DBMS LAB OBSERVATION:

Experiment By. `Darshan Suresh` :

Aim:

To demonstrate creation, manipulation, and retrieval of data in relational databases using **MySQL** and **PL/SQL**, including implementation of **tables, constraints, queries, joins, and stored procedures**.

Pre-requisites:

- Basic understanding of **SQL commands** (DDL, DML, DCL, TCL).
- MySQL server or Oracle Database installed.
- SQL client tools like **MySQL Workbench**, **Oracle SQL Developer**, or **PL/SQL Developer**.
- Knowledge of **primary keys, foreign keys, constraints**, and basic programming constructs for PL/SQL.

Theory/Concepts:

- **SQL (Structured Query Language):**
  Language used to interact with relational databases.
  * **DDL (Data Definition Language):** `CREATE`, `ALTER`, `DROP`.
  * **DML (Data Manipulation Language):** `INSERT`, `UPDATE`, `DELETE`, `SELECT`.
  * **DCL (Data Control Language):** `GRANT`, `REVOKE`.
  * **TCL (Transaction Control Language):** `COMMIT`, `ROLLBACK`, `SAVEPOINT`.

- **PL/SQL (Procedural Language for SQL):**
  * Procedural extension for SQL in Oracle.
  * Supports **variables, loops, conditional statements**, and **stored procedures/functions**.

- **Constraints:**
  `PRIMARY KEY`, `FOREIGN KEY`, `UNIQUE`, `NOT NULL`, `CHECK`.

- **Joins:**
  `INNER JOIN`, `LEFT JOIN`, `RIGHT JOIN`, `FULL OUTER JOIN`.

- **Stored Procedures & Functions (PL/SQL):**
  Encapsulate SQL queries and logic for reusability.

References (Optional):

- "Database System Concepts" by Silberschatz, Korth, Sudarshan

Oracle PL/SQL Documentation:https://docs.oracle.com/en/database/oracle/oracle-database/

- MySQL Documentation: https://dev.mysql.com/doc/
- W3Schools SQL Tutorial: https://www.w3schools.com/sql/

On Sep 1, 2025

Queries/Code:

01) sqlcommand prompt -> connect system -> enter the password

02) Create a customer table named `customer_2025` with the following specifications:

- customerid as primary key (integer)
- firstname and lastname as mandatory fields (VARCHAR 50 characters)
- middlename as optional field (VARCHAR 50 characters)
- date of birth as DATE type
- phone number with unique constraint (VARCHAR 15 characters)
- address field (VARCHAR 200 characters)
- balance field using DECIMAL(10,2) with CHECK constraint between 1500.00 and 3000.00

```
CREATE TABLE customer_2025 (
    customerid INT PRIMARY KEY,
    firstname VARCHAR(50) NOT NULL,
    middlename VARCHAR(50),
    lastname VARCHAR(50) NOT NULL,
    dob DATE,
    phonenumber VARCHAR(15) UNIQUE,
    address VARCHAR(200),
    balance DECIMAL(10,2),
    CHECK (balance BETWEEN 1500.00 AND 3000.00)
);
```

```
SQL> CREATE TABLE customer_2025 (
         customerid INT PRIMARY KEY,
         firstname VARCHAR(50) NOT NULL,
         middlename VARCHAR(50), ...
Show more...


Table CUSTOMER_2025 created.

Elapsed: 00:00:00.025
```

03) Verifying Table Creation:

```
SELECT table_name FROM user_tables WHERE table_name =
'CUSTOMER_2025';
```

	TABLE_NAME
1	CUSTOMER_2025

04) Display Table Structure:

```
DESCRIBE customer_2025;
```

```
SQL> DESCRIBE customer_2025

Name          Null?    Type
------------- -------- -------------
CUSTOMERID    NOT NULL NUMBER(38)
FIRSTNAME     NOT NULL VARCHAR2(50)
MIDDLENAME             VARCHAR2(50)
LASTNAME      NOT NULL VARCHAR2(50)
DOB                    DATE
PHONENUMBER            VARCHAR2(15)
ADDRESS                VARCHAR2(200)
BALANCE                NUMBER(10,2)
```

05) Insert Sample Data:

```
 INTO customer_2025 VALUES
  (1, 'Darshan', NULL, 'Suresh',
TO_DATE('2003-07-12','YYYY-MM-DD'),
   '9876543210', 'Mysuru, Karnataka', 3000.00)

 INTO customer_2025 VALUES
  (2, 'Aarav', 'Rajesh', 'Sharma',
TO_DATE('1995-04-20','YYYY-MM-DD'),
   '9876543211', 'Bengaluru, Karnataka', 2750.50)
```

```
  INTO customer_2025 VALUES
  (3, 'Priya', 'Anita', 'Reddy',
TO_DATE('1998-09-10','YYYY-MM-DD'),
   '9876543212', 'Hyderabad, Telangana', 1900.75)

  INTO customer_2025 VALUES
  (4, 'Vikram', 'Arun', 'Nair',
TO_DATE('1992-01-18','YYYY-MM-DD'),
   '9876543213', 'Kochi, Kerala', 2950.00)

  INTO customer_2025 VALUES
  (5, 'Meera', 'Lakshmi', 'Iyer',
TO_DATE('1997-11-05','YYYY-MM-DD'),
   '9876543214', 'Chennai, Tamil Nadu', 2100.25)
```

```
SQL> INSERT ALL
       INTO customer_2025 VALUES
       (1, 'Darshan', NULL, 'Suresh', TO_DATE('2003-07-12','YYYY-MM-DD'),
        '9876543210', 'Mysuru, Karnataka', 3000.00)...
Show more...



5 rows inserted.
```

06) Display all the data:

```
SELECT * FROM customer_2025;
```

	CUSTOMERID	FIRSTNAME	MIDDLENAME	LASTNAME	DOB	PHONENUMBER	ADDRESS	BALANCE
1	1	Darshan	*(null)*	Suresh	7/12/2003, 12:00:0(	9876543210	Mysuru, Karnataka	3000
2	2	Aarav	Rajesh	Sharma	4/20/1995, 12:00:0(	9876543211	Bengaluru, Karnataka	2750.5
3	3	Priya	Anita	Reddy	9/10/1998, 12:00:0(	9876543212	Hyderabad, Telangan	1900.75
4	4	Vikram	Arun	Nair	1/18/1992, 12:00:0(	9876543213	Kochi, Kerala	2950
5	5	Meera	Lakshmi	Iyer	11/5/1997, 12:00:0(	9876543214	Chennai, Tamil Nadu	2100.25

07) Create a cart table named cart_2025 with the following specifications:

- order_number as primary key (integer)
- product_name as a mandatory field (VARCHAR 255 characters)
- product_id as a mandatory field (integer)
- customerid as a mandatory field (integer) referencing customer_2025(customerid)
- order_date as DATE type with default value as current date
- quantity as integer with a CHECK constraint to ensure it is greater than 0
- price as DECIMAL(10,2) with a CHECK constraint to ensure it is not negative

```
CREATE TABLE cart_2025 (
    order_number INT PRIMARY KEY,
    product_name VARCHAR(255) NOT NULL,
    product_id INT NOT NULL,
    customerid INT NOT NULL,
    order_date DATE DEFAULT SYSDATE NOT NULL,
    quantity INT NOT NULL CHECK (quantity > 0),
    price DECIMAL(10,2) NOT NULL CHECK (price >= 0),
    FOREIGN KEY (customerid) REFERENCES
customer_2025(customerid)
);
```

```
SQL> CREATE TABLE cart_2025 (
         order_number INT PRIMARY KEY,
         product_name VARCHAR(255) NOT NULL,
         product_id INT NOT NULL,...
 Show more...



 Table CART_2025 created.
```

08) Verifying Table Creation:

```
SELECT table_name FROM user_tables WHERE table_name =
'cart_2025';
```

	TABLE_NAME
1	CART_2025

09) Display Table Structure:

```
DESCRIBE cart_2025;
```

```
SQL> DESCRIBE cart_2025

Name          Null?    Type
------------ -------- -------------
ORDER_NUMBER NOT NULL NUMBER(38)
PRODUCT_NAME NOT NULL VARCHAR2(255)
PRODUCT_ID   NOT NULL NUMBER(38)
CUSTOMERID   NOT NULL NUMBER(38)
ORDER_DATE   NOT NULL DATE
QUANTITY     NOT NULL NUMBER(38)
PRICE        NOT NULL NUMBER(10,2)
```

10) Insert Sample Data:

```
INSERT ALL
  INTO cart_2025 VALUES
    (1, 'Laptop', 101, 1, TO_DATE('2025-09-23','YYYY-MM-DD'), 1,
75000.00)
  INTO cart_2025 VALUES
    (2, 'Smartphone', 102, 2,
TO_DATE('2025-09-22','YYYY-MM-DD'), 2, 50000.00)
  INTO cart_2025 VALUES
```

```
    (3, 'Headphones', 103, 3,
TO_DATE('2025-09-21','YYYY-MM-DD'), 1, 2500.00)
  INTO cart_2025 VALUES
    (4, 'Keyboard', 104, 4, TO_DATE('2025-09-20','YYYY-MM-DD'),
1, 1200.00)
  INTO cart_2025 VALUES
    (5, 'Mouse', 105, 5, TO_DATE('2025-09-19','YYYY-MM-DD'), 2,
1500.00)
SELECT * FROM dual;
```

```
SQL> INSERT ALL
       INTO cart_2025 VALUES
         (1, 'Laptop', 101, 1, TO_DATE('2025-09-23','YYYY-MM-DD'), 1, 75000.00)
       INTO cart_2025 VALUES ...
Show more...



5 rows inserted.
```

11) Display all the data:

```
SELECT * FROM cart_2025;
```

	ORDER_NUMBER	PRODUCT_NAME	PRODUCT_ID	CUSTOMERID	ORDER_DATE	QUANTITY	PRICE
1	1	Laptop	101	1	9/23/2025, 12:00:0(	1	75000
2	2	Smartphone	102	2	9/22/2025, 12:00:0(	2	50000
3	3	Headphones	103	3	9/21/2025, 12:00:0(	1	2500
4	4	Keyboard	104	4	9/20/2025, 12:00:0(	1	1200
5	5	Mouse	105	5	9/19/2025, 12:00:0(	2	1500

12) Display the column customerid, customer first name, balance from the table customer_2025?

```
SELECT customerid, firstname, balance
FROM customer_2025;
```

	CUSTOMERID	FIRSTNAME	BALANCE
1	1	Darshan	3000
2	2	Aarav	2750.5
3	3	Priya	1900.75
4	4	Vikram	2950
5	5	Meera	2100.25

13) Display all the columns for the customer whose id is 1?

```
Select * from customer_2025 where customerid='1'
```

	CUSTOMERID	FIRSTNAME	MIDDLENAME	LASTNAME	DOB	PHONENUMBER	ADDRESS	BALANCE
1	1	Darshan	*(null)*	Suresh	7/12/2003, 12:00:0(	9876543210	Mysuru, Karnataka	3000

14) Display customerid, customer address, dob of customer whose id is 2?

```
select customerid, firstname, dob from customer_2025 where
customerid = 2;
```

	CUSTOMERID	FIRSTNAME	DOB
1	2	Aarav	4/20/1995, 12:00:0(

15) Display the first name of customer whose balance is less than 3000?

```
select firstname from customer_2025 where balance < 3000;
```

	FIRSTNAME
1	Aarav
2	Priya
3	Vikram
4	Meera

16) Display customer id and address of customer whose balance is not equal to 1500?

```
select customerid, address from customer_2025 where balance != 1500;
```

	CUSTOMERID	ADDRESS
1	1	Mysuru, Karnataka
2	2	Bengaluru, Karnataka
3	3	Hyderabad, Telangan
4	4	Kochi, Kerala
5	5	Chennai, Tamil Nadu

17) Display the customer id matching to DOB

```
Select customerid  from customer_2025 where dob=TO_DATE('2003-07-12','YYYY-MM-DD')
```

	CUSTOMERID
1	1

18) Display first name of customer whose name is Vikram and the balance is 2950?

```
Select firstname  from customer_2025 where firstname='Vikram'
And balance=2950;
```

	FIRSTNAME
1	Vikram

On Sep 8, 2025

Queries/Code:

19) Display customer id, firstname, product name from customer_2025 and cart table having common customer id

```
SELECT c.CUSTOMERID,
       c.FIRSTNAME,
       ca.PRODUCT_NAME
FROM CUSTOMER_2025 c
INNER JOIN CART_2025 ca
    ON c.CUSTOMERID = ca.CUSTOMERID;
```

	CUSTOMERID	FIRSTNAME	PRODUCT_NAME
1	1	Darshan	Laptop
2	2	Aarav	Smartphone
3	3	Priya	Headphones
4	4	Vikram	Keyboard
5	5	Meera	Mouse

20) Display customerid, customer firstname, customer lastname, and order date from customer by applying right outer join on both tables

```
SELECT c.CUSTOMERID,
       c.FIRSTNAME,
       c.LASTNAME,
       ca.ORDER_DATE
FROM CUSTOMER_2025 c
RIGHT OUTER JOIN CART_2025 ca
    ON c.CUSTOMERID = ca.ORDER_NUMBER;
```

	CUSTOMERID	FIRSTNAME	LASTNAME	ORDER_DATE
1	1	Darshan	Suresh	9/23/2025, 12:00:0(
2	2	Aarav	Sharma	9/22/2025, 12:00:0(
3	3	Priya	Reddy	9/21/2025, 12:00:0(
4	4	Vikram	Nair	9/20/2025, 12:00:0(
5	5	Meera	Iyer	9/19/2025, 12:00:0(

21) Display all the matched and unmatched data from tables customer and cart?

```
SELECT c.CUSTOMERID,
       c.FIRSTNAME || ' ' || c.LASTNAME AS NAME,
       c.BALANCE,
       ca.PRODUCT_ID,
       ca.PRODUCT_NAME,
       ca.PRICE
FROM CUSTOMER_2025 c
FULL OUTER JOIN CART_2025 ca
    ON c.CUSTOMERID = ca.ORDER_NUMBER;
```

	CUSTOMERID	NAME	BALANCE	PRODUCT_ID	PRODUCT_NAME	PRICE
1	1	Darshan Suresh	3000	101	Laptop	75000
2	2	Aarav Sharma	2750.5	102	Smartphone	50000
3	3	Priya Reddy	1900.75	103	Headphones	2500
4	4	Vikram Nair	2950	104	Keyboard	1200
5	5	Meera Iyer	2100.25	105	Mouse	1500

22) EXISTS statement

```
SELECT c.FIRSTNAME
FROM CUSTOMER_2025 c
WHERE EXISTS (
    SELECT 1
    FROM CART_2025 ca
    WHERE c.CUSTOMERID = ca.ORDER_NUMBER
```

```
);
```

	FIRSTNAME
1	Darshan
2	Aarav
3	Priya
4	Vikram
5	Meera

23) Display firstname whose DOB lies between 1 Jan 1200 and 1 Jan 2020?

```
SELECT FIRSTNAME
FROM CUSTOMER_2025
WHERE DOB BETWEEN DATE '1200-01-01' AND DATE '2000-01-01';
```

	FIRSTNAME
1	Aarav
2	Priya
3	Vikram
4	Meera

24) Display customer_id and firstname where balance is either 2000, 2200, or 2500?

```
SELECT CUSTOMERID, FIRSTNAME
FROM CUSTOMER_2025
WHERE BALANCE IN (2000, 2200, 3000);
```

	CUSTOMERID	FIRSTNAME
1	1	Darshan

25) Display firstname where it ends with letter ‘r’

```
SELECT FIRSTNAME
FROM CUSTOMER_2025
WHERE FIRSTNAME LIKE '%n';
```

	FIRSTNAME
1	Darshan

26) Display address where it ends with letter ‘e’

```
SELECT ADDRESS
FROM CUSTOMER_2025
WHERE ADDRESS LIKE '%a';
```

	ADDRESS
1	Mysuru, Karnataka
2	Bengaluru, Karnataka
3	Hyderabad, Telangan
4	Kochi, Kerala

28) Display firstname of customer whose last 2 letters are ‘nu’?

```
SELECT FIRSTNAME
FROM CUSTOMER_2025
WHERE FIRSTNAME LIKE '%nu';
```

	FIRSTNAME
1	Aarav

29) Calculate the total balance from the customer table?

```
SELECT SUM(BALANCE) AS TOTAL_BALANCE
FROM CUSTOMER_2025;
```

	TOTAL_BALANCE
1	12701.5

30) Calculate average balance from customer table?

```
SELECT AVG(BALANCE) AS AVERAGE_BALANCE
FROM CUSTOMER_2025;
```

	AVERAGE_BALANCE
1	2540.3

31) Find max balance from customer table?

```
SELECT MAX(BALANCE) AS MAX_BALANCE
FROM CUSTOMER_2025;
```

	MAX_BALANCE
1	3000

32) **Find min balance from customer table?**

```
SELECT MIN(BALANCE) AS MIN_BALANCE
FROM CUSTOMER_2025;
```

	MIN_BALANCE
1	1900.75

33) Count number of records in customer table?

```
SELECT COUNT(*) AS TOTAL_CUSTOMERS
FROM CUSTOMER_2025;
```

	TOTAL_CUSTOMERS
1	5

34) Concatenate the firstname and lastname of customer?

```
SELECT FIRSTNAME || ' ' || LASTNAME AS FULLNAME
FROM CUSTOMER_2025;
```

	FULLNAME
1	Darshan Suresh
2	Aarav Sharma
3	Priya Reddy
4	Vikram Nair
5	Meera Iyer

35) **Capitalize the firstname of customer**

```
SELECT UPPER(FIRSTNAME) AS CAPITALIZED_FIRSTNAME
FROM CUSTOMER_2025;
```

	CAPITALIZED_FIRST
1	DARSHAN
2	AARAV
3	PRIYA
4	VIKRAM
5	MEERA

36)**Convert customer lastname to lowercase**

```
SELECT LOWER(LASTNAME) AS LOWERCASE_LASTNAME
```

```
FROM CUSTOMER_2025;
```

	LOWERCASE_LASTI
1	suresh
2	sharma
3	reddy
4	nair
5	iyer

37)Find the length of customer firstname

```
SELECT FIRSTNAME, LENGTH(FIRSTNAME) AS NAME_LENGTH
FROM CUSTOMER_2025;
```

	FIRSTNAME	NAME_LENGTH
1	Darshan	7
2	Aarav	5
3	Priya	5
4	Vikram	6
5	Meera	5

38) Find substring from customer firstname beginning?

```
SELECT SUBSTR(FIRSTNAME, 1, 3) AS SUBSTRING_FIRSTNAME
FROM CUSTOMER_2025;
```

	SUBSTRING_FIRSTN
1	Dar
2	Aar
3	Pri
4	Vik
5	Mee

On Sep 15, 2025

Queries/Code:

39)

```
SELECT *
FROM CART_2025
WHERE CUSTOMERID IN (
    SELECT CUSTOMERID
    FROM CUSTOMER_2025
    WHERE FIRSTNAME IN ('Darshan','Priya')
);
```

	ORDER_NUMBER	PRODUCT_NAME	PRODUCT_ID	CUSTOMERID	ORDER_DATE	QUANTITY	PRICE
1	1	Laptop	101	1	9/23/2025, 12:00:0(	1	75000
2	3	Headphones	103	3	9/21/2025, 12:00:0(	1	2500

40) **List customerid, firstname, lastname who ordered product ‘Laptop’ or ‘Keyboard’.**

```
SELECT DISTINCT c.CUSTOMERID,
       c.FIRSTNAME,
       c.LASTNAME
FROM CUSTOMER_2025 c
JOIN CART_2025 ca
    ON c.CUSTOMERID = ca.CUSTOMERID
WHERE ca.PRODUCT_NAME IN ('Laptop', 'Keyboard');
```

	CUSTOMERID	FIRSTNAME	LASTNAME
1	1	Darshan	Suresh
2	4	Vikram	Nair

41) **List customerid, phonenumber who ordered products based on the given dates?.**

```
SELECT customerid, phonenumber
FROM customer_2025
WHERE customerid IN (
    SELECT customerid
    FROM cart_2025
    WHERE ORDER_DATE IN (
        TO_DATE('04/21/2025','MM/DD/YYYY'),
        TO_DATE('04/12/2025','MM/DD/YYYY'),
        TO_DATE('09/26/2025','MM/DD/YYYY')
    )
);
```

CUSTOMERID	PHONENUMBER
No items to display.	

42) **List customerid, firstname, lastname who ordered product 'Keyboard' concatenate 'Mouse'.**

```
select * from customer_2025 where CUSTOMERID in (Select
product_id from CART_2025 where product_name = 'Keyboard' ||
'Mouse');
```

CUSTOMERID	FIRSTNAME	MIDDLENAME	LASTNAME	DOB	PHONENUMBER	ADDRESS	BALANCE
No items to display.							

43) **List firstname who has firstname 'Darshan' and balance '3000'.**

```
Select firstname  from customer_2025 where firstname='Darshan'
And balance=3000
```

	FIRSTNAME
1	Darshan

44) Adding new rows, Passport Number and District to the Customer_2025 table?

```
Alter table customer_2025 add (passportno VARCHAR(20), district
varchar(20))
```

```
SQL> Alter table customer_2025 add (passportno VARCHAR(20), district varchar(20))


Table CUSTOMER_2025 altered.

Elapsed: 00:00:00.038
```

45) Updating new rows with new data?

```
UPDATE customer_2025
SET
    passportno = CASE customerid
                    WHEN 1 THEN 'P100001'
                    WHEN 2 THEN 'P100002'
                    WHEN 3 THEN 'P100003'
                    WHEN 4 THEN 'P100004'
                    WHEN 5 THEN 'P100005'
                 END,
    district = CASE customerid
                    WHEN 1 THEN 'Mysuru'
                    WHEN 2 THEN 'Bengaluru'
                    WHEN 3 THEN 'Hyderabad'
                    WHEN 4 THEN 'Kochi'
                    WHEN 5 THEN 'Chennai'
               END
```

```
WHERE customerid IN (1,2,3,4,5);
```

```
SQL> UPDATE customer_2025
     SET
         passportno = CASE customerid
                          WHEN 1 THEN 'P100001'...
Show more...
```

```
5 rows updated.

Elapsed: 00:00:00.010
```

On Sep 22, 2025

Queries/Code:

46) Creating a VIEW for customerid, productname using customer_2025 and cart_2025 tables?

```
CREATE VIEW customer_cartdata AS
SELECT
    cust.customerid AS "Customer Id",
    cart.product_name
FROM customer_2025 cust
JOIN cart_2025 cart
    ON cust.customerid = cart.customerid;
```

```
SQL> CREATE VIEW customer_cartdata AS
     SELECT
         cust.customerid AS "Customer Id",
         cart.product_name...
Show more...



View CUSTOMER_CARTDATA created.

Elapsed: 00:00:00.010
```

47) Dropping View table customer_cartdata?

```
DROP VIEW customer_cartdata;
```

```
SQL> DROP VIEW customer_cartdata


View CUSTOMER_CARTDATA dropped.

Elapsed: 00:00:00.020
```

48) Query Inside a Query to display firstname, district from customer_2025 table considering balance of max(balance) from customer_2025 table?

```
select firstname, address, balance, district from CUSTOMER_2025
where balance = (Select max(Balance) from customer_2025);
```

	FIRSTNAME	ADDRESS	BALANCE	DISTRICT
1	Darshan	Mysuru, Karnataka	3000	Mysuru

49) Usage of Commit

```
commit;
```

```
SQL> commit


Commit complete.

Elapsed: 00:00:00.001
```

About Commit:

1. Purpose

- `COMMIT` is used to **save all changes** made in the current transaction to the database permanently.
- Without `COMMIT`, changes are **temporary** and can be undone.

---

2. When to Use

- After `INSERT`, `UPDATE`, or `DELETE` statements when you are sure the changes should be permanent.
- Before ending a session to ensure all modifications are saved.

---

3. Transaction Control

- `COMMIT` **ends the current transaction** and starts a new one.
- After `COMMIT`, you **cannot roll back** the changes.

---

4. Syntax

```
COMMIT;
```

---

5. Auto-commit vs Manual Commit

- **Auto-commit:** Some tools automatically commit after each statement (like in MySQL Workbench by default).

- **Manual commit:** Oracle requires an explicit `COMMIT` unless auto-commit is enabled.

---

6. Related Commands

- `ROLLBACK` → Undo changes in the current transaction.
- `SAVEPOINT` → Mark a point in a transaction to roll back to, without undoing the entire transaction.

---

7. Best Practices

- Commit **only when changes are verified** to prevent errors.
- Use `COMMIT` after a logical unit of work (like updating multiple related tables).

50) PL/SQL

```
SET SERVEROUTPUT ON

DECLARE

    v_customer_id   customer_2025.customerid%TYPE := 1;
    v_first_name    customer_2025.firstname%TYPE;
    v_last_name     customer_2025.lastname%TYPE;
    v_balance       customer_2025.balance%TYPE;

BEGIN

    SELECT
        firstname,
        lastname,
        balance
```

```
    INTO
        v_first_name,
        v_last_name,
        v_balance
    FROM
        customer_2025
    WHERE
        customerid = v_customer_id;

    DBMS_OUTPUT.PUT_LINE('Customer Information:');
    DBMS_OUTPUT.PUT_LINE('----------------------');
    DBMS_OUTPUT.PUT_LINE('Name: ' || v_first_name || ' ' ||
v_last_name);
    DBMS_OUTPUT.PUT_LINE('Balance: ' || v_balance);

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Error: No customer found with ID '
|| v_customer_id);
    -- Handle any other potential errors
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' ||
SQLERRM);
END;
```

```
Customer Information:
----------------------
Name: Darshan Suresh
Balance: 3000


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.011
```

On Sep 29, 2025

Queries/Code:

51)
```
create table parof_2025(parent varchar(30), name varchar(30));
```

52)
```
CREATE TABLE ancestor_2025 (
    parent VARCHAR(30),
    name VARCHAR(30),
    grandparent VARCHAR(30),
    child VARCHAR(30)
)
```

53)
```
insert into PAROF_2025(parent,name) values('alice','carol');
insert into PAROF_2025(parent,name) values('bob','carol');
insert into PAROF_2025(parent,name) values('carol','dave');
insert into PAROF_2025(parent,name) values('carol','george');
insert into PAROF_2025(parent,name) values('dave','mary');
insert into PAROF_2025(parent,name) values('eve','mary');
insert into PAROF_2025(parent,name) values('mary','frank');
```

54)
```
 select * from parof_2025;
```

55)
```
WITH ancestor_2025 (parent) AS (
    SELECT parent FROM parof_2025 WHERE name = 'frank'
    UNION ALL
    SELECT p.parent
    FROM ancestor_2025 a
```

```
    JOIN parof_2025 p ON a.parent = p.name
)
SELECT parent FROM ancestor_2025
```

56) Query with Countup (dual is oracle based dummy table)

```
WITH countup (n) AS ( SELECT 1 FROM dual UNION ALL SELECT n + 1
FROM countup WHERE n + 1 < 3)
SELECT n FROM countup;
```

57) Leap Year

```
SET SERVEROUTPUT ON
DECLARE
 yr NUMBER := &year;
 BEGIN
 IF MOD(yr,400)=0 OR (MOD(yr,4)=0 AND MOD(yr,100)<>0) THEN
 DBMS_OUTPUT.PUT_LINE(yr || ' is a Leap Year'); ELSE
 DBMS_OUTPUT.PUT_LINE(yr || ' is NOT a Leap Year'); END IF;
END;
/
```

58) Arithmetic, Relational, Logical Operators

```
SET SERVEROUTPUT ON
DECLARE
 a NUMBER := 20;
 b NUMBER := 10;
BEGIN
 DBMS_OUTPUT.PUT_LINE('Addition: ' || (a+b));
DBMS_OUTPUT.PUT_LINE('Subtraction: ' || (a-b));
DBMS_OUTPUT.PUT_LINE('Multiplication: ' || (a*b));
DBMS_OUTPUT.PUT_LINE('Division: ' || (a/b));
 IF a > b THEN
 DBMS_OUTPUT.PUT_LINE('a is greater than b'); END IF;
 IF (a > 10 AND b < 15) THEN
 DBMS_OUTPUT.PUT_LINE('Logical AND condition is TRUE'); END IF;
```

```
END;
/
```

59) Character, Number, Special Character

```
SET SERVEROUTPUT ON;
DECLARE
  ch CHAR(1) := '&input_char';
BEGIN
  IF ch BETWEEN '0' AND '9' THEN
    DBMS_OUTPUT.PUT_LINE(ch || ' IS A NUMBER');
  ELSIF ch BETWEEN 'A' AND 'Z' OR ch BETWEEN 'a' AND 'z' THEN
    DBMS_OUTPUT.PUT_LINE(ch || ' IS A CHARACTER');
  ELSE
    DBMS_OUTPUT.PUT_LINE(ch || ' IS A SPECIAL CHARACTER');
  END IF;
END;
/
```

60) Largest of 2 Numbers

```
SET SERVEROUTPUT ON;
DECLARE
a NUMBER := &a;
b NUMBER := &b;
BEGIN
    IF a>b THEN
    DBMS_OUTPUT.PUT_LINE(a || ' is Larger');
    ELSE
    DBMS_OUTPUT.PUT_LINE(b || ' is Larger');
    END IF;
END;
/
```

61) Largest of 3 Numbers

```
SET SERVEROUTPUT ON;
DECLARE
  a NUMBER := &a;
```

```
  b NUMBER := &b;
  c NUMBER := &c;
BEGIN
  IF a > b AND a > c THEN
    DBMS_OUTPUT.PUT_LINE(a || ' is Largest');
  ELSIF b > a AND b > c THEN
    DBMS_OUTPUT.PUT_LINE(b || ' is Largest');
  ELSE
    DBMS_OUTPUT.PUT_LINE(c || ' is Largest');
  END IF;
END;
/
```

62) Odd or Even

```
SET SERVEROUTPUT ON;
DECLARE
  N NUMBER := &num;
BEGIN
  IF MOD(n,2)=0 THEN
    DBMS_OUTPUT.PUT_LINE(N || ' is Even');
  ELSE
    DBMS_OUTPUT.PUT_LINE(N || ' is Odd');
  END IF;
END;
/
```

63) Grade Using Nested IF

```
SET SERVEROUTPUT ON;

DECLARE
 marks NUMBER := &marks;
 BEGIN
 IF marks >= 90 THEN
 DBMS_OUTPUT.PUT_LINE('Grade A');
```

```
 ELSIF marks >= 75 THEN
 DBMS_OUTPUT.PUT_LINE('Grade B');
 ELSIF marks >= 50 THEN
 DBMS_OUTPUT.PUT_LINE('Grade C');
 ELSE
 DBMS_OUTPUT.PUT_LINE('Fail');
 END IF;
END;
/
```

64) Voting Eligibility

```
SET SERVEROUTPUT ON;

DECLARE
 age NUMBER := &age;
 BEGIN
 IF age >= 18 THEN
 DBMS_OUTPUT.PUT_LINE('Eligible to Vote');
 ELSE
 DBMS_OUTPUT.PUT_LINE('Not Eligible to Vote');
 END IF;
END;
/
```

65) Sum of N Number (FOR loop)

```
SET SERVEROUTPUT ON;

DECLARE
 n NUMBER := &n;
 sum NUMBER := 0;
BEGIN
 FOR i IN 1..n LOOP sum := sum + i;
 END LOOP;
 DBMS_OUTPUT.PUT_LINE('Sum = ' || sum);
```

```
 END;
/
```

66) Multiplication Table

```
SET SERVEROUTPUT ON;
DECLARE
 n NUMBER := 5;
BEGIN
 FOR i IN 1..10 LOOP
 DBMS_OUTPUT.PUT_LINE(n || ' x ' || i || ' = ' || n*i); END
LOOP;
END;
```

67) Sum of Even, Odd, Prime Numbers

```
SET SERVEROUTPUT ON;
DECLARE
 n NUMBER := &n;
 sum_even NUMBER := 0; sum_odd NUMBER := 0; sum_prime NUMBER :=
0; flag NUMBER;
BEGIN
 FOR i IN 1..n LOOP IF MOD(i,2)=0 THEN sum_even := sum_even + i;
ELSE sum_odd := sum_odd + i;
 END IF;
 -- Prime check
 flag := 0;
 IF i > 1 THEN
 FOR j IN 2..i-1 LOOP IF MOD(i,j)=0 THEN flag := 1; END IF;
 END LOOP;
 IF flag=0 THEN sum_prime := sum_prime + i; END IF;
 END IF;
 END LOOP;
 DBMS_OUTPUT.PUT_LINE('Sum Even = ' || sum_even);
DBMS_OUTPUT.PUT_LINE('Sum Odd = ' || sum_odd);
DBMS_OUTPUT.PUT_LINE('Sum Prime = ' || sum_prime);
```

```
END;
/
```

68) Fibonacci Series

```
SET SERVEROUTPUT ON;
DECLARE
 n NUMBER := &n;
 a NUMBER := 0;
 b NUMBER := 1;
 c NUMBER;
BEGIN
 DBMS_OUTPUT.PUT_LINE(a); DBMS_OUTPUT.PUT_LINE(b); FOR i IN 3..n
LOOP c := a + b;
 DBMS_OUTPUT.PUT_LINE(c); a := b; b := c;
 END LOOP;
END;
/
```

7

0
1
1
2
3
5
8

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.008

69) Factorial

```
SET SERVEROUTPUT ON;
DECLARE
 n NUMBER := &n;
 fact NUMBER := 1;BEGIN
```

```
 FOR i IN 1..n LOOP fact := fact * i; END LOOP;
 DBMS_OUTPUT.PUT_LINE('Factorial = ' || fact);END;
/
```

Factorial = 120

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.007

70) Reverse a Number

```
SET SERVEROUTPUT ON;
DECLARE
 n NUMBER := &n;
 rev NUMBER := 0;
 rem NUMBER;
BEGIN
 WHILE n > 0 LOOP
 rem := MOD(n,10); rev := rev*10 + rem; n := TRUNC(n/10); END
LOOP;
 DBMS_OUTPUT.PUT_LINE('Reverse = ' || rev);END;
/
```

Reverse = 4321

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.007

71) Write a PL/SQL block using IF to determine the output of the following code:

```
SET SERVEROUTPUT ON;
Declare
num1 number:= 10;
num2 number:= 20;
BEGIN
if num1 > num2 then
```

```
dbms_output.put_line('num1 small');end
if;dbms_output.put_line('I am Not in if');
end;
/
```

I am Not in if

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.005

72) Write a PL/SQL block using IF..Then..Else to determine the output of the following code:

```
SET SERVEROUTPUT ON;
Declare
num1 number:= 10;
num2 number:= 20;
BEGIN
if num1 < num2 then
dbms_output.put_line('i am in if block');
ELSE
dbms_output.put_line('i am in else Block');
end if;
dbms_output.put_line('i am not in if or else Block');
end;
/
```

i am in if block
i am not in if or else Block

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.007

73) Write a PL/SQL block using NESTED..IF..Then to determine the output of the following

```
SET SERVEROUTPUT ON;
Declare
num1 number:= 10;
num2 number:= 20;
num3 number:= 20;
BEGIN
if num1 < num2 then
dbms_output.put_line('num1 small num2'); if num1 < num3 then
 dbms_output.put_line('num1 small num3 also');end if;end
if;dbms_output.put_line('after end if');
end;
/
```

num1 small num2
num1 small num3 also
after end if

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.008

74) Write a PL/SQL block using IF..Then..ELSEIF..LADDER to determine the output of the following code:

```
SET SERVEROUTPUT ON;
Declare
  num1 number := 10;
  num2 number := 20;
```

```
BEGIN
  if num1 < num2 then
    dbms_output.put_line('num1 small');
  ELSEIF num1 = num2 then
    dbms_output.put_line('both equal');
  ELSE
    dbms_output.put_line('num2 greater');
  end if;
  dbms_output.put_line('after end if');
END;
/
```

75) Write a PL/SQL block using EXIT to determine the output of the following code:

```
SET SERVEROUTPUT ON;
DECLARE
 counter NUMBER := 1;
BEGIN
 LOOP
 DBMS_OUTPUT.PUT_LINE('This is iteration number ' || counter);
 IF counter = 3 THEN
 EXIT;
 END IF;
 counter := counter + 1; END LOOP;
END;
/
```

This is iteration number 1
This is iteration number 2
This is iteration number 3

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.006

76) Write a PL/SQL block using EXIT..When.. to determine the output of the following code:

```
SET SERVEROUTPUT ON;
DECLARE
 counter NUMBER := 1;
BEGIN
 LOOP
 DBMS_OUTPUT.PUT_LINE('I LIKE DBMS PROGRAMMING'); counter :=
counter + 1;
 EXIT WHEN counter > 5;
 END LOOP;
END;
/
```

I LIKE DBMS PROGRAMMING
I LIKE DBMS PROGRAMMING
I LIKE DBMS PROGRAMMING
I LIKE DBMS PROGRAMMING
I LIKE DBMS PROGRAMMING

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.005

77) Write a PL/SQL block using CASE to determine the grade of a student based on marks:
a. ≥90 → "Excellent"
b. 75–89 → "Good"
c. 60–74 → "Average"
d. <60 → "Poor"

```
SET SERVEROUTPUT ON;
DECLARE
 p_marks NUMBER := 82;
 v_grade VARCHAR2(20);
BEGIN
 CASE
 WHEN p_marks >= 90 THEN v_grade := 'Excellent';
```

```
 WHEN p_marks BETWEEN 75 AND 89 THEN v_grade := 'Good';
 WHEN p_marks BETWEEN 60 AND 74 THEN v_grade := 'Average';
 WHEN p_marks < 60 THEN v_grade := 'Poor';
 ELSE
 v_grade := 'Invalid Marks';
 END CASE;
 DBMS_OUTPUT.PUT_LINE('Marks: ' || p_marks || ', Grade: ' ||
v_grade);
END;
/
```

```
Marks: 82, Grade: Good

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.008
```

78) Write a PL/SQL block using CASE to calculate bonus for employees:
a. Manager → 20% of salary
b. Developer → 15% of salary
c. Clerk → 10% of salary
d. Others → 5% of salary

```
SET SERVEROUTPUT ON;
DECLARE
  v_job    VARCHAR2(20) := 'Manager'; -- Specify a job to get a
specific result
  v_salary NUMBER := 50000;
  v_bonus  NUMBER;
BEGIN
  CASE v_job
    WHEN 'Manager'   THEN v_bonus := v_salary * 0.20;
    WHEN 'Developer' THEN v_bonus := v_salary * 0.15;
    WHEN 'Clerk'     THEN v_bonus := v_salary * 0.10;
    ELSE                  v_bonus := v_salary * 0.05;
  END CASE;
```

```
  DBMS_OUTPUT.PUT_LINE('Job Role: ' || v_job);
  DBMS_OUTPUT.PUT_LINE('Salary: ' || v_salary);
  DBMS_OUTPUT.PUT_LINE('Bonus: ' || v_bonus);
END;
/
```

Marks: 82, Grade: Good

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.008

79) Write a PL/SQL program using a FOR loop to print the multiplication table of a give number (e.g., 5).

```
SET SERVEROUTPUT ON;
DECLARE
 v_number NUMBER := 5; -- The number for which to print the
multiplication table
BEGIN
 DBMS_OUTPUT.PUT_LINE('Multiplication Table of ' || v_number);
-- Added semicolon here
 DBMS_OUTPUT.PUT_LINE('--------------------------');
 FOR i IN 1..10 LOOP
  DBMS_OUTPUT.PUT_LINE(v_number || ' * ' || i || ' = ' ||
(v_number * i));
 END LOOP;
END;
/
```

Multiplication Table of 5
---
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25

5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.007

80) Write a PL/SQL block that uses a FOR loop to calculate the sum of the first 100 natural numbers.

```
SET SERVEROUTPUT ON;
DECLARE
  v_sum NUMBER := 0; -- Declare a variable to store the sum,
initialized to 0
BEGIN
  -- Loop from 1 to 100 (inclusive)
  FOR i IN 1..100 LOOP
    v_sum := v_sum + i; -- Add the current number to the sum
  END LOOP;

  -- Display the final sum
  DBMS_OUTPUT.PUT_LINE('The sum of the first 100 natural numbers
is: ' || v_sum);
END;
/
```

The sum of the first 100 natural numbers is: 5050

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.006

81) Write a PL/SQL block that uses a Nested FOR loop with continues iteratn to calculate the sum of the first 100 natural numbers

```
SET SERVEROUTPUT ON;
```

```
BEGIN
 FOR outer_counter IN 1..3 LOOP
  DBMS_OUTPUT.PUT_LINE('Outer Loop - Iteration ' ||
outer_counter); -- Added semicolon
  FOR inner_counter IN 1..2 LOOP
   DBMS_OUTPUT.PUT_LINE('Inner Loop - Iteration ' ||
inner_counter); -- Added semicolon
  END LOOP;
 END LOOP;
END;
/
```

Outer Loop - Iteration 1
Inner Loop - Iteration 1
Inner Loop - Iteration 2
Outer Loop - Iteration 2
Inner Loop - Iteration 1
Inner Loop - Iteration 2
Outer Loop - Iteration 3
Inner Loop - Iteration 1
Inner Loop - Iteration 2

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.006

82) Write a PL/SQL program using a WHILE loop to reverse a given number.

```
SET SERVEROUTPUT ON;
DECLARE
  v_number NUMBER := 12345; -- The number to be reversed
  v_reversed_num NUMBER := 0; -- Stores the reversed number
  v_remainder NUMBER; -- Stores the remainder after division
BEGIN
  DBMS_OUTPUT.PUT_LINE('Original number: ' || v_number);

  WHILE v_number > 0 LOOP
    -- Get the last digit (remainder when divided by 10)
```

```
    v_remainder := MOD(v_number, 10);

    -- Build the reversed number
    v_reversed_num := (v_reversed_num * 10) + v_remainder;

    -- Remove the last digit from the original number
    v_number := TRUNC(v_number / 10);
  END LOOP;

  DBMS_OUTPUT.PUT_LINE('Reversed number: ' || v_reversed_num);
END;
/
```

Original number: 12345
Reversed number: 54321

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.008

83) Write a PL/SQL block that uses a WHILE loop to find the factorial of a given number

```
SET SERVEROUTPUT ON;
DECLARE
  p_number NUMBER := 5; -- The number for which to calculate the
factorial
  v_factorial NUMBER := 1; -- Variable to store the calculated
factorial
  v_counter NUMBER; -- Counter for the loop
BEGIN
  -- Initialize the counter with the input numbe
  v_counter := p_number;

  -- Handle edge case for 0! which is 1
  IF p_number = 0 THEN
    v_factorial := 1;
  ELSE
    -- Loop while the counter is greater than 0
```

```
    WHILE v_counter > 0 LOOP
      v_factorial := v_factorial * v_counter; -- Multiply
factorial by current countervalue
      v_counter := v_counter - 1; -- Decrement the counter
    END LOOP;
  END IF;

  -- Display the result
  DBMS_OUTPUT.PUT_LINE('Factorial of ' || p_number || ' is: ' ||
v_factorial);
END;
/
```

Factorial of 5 is: 120

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.003

84) Write a PL/SQL functio that accepts a student's roll number and returns their total marks from the StudentMarks table.

```
-- Disable server output for object creation to keep logs clean
SET SERVEROUTPUT OFF;

-- Drop the function if it already exists to allow for
recreation
DROP FUNCTION GET_STUDENT_TOTAL_MARKS;

-- Drop the table if it already exists to start with a clean
state
DROP TABLE Student_Marks;

-- Create the Student_Marks table
CREATE TABLE Student_Marks (
    ROLL_NUMBER NUMBER,
    MARKS NUMBER
);
```

```sql
-- Insert sample data
INSERT INTO Student_Marks (ROLL_NUMBER, MARKS) VALUES (101, 85);
INSERT INTO Student_Marks (ROLL_NUMBER, MARKS) VALUES (101, 92);
INSERT INTO Student_Marks (ROLL_NUMBER, MARKS) VALUES (102, 78);
INSERT INTO Student_Marks (ROLL_NUMBER, MARKS) VALUES (102, 88);
INSERT INTO Student_Marks (ROLL_NUMBER, MARKS) VALUES (103, 65);
COMMIT;

-- Create the PL/SQL function
CREATE OR REPLACE FUNCTION GET_STUDENT_TOTAL_MARKS (
  p_roll_number IN NUMBER
) RETURN NUMBER
IS
  v_total_marks NUMBER;
BEGIN
  SELECT NVL(SUM(MARKS), 0)
  INTO v_total_marks
  FROM Student_Marks
  WHERE ROLL_NUMBER = p_roll_number;

  RETURN v_total_marks;
EXCEPTION
  WHEN OTHERS THEN
    RAISE;
END;
/

-- Re-enable server output to display the results of the test
block
SET SERVEROUTPUT ON;

-- Test the function
DECLARE
    v_total_marks NUMBER;
BEGIN
```

```
    v_total_marks := GET_STUDENT_TOTAL_MARKS(101);
    DBMS_OUTPUT.PUT_LINE('Total marks for student 101: ' ||
v_total_marks);

    v_total_marks := GET_STUDENT_TOTAL_MARKS(102);
    DBMS_OUTPUT.PUT_LINE('Total marks for student 102: ' ||
v_total_marks);

    v_total_marks := GET_STUDENT_TOTAL_MARKS(104); -- Roll
number not in table
    DBMS_OUTPUT.PUT_LINE('Total marks for student 104: ' ||
v_total_marks);
END;
/
```

Total marks for student 101: 177
Total marks for student 102: 166
Total marks for student 104: 0

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.014

85) Write a function that accepts an employee ID and returns their annual salary (monthly salary × 12).

```
-- Disable server output for object creation to keep logs clean
SET SERVEROUTPUT OFF;

-- Drop the function if it already exists (to allow recreation)
DROP FUNCTION GET_EMPLOYEE_ANNUAL_SALARY;
```

```sql
-- Drop the table if it already exists (to start with a clean
state)
DROP TABLE Employees;

-- 1. Create the Employees table
CREATE TABLE Employees (
    EMPLOYEE_ID VARCHAR2(20) PRIMARY KEY,
    MONTHLY_SALARY NUMBER(10, 2)
);

-- 2. Insert sample employee data
INSERT INTO Employees (EMPLOYEE_ID, MONTHLY_SALARY) VALUES
('EMP001', 5000.00);
INSERT INTO Employees (EMPLOYEE_ID, MONTHLY_SALARY) VALUES
('EMP002', 6500.00);
INSERT INTO Employees (EMPLOYEE_ID, MONTHLY_SALARY) VALUES
('EMP003', 4800.00);
COMMIT;

-- 3. Create the PL/SQL function to get annual salary
CREATE OR REPLACE FUNCTION GET_EMPLOYEE_ANNUAL_SALARY (
  p_employee_id IN VARCHAR2
) RETURN NUMBER
IS
  v_monthly_salary NUMBER(10, 2);
  v_annual_salary  NUMBER(12, 2); -- Larger number to store
annual salary
BEGIN
  SELECT MONTHLY_SALARY
  INTO v_monthly_salary
  FROM Employees
  WHERE EMPLOYEE_ID = p_employee_id;

  v_annual_salary := v_monthly_salary * 12;

  RETURN v_annual_salary;
```

```
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    -- Handle the case where no employee is found for the given
ID
    DBMS_OUTPUT.PUT_LINE('Error: Employee with ID ' ||
p_employee_id || ' not found.');
    RETURN NULL; -- Or you could return 0, depending on desired
behavior
  WHEN OTHERS THEN
    -- Handle other potential errors (e.g., data type issues,
unexpected exceptions)
    DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' ||
SQLERRM);
    RAISE; -- Re-raise the exception to propagate it to the
caller
END;
/

-- Re-enable server output to display the results of the test
block
SET SERVEROUTPUT ON;

-- 4. Test the PL/SQL function
DECLARE
    v_annual_salary NUMBER;
    employee_id_1 VARCHAR2(20) := 'EMP001';
    employee_id_2 VARCHAR2(20) := 'EMP004'; -- Employee not
found
BEGIN
    -- Test with an existing employee
    v_annual_salary :=
GET_EMPLOYEE_ANNUAL_SALARY(employee_id_1);
    IF v_annual_salary IS NOT NULL THEN
```

```
        DBMS_OUTPUT.PUT_LINE('The annual salary for employee '
|| employee_id_1 || ' is: $' || TO_CHAR(v_annual_salary,
'FM999,999,990.00'));
    END IF;

    -- Test with a non-existent employee
    v_annual_salary :=
GET_EMPLOYEE_ANNUAL_SALARY(employee_id_2);
    IF v_annual_salary IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE('The annual salary for employee '
|| employee_id_2 || ' is: $' || TO_CHAR(v_annual_salary,
'FM999,999,990.00'));
    END IF;

END;
/
```

The annual salary for employee EMP001 is: $60,000.00
Error: Employee with ID EMP004 not found.

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.012

86) Write a PL/SQL procedure to insert a new record into the Employees(emp_id, name, salary) table.

```
-- Disable server output for object creation to keep logs clean
SET SERVEROUTPUT OFF;

-- Drop the procedure and table if they already exist
DROP PROCEDURE insert_employee;
DROP TABLE Employees;
```

```sql
-- Create the Employees table with a PRIMARY KEY
CREATE TABLE Employees (
    emp_id NUMBER PRIMARY KEY,
    name VARCHAR2(100),
    salary NUMBER
);

-- Re-enable server output to display the results of the test
block
SET SERVEROUTPUT ON;

-- Create the PL/SQL procedure to insert a new employee
CREATE OR REPLACE PROCEDURE insert_employee (
    p_emp_id IN NUMBER,
    p_name IN VARCHAR2,
    p_salary IN NUMBER
)
IS
BEGIN
    INSERT INTO Employees (emp_id, name, salary)
    VALUES (p_emp_id, p_name, p_salary);

    COMMIT; -- Commit the transaction to make the changes
permanent

    DBMS_OUTPUT.PUT_LINE('Employee ' || p_name || ' with ID ' ||
p_emp_id || ' inserted successfully.');
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        DBMS_OUTPUT.PUT_LINE('Error: Employee with ID ' ||
p_emp_id || ' already exists.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' ||
SQLERRM);
```

```
        ROLLBACK; -- Rollback the transaction in case of other
errors
END insert_employee;
/

-- Test Block for the insert_employee procedure
DECLARE
    v_new_emp_id NUMBER := 101;
    v_new_name VARCHAR2(100) := 'John Doe';
    v_new_salary NUMBER := 50000;

    v_duplicate_emp_id NUMBER := 101; -- To test the duplicate
key scenario
BEGIN
    -- Test a successful insertion
    DBMS_OUTPUT.PUT_LINE('--- Testing successful insertion
---');
    insert_employee(p_emp_id => v_new_emp_id, p_name =>
v_new_name, p_salary => v_new_salary);

    -- Test insertion of a duplicate record
    DBMS_OUTPUT.PUT_LINE(CHR(10) || '--- Testing duplicate key
insertion ---');
    insert_employee(p_emp_id => v_duplicate_emp_id, p_name =>
v_new_name, p_salary => v_new_salary);

    DBMS_OUTPUT.PUT_LINE(CHR(10) || '--- Current Employees table
data ---');
    FOR rec IN (SELECT * FROM Employees) LOOP
        DBMS_OUTPUT.PUT_LINE('Emp ID: ' || rec.emp_id || ',
Name: ' || rec.name || ', Salary: ' || rec.salary);
    END LOOP;
END;
/

--- Testing successful insertion ---
```

```
Employee John Doe with ID 101 inserted successfully.

--- Testing duplicate key insertion ---
Error: Employee with ID 101 already exists.

--- Current Employees table data ---
Emp ID: 101, Name: John Doe, Salary: 50000


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.037
```

87) Write a procedure that accepts an account number and an amount, and updates the balance after depositing the amoun

```
SET SERVEROUTPUT OFF;

DROP PROCEDURE deposit_into_account;

DROP TABLE Accounts;

CREATE TABLE Accounts (
    account_number VARCHAR2(20) PRIMARY KEY,
    balance NUMBER(10, 2) DEFAULT 0
);

INSERT INTO Accounts (account_number, balance) VALUES ('ACC001',
1000.00);
INSERT INTO Accounts (account_number, balance) VALUES ('ACC002',
500.00);
COMMIT;
```

```
CREATE OR REPLACE PROCEDURE deposit_into_account (
    p_account_number IN VARCHAR2,
    p_amount IN NUMBER
)
IS
    v_rows_updated NUMBER;
BEGIN
    UPDATE Accounts
    SET balance = balance + p_amount
    WHERE account_number = p_account_number;

    v_rows_updated := SQL%ROWCOUNT;

    IF v_rows_updated = 0 THEN

        DBMS_OUTPUT.PUT_LINE('Error: Account number ' ||
p_account_number || ' not found.');
        ROLLBACK;
    ELSE
        COMMIT;
        DBMS_OUTPUT.PUT_LINE('Amount ' || p_amount || '
deposited into account ' || p_account_number || '.');

    END IF;

EXCEPTION
    WHEN OTHERS THEN

        DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' ||
SQLERRM);
        ROLLBACK;
END deposit_into_account;
/

SET SERVEROUTPUT ON;
```

```
DECLARE
    v_account_number_1 VARCHAR2(20) := 'ACC001';
    v_deposit_amount_1 NUMBER := 250.75;

    v_account_number_2 VARCHAR2(20) := 'ACC003';
    v_deposit_amount_2 NUMBER := 100.00;

    v_current_balance NUMBER;
BEGIN
    DBMS_OUTPUT.PUT_LINE('--- Testing deposit into existing
account (' || v_account_number_1 || ') ---');
    -- Get initial balance
    SELECT balance INTO v_current_balance FROM Accounts WHERE
account_number = v_account_number_1;
    DBMS_OUTPUT.PUT_LINE('Initial balance for ' ||
v_account_number_1 || ': ' || v_current_balance);

    -- Call the procedure to deposit money
    deposit_into_account(p_account_number => v_account_number_1,
p_amount => v_deposit_amount_1);

    -- Verify new balance
    SELECT balance INTO v_current_balance FROM Accounts WHERE
account_number = v_account_number_1;
    DBMS_OUTPUT.PUT_LINE('New balance for ' ||
v_account_number_1 || ': ' || v_current_balance);

    DBMS_OUTPUT.PUT_LINE(CHR(10) || '--- Testing deposit into
non-existent account (' || v_account_number_2 || ') ---');
    -- Call the procedure for a non-existent account
    deposit_into_account(p_account_number => v_account_number_2,
p_amount => v_deposit_amount_2);

END;
/
```

--- Testing deposit into existing account (ACC001) ---
Initial balance for ACC001: 1000
Amount 250.75 deposited into account ACC001.
New balance for ACC001: 1250.75

--- Testing deposit into non-existent account (ACC003) ---
Error: Account number ACC003 not found.

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.009

On Oct 6, 2025

Queries/Code:

VARRAY of Integers – Sum, Max & Update Example

Create Type

```
CREATE TYPE score_array AS VARRAY(8) OF NUMBER;
```

PL/SQL Program

```
DECLARE
  scores score_array;
  total NUMBER := 0;
  highest NUMBER := 0;
BEGIN
  scores := score_array(12, 25, 37, 46, 59);
  FOR i IN 1..scores.COUNT LOOP
    total := total + scores(i);
    IF scores(i) > highest THEN
      highest := scores(i);
    END IF;
    DBMS_OUTPUT.PUT_LINE('Score ' || i || ': ' || scores(i));
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('Total Score = ' || total);
  DBMS_OUTPUT.PUT_LINE('Highest Score = ' || highest);
  scores(2) := 99;
  DBMS_OUTPUT.PUT_LINE('--- After Update ---');
  FOR i IN 1..scores.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE('Updated Score ' || i || ': ' || scores(i));
  END LOOP;
END;
```

Output

```
Score 1: 12
Score 2: 25
Score 3: 37
Score 4: 46
Score 5: 59
Total Score = 179
Highest Score = 59
--- After Update ---
Updated Score 1: 12
Updated Score 2: 99
Updated Score 3: 37
Updated Score 4: 46
Updated Score 5: 59
```

VARRAY of Strings – Insert, Update & Delete Simulation

Create Type

```
CREATE TYPE city_array AS VARRAY(6) OF VARCHAR2(20);
```

PL/SQL Program

```
DECLARE
  cities city_array := city_array('Mysuru', 'Bengaluru', 'Chennai');
BEGIN
  DBMS_OUTPUT.PUT_LINE('--- Original Cities ---');
  FOR i IN 1..cities.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE('City ' || i || ': ' || cities(i));
  END LOOP;
  cities.EXTEND;
  cities(4) := 'Hyderabad';
  DBMS_OUTPUT.PUT_LINE('--- After Inserting Hyderabad ---');
  FOR i IN 1..cities.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE('City ' || i || ': ' || cities(i));
  END LOOP;
```

```
  cities(3) := 'Pune';
  DBMS_OUTPUT.PUT_LINE('--- After Update ---');
  FOR i IN 1..cities.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE('Updated City ' || i || ': ' || cities(i));
  END LOOP;
END;
```

Output

```
--- Original Cities ---
City 1: Mysuru
City 2: Bengaluru
City 3: Chennai
--- After Inserting Hyderabad ---
City 1: Mysuru
City 2: Bengaluru
City 3: Chennai
City 4: Hyderabad
--- After Update ---
Updated City 1: Mysuru
Updated City 2: Bengaluru
Updated City 3: Pune
Updated City 4: Hyderabad
```

Multiple VARRAY Types – Employee Salary Sheet

```
DECLARE
  TYPE emp_names IS VARRAY(5) OF VARCHAR2(15);
  TYPE emp_salary IS VARRAY(5) OF NUMBER;
  names emp_names;
  salary emp_salary;
BEGIN
  names := emp_names('Arjun', 'Meera', 'Ravi', 'Sneha', 'Kiran');
  salary := emp_salary(45000, 52000, 61000, 48000, 70000);
  DBMS_OUTPUT.PUT_LINE('Employee Salary Sheet');
  DBMS_OUTPUT.PUT_LINE('----------------------');
```

```
  FOR i IN 1..names.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE(names(i) || ' earns ₹' || salary(i));
  END LOOP;
END;
```

Output

```
Employee Salary Sheet
---------------------
Arjun earns ₹45000
Meera earns ₹52000
Ravi earns ₹61000
Sneha earns ₹48000
Kiran earns ₹70000
```

VARRAY with %ROWTYPE + Cursor – Product List Example

Assume table:

```
CREATE TABLE products (
  pid NUMBER,
  pname VARCHAR2(20)
);
```

PL/SQL Program

```
DECLARE
  CURSOR c_prod IS SELECT pname FROM products;
  TYPE prod_list IS VARRAY(10) OF products.pname%TYPE;
  p_array prod_list := prod_list();
  counter NUMBER := 0;
BEGIN
  FOR p IN c_prod LOOP
    counter := counter + 1;
    p_array.EXTEND;
    p_array(counter) := p.pname;
    DBMS_OUTPUT.PUT_LINE('Product ' || counter || ': ' || p_array(counter));
```

```
  END LOOP;
END;
```

Output (Example)

Product 1: Keyboard
Product 2: Mouse
Product 3: Monitor
Product 4: CPU

Custom Procedure Example – Calculate Discount

Procedure

```
CREATE OR REPLACE PROCEDURE calculate_discount(
  price IN NUMBER,
  discount IN NUMBER
)
AS
  final_price NUMBER;
BEGIN
  final_price := price - (price * discount / 100);
  DBMS_OUTPUT.PUT_LINE('Original Price: ' || price);
  DBMS_OUTPUT.PUT_LINE('Discount: ' || discount || '%');
  DBMS_OUTPUT.PUT_LINE('Final Price: ' || final_price);
END;
```

Execute Procedure

```
BEGIN
  calculate_discount(1200, 15);
END;
```

Output

Original Price: 1200
Discount: 15%
Final Price: 1020

On Nov 3, 2025

Queries/Code:

Oracle Live

```
-- Cart table
CREATE TABLE Carts (
  id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  customer_id NUMBER REFERENCES Customers(id),
  total_value NUMBER(12,2),
  created_at TIMESTAMP DEFAULT SYSTIMESTAMP,
  updated_at TIMESTAMP
);
->
SQL> CREATE TABLE Carts (
       id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
       customer_id NUMBER REFERENCES Customers(id),
       total_value NUMBER(12,2),...

Show more...


Table CARTS created.

Elapsed: 00:00:00.032
```

```
-- Trigger for Carts
CREATE TRIGGER trg_carts_before_update
BEFORE UPDATE ON Carts
FOR EACH ROW
BEGIN
  :NEW.updated_at := SYSTIMESTAMP;
END;
/
->
SQL> CREATE TRIGGER trg_carts_before_update
     BEFORE UPDATE ON Carts
     FOR EACH ROW
     BEGIN...

Show more...


Trigger TRG_CARTS_BEFORE_UPDATE compiled

Elapsed: 00:00:00.074
```

```
CREATE OR REPLACE TRIGGER trg_display_cart_value_changes
AFTER INSERT OR UPDATE OR DELETE ON Carts
FOR EACH ROW
DECLARE
  val_old NUMBER;
  val_new NUMBER;
  val_diff NUMBER;
  act VARCHAR2(10);
BEGIN
  IF INSERTING THEN
    act := 'INSERT';
    val_old := NULL;
    val_new := :NEW.total_value;
    val_diff := NVL(:NEW.total_value,0);

    DBMS_OUTPUT.PUT_LINE('Action: ' || act);
    DBMS_OUTPUT.PUT_LINE('Cart ID: ' || :NEW.id);
    DBMS_OUTPUT.PUT_LINE('Customer ID: ' || :NEW.customer_id);
    DBMS_OUTPUT.PUT_LINE('Created At: ' || TO_CHAR(:NEW.created_at,
'YYYY-MM-DD HH24:MI:SS'));
    DBMS_OUTPUT.PUT_LINE('Old Value: NULL');
    DBMS_OUTPUT.PUT_LINE('New Value: ' || TO_CHAR(:NEW.total_value));
    DBMS_OUTPUT.PUT_LINE('Value Difference: ' || TO_CHAR(val_diff));

  ELSIF UPDATING THEN
    act := 'UPDATE';
    val_old := :OLD.total_value;
    val_new := :NEW.total_value;
    val_diff := NVL(:NEW.total_value,0) - NVL(:OLD.total_value,0);

    DBMS_OUTPUT.PUT_LINE('Action: ' || act);
    DBMS_OUTPUT.PUT_LINE('Cart ID: ' || :NEW.id);
    DBMS_OUTPUT.PUT_LINE('Customer ID: ' || :NEW.customer_id);
    DBMS_OUTPUT.PUT_LINE('Created At: ' || TO_CHAR(:NEW.created_at,
'YYYY-MM-DD HH24:MI:SS'));
    DBMS_OUTPUT.PUT_LINE('Updated At: ' || TO_CHAR(:NEW.updated_at,
'YYYY-MM-DD HH24:MI:SS'));
    DBMS_OUTPUT.PUT_LINE('Old Value: ' || TO_CHAR(:OLD.total_value));
    DBMS_OUTPUT.PUT_LINE('New Value: ' || TO_CHAR(:NEW.total_value));
```

```
    DBMS_OUTPUT.PUT_LINE('Value Difference: ' || TO_CHAR(val_diff));

  ELSIF DELETING THEN
    act := 'DELETE';
    val_old := :OLD.total_value;
    val_new := NULL;
    val_diff := -NVL(:OLD.total_value,0);

    DBMS_OUTPUT.PUT_LINE('Action: ' || act);
    DBMS_OUTPUT.PUT_LINE('Cart ID: ' || :OLD.id);
    DBMS_OUTPUT.PUT_LINE('Customer ID: ' || :OLD.customer_id);
    DBMS_OUTPUT.PUT_LINE('Created At: ' || TO_CHAR(:OLD.created_at,
'YYYY-MM-DD HH24:MI:SS'));
    DBMS_OUTPUT.PUT_LINE('Old Value: ' || TO_CHAR(:OLD.total_value));
    DBMS_OUTPUT.PUT_LINE('New Value: NULL');
    DBMS_OUTPUT.PUT_LINE('Value Difference: ' || TO_CHAR(val_diff));
  END IF;
END;
/
->
SQL> CREATE OR REPLACE TRIGGER trg_display_cart_value_changes
     AFTER INSERT OR UPDATE OR DELETE ON Carts
     FOR EACH ROW
     DECLARE...

Show more...


Trigger TRG_DISPLAY_CART_VALUE_CHANGES compiled

Elapsed: 00:00:00.075
```

```
-- a) Insert a cart
INSERT INTO Carts (customer_id, total_value) VALUES (1, 1200);
->
SQL> INSERT INTO Carts (customer_id, total_value) VALUES (1, 1200)

Action: INSERT
Cart ID: 21
Customer ID: 1
Created At: 2025-11-03 11:27:57
Old Value: NULL
New Value: 1200
Value Difference: 1200

1 row inserted.
```

```
Elapsed: 00:00:00.011
```

-- b) View carts
SELECT * FROM Carts;

```
->SQL> SELECT * FROM Carts
ID CUSTOMER_ID TOTAL_VALUE CREATED_AT                    UPDATED_AT
-- ----------- ----------- ---------------------------- ----------
1  1           1200        2025-11-03T11:19:47.46714Z

Elapsed: 00:00:00.001
1 rows selected.
```

-- c) Update cart value
UPDATE Carts SET total_value = 1500 WHERE id = 1;

```
->
SQL> UPDATE Carts SET total_value = 1500 WHERE id = 1

Action: UPDATE
Cart ID: 1
Old Value: 1200
New Value: 1500
Value Difference: 300

1 row updated.

Elapsed: 00:00:00.009
```

-- d) View carts after update
SELECT * FROM Carts;

```
->
SQL> SELECT * FROM Carts

ID CUSTOMER_ID TOTAL_VALUE CREATED_AT                    UPDATED_AT
-- ----------- ----------- ---------------------------- ----------------------------
1  1           1500        2025-11-03T11:19:47.46714Z   2025-11-03T11:20:38.655353Z

Elapsed: 00:00:00.002
1 rows selected.
```

-- f) Delete cart
DELETE FROM Carts WHERE id = 1;

```
->
SQL> DELETE FROM Carts WHERE id = 1

Action: DELETE
Cart ID: 1
Old Value: 1500
New Value: NULL
Value Difference: -1500
```

```
1 row deleted.

Elapsed: 00:00:00.004
```

On Nov 10, 2025

Queries/Code:

01) Complex Types (Structured Types):

```
CREATE TYPE AddressType AS OBJECT (
  street VARCHAR(100),
  city VARCHAR(50),
  zip_code VARCHAR(10)
);
```

```
Type ADDRESSTYPE compiled
Elapsed: 00:00:00.104
```

```
CREATE TABLE Customers1011 (
  customer_id INT PRIMARY KEY,
  name VARCHAR(100),
  address AddressType
);
```

```
Table CUSTOMERS1011 created.
Elapsed: 00:00:00.020
```

```
INSERT INTO CUSTOMERS1011 (customer_id, name, address) VALUES (
  1,
  'Darshan Suresh',
  AddressType('45, Residency Road', 'Bengaluru', '560025')
);
```

```
INSERT INTO CUSTOMERS1011 (customer_id, name, address) VALUES (
  2,
  'Pooja Sharma',
  AddressType('B-10, Malviya Nagar', 'New Delhi', '110017')
);
```

```
SELECT c.name, c.address.city FROM CUSTOMERS1011 c;
```

	NAME	ADDRESS.CITY
1	Darshan Suresh	Bengaluru
2	Pooja Sharma	New Delhi

KINDLY RUN IN POSTGRE SQL

(https://onecompiler.com/postgresql/4448qaftv)

02) Table Inheritance Examples (PostgreSQL Style)

```
CREATE TABLE Persons (
 person_id INT PRIMARY KEY,
 name VARCHAR(100)
 );
  CREATE TABLE Employees (
 employee_id INT PRIMARY KEY,
 salary DECIMAL(10, 2)
 ) INHERITS (Persons);

INSERT INTO Persons (person_id, name) VALUES (101, 'Darshan Suresh');

INSERT INTO Employees (person_id, name, employee_id, salary) VALUES (
  201,
  'Alok',
  201,
  85000.00
);
INSERT INTO Employees (person_id, name, employee_id, salary) VALUES (
  202,
  'Sunita',
  202,
  125000.00
);

SELECT * FROM Persons;
```

```
 person_id |      name
-----------+----------------
       101 | Darshan Suresh
       201 | Alok
       202 | Sunita
(3 rows)
```

```
SELECT * FROM ONLY Persons;
```

```
 person_id |      name
-----------+----------------
       101 | Darshan Suresh
(1 row)
```

Using UNION:

```
CREATE TYPE Person1011 AS OBJECT (
    person_id NUMBER,
    full_name VARCHAR2(100),
    email_id VARCHAR2(100)
) NOT FINAL;

CREATE TYPE Student1011 UNDER Person1011 (
    roll_no VARCHAR2(15),
    course_name VARCHAR2(50),
    cgpa NUMBER
);

CREATE TYPE Professor1011 UNDER Person1011 (
    emp_code VARCHAR2(15),
    department VARCHAR2(50),
    salary NUMBER
);

CREATE TABLE Peoples1011 OF Person1011;
CREATE TABLE Students1011 OF Student1011;
CREATE TABLE Professors1011 OF Professor1011;

INSERT INTO Peoples1011 VALUES (Person1011(1, 'Darshan Suresh',
'darshansuresh1804@gmail.com'));
INSERT INTO Students1011 VALUES (Student1011(2, 'Sneha', 'sneha@amrita.edu', 'S102',
'BCA', 8.9));
INSERT INTO Students1011 VALUES (Student1011(3, 'Rohit', 'rohit.verma@nit.edu',
'S103', 'Mechanical', 9.2));
INSERT INTO Professors1011 VALUES (Professor1011(4, 'Dr. Leela Krishnan',
'leela.krishnan@iitm.ac.in', 'P210', 'AI and Data Science', 120000)
);
INSERT INTO Professors1011 VALUES (Professor1011(5, 'Kiran', 'kiran@iiitb.ac.in',
'P211', 'Cybersecurity', 95000));

SELECT VALUE(p) FROM People1011 p
UNION ALL
SELECT VALUE(s) FROM Students1011 s
UNION ALL
SELECT VALUE(t) FROM Professors1011 t;
```

Column "VALUE(P)" row 1

{"person_id":5,"full_name":"Dr. Kiran Rao","email_id":"kiran.rao@iiitb.ac.in"}

03)

```
-- STEP 1: Create VARRAY types first
CREATE TYPE Contact_List AS VARRAY(5) OF VARCHAR2(15);
/
CREATE TYPE Tech_Skill_List AS VARRAY(10) OF VARCHAR2(50);
/

-- STEP 2: Create the main table using those types
CREATE TABLE IT_Freelancers (
    freelancer_id NUMBER PRIMARY KEY,
    full_name VARCHAR2(100),
    city VARCHAR2(50),
    contacts Contact_List,
    skills Tech_Skill_List
);
/

-- STEP 3: Insert records
INSERT INTO IT_Freelancers VALUES (
    1,
    'Rakesh Shetty',
    'Mangalore',
    Contact_List('9876543210', '9887654321'),
    Tech_Skill_List('Python', 'Flask', 'SQL', 'ML')
);

INSERT INTO IT_Freelancers VALUES (
    2,
    'Priya Gupta',
    'Delhi',
    Contact_List('9998877766'),
    Tech_Skill_List('Java', 'Spring Boot', 'Microservices')
);

INSERT INTO IT_Freelancers VALUES (
    3,
    'Vivek Menon',
    'Kochi',
    Contact_List('8887776665'),
    Tech_Skill_List('C++', 'Linux', 'Networking')
);
COMMIT;

-- STEP 4: Query
SELECT f.full_name, f.city, p.COLUMN_VALUE AS phone, s.COLUMN_VALUE AS skill
```

```
FROM IT_Freelancers f, TABLE(f.contacts) p, TABLE(f.skills) s;
```

"FULL_NAME"	"CITY"	"PHONE"	"SKILL"
"Priya Gupta"	"Delhi"	"9998877766"	"Java"
"Priya Gupta"	"Delhi"	"9998877766"	"Spring Boot"
"Priya Gupta"	"Delhi"	"9998877766"	"Microservices"
"Rakesh Shetty"	"Mangalore"	"9876543210"	"Python"
"Rakesh Shetty"	"Mangalore"	"9876543210"	"Flask"
"Rakesh Shetty"	"Mangalore"	"9876543210"	"SQL"
"Rakesh Shetty"	"Mangalore"	"9876543210"	"ML"
"Rakesh Shetty"	"Mangalore"	"9887654321"	"Python"
"Rakesh Shetty"	"Mangalore"	"9887654321"	"Flask"
"Rakesh Shetty"	"Mangalore"	"9887654321"	"SQL"
"Rakesh Shetty"	"Mangalore"	"9887654321"	"ML"

04)

```
-- Define Nested Object Type
CREATE TYPE Govt_Project AS OBJECT (
    proj_id NUMBER,
    proj_name VARCHAR2(100),
    start_date DATE,
    end_date DATE
);
/

CREATE TYPE Project_List AS TABLE OF Govt_Project;
/

-- Department Table
CREATE TABLE Gov_Departments (
    dept_code NUMBER PRIMARY KEY,
    dept_name VARCHAR2(100),
    projects Project_List
) NESTED TABLE projects STORE AS projects_storage;

-- Insert Data
INSERT INTO Gov_Departments VALUES (
    101,
    'Department of Science & Technology',
    Project_List(
        Govt_Project(1, 'Digital India AI Mission', DATE '2024-04-01', DATE
'2026-03-31'),
        Govt_Project(2, 'Drone Policy Framework', DATE '2023-01-15', DATE
'2024-12-31')
```

```
    )
);

INSERT INTO Gov_Departments VALUES (
    102,
    'Ministry of Health',
    Project_List(
        Govt_Project(3, 'Telemedicine Outreach', DATE '2023-06-01', DATE '2025-05-31')
    )
);

-- Query Nested Data
SELECT d.dept_name, p.proj_name, p.start_date, p.end_date
FROM Gov_Departments d, TABLE(d.projects) p;
```

"DEPT_NAME" "PROJ_NAME" "START_DATE" "END_DATE"
"Department of Science Drone Policy Framework" "Digital India AI Mission"
"2024-04-01T00:00:00Z" "2026-03-31T00:00:00Z"
"Department of Science Drone Policy Framework" "Drone Policy Framework"
"2023-01-15T00:00:00Z" "2024-12-31T00:00:00Z"
"Ministry of Health" "Telemedicine Outreach" "2023-06-01T00:00:00Z" "2025-05-31T00:00:00Z"

05)

```
-- Define Basic Employee Type
CREATE TYPE Emp_Info AS OBJECT (
    emp_id NUMBER,
    emp_name VARCHAR2(100),
    date_of_birth DATE,
    MEMBER FUNCTION get_age RETURN NUMBER
);
/

CREATE TYPE BODY Emp_Info AS
    MEMBER FUNCTION get_age RETURN NUMBER IS
    BEGIN
        RETURN FLOOR(MONTHS_BETWEEN(SYSDATE, date_of_birth) / 12);
    END;
END;
/

-- Define Address Type
CREATE TYPE Emp_Address AS OBJECT (
    flat_no VARCHAR2(10),
    area VARCHAR2(100),
    city VARCHAR2(50),
```

```
    state VARCHAR2(50),
    pincode VARCHAR2(10)
);
/

-- Define Dependent & Table of Dependents
CREATE TYPE Dependent_Info AS OBJECT (
    dep_id NUMBER,
    dep_name VARCHAR2(100),
    relationship VARCHAR2(50),
    dob DATE
);
/

CREATE TYPE Dependent_Table AS TABLE OF Dependent_Info;
/

-- Create Corporate Employee Table
CREATE TABLE Corporate_Employees (
    emp_id NUMBER PRIMARY KEY,
    emp_details Emp_Info,
    emp_address Emp_Address,
    dependents Dependent_Table,
    salary NUMBER
) NESTED TABLE dependents STORE AS dependents_storage;

-- Insert Indian Employees
INSERT INTO Corporate_Employees VALUES (
    301,
    Emp_Info(301, 'Rahul Dev', DATE '1988-09-25'),
    Emp_Address('A-203', 'Koramangala', 'Bengaluru', 'Karnataka', '560095'),
    Dependent_Table(
        Dependent_Info(1, 'Sunita Dev', 'Spouse', DATE '1990-04-12'),
        Dependent_Info(2, 'Aarav Dev', 'Son', DATE '2015-07-20')
    ),
    95000
);

INSERT INTO Corporate_Employees VALUES (
    302,
    Emp_Info(302, 'Nisha Rao', DATE '1992-02-18'),
    Emp_Address('Flat 8B', 'Powai', 'Mumbai', 'Maharashtra', '400076'),
    Dependent_Table(
        Dependent_Info(1, 'Rohan Rao', 'Spouse', DATE '1990-10-05')
    ),
    87000
);
```

```
-- Query Dependents
SELECT e.emp_details.emp_name AS Employee, e.emp_details.get_age() AS Age,
       d.dep_name AS Dependent, d.relationship
FROM Corporate_Employees e, TABLE(e.dependents) d;
```

```
"EMPLOYEE"  "AGE" "DEPENDENT" "RELATIONSHIP"
 "Rahul Dev" 37     "Sunita Dev"   "Spouse"
 "Rahul Dev" 37     "Aarav Dev" "Son"
 "Nisha Rao" 33     "Rohan Rao" "Spouse"
```

Main LAB Exercise:

🇮🇳 UNIVERSITY DATABASE (Full Lab Implementation)

🔹 Step 1 – Create Basic Address Type

```
CREATE TYPE Address_Type AS OBJECT (
    street VARCHAR2(100),
    city VARCHAR2(50),
    state VARCHAR2(50),
    pincode VARCHAR2(10),
    country VARCHAR2(50)
);
/
```

🔹 **Step 2 – Create Base Person Type (NOT FINAL → allows inheritance)**

```
CREATE TYPE Person_Type AS OBJECT (
    person_id NUMBER,
    first_name VARCHAR2(50),
    last_name VARCHAR2(50),
    date_of_birth DATE,
    email VARCHAR2(100),
    phone VARCHAR2(15),
    address Address_Type,

    MEMBER FUNCTION get_age RETURN NUMBER,
    MEMBER FUNCTION get_full_name RETURN VARCHAR2
) NOT FINAL;
/
```

🔹 **Step 3 – Implement the Person Type Body**

```
CREATE TYPE BODY Person_Type AS
    MEMBER FUNCTION get_age RETURN NUMBER IS
    BEGIN
```

```
        RETURN FLOOR(MONTHS_BETWEEN(SYSDATE, date_of_birth) /
12);
    END get_age;

    MEMBER FUNCTION get_full_name RETURN VARCHAR2 IS
    BEGIN
        RETURN first_name || ' ' || last_name;
    END get_full_name;
END;
/
```

🔹 Step 4 – Define Course and Research Project Object Types

VARRAY for Courses (Student)

```
CREATE TYPE Course_Type AS OBJECT (
    course_code VARCHAR2(10),
    course_name VARCHAR2(100),
    credits NUMBER,
    semester VARCHAR2(20),
    grade VARCHAR2(2)
);
/
```

```
CREATE TYPE Course_List AS VARRAY(6) OF Course_Type;
/
```

Nested Table for Research Projects (Professor)

```
CREATE TYPE Research_Project_Type AS OBJECT (
    project_id NUMBER,
    project_title VARCHAR2(100),
    funding_amount NUMBER,
    start_date DATE,
    end_date DATE,
    status VARCHAR2(20)
);
/

CREATE TYPE Project_List AS TABLE OF Research_Project_Type;
/
```

🔹 Step 5 – Create Inherited Types

Student Type

```
CREATE TYPE Student_Type UNDER Person_Type (
```

```
    student_roll VARCHAR2(15),
    course VARCHAR2(50),
    semester VARCHAR2(20),
    gpa NUMBER,
    courses Course_List,

    OVERRIDING MEMBER FUNCTION get_full_name RETURN VARCHAR2
);
/

CREATE TYPE BODY Student_Type AS
    OVERRIDING MEMBER FUNCTION get_full_name RETURN VARCHAR2 IS
    BEGIN
        RETURN 'Student: ' || first_name || ' ' || last_name ||
' (' || student_roll || ')';
    END;
END;
/
```

Professor Type

```
CREATE TYPE Professor_Type UNDER Person_Type (
```

```
    emp_code VARCHAR2(15),
    department VARCHAR2(50),
    salary NUMBER,
    hire_date DATE,
    research_projects Project_List,

    OVERRIDING MEMBER FUNCTION get_full_name RETURN VARCHAR2,
    MEMBER FUNCTION get_years_of_service RETURN NUMBER,
    MEMBER FUNCTION get_total_funding RETURN NUMBER
);
/

CREATE TYPE BODY Professor_Type AS
    OVERRIDING MEMBER FUNCTION get_full_name RETURN VARCHAR2 IS
    BEGIN
        RETURN 'Prof. ' || first_name || ' ' || last_name || '
(' || emp_code || ')';
    END;

    MEMBER FUNCTION get_years_of_service RETURN NUMBER IS
    BEGIN
        RETURN FLOOR(MONTHS_BETWEEN(SYSDATE, hire_date)/12);
```

```
    END;

    MEMBER FUNCTION get_total_funding RETURN NUMBER IS
        total NUMBER := 0;
    BEGIN
        IF research_projects IS NOT NULL THEN
            FOR i IN 1..research_projects.COUNT LOOP
                total := total +
research_projects(i).funding_amount;
            END LOOP;
        END IF;
        RETURN total;
    END;
END;
/
```

Staff Type

```
CREATE TYPE Staff_Type UNDER Person_Type (
    staff_code VARCHAR2(15),
    position VARCHAR2(50),
    department VARCHAR2(50),
```

```
    salary NUMBER,
    hire_date DATE,

    OVERRIDING MEMBER FUNCTION get_full_name RETURN VARCHAR2,
    MEMBER FUNCTION get_years_of_service RETURN NUMBER
);
/

CREATE TYPE BODY Staff_Type AS
    OVERRIDING MEMBER FUNCTION get_full_name RETURN VARCHAR2 IS
    BEGIN
        RETURN 'Staff: ' || first_name || ' ' || last_name || '
(' || staff_code || ')';
    END;

    MEMBER FUNCTION get_years_of_service RETURN NUMBER IS
    BEGIN
        RETURN FLOOR(MONTHS_BETWEEN(SYSDATE, hire_date)/12);
    END;
END;
/
```

🔹 Step 6 – Create Tables for Each Type

```
CREATE TABLE Students_India OF Student_Type;

/

CREATE TABLE Professors_India OF Professor_Type

    NESTED TABLE research_projects STORE AS research_nt;

/

CREATE TABLE Staff_India OF Staff_Type;

/
```

🔹 Step 7 – Insert Sample Data

Students

```
INSERT INTO Students_India VALUES (

    Student_Type(

        1001, 'Rohan', 'Nair', DATE '2001-04-15',
'rohan.nair@amrita.edu', '9876543210',

        Address_Type('12 MG Road', 'Bengaluru', 'Karnataka',
'560001', 'India'),

        'CS2021A01', 'BCA', 'Semester 5', 8.7,

        Course_List(

            Course_Type('CS101', 'Intro to Programming', 4, 'Sem
1', 'A'),
```

```
            Course_Type('CS202', 'Data Structures', 4, 'Sem 2',
'A'),
            Course_Type('CS305', 'Machine Learning', 3, 'Sem 5',
'A-')
        )
    )
);

INSERT INTO Students_India VALUES (
    Student_Type(
        1002, 'Ananya', 'Reddy', DATE '2000-10-22',
'ananya.reddy@amrita.edu', '9898765432',
        Address_Type('7 Residency Road', 'Hyderabad',
'Telangana', '500081', 'India'),
        'CS2021A02', 'B.Tech CSE', 'Semester 5', 9.1,
        Course_List(
            Course_Type('CS201', 'OOP with Java', 4, 'Sem 3',
'A'),
            Course_Type('CS310', 'Operating Systems', 3, 'Sem
4', 'A-')
        )
    )
);
```

Professors

```
INSERT INTO Professors_India VALUES (

    Professor_Type(

        2001, 'Dr. Meera', 'Iyer', DATE '1975-08-10',
'meera.iyer@amrita.edu', '9876500001',

        Address_Type('5A Faculty Quarters', 'Mysuru',
'Karnataka', '570006', 'India'),

        'EMP100', 'Computer Science', 98000, DATE '2012-07-15',

        Project_List(

            Research_Project_Type(1, 'AI for Healthcare',
250000, DATE '2023-01-01', DATE '2024-12-31', 'Active'),

            Research_Project_Type(2, 'Smart Campus IoT', 180000,
DATE '2022-06-01', DATE '2023-12-31', 'Completed')

        )

    )

);



INSERT INTO Professors_India VALUES (

    Professor_Type(

        2002, 'Dr. Suresh', 'Patel', DATE '1970-05-25',
'suresh.patel@amrita.edu', '9876511112',

        Address_Type('Faculty Block C', 'Coimbatore', 'Tamil
Nadu', '641112', 'India'),
```

```
        'EMP101', 'Electrical Engineering', 120000, DATE
'2010-02-10',
        Project_List(
            Research_Project_Type(3, 'Solar Microgrids', 300000,
DATE '2024-01-10', NULL, 'Active')
        )
    )
);
```

Staff

```
INSERT INTO Staff_India VALUES (
    Staff_Type(
        3001, 'Kavita', 'Menon', DATE '1985-07-19',
'kavita.menon@amrita.edu', '9812345678',
        Address_Type('13 University Street', 'Kochi', 'Kerala',
'682020', 'India'),
        'STF001', 'Admin Assistant', 'Computer Science', 48000,
DATE '2018-03-10'
    )
);


INSERT INTO Staff_India VALUES (
    Staff_Type(
```

```
        3002, 'Arun', 'Deshmukh', DATE '1980-03-25',
'arun.deshmukh@amrita.edu', '9823456789',

        Address_Type('9A Campus Road', 'Mumbai', 'Maharashtra',
'400076', 'India'),

        'STF002', 'IT Technician', 'IT Department', 56000, DATE
'2016-05-05'

    )

);
```

🔹 Step 8 – Example Queries

1️⃣ List All Students with Their Courses

```
SELECT

    s.student_roll,

    s.get_full_name() AS student_name,

    s.course,

    s.gpa,

    c.course_code,

    c.course_name,

    c.grade

FROM Students_India s, TABLE(s.courses) c

ORDER BY s.student_roll;
```

2️⃣ Professors with Research Projects

```
SELECT
    p.emp_code,
    p.get_full_name() AS professor_name,
    p.department,
    r.project_title,
    r.funding_amount,
    r.status
FROM Professors_India p, TABLE(p.research_projects) r
ORDER BY p.emp_code;
```

3️⃣ All University Members (Union of Roles)

```
SELECT 'Student' AS role, student_roll AS id, get_full_name() AS name, course AS dept FROM Students_India
UNION ALL
SELECT 'Professor', emp_code, get_full_name(), department FROM Professors_India
UNION ALL
SELECT 'Staff', staff_code, get_full_name(), department FROM Staff_India;
```

4️⃣ Funding Summary for Professors

```
SELECT
```

```
    emp_code,
    get_full_name() AS professor_name,
    department,
    get_total_funding() AS total_funding,
    get_years_of_service() AS years_of_service
FROM Professors_India;
```

On Nov 24, 2025

Queries/Code:

```
-- 1. Create the Parent Object Type
CREATE TYPE person_obj1 AS OBJECT (
    person_id NUMBER,
    name VARCHAR2(100),
    email VARCHAR2(100)
) NOT FINAL;
/
```

```
SQL> CREATE TYPE person_obj1 AS OBJECT (
         person_id NUMBER,
         name VARCHAR2(100),
         email VARCHAR2(100)...
```

Show more...

```
Type PERSON_OBJ1 compiled

Elapsed: 00:00:00.095
```

```
-- 2. Create the Child Types (Inheriting from person_obj1)
CREATE TYPE student_obj1 UNDER person_obj1 (
    student_id NUMBER,
    major VARCHAR2(50),
    gpa NUMBER
);
/
```

```
SQL> CREATE TYPE student_obj1 UNDER person_obj1 (
         student_id NUMBER,
         major VARCHAR2(50),
         gpa NUMBER...
```

```
Show more...


Type STUDENT_OBJ1 compiled

Elapsed: 00:00:00.113

CREATE TYPE teacher_obj1 UNDER person_obj1 (
    teacher_id NUMBER,
    department VARCHAR2(50),
    salary NUMBER
);
/

SQL> CREATE TYPE teacher_obj1 UNDER person_obj1 (
         teacher_id NUMBER,
         department VARCHAR2(50),
         salary NUMBER...

Show more...


Type TEACHER_OBJ1 compiled

Elapsed: 00:00:00.109

-- 3. Create the Object Tables
CREATE TABLE student_tab OF student_obj1;
CREATE TABLE teacher_tab OF teacher_obj1;


Table STUDENT_TAB created.

Elapsed: 00:00:00.037

Table TEACHER_TAB created.
```

```
Elapsed: 00:00:00.023

-- Insert students
INSERT INTO student_tab VALUES (student_obj1(1,'Darshan
Suresh','darshansuresh1804@gmail.com',101,'CSE',8.5));
INSERT INTO student_tab VALUES
(student_obj1(2,'Meenakshi','meena1997@gmail.com',102,'ECE',9.1)
);
INSERT INTO student_tab VALUES (student_obj1(3,'Kiran
Kumar','kirankumar2004@gmail.com',103,'CSE',8.2));



-- Insert teachers
INSERT INTO teacher_tab VALUES (teacher_obj1(10,'Sudharshan
Duth','sudharshanduth@college.edu',501,'CSE',55000));
INSERT INTO teacher_tab VALUES (teacher_obj1(11,'Lakshmi
Kumari','lakshmikumari@college.edu',502,'ECE',60000));

COMMIT; -- Commits the changes to the database

Commit complete.

Elapsed: 00:00:00.001
```

BASIC QUERIES:

```
1. Select all students
SELECT * FROM student_tab;
```

	PERSON_ID	NAME	EMAIL	STUDENT_ID	MAJOR	GPA
1	2	Meenakshi	meena1997@gmail.(	102	ECE	9.1
2	1	Darshan Suresh	darshansuresh1804(	101	CSE	8.5
3	3	Kiran Kumar	kirankumar2004@gr	103	CSE	8.2

2. Select all teachers

```
SELECT * FROM teacher_tab;
```

	PERSON_ID	NAME	EMAIL	TEACHER_ID	DEPARTMENT	SALARY
1	10	Sudharshan Duth	sudharshanduth@co	501	CSE	55000
2	11	Lakshmi Kumari	lakshmikumari@colle	502	ECE	60000

3. Select only student names and GPA

```
SELECT s.name, s.gpa
FROM student_tab s;
```

	NAME	GPA
1	Meenakshi	9.1
2	Darshan Suresh	8.5
3	Kiran Kumar	8.2

4. Select only teacher names and salary

```
SELECT t.name, t.salary
FROM teacher_tab t;
```

	NAME	SALARY
1	Sudharshan Duth	55000
2	Lakshmi Kumari	60000

FILTERING QUERIES:

5. Students with GPA above 8.5

```
SELECT name, student_id, gpa
FROM student_tab
WHERE gpa > 8.5;
```

	NAME	STUDENT_ID	GPA
1	Meenakshi	102	9.1

6. Teachers with salary greater than 55,000

```
SELECT name, department, salary
FROM teacher_tab
WHERE salary > 55000;
```

	NAME	DEPARTMENT	SALARY
1	Lakshmi Kumar	ECE	60000

7. Students from CSE department

```
SELECT name, student_id
FROM student_tab
WHERE major = 'CSE';
```

	NAME	STUDENT_ID
1	Darshan Suresh	101
2	Kiran Kumar	103

8. Teachers from ECE department

```
SELECT name, teacher_id
FROM teacher_tab
WHERE department = 'ECE';
```

	NAME	TEACHER_ID
1	Lakshmi Kumari	502

USING INHERITED:

9. Select person_id, name, email for all students

```
SELECT person_id, name, email
FROM student_tab;
```

	PERSON_ID	NAME	EMAIL
1	2	Meenakshi	meena1997@gmail.(
2	1	Darshan Suresh	darshansuresh1804(
3	3	Kiran Kumar	kirankumar2004@gr

```
10. Select person attributes for teachers
SELECT person_id, name, email
FROM teacher_tab;
```

	PERSON_ID	NAME	EMAIL
1	10	Sudharshan Duth	sudharshanduth@co
2	11	Lakshmi Kumari	lakshmikumari@colle

ADVANCED INHERITANCE QUERIES:

```
11. SELECT the full object instance
SELECT VALUE(s)
FROM student_tab s;
```

	VALUE(S)
1	{"person_id":2,"name":"Meenakshi","email":"meena1997@gmail.com","student_id":102,"major":"ECE","gpa":9.1}
2	{"person_id":1,"name":"Darshan Suresh","email":"darshansuresh1804@gmail.com","student_id":101,"major":"CSE","gpa":8.5}
3	{"person_id":3,"name":"Kiran Kumar","email":"kirankumar2004@gmail.com","student_id":103,"major":"CSE","gpa":8.2}

```
SELECT VALUE(t)
FROM teacher_tab t;
```

	VALUE(T)
1	{"person_id":10,"name":"Sudharshan Duth","email":"sudharshanduth@college.edu","teacher_id":501,"department":"CSE","salary":55(
2	{"person_id":11,"name":"Lakshmi Kumari","email":"lakshmikumari@college.edu","teacher_id":502,"department":"ECE","salary":60000}

TYPE CHECKING:

12. Treat as Parent → Child

```
SELECT TREAT(VALUE(s) AS person_obj1).name AS person_name
FROM student_tab s;
```

	PERSON_NAME
1	Meenakshi
2	Darshan Suresh
3	Kiran Kumar

13. Find only student records from parent-type table (if merged)
(Useful if both stored in one big table)

```
SELECT s.name, s.major
FROM student_tab s
WHERE VALUE(s) IS OF (student_obj1);
```

	NAME	MAJOR
1	Meenakshi	ECE
2	Darshan Suresh	CSE
3	Kiran Kumar	CSE

PATTERN SEARCH :

14. Students with gmail address

```
SELECT name, email
FROM student_tab
WHERE email LIKE '%gmail%';
```

	NAME	EMAIL
1	Meenakshi	meena1997@gmail.com
2	Darshan Suresh	darshansuresh1804@gmail.com
3	Kiran Kumar	kirankumar2004@gmail.com

15. Teachers whose name starts with 'Dr'

```
SELECT name
FROM teacher_tab
WHERE name LIKE 'Dr%';
```

NAME

No items to display.

SORTING & AGGREGATION:

16. Order students by GPA (descending)

```
SELECT name, gpa
FROM student_tab
```

```
ORDER BY gpa DESC;
```

	NAME	GPA
1	Meenakshi	9.1
2	Darshan Suresh	8.5
3	Kiran Kumar	8.2

17. Find highest GPA

```
SELECT MAX(gpa) AS highest_gpa FROM student_tab;
```

	HIGHEST_GPA
1	9.1

18. Average teacher salary

```
SELECT AVG(salary) AS avg_salary
FROM teacher_tab;
```

	AVG_SALARY
1	57500

JOIN-LIKE BEHAVIOR:

19. Join students with teachers by department/major (if linked)

(CSE student with CSE teacher)

```
SELECT s.name AS student, t.name AS teacher
FROM student_tab s, teacher_tab t
WHERE s.major = t.department;
```

	STUDENT	TEACHER
1	Meenakshi	Lakshmi Kumari
2	Darshan Suresh	Sudharshan Duth
3	Kiran Kumar	Sudharshan Duth

OTHERS:

20. Count students and teachers by specialization/department

Count students per major:

```
SELECT major, COUNT(*)
FROM student_tab
GROUP BY major;
```

Count teachers per department:

```
SELECT department, COUNT(*)
FROM teacher_tab
GROUP BY department;
```

	MAJOR	COUNT(*)
1	ECE	1
2	CSE	2

21. Students with GPA between 8 and 9

```
SELECT name, gpa
FROM student_tab
```

```
WHERE gpa BETWEEN 8 AND 9;
```

	NAME	GPA
1	Darshan Suresh	8.5
2	Kiran Kumar	8.2

22. Teachers with even teacher_id

```
SELECT name, teacher_id
FROM teacher_tab
WHERE MOD(teacher_id, 2) = 0;
```

	NAME	TEACHER_ID
1	Lakshmi Kumari	502

On Dec 8, 2025

Queries/Code:

1. Create Table With JSON Column

```
CREATE TABLE ds_json_profiles (
     profile_id VARCHAR2(30) PRIMARY KEY,
     created_on TIMESTAMP WITH TIME ZONE DEFAULT SYSTIMESTAMP,
     profile_doc JSON
);
```

Output:

```
SQL> CREATE TABLE ds_json_profiles (
        profile_id VARCHAR2(30) PRIMARY KEY,
        created_on TIMESTAMP WITH TIME ZONE DEFAULT SYSTIMESTAMP,
        profile_doc JSON...
Show more...


Table DS_JSON_PROFILES created.

Elapsed: 00:00:00.021
```

2. Insert JSON Records

```
INSERT INTO ds_json_profiles VALUES (
     'U001',
     SYSTIMESTAMP,
     '{"name":"Darshan
Suresh","age":22,"skills":["SQL","Linux"],"active":true}'
);
```

Output:

```
SQL> INSERT INTO ds_json_profiles VALUES (
        'U001',
        SYSTIMESTAMP,
        '{"name":"Darshan Suresh","age":22,"skills":["SQL","Linux"],"active":true}'...
Show more...


1 row inserted.

Elapsed: 00:00:00.047
```

```
INSERT INTO ds_json_profiles VALUES (
     'U002',
     SYSTIMESTAMP,
'{"name":"Bhavya","city":{"name":"Mysuru","pin":570001},"roles":
["admin","editor"]}'
);
```

Output:

```
SQL> INSERT INTO ds_json_profiles VALUES (
        'U002',
        SYSTIMESTAMP,   '{"name":"Bhavya","city":{"name":"Mysuru","pin":570001},"roles":["admin","editor"]}'
     )


1 row inserted.

Elapsed: 00:00:00.003
```

```
INSERT INTO ds_json_profiles VALUES (
     'U004',
     SYSTIMESTAMP,
     '{
     "name":"Divya",
     "Devices":[
          {"type":"mobile","os":"android"},
            {"type":"laptop","os":"windows"}
     ]
     }'
);
```

Output:

```
SQL> INSERT INTO ds_json_profiles VALUES (
        'U004',
        SYSTIMESTAMP,
        '{...
Show more...



1 row inserted.

Elapsed: 00:00:00.003
```

3. SELECT All Rows (OUTPUT)

```
SELECT profile_id, profile_doc FROM ds_json_profiles;
```

Output:

	PROFILE_ID	PROFILE_DOC
1	U001	{"name":"Darshan Suresh","age":22,"skills":["SQL","Linux"],"active":true}
2	U002	{"name":"Bhavya","city":{"name":"Mysuru","pin":570001},"roles":["admin","editor"]}
3	U004	{"name":"Divya","Devices":[{"type":"mobile","os":"android"},{"type":"laptop","os":"windows"}]}

4. Extract JSON Value (Using Dot Notation)

Get name of each profile

```
SELECT profile_id,
       JSON_VALUE(profile_doc, '$.name') AS user_name
FROM ds_json_profiles;
```

Output:

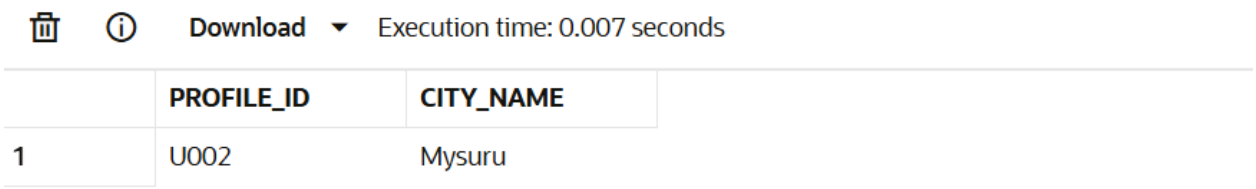
	PROFILE_ID	USER_NAME
1	U001	Darshan Suresh
2	U002	Bhavya
3	U004	Divya

5. Extract Nested JSON Value

Get city name from nested object (for U002)

```
SELECT profile_id,
       JSON_VALUE(profile_doc, '$.city.name') AS city_name
FROM ds_json_profiles
WHERE profile_id = 'U002';
```

Output:



	PROFILE_ID	CITY_NAME
1	U002	Mysuru

6. Extract Array Item

Example: List the `type` of every device for Divya

```
SELECT d.*
FROM ds_json_profiles p,
     JSON_TABLE(
     P.profile_doc,
     '$.devices[*]' COLUMNS (
          type VARCHAR2(20) PATH '$.type',
          os   VARCHAR2(20) PATH '$.os'
     )
     ) d
WHERE profile_id = 'U004';
```

Output:

"TYPE"	"OS"
"mobile"	"android"
"laptop"	"windows"

7. Filter JSON Records

Find all users with active=true

```
SELECT *
FROM ds_json_profiles
WHERE profile_doc LIKE '%"active":true%';
```

Output:

	PROFILE_ID	CREATED_ON	PROFILE_DOC
1	U001	2025-12-08T15:08:59.287413Z	{"name":"Darshan Suresh","age":22,"skills":["SQL","Linux"],"active":true}

8. Update JSON Value

Example: change "Mysuru" → "Bengaluru"

```
UPDATE ds_json_profiles
SET profile_doc = REPLACE(profile_doc, '"name":"Mysuru"',
'"name":"Bengaluru"')
WHERE profile_id = 'U002';
```

Output:

```
SQL> UPDATE ds_json_profiles
     SET profile_doc = REPLACE(profile_doc, '"name":"Mysuru"', '"name":"Bengaluru"')
     WHERE profile_id = 'U002'


1 row updated.

Elapsed: 00:00:00.005
```

```
Select * from ds_json_profiles;
```

	PROFILE_ID	CREATED_ON	PROFILE_DOC
1	U001	2025-12-08T15:08:59.287413Z	{"name":"Darshan Suresh","age":22,"skills":["SQL","Linux"],"active":true}
2	U002	2025-12-08T15:09:20.746197Z	{"name":"Bhavya","city":{"name":"Bengaluru","pin":570001},"roles":["admin","editor"]}
3	U004	2025-12-08T15:09:39.5066Z	{"name":"Divya","Devices":[{"type":"mobile","os":"android"},{"type":"laptop","os":"windows"}]}