Name: Bibhu kumar mishra
Abhishek Kumar

PRACTICAL - 1

Create table Employee, product Jan, product Feb.

```
CREATE TABLE Employee (
  employee_id INT PRIMARY KEY,
  first_name VARCHAR(50),
  last_name VARCHAR(50),
  age INT,
  salary DECIMAL(10, 2)
);

CREATE TABLE product_jan (
  product_id INT PRIMARY KEY,
  product_name VARCHAR(100),
  price DECIMAL(10, 2)
);

CREATE TABLE product_feb (
  product_id INT PRIMARY KEY,
  product_name VARCHAR(100),
  price DECIMAL(10, 2)
);
```

```
INSERT INTO Employee (employee_id, first_name, last_name, age, salary)
VALUES
```

```
(1, 'Alice', 'Johnson', 29, 55000.00),
 (2, 'Bob', 'Smith', 35, 72000.50),
 (3, 'Charlie', 'Nguyen', 41, 68000.00),
(4, 'Diana', 'Patel', 26, 49000.75),
 (5, 'Ethan', 'Brown', 38, 81000.00);

INSERT INTO product_jan (product_id, product_name, price)
VALUES

 (101, 'Laptop', 1200.00),
(102, 'Mouse', 25.50),
(103, 'Keyboard', 75.00);

INSERT INTO product_feb (product_id, product_name, price)
VALUES
(101, 'Laptop', 1150.00),
(103, 'Keyboard', 75.00),
(106, 'Headphones', 99.99);
```

----------OUTPUT ----------

employee_id	first_name	last_name	age	salary
1	Alice	Johnson	29	55000
2	Bob	Smith	35	72000.5
3	Charlie	Nguyen	41	68000
4	Diana	Patel	26	49000.75
5	Ethan	Brown	38	81000

product_id	product_name	price
101	Laptop	1200
102	Mouse	25.5
103	Keyboard	75

product_id	product_name	price
101	Laptop	1150
103	Keyboard	75

106	Headphones	99.99

PRACTICAL - 2

Perform DML operations on the employee table.

1. INSERT Operation:

```
INSERT INTO Employee (employee_id, first_name, last_name, age, salary)
VALUES

(1, 'Alice', 'Johnson', 29, 55000.00),
 (2, 'Bob', 'Smith', 35, 72000.50),
 (3, 'Charlie', 'Nguyen', 41, 68000.00),
(4, 'Diana', 'Patel', 26, 49000.75),
 (5, 'Ethan', 'Brown', 38, 81000.00);
```

2. UPDATE Operation:

```
-- Increase salary of employee with ID 3
UPDATE Employee
SET salary = salary + 5000
WHERE employee_id = 3;

-- Update age for an employee
UPDATE Employee
SET age = 30
WHERE first_name = 'Alice';
```

3. DELETE Operation:

```
-- Delete one employee by ID
DELETE FROM Employee
WHERE employee_id = 4;

-- Delete all employees with salary less than 50,000
DELETE FROM Employee
WHERE salary < 50000;
```

4. SELECT Operation

-- View all employees SELECT * FROM Employee;

```
-- View employees with salary greater than 60,000
SELECT first_name, last_name, salary
FROM Employee
WHERE salary > 60000;

-- Sort employees by salary (highest first)
SELECT * FROM Employee
ORDER BY salary DESC;

-- Find average salary
SELECT AVG(salary) AS avg_salary FROM Employee;
```

----------OUTPUT ----------

1. INSERT Operation:

employee_id	first_name	last_name	age	salary
1	Alice	Johnson	29	55000
2	Bob	Smith	35	72000.5
3	Charlie	Nguyen	41	68000
4	Diana	Patel	26	49000.75
5	Ethan	Brown	38	81000

2. UPDATE Operation:

employee_id	first_name	last_name	age	salary
1	Alice	Johnson	30	55000
2	Bob	Smith	35	72000.5
3	Charlie	Nguyen	41	78000
4	Diana	Patel	26	49000.75
5	Ethan	Brown	38	81000

3. DELETE Operation

employee_id	first_name	last_name	age	salary
1	Alice	Johnson	30	55000

2	Bob	Smith	35	72000.5
3	Charlie	Nguyen	41	78000
5	Ethan	Brown	38	81000

4. SELECT Operation:

employee_id	first_name	last_name	age	salary
1	Alice	Johnson	30	55000
2	Bob	Smith	35	72000.5
3	Charlie	Nguyen	41	78000
5	Ethan	Brown	38	81000

first_name	last_name	salary
Bob	Smith	72000.5
Charlie	Nguyen	78000
Ethan	Brown	81000

employee_id	first_name	last_name	age	salary
5	Ethan	Brown	38	81000
3	Charlie	Nguyen	41	78000
2	Bob	Smith	35	72000.5
1	Alice	Johnson	30	55000

avg_salary
71500.125

Name: Bibhu kumar mishra
Abhishek Kumar

PRACTICAL - 3

Perform aggregate function on the employee table.

1. **COUNT()**

```
SELECT COUNT(*) AS total_employees FROM employee;
```

2. **SUM()**

```
SELECT SUM(salary) AS total_salary FROM employee;
```

3. **AVG()**

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

4. **MIN() / MAX()**

```
SELECT MIN(salary) AS min_salary, MAX(salary) AS max_salary FROM employee;
```

----------OUTPUT ----------

1. **COUNT()**

total_employees
5

2. **SUM()**

total_salary
325001.25

3. **AVG()**

avg_salary

65000.25

4. **MIN() / MAX()**

min_salary	max_salary
49000.75	81000

Name: Bibhu kumar mishra
Abhishek Kumar

PRACTICAL – 4

Perform Built-in(numeric) Function on employee table.

1. **ABS()** – Absolute Value

```
SELECT employee_id, first_name, salary, ABS(salary - 60000) AS
abs_difference FROM Employee;
```

2. **CEILING()** – Rounds up to the nearest integer

```
SELECT employee_id, first_name, salary, CEIL(salary) AS ceil_salary FROM
Employee;
```

3. **FLOOR()** – Rounds down to the nearest integer

```
SELECT employee_id, first_name, salary, FLOOR(salary) AS floor_salary
FROM Employee;
```

4. **ROUND()** – Rounds a number to a specified number of decimal places

```
SELECT employee_id, first_name, salary, ROUND(salary, 0) AS
rounded_salary FROM Employee;
```

5. **POWER()** – Raises a number to a specified power

```
SELECT employee_id, first_name, POWER(age, 2) AS age_squared FROM
Employee;
```

6. **SQRT()** – Square Root

```
SELECT employee_id, first_name, SQRT(salary) AS sqrt_salary FROM
Employee;
```

7. **AVG(), SUM(), MIN(), MAX(), COUNT()** – Aggregate numeric functions

```
SELECT
  COUNT(*) AS total_employees,
  AVG(salary) AS avg_salary,
  SUM(salary) AS total_salary,
  MIN(salary) AS min_salary,
  MAX(salary) AS max_salary
FROM Employee;
```

8. **MOD()** – Modulus (remainder after division)

```
SELECT employee_id, first_name, age % 2 AS age_mod_2 FROM Employee;
```

----------OUTPUT ----------

1. **ABS()** – Absolute Value

employee_id	first_name	salary	abs_difference
1	Alice	55000	5000
2	Bob	72000.5	12000.5
3	Charlie	68000	8000
4	Diana	49000.75	10999.25
5	Ethan	81000	21000

2. **CEILING()** – Rounds up to the nearest integer

employee_id	first_name	salary	ceil_salary
1	Alice	55000	55000
2	Bob	72000.5	72001
3	Charlie	68000	68000
4	Diana	49000.75	49001
5	Ethan	81000	81000

3. **FLOOR()** – Rounds down to the nearest integer

employee_id	first_name	salary	floor_salary
1	Alice	55000	55000
2	Bob	72000.5	72000
3	Charlie	68000	68000
4	Diana	49000.75	49000
5	Ethan	81000	81000

4. **ROUND()** – Rounds a number to a specified number of decimal places

employee_id	first_name	salary	rounded_salary
1	Alice	55000	55000
2	Bob	72000.5	72001
3	Charlie	68000	68000
4	Diana	49000.75	49001
5	Ethan	81000	81000

5. **POWER()** – Raises a number to a specified power

employee_id	first_name	age_squared
1	Alice	841
2	Bob	1225
3	Charlie	1681
4	Diana	676
5	Ethan	1444

6. **SQRT()** – Square Root

employee_id	first_name	sqrt_salary
1	Alice	234.5207879911715
2	Bob	268.32908899334785
3	Charlie	260.76809620810593
4	Diana	221.3611302826221
5	Ethan	284.60498941515414

7. **AVG(), SUM(), MIN(), MAX(), COUNT()** – Aggregate numeric functions

total_employees	avg_salary	total_salary	min_salary	max_salary
5	65000.25	325001.25	49000.75	81000

8. **MOD()** – Modulus (remainder after division)

employee_id	first_name	age_mod_2
1	Alice	1
2	Bob	1
3	Charlie	1
4	Diana	0
5	Ethan	0

Name: Bibhu kumar mishra
Abhishek Kumar

Sub: DBMS

Roll: MCA-B21, MCA-B04

Name: Bibhu kumar mishra
Abhishek Kumar

PRACTICAL - 5

Perform Built-in(String) Function on employee table.

1. **UPPER()** – Convert to uppercase

```
SELECT employee_id,
    UPPER(first_name) AS upper_first_name,
    UPPER(last_name) AS upper_last_name
FROM Employee;
```

2. **LOWER()** – Convert to lowercase

```
SELECT
    LOWER(first_name) AS lower_first_name,
    LOWER(last_name) AS lower_last_name
FROM Employee;
```

3. **CONCAT()** – Combine first and last name

```
SELECT
    CONCAT(first_name, ' ', last_name) AS full_name
FROM Employee;
```

4. **LENGTH()** – Find the number of characters

```
SELECT
    first_name,
    LENGTH(first_name) AS name_length
FROM Employee;
```

5. **SUBSTRING()** – Extract part of a string

```
SELECT
    first_name,
    SUBSTRING(first_name, 1, 3) AS short_name
FROM Employee;
```

6. **REPLACE()** – Replace characters

```
SELECT
        first_name,
        REPLACE(first_name, 'a', '@') AS replaced_name
FROM Employee;
```

7. **REVERSE()** – Reverse the string

```
SELECT
        first_name,
        REVERSE(first_name) AS reversed_name
FROM Employee;
```

8. **LTRIM() / RTRIM()** – Remove spaces from sides

```
SELECT
  LTRIM('   ' || first_name) AS left_trimmed,
  RTRIM(first_name || '   ') AS right_trimmed
FROM Employee;
```

9. **INSTR() / LOCATE() –** Find position of a substring

```
SELECT
  first_name,
  INSTR(first_name, 'a') AS position_of_a
FROM Employee;
```

10. **CONCAT_WS()** – Concatenate with a separator

```
SELECT
  CONCAT_WS(' - ', first_name, last_name, salary) AS employee_info
FROM Employee;
```

Name: Bibhu kumar mishra
Abhishek Kumar

----------**OUTPUT** ----------

1. **UPPER()** – Convert to uppercase

employee_id	upper_first_name	upper_last_name
1	ALICE	JOHNSON
2	BOB	SMITH
3	CHARLIE	NGUYEN
4	DIANA	PATEL
5	ETHAN	BROWN

2. **LOWER()** – Convert to lowercase

lower_first_name	lower_last_name
alice	johnson
bob	smith
charlie	nguyen
diana	patel
ethan	brown

3. **CONCAT()** – Combine first and last name

4. **LENGTH()** – Find the number of characters

first_name	name_length
Alice	5
Bob	3

Charlie	7
Diana	5
Ethan	5

5. **SUBSTRING()** – Extract part of a string

first_name	short_name
Alice	Ali
Bob	Bob
Charlie	Cha
Diana	Dia
Ethan	Eth

6. **REPLACE()** – Replace characters

first_name	replaced_name
Alice	Alice
Bob	Bob
Charlie	Ch@rlie
Diana	Di@n@
Ethan	Eth@n

7. **REVERSE()** – Reverse the string

first_name	reversed_name

Alice	ecilA
Bob	boB
Charlie	eilrahC
Diana	anaiD
Ethan	nahtE

8. **LTRIM() / RTRIM()** – Remove spaces from sides

left_trimmed	right_trimmed
Alice	Alice
Bob	Bob
Charlie	Charlie
Diana	Diana
Ethan	Ethan

9. **INSTR() / LOCATE()** – Find position of a substring

first_name	position_of_a
Alice	0
Bob	0
Charlie	3
Diana	3

Ethan	4

10. **CONCAT_WS()** – Concatenate with a separator

Name: Bibhu kumar mishra
Abhishek Kumar

PRACTICAL - 6

Perform Built-in (DateTime) Function on employee table.

1: Add a Date Column

```
ALTER TABLE Employee
ADD hire_date DATE;
```

2: Update the Table with Sample Dates

```
UPDATE Employee
SET hire_date = CASE employee_id
  WHEN 1 THEN '2021-03-15'
  WHEN 2 THEN '2019-07-20'
  WHEN 3 THEN '2018-11-05'
  WHEN 4 THEN '2023-01-10'
  WHEN 5 THEN '2020-09-30'
END;
```

3: Use Built-in DateTime Functions

3.1. CURRENT_DATE / GETDATE()

```
SELECT employee_id, first_name, last_name, hire_date,
CURRENT_DATE AS current_date
FROM Employee;
```

3.2. DATEDIFF()

```
SELECT
  first_name,
  last_name,
  hire_date,
  DATEDIFF(YEAR, hire_date, CURRENT_DATE) AS years_worked
FROM Employee;
```

3.3. DATEADD()

```
SELECT
  first_name,
  hire_date,
  DATEADD(YEAR, 1, hire_date) AS next_appraisal_date
FROM Employee;
```

3.4. EXTRACT()

```
SELECT
  first_name,
  hire_date,
  EXTRACT(YEAR FROM hire_date) AS hire_year,
  EXTRACT(MONTH FROM hire_date) AS hire_month
FROM Employee;
```

3.5. DAY(), MONTH(), YEAR()

```
SELECT
  first_name,
  hire_date,
  DAY(hire_date) AS day_hired,
  MONTH(hire_date) AS month_hired,
  YEAR(hire_date) AS year_hired
FROM Employee;
```

----------OUTPUT ----------

3.1. CURRENT_DATE / GETDATE()

employee_id	first_name	last_name	hire_date	current_date
1	Alice	Johnson	2021-03-15	2025-11-11
2	Bob	Smith	2019-07-20	2025-11-11
3	Charlie	Nguyen	2018-11-05	2025-11-11
4	Diana	Patel	2023-01-10	2025-11-11
5	Ethan	Brown	2020-09-30	2025-11-11

Name: Bibhu kumar mishra
Abhishek Kumar

Sub: DBMS

Roll: MCA-B21, MCA-B04

3.2. DATEDIFF()

first_name	last_name	hire_date	years_worked
Alice	Johnson	2021-03-15	4
Bob	Smith	2019-07-20	6
Charlie	Nguyen	2018-11-05	7
Diana	Patel	2023-01-10	2
Ethan	Brown	2020-09-30	5

3.3. DATEADD()

first_name	hire_date	next_appraisal_date
Alice	2021-03-15	2022-03-15
Bob	2019-07-20	2020-07-20
Charlie	2018-11-05	2019-11-05
Diana	2023-01-10	2024-01-10
Ethan	2020-09-30	2021-09-30

3.4. EXTRACT()

first_name	hire_date	hire_year	hire_month
Alice	2021-03-15	2021	3
Bob	2019-07-20	2019	7
Charlie	2018-11-05	2018	11
Diana	2023-01-10	2023	1
Ethan	2020-09-30	2020	9

3.5. DAY(), MONTH(), YEAR()

first_name	hire_date	day_hired	month_hired	year_hired
Alice	2021-03-15	15	3	2021
Bob	2019-07-20	20	7	2019
Charlie	2018-11-05	5	11	2018

Diana	2023-01-10	10	1	2023
Ethan	2020-09-30	30	9	2020

Name: Bibhu kumar mishra
Abhishek Kumar

PRACTICAL - 7

Perform set operations on table product jan and product feb and

1. UNION (All distinct rows)

```
SELECT * FROM product_jan
UNION
SELECT * FROM product_feb;
```

2. UNION ALL

```
SELECT * FROM product_jan
UNION ALL
SELECT * FROM product_feb;
```

3. INTERSECT (Common records)

```
SELECT j.product_id, j.product_name, j.price
FROM product_jan j
INNER JOIN product_feb f
ON j.product_id = f.product_id
  AND j.product_name = f.product_name
  AND j.price = f.price;
```

4. EXCEPT / MINUS (Records in Jan but not in Feb)

```
SELECT j.*
FROM product_jan j
LEFT JOIN product_feb f
ON j.product_id = f.product_id
WHERE f.product_id IS NULL;
```

Name: Bibhu kumar mishra
Abhishek Kumar

5. Products sold only in February (not in January)

```
SELECT f.*
FROM product_feb f
LEFT JOIN product_jan j
ON f.product_id = j.product_id
WHERE j.product_id IS NULL;
```

----------OUTPUT ----------

1. UNION (All distinct rows)

product_id	product_name	price
101	Laptop	1150
101	Laptop	1200
102	Mouse	25.5
103	Keyboard	75
106	Headphones	99.99

2. UNION ALL

product_id	product_name	price
101	Laptop	1200
102	Mouse	25.5
103	Keyboard	75
101	Laptop	1150
103	Keyboard	75

106	Headphones	99.99

3. INTERSECT (Common records)

product_id	product_name	price
103	Keyboard	75

4. EXCEPT / MINUS (Records in Jan but not in Feb)

product_id	product_name	price
102	Mouse	25.5

5. Products sold only in February (not in January)

product_id	product_name	price
106	Headphones	99.99

Name: Bibhu kumar mishra
Abhishek Kumar

PRACTICAL - 8

Perform alter, drop, truncate on product jan and product feb

1. **ALTER TABLE**

```
ALTER TABLE product_jan
    ADD COLUMN category VARCHAR(50);

    ALTER TABLE product_feb
    ADD COLUMN category VARCHAR(50);
```

2. **TRUNCATE TABLE**

```
TRUNCATE TABLE product_jan;
TRUNCATE TABLE product_feb;
```

3. **DROP TABLE**

```
DROP TABLE product_jan;
    DROP TABLE product_feb;
```

----------OUTPUT ----------

1. **ALTER TABLE**

product_id	product_name	price	category
101	Laptop	1200	
102	Mouse	25.5	
103	Keyboard	75	

product_id	product_name	price	category
101	Laptop	1150	
103	Keyboard	75	
106	Headphones	99.99	

2. **TRUNCATE TABLE**

product_id	product_name	price	category
(no rows — table is empty)			

product_id	product_name	price	category
(no rows — table is empty)			

3. **DROP TABLE**

Error Code: 1146. Table 'your_database.product_jan' doesn't exist

Error Code: 1146. Table 'your_database.product_feb' doesn't exist

Name: Bibhu kumar mishra
Abhishek Kumar

PRACTICAL - 9

Perform Group by, having and order by on employee table.

Group employees by age and calculate the average salary per age

```
SELECT
  age,
  COUNT(*) AS num_employees,
  AVG(salary) AS avg_salary
FROM Employee
GROUP BY age
HAVING AVG(salary) > 50000    -- filter groups with average salary > 50,000
ORDER BY avg_salary DESC;     -- sort by average salary descending
```

----------OUTPUT ----------

age	num_employees	avg_salary
38	1	81000
35	1	72000.5
41	1	68000
29	1	55000

Name: Bibhu kumar mishra
Abhishek Kumar

PRACTICAL - 10

Perform exist, all, any on employee table.

a) EXISTS

```
SELECT *
FROM Employee e
WHERE EXISTS (
  SELECT 1
  FROM Employee
  WHERE salary > 80000
);
```

b) ALL

```
SELECT *
FROM Employee
WHERE salary >= (
  SELECT MAX(salary)
  FROM Employee
  WHERE salary < 80000
);
```

c) ANY / SOME

```
SELECT *
FROM Employee e
WHERE salary > (
  SELECT salary
  FROM Employee
  WHERE age < 30
);
```

Name: Bibhu kumar mishra
Abhishek Kumar

----------OUTPUT ----------

a) EXISTS

employee_id	first_name	last_name	age	salary	hire_date
1	Alice	Johnson	29	55000	2021-03-15
2	Bob	Smith	35	72000.5	2019-07-20
3	Charlie	Nguyen	41	68000	2018-11-05
4	Diana	Patel	26	49000.75	2023-01-10
5	Ethan	Brown	38	81000	2020-09-30

b) ALL

employee_id	first_name	last_name	age	salary	hire_date
2	Bob	Smith	35	72000.5	2019-07-20
5	Ethan	Brown	38	81000	2020-09-30

c) ANY / SOME

employee_id	first_name	last_name	age	salary	hire_date
2	Bob	Smith	35	72000.5	2019-07-20
3	Charlie	Nguyen	41	68000	2018-11-05
5	Ethan	Brown	38	81000	2020-09-30

Name: Bibhu kumar mishra
Abhishek Kumar

PRACTICAL - 11

Perform joins on product and vendor table.

```
CREATE TABLE Vendor (
  vendor_id INT PRIMARY KEY,
  vendor_name VARCHAR(100)
);

INSERT INTO Vendor (vendor_id, vendor_name)
VALUES
(1, 'Tech World'),
(2, 'Gadget Hub'),
(3, 'Accessory King');


ALTER TABLE product_jan ADD COLUMN vendor_id INT;
ALTER TABLE product_feb ADD COLUMN vendor_id INT;

-- Sample data
UPDATE product_jan
SET vendor_id = CASE product_id
  WHEN 101 THEN 1
  WHEN 102 THEN 3
  WHEN 103 THEN 3
END;

UPDATE product_feb
SET vendor_id = CASE product_id
  WHEN 101 THEN 1
  WHEN 103 THEN 3
  WHEN 106 THEN 2
END;
```

a) INNER JOIN

```
SELECT p.product_id, p.product_name, p.price, v.vendor_name
FROM product_jan p
INNER JOIN Vendor v ON p.vendor_id = v.vendor_id;
```

b) LEFT JOIN

```
SELECT p.product_id, p.product_name, p.price, v.vendor_name
FROM product_jan p
LEFT JOIN Vendor v ON p.vendor_id = v.vendor_id;
```

c) RIGHT JOIN

- Equivalent to a RIGHT JOIN from product_jan to Vendor

```
SELECT p.product_id, p.product_name, p.price, v.vendor_name
FROM Vendor v
LEFT JOIN product_jan p ON p.vendor_id = v.vendor_id;
```

d) FULL OUTER JOIN (MySQL doesn't support directly)

- Workaround using UNION:

```
SELECT p.product_id, p.product_name, p.price, v.vendor_name
FROM Vendor v
LEFT JOIN product_jan p ON p.vendor_id = v.vendor_id
UNION SELECT p.product_id, p.product_name, p.price, v.vendor_name
FROM product_jan p
LEFT JOIN Vendor v ON p.vendor_id = v.vendor_id WHERE v.vendor_id IS
NULL;
```

----------OUTPUT ----------

Name: Bibhu kumar mishra
Abhishek Kumar

a) INNER JOIN

product_id	product_name	price	vendor_name
101	Laptop	1200	Tech World
102	Mouse	25.5	Accessory King
103	Keyboard	75	Accessory King

b) LEFT JOIN

product_id	product_name	price	vendor_name
101	Laptop	1200	Tech World
102	Mouse	25.5	Accessory King
103	Keyboard	75	Accessory King

c) RIGHT JOIN

product_id	product_name	price	vendor_name
101	Laptop	1200	Tech World
			Gadget Hub
102	Mouse	25.5	Accessory King
103	Keyboard	75	Accessory King

d) FULL OUTER JOIN (MySQL doesn't support directly)

product_id	product_name	price	vendor_name
			Gadget Hub
101	Laptop	1200	Tech World

102	Mouse	25.5	Accessory King
103	Keyboard	75	Accessory King

Name: Bibhu kumar mishra
Abhishek Kumar

PRACTICAL - 12

Create view on employee table.

```
CREATE VIEW Employee_Salary_View AS
SELECT
  employee_id,
  first_name,
  last_name,
  salary
FROM Employee;


SELECT * FROM Employee_Salary_View;
```

----------OUTPUT ----------

employee_id	first_name	last_name	salary
1	Alice	Johnson	55000
2	Bob	Smith	72000.5
3	Charlie	Nguyen	68000
4	Diana	Patel	49000.75
5	Ethan	Brown	81000

Name: Bibhu kumar mishra
Abhishek Kumar

PRACTICAL - 13

Perform TCL operation on employee table.

1. **Start a transaction**

```
BEGIN TRANSACTION;
```

2. **Make changes (INSERT, UPDATE, DELETE)**

```
UPDATE Employee
SET salary = salary + 5000
WHERE employee_id = 2;

INSERT INTO Employee (employee_id, first_name, last_name, age, salary)
VALUES (6, 'Fiona', 'Lee', 30, 60000);
```

3. **Check the changes before committing**

```
SELECT * FROM Employee;
```

4. **Commit the transaction (make changes permanent)**

```
COMMIT;
```

5. **Rollback example (undo changes)**

```
BEGIN TRANSACTION;

UPDATE Employee
SET salary = salary - 1000
WHERE employee_id = 3;
```

ROLLBACK;

6. Using SAVEPOINT

```
BEGIN TRANSACTION;

UPDATE Employee
SET salary = salary + 2000
WHERE employee_id = 1;

SAVEPOINT bonus_applied;

UPDATE Employee
SET salary = salary + 3000
WHERE employee_id = 4;


ROLLBACK TO SAVEPOINT bonus_applied;

COMMIT;
```

----------OUTPUT ----------

3. Check the changes before committing

employee_id	first_name	last_name	age	salary
1	Alice	Johnson	29	55000
2	Bob	Smith	35	77000.5
3	Charlie	Nguyen	41	68000
4	Diana	Patel	26	49000.75
5	Ethan	Brown	38	81000

6	Fiona	Lee	30	60000

Name: Bibhu kumar mishra
Abhishek Kumar

PRACTICAL – 14

write PL/SQL program to swap two numbers

```
DECLARE
  num1 NUMBER := 10; -- First number
  num2 NUMBER := 20; -- Second number
  temp NUMBER;      -- Temporary variable to hold value during swap
BEGIN
  -- Displaying the values before swapping
  DBMS_OUTPUT.PUT_LINE('Before Swap:');
  DBMS_OUTPUT.PUT_LINE('num1 = ' || num1);
  DBMS_OUTPUT.PUT_LINE('num2 = ' || num2);
  -- Swapping logic using a temporary variable
  temp := num1;
  num1 := num2;
  num2 := temp;
  -- Displaying the values after swapping
  DBMS_OUTPUT.PUT_LINE('After Swap:');
  DBMS_OUTPUT.PUT_LINE('num1 = ' || num1);
  DBMS_OUTPUT.PUT_LINE('num2 = ' || num2);
END;
/
```

----------OUTPUT ----------

Before Swap:

num1 = 10

num2 = 20

After Swap:

num1 = 20

num2 = 10

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.008

Name: Bibhu kumar mishra
Abhishek Kumar

PRACTICAL – 15

Write PL/SQL program to reverse the string.

```
DECLARE
  original_string VARCHAR2(100) := 'Hello, World!';  -- String to be reversed
  reversed_string VARCHAR2(100);  -- To hold the reversed string
  i NUMBER;  -- Loop counter
BEGIN
  -- Initialize the reversed_string as an empty string
  reversed_string := '';

  -- Loop through the original string in reverse order
  FOR i IN REVERSE 1..LENGTH(original_string) LOOP
    reversed_string := reversed_string || SUBSTR(original_string, i, 1);
  END LOOP;

  -- Output the original and reversed strings
  DBMS_OUTPUT.PUT_LINE('Original String: ' || original_string);
  DBMS_OUTPUT.PUT_LINE('Reversed String: ' || reversed_string);
END;
/
```

----------OUTPUT ----------

```
Original String: Hello, World!
Reversed String: !dlroW ,olleH


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.009
```

Name: Bibhu kumar mishra
Abhishek Kumar

PRACTICAL – 16

Write PL/SQL program to find the greatest of two numbers by taking user input.

```
SET SERVEROUTPUT ON;

DECLARE
  num1 NUMBER;
  num2 NUMBER;
BEGIN
  -- Take input from the user
  num1 := &num1;  -- Prompts user to enter first number
  num2 := &num2;  -- Prompts user to enter second number

  -- Compare and display the greatest number
  IF num1 > num2 THEN
    DBMS_OUTPUT.PUT_LINE('The greatest number is: ' || num1);
  ELSIF num2 > num1 THEN
    DBMS_OUTPUT.PUT_LINE('The greatest number is: ' || num2);
  ELSE
    DBMS_OUTPUT.PUT_LINE('Both numbers are equal.');
  END IF;
END;
/
```

----------OUTPUT ----------

```
Num1
Enter a value: 78
Num2
Enter a value: 55

The greatest number is: 78


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.007
```

Name: Bibhu kumar mishra
Abhishek Kumar

PRACTICAL – 17

Write PL/SQL program to find a palindrome of number by taking user input.

```
SET SERVEROUTPUT ON;

DECLARE
  num      NUMBER;   -- Original number entered by user
  original  NUMBER;   -- Copy of original number
  reversed  NUMBER := 0; -- To store reversed number
  remainder NUMBER;   -- To store each digit during reversal
BEGIN
  -- Taking input from user
  num := &num;
  original := num;

  -- Reverse the number
  WHILE num > 0 LOOP
    remainder := MOD(num, 10);             -- Extract last digit
    reversed  := (reversed * 10) + remainder; -- Build reversed number
    num := TRUNC(num / 10);               -- Remove last digit
  END LOOP;

  -- Check if the number is palindrome
  IF original = reversed THEN
    DBMS_OUTPUT.PUT_LINE('The number ' || original || ' is a palindrome.');
  ELSE
    DBMS_OUTPUT.PUT_LINE('The number ' || original || ' is NOT a
palindrome.');
  END IF;
END;
/
```

Name: Bibhu kumar mishra
Abhishek Kumar

----------OUTPUT ----------

```
The number 56146 is NOT a palindrome.


PL/SQL procedure successfully completed.

Elapsed: 00:00:00.002
```

Name: Bibhu kumar mishra
Abhishek Kumar

PRACTICAL – 18

Create a procedure to insert a new record into the employe table.

```
SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE add_employee (
  p_employee_id IN Employee.employee_id%TYPE,
  p_first_name  IN Employee.first_name%TYPE,
  p_last_name   IN Employee.last_name%TYPE,
  p_age       IN Employee.age%TYPE,
  p_salary    IN Employee.salary%TYPE
)
IS
BEGIN
  -- Insert a new employee record
  INSERT INTO Employee (employee_id, first_name, last_name, age, salary)
  VALUES (p_employee_id, p_first_name, p_last_name, p_age, p_salary);

  DBMS_OUTPUT.PUT_LINE('Employee record added successfully: ' ||
p_first_name || ' ' || p_last_name);

EXCEPTION
  WHEN DUP_VAL_ON_INDEX THEN
    DBMS_OUTPUT.PUT_LINE('Error: Employee ID ' || p_employee_id || '
already exists.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Unexpected error: ' || SQLERRM);
END;
/
```

Name: Bibhu kumar mishra
Abhishek Kumar

----------OUTPUT ----------

Procedure ADD_EMPLOYEE compiled

No errors.
Elapsed: 00:00:00.020

Name: Bibhu kumar mishra
Abhishek Kumar

PRACTICAL – 19

Write a PL/SQL program using an explicit cursor to display the name and salary of each employee from the Employee table.

```
DECLARE -- Define variables to hold cursor data v_first_name
Employee.first_name%TYPE; v_last_name
Employee.last_name%TYPE; v_salary Employee.salary%TYPE;

-- Define the explicit cursor
CURSOR emp_cursor IS
  SELECT first_name, last_name, salary
  FROM Employee;


BEGIN -- Open the cursor OPEN emp_cursor;

LOOP
  -- Fetch one record at a time
  FETCH emp_cursor INTO v_first_name, v_last_name, v_salary;

  -- Exit loop when no more rows
  EXIT WHEN emp_cursor%NOTFOUND;

  -- Display the result
  DBMS_OUTPUT.PUT_LINE('Employee: ' || v_first_name || ' ' ||
v_last_name ||
                ' | Salary: $' || TO_CHAR(v_salary, '999,999.99'));
END LOOP;

-- Close the cursor
CLOSE emp_cursor;


END;
```

----------OUTPUT ----------

Employee: Alice Johnson | Salary: $ 55,000.00

Employee: Bob Smith | Salary: $ 72,000.50
Employee: Charlie Nguyen | Salary: $ 68,000.00
Employee: Diana Patel | Salary: $ 49,000.75
Employee: Ethan Brown | Salary: $ 81,000.00

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.038

Name: Bibhu kumar mishra
Abhishek Kumar

PRACTICAL – 20

Write a PL/SQL stored function named getTotalSal that returns the totalsalary of all employee whose age is greater than 20.

```
CREATE OR REPLACE FUNCTION getTotalSal
RETURN NUMBER
IS
  v_total_salary NUMBER := 0;
BEGIN
  -- Calculate total salary for employees with age > 20
  SELECT SUM(salary)
  INTO v_total_salary
  FROM Employee
  WHERE age > 20;

  -- If no rows match, handle NULL result
  IF v_total_salary IS NULL THEN
    v_total_salary := 0;
  END IF;

  RETURN v_total_salary;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
    RETURN 0;
END;
/
```

----------OUTPUT ----------

Function GETTOTALSAL compiled

No errors.

Elapsed: 00:00:00.016