BASKET\_SIZE**

**FROM FRUIT**

**NATURAL JOIN BASKET;**

**Viii.ANTI JOIN:**

**SELECT FRUIT\_NAME**

**FROM FRUIT**

**WHERE BASKET\_ID NOT IN (SELECT BASKET\_ID FROM BASKET);**

**ix.SEMI JOIN:**

**SELECT FRUIT\_NAME**

**FROM FRUIT**

**WHERE EXISTS (SELECT 1 FROM BASKET WHERE FRUIT.BASKET\_ID = BASKET.BASKET\_ID);**

# **DISTINCT:**

SELECT DISTINCT FRUIT\_NAME

FROM FRUIT;

# **LIMIT:**

SELECT \* FROM FRUIT

LIMIT 5;

# **OFFSET:**

SELECT \* FROM FRUIT

LIMIT 5 OFFSET 10;

# **TOP:**

SELECT TOP 5 \* FROM FRUIT;

# **DCL (Data Control Language):**

# **COMMANDS:**

# **GRANT:**

GRANT SELECT, INSERT ON MART TO user\_name;

# **REVOKE:**

REVOKE SELECT, INSERT ON MART FROM user\_name;

# **TCL (Transactional Control Language):**

# **COMMANDS:**

# **COMMIT:**

UPDATE BASKET

SET BASKET\_QUANTITY = 200

WHERE BASKET\_ID = 1;

COMMIT;

# **ROLLBACK:**

UPDATE BASKET

SET BASKET\_QUANTITY = 200

WHERE BASKET\_ID = 1;

ROLLBACK;

# **SAVEPOINT:**

SAVEPOINT sp1;

UPDATE BASKET

SET BASKET\_QUANTITY = 200

WHERE BASKET\_ID = 1;

SAVEPOINT sp2;

UPDATE BASKET

SET BASKET\_QUANTITY = 300

WHERE BASKET\_ID = 2;

ROLLBACK TO sp1;

# **SET TRANSACTION:**

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

UPDATE MART

SET MART\_ADDRESS = '789 Oak St'

WHERE MART\_ID = 1;

## **OBJECTS IN SQL:**

These are objects of SQL:

# **Tables:**

CREATE TABLE Employees (

Employee ID INT PRIMARY KEY,

Name VARCHAR (50) NOT NULL,

Department VARCHAR (50),

Salary DECIMAL (10, 2)

);

# **Views:**

CREATE VIEW HighSalaryEmployees AS

SELECT Name, Salary

FROM Employees

WHERE Salary > 50000;

# **Indexes:**

CREATE INDEX idx\_salary ON Employees(Salary);

# **Schemas:**

CREATE SCHEMA Sales AUTHORIZATION sales\_manager;

# **Sequences:**

CREATE SEQUENCE emp\_seq

START WITH 1

INCREMENT BY 1;

# **Triggers:**

CREATE TRIGGER trg\_salary\_check

BEFORE INSERT ON Employees

FOR EACH ROW

WHEN (NEW.Salary < 0)

BEGIN

RAISE\_APPLICATION\_ERROR(-20001, 'Salary cannot be negative.');

END;

# **Stored Procedures:**

CREATE PROCEDURE UpdateSalary (emp\_id INT, new\_salary DECIMAL)

AS

BEGIN

UPDATE Employees

SET Salary = new\_salary

WHERE Employee\_ID = emp\_id;

END;

# **Functions:**

CREATE FUNCTION CalculateBonus (salary DECIMAL)

RETURNS DECIMAL

AS

BEGIN

RETURN salary \* 0.1;

END;

# **Constraints (e.g., PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK, NOT NULL):**

CREATE TABLE Departments (

Dept\_ID INT PRIMARY KEY,

Dept\_Name VARCHAR(50) NOT NULL,

Manager\_ID INT UNIQUE,

CONSTRAINT chk\_name CHECK (Dept\_Name != '')

);

# **Synonyms:**

CREATE SYNONYM EmpView FOR HighSalaryEmployees;

# **Users and Roles (for database security management):**

CREATE USER john IDENTIFIED BY password123;

GRANT CONNECT, CREATE TABLE TO john;

CREATE ROLE SalesRole;

GRANT SELECT, INSERT ON Employees TO SalesRole;

GRANT SalesRole TO john;

# **PL/SQL COMMANDS & OBJECTS IN SQL :**

These are objects of PL/SQL:

# **Anonymous Blocks:**

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Hello, PL/SQL!');

END;

# **Stored Procedures:**

CREATE PROCEDURE UpdateEmployeeSalary (emp\_id INT, increment DECIMAL)

AS

BEGIN

UPDATE Employees

SET Salary = Salary + increment

WHERE Employee\_ID = emp\_id;

END;

# **Functions:**

CREATE FUNCTION GetTotalSalary

RETURN DECIMAL

AS

total\_salary DECIMAL;

BEGIN

SELECT SUM(Salary) INTO total\_salary FROM Employees;

RETURN total\_salary;

END;

# **Packages:**

CREATE PACKAGE EmployeePackage AS

PROCEDURE UpdateEmployeeSalary(emp\_id INT, increment DECIMAL);

FUNCTION GetEmployeeCount RETURN INT;

END EmployeePackage;

CREATE PACKAGE BODY EmployeePackage AS

PROCEDURE UpdateEmployeeSalary(emp\_id INT, increment DECIMAL) AS

BEGIN

UPDATE Employees

SET Salary = Salary + increment

WHERE Employee\_ID = emp\_id;

END;

FUNCTION GetEmployeeCount RETURN INT AS

emp\_count INT;

BEGIN

SELECT COUNT(\*) INTO emp\_count FROM Employees;

RETURN emp\_count;

END;

END EmployeePackage;

# **Triggers:**

CREATE TRIGGER trg\_before\_insert

BEFORE INSERT ON Employees

FOR EACH ROW

BEGIN

IF :NEW.Salary < 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Salary cannot be negative.');

END IF;

END;

# **Views:**

CREATE VIEW ActiveEmployees AS

SELECT Name, Department, Salary

FROM Employees

WHERE Salary > 0;

# **Cursors (Implicit and Explicit):**

BEGIN

SELECT Salary INTO total\_salary FROM Employees WHERE Employee\_ID = 1;

DBMS\_OUTPUT.PUT\_LINE('Total Salary: ' || total\_salary);

END;

DECLARE

CURSOR emp\_cursor IS SELECT Name, Salary FROM Employees;

emp\_name Employees.Name%TYPE;

emp\_salary Employees.Salary%TYPE;

BEGIN

OPEN emp\_cursor;

LOOP

FETCH emp\_cursor INTO emp\_name, emp\_salary;

EXIT WHEN emp\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(emp\_name || ': ' || emp\_salary);

END LOOP;

CLOSE emp\_cursor;

END;

# **Records:**

DECLARE

emp\_rec Employees%ROWTYPE;

BEGIN

SELECT \* INTO emp\_rec FROM Employees WHERE Employee\_ID = 1;

DBMS\_OUTPUT.PUT\_LINE('Name: ' || emp\_rec.Name || ', Salary: ' || emp\_rec.Salary);

END;

# **Exceptions:**

BEGIN

UPDATE Employees SET Salary = -1 WHERE Employee\_ID = 1;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

END;

# **Variables:**

DECLARE

emp\_name VARCHAR2(50);

emp\_salary DECIMAL(10, 2);

BEGIN

emp\_name := 'John Doe';

emp\_salary := 50000;

DBMS\_OUTPUT.PUT\_LINE(emp\_name || ': ' || emp\_salary);

END;

# **Collections (Associative Arrays, Nested Tables, Varrays):**

DECLARE

TYPE salary\_array IS TABLE OF DECIMAL INDEX BY PLS\_INTEGER;

salaries salary\_array;

BEGIN

salaries(1) := 50000;

salaries(2) := 60000;

DBMS\_OUTPUT.PUT\_LINE('First Salary: ' || salaries(1));

END;

--NESTED TABLE

DECLARE

TYPE salary\_table IS TABLE OF DECIMAL;

salaries salary\_table := salary\_table(50000, 60000, 70000);

BEGIN

FOR i IN 1..salaries.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE('Salary ' || i || ': ' || salaries(i));

END LOOP;

END;

DECLARE

TYPE salary\_varray IS VARRAY(3) OF DECIMAL;

salaries salary\_varray := salary\_varray(50000, 60000, 70000);

BEGIN

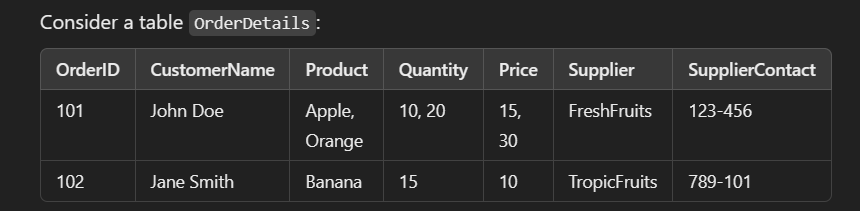
FOR i IN 1..salaries.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE('Salary ' || i || ': ' || salaries(i));

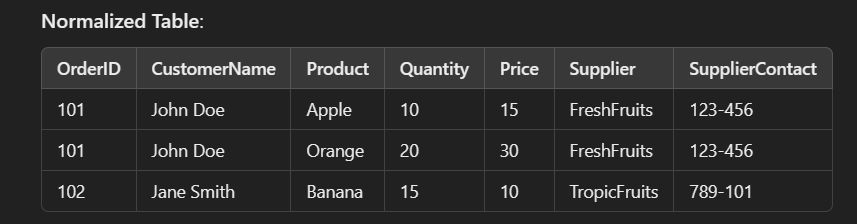
END LOOP;

END;

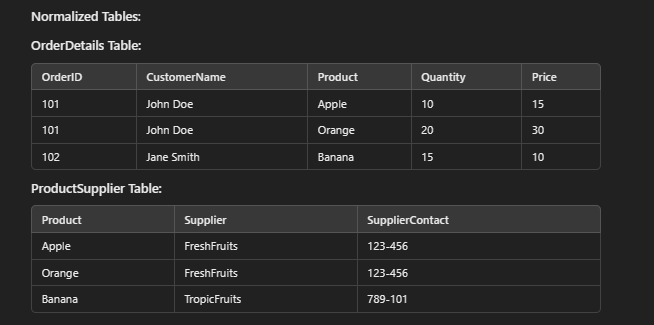
# **NORMALIZATION:**



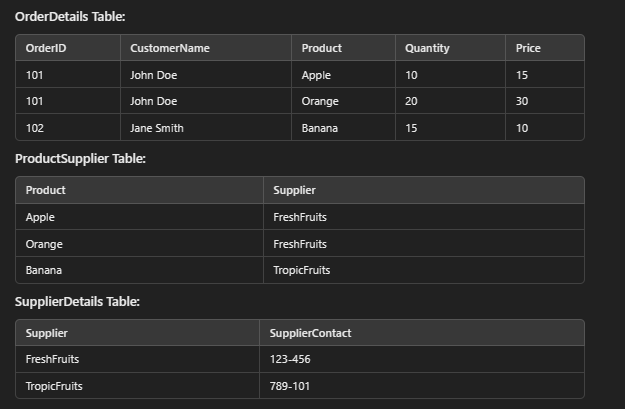
1NF:



2NF



3NF



ANS:

1NF:

CREATE TABLE OrderDetails\_1NF (

OrderID INT,

CustomerName VARCHAR(50),

Product VARCHAR(50),

Quantity INT,

Price DECIMAL(10, 2),

Supplier VARCHAR(50),

SupplierContact VARCHAR(20)

);

INSERT INTO OrderDetails\_1NF

VALUES

(101, 'John Doe', 'Apple', 10, 15, 'FreshFruits', '123-456'),

(101, 'John Doe', 'Orange', 20, 30, 'FreshFruits', '123-456'),

(102, 'Jane Smith', 'Banana', 15, 10, 'TropicFruits', '789-101');

2NF:

CREATE TABLE OrderDetails\_2NF (

OrderID INT,

CustomerName VARCHAR(50),

Product VARCHAR(50),

Quantity INT,

Price DECIMAL(10, 2),

PRIMARY KEY (OrderID, Product)

);

CREATE TABLE ProductSupplier (

Product VARCHAR(50) PRIMARY KEY,

Supplier VARCHAR(50),

SupplierContact VARCHAR(20)

);

INSERT INTO OrderDetails\_2NF

VALUES

(101, 'John Doe', 'Apple', 10, 15),

(101, 'John Doe', 'Orange', 20, 30),

(102, 'Jane Smith', 'Banana', 15, 10);

INSERT INTO ProductSupplier

VALUES

('Apple', 'FreshFruits', '123-456'),

('Orange', 'FreshFruits', '123-456'),

('Banana', 'TropicFruits', '789-101');

**3NF**

CREATE TABLE SupplierDetails (

Supplier VARCHAR(50) PRIMARY KEY,

SupplierContact VARCHAR(20)

);

INSERT INTO SupplierDetails

VALUES

('FreshFruits', '123-456'),

('TropicFruits', '789-101');

ALTER TABLE ProductSupplier

DROP COLUMN SupplierContact;

INSERT INTO ProductSupplier

VALUES

('Apple', 'FreshFruits'),

('Orange', 'FreshFruits'),

('Banana', 'TropicFruits');

