**PRACTICAL : 2**

**AIM : To understand & apply the following SQL concepts:**

* **Logical Operators: AND, OR, NOT**
* **BETWEEN...AND, NOT BETWEEN...AND**
* **LIKE Predicate**
* **IN, NOT IN Predicates**
* **GROUP BY**
* **Scalar / Aggregate Functions**

1. **Logical Operators: AND, OR, NOT**

▶ **AND – All conditions must be true**

**SELECT** \* **FROM** Employees

**WHERE** Department = 'HR' **AND** Salary > 50000;

**Explanation**: Fetch employees who are in HR **and** earn more than 50,000.

▶ **OR – At least one condition must be true**

**SELECT** \* **FROM** Employees

**WHERE** Department = 'HR' **OR** Salary > 50000;

**Explanation**: Fetch employees who are in HR **or** earn more than 50,000.

▶ **NOT – Negates a condition SELECT** \* **FROM** Employees **WHERE NOT** Department = 'HR';

**Explanation**: Fetch employees **not** in the HR department.

1. **BETWEEN … AND | NOT BETWEEN … AND**

▶ **BETWEEN**

**SELECT** \* **FROM** Products

**WHERE** Price **BETWEEN** 100 **AND** 500;

**Explanation**: Fetch products with price **from 100 to 500** (inclusive).

▶ **NOT BETWEEN**

**SELECT** \* **FROM** Products

**WHERE** Price **NOT BETWEEN** 100 **AND** 500;

**Explanation**: Fetch products with price **less than 100 or more than 500**.

1. **LIKE Predicate Wildcards:**

% → zero or more characters

\_ → exactly one character

▶ **Examples:**

*-- Starts with 'A'*

**SELECT** \* **FROM** Customers

**WHERE** Name **LIKE** 'A%';

*-- Ends with 'son'*

**SELECT** \* **FROM** Customers

**WHERE** Name **LIKE** '%son';

*-- Contains 'mit'*

**SELECT** \* **FROM** Customers

**WHERE** Name **LIKE** '%mit%';

*-- Exactly 5-letter names starting with 'A'* **SELECT** \* **FROM** Customers **WHERE** Name **LIKE** 'A ';

1. **IN and NOT IN Predicates**

▶ **IN**

**SELECT** \* **FROM** Students

**WHERE** Grade **IN** ('A', 'B', 'C');

**Explanation**: Fetch students with grade A, B, or C.

▶ **NOT IN**

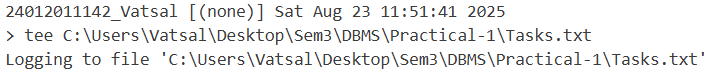
**SELECT** \* **FROM** Students

**WHERE** Grade **NOT IN** ('F', 'D');

**Explanation**: Fetch students whose grades are **not** F or D.

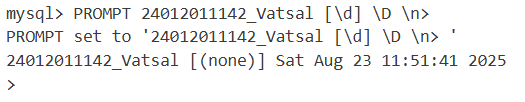
**PREREQUISITE STEPS**

**Step 1 : Use tee PATH\TO\FOLDER\FILENAME.txt in MySQL CLI to start saving all commands and outputs of the current session into a file until stopped with notee or else session is closed.**

****

* **SYNTAX : tee PATH\TO\FOLDER\FILENAME.txt**
* tee in MySQL Command-Line Client enables logging of all commands and their results into a file during the same session. From this point onward, all entered commands and their outputs will be written to the specified file.
* This is useful for: Record keeping, Creating reports of SQL execution, Debugging query results later. The log continues until you turn it off with “notee” Command

**Step 2 : Changing the MySQL Prompt**

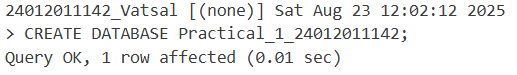
****

PROMPT changes how the MySQL CLI prompt looks.

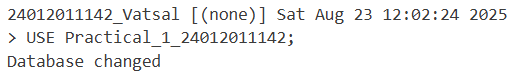
Components:

* FULLENROLLMENT\_FULLNAME → Custom label or project name
* [\d] → Shows the current database in square brackets
* \D → Shows the current date
* \n → Inserts a newline before
* > → The actual command prompt symbol

**Step 3 : Either Create a new Database or Use Existing Database**



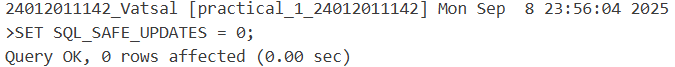
**Step 4 : Select a Existing Database**



**REMOVE Note : (also remove these types of notes after understanding)   
note : option a : you can use existing database created for practical 1 as mentioned above (only if it is performed on your personal system & if you have successfully completed,**

**REMOVE Note : option b : otherwise you need to create new database for practical 2 then either use earlier typed queries or create new tables with data**

**REMOVE Note : for the first question (drop address column from account table (if not created, create it first)), you will require address column to drop it, if you have performed following option a, there will be no issue, but if you have done it using option b you need to firstly add new column address : “ALTER TABLE ACCOUNT\_FULLENROLLMENT ADD address VARCHAR(20);”, then you will be able to drop column for q-1.**

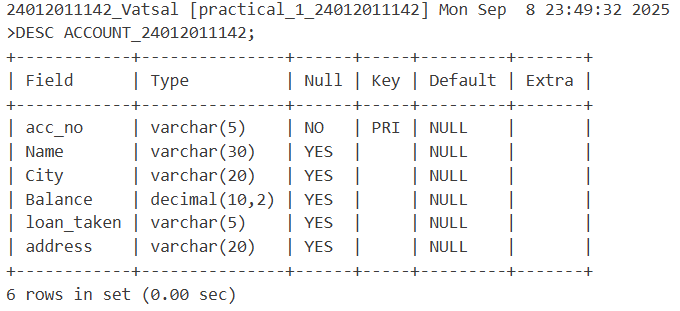
****

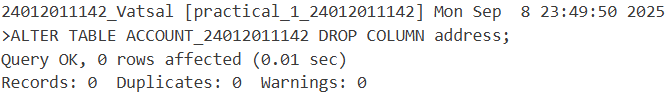
* **Purpose:** Disables safe update mode in MySQL for the current session.
* **Safe Mode Function:** Normally blocks UPDATE or DELETE without WHERE, indexed column, or LIMIT.
* **Effect of Disabling:** Allows unrestricted updates or deletions, even on all rows.
* **Common Use:** Bulk updates/deletions without strict conditions.
* **Scope:** Applies only to the current session; resets after disconnecting.
* **Caution:** Increases risk of accidental data loss—use carefully.
* **Re-enable:** Use SET SQL\_SAFE\_UPDATES = 1; when finished.

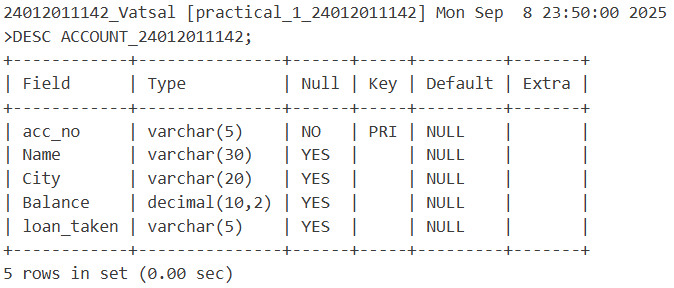
**TASK**

**Answer following Queries based on above 3 tables.**

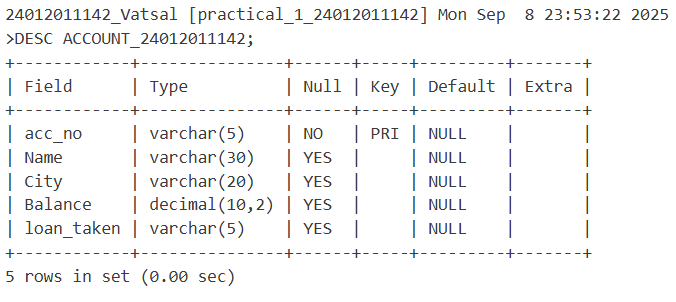
1. **Drop address column from ACCOUNT\_24012011142 table (if not created, create it first).**

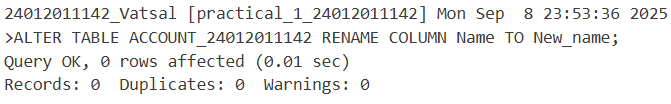
****

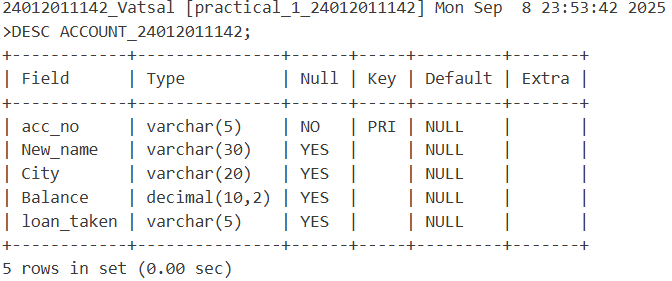
****

****

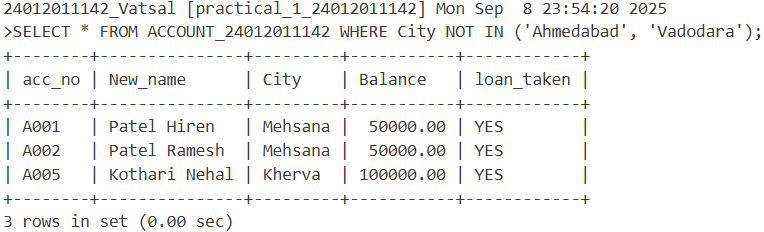
1. **Rename Name to New\_name in Account table.**

****

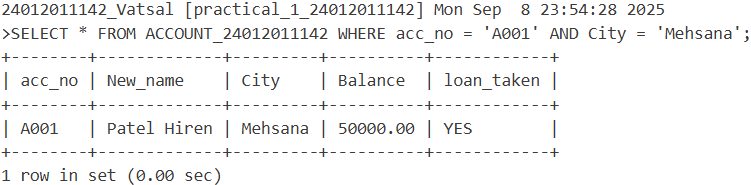
****

****

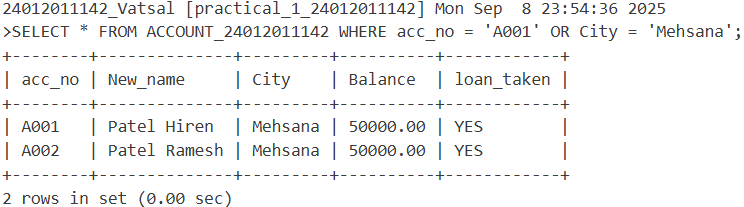
1. **Retrieve specific information for the account holder who are not in ‘Ahmedabad’ or ‘Vadodara’.**

****

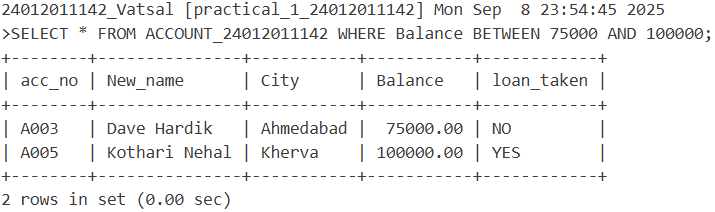
1. **Display only those data whose account number is ‘A001’ and city is ‘Mehsana’.**

****

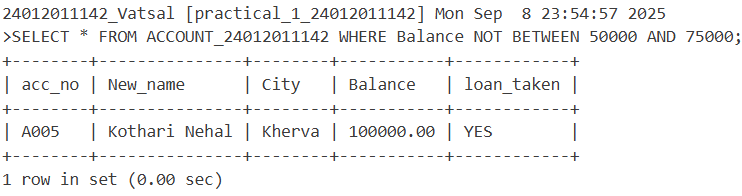
1. **Display only those data whose account number is ‘A001’ or city is ‘Mehsana’.**

****

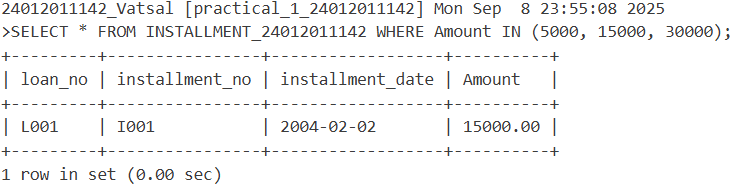
1. **Retrieve those records of Account holders whose balance between 75000 and 100000.**

****

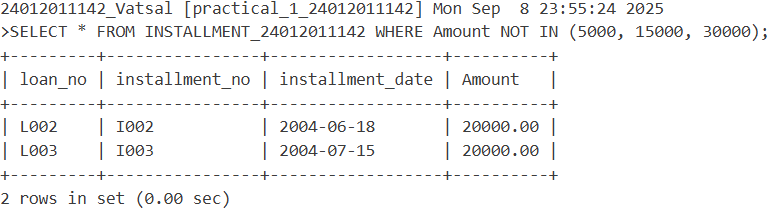
1. **Retrieve those records of Account holders whose balance not between 50000 and 75000.**

****

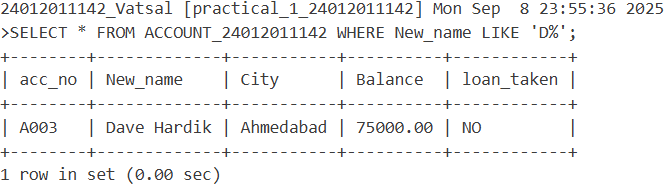
1. **Display only those records whose amount is 5000, 15000, 30000 from installment table.**

****

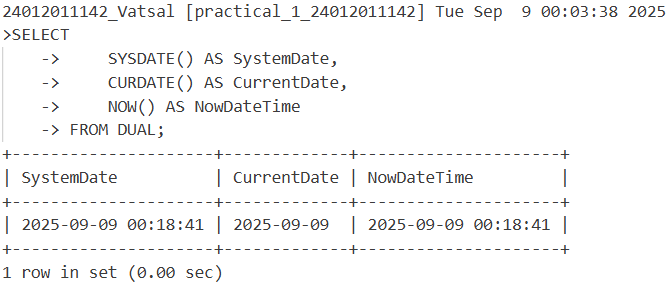
1. **Display only those records whose amount is not in 5000, 15000, 30000 from installment table.**

****

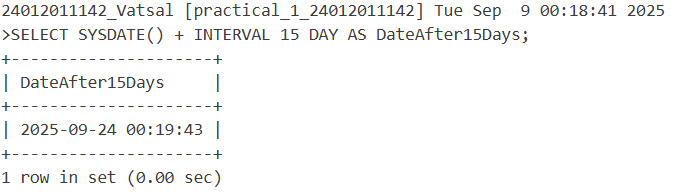
1. **Display those records of account holders whose name starts with ‘D’.**

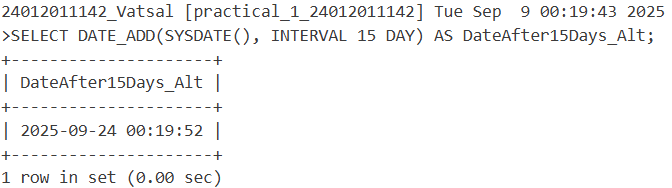
****

1. **Display System date.**

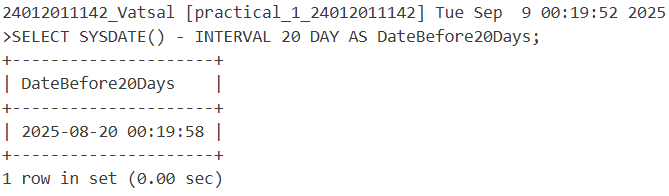
****

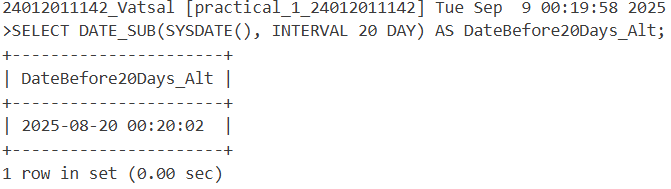
1. **Find the date,15 days after today’s date.**

****

****

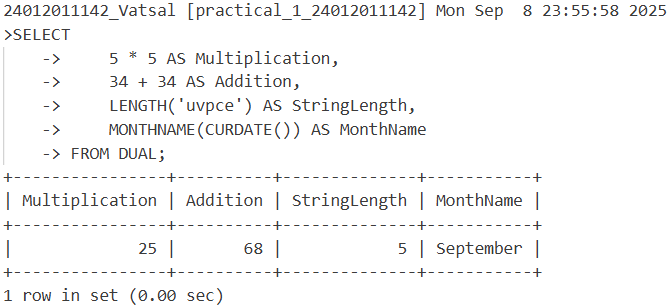
1. **Find the date,20 days before today’s date.**

****

****

1. **Perform the following operation using DUAL table.**

**5\*5, 34+34, 1000/300, length of ‘uvpce’, display only month of system date**

****

**Function Based Queries :**

1. **GROUP BY**

**The GROUP BY clause is used to group rows that have the same values in specified columns. It is commonly used with aggregate functions such as SUM(), AVG(), COUNT(), MAX(), and MIN().**

**Syntax:**

**SELECT column\_name, AGGREGATE\_FUNCTION(column\_name)**

**FROM table\_name**

**GROUP BY column\_name;**

**Example:**

**-- Total salary by department**

**SELECT Department, SUM(Salary) AS Total\_Salary**

**FROM Employees**

**GROUP BY Department;**

**-- Average marks by subject**

**SELECT Subject, AVG(Marks) AS Average\_Marks**

**FROM Results**

**GROUP BY Subject;**

1. **SCALAR FUNCTIONS**

**Scalar functions return a single value based on the input value. They operate on each row individually.**

**Example:**

**-- Convert name to uppercase**

**SELECT UPPER(Name) AS Upper\_Name FROM Customers;**

**-- Round salary to nearest thousand**

**SELECT Name, ROUND(Salary, -3) AS Rounded\_Salary FROM Employees;**

**-- Length of name**

**SELECT Name, LENGTH(Name) AS Name\_Length FROM Customers;**

**-- Current date and time**

**SELECT NOW() AS Current\_Timestamp;**

* Common Scalar Functions:

|  |  |
| --- | --- |
| **Function** | **Description** |
| UPPER() | Converts text to uppercase |
| LOWER() | Converts text to lowercase |
| LEN() or LENGTH() | Returns length of a string |
| ROUND() | Rounds numeric value |
| NOW() | Returns current date & time |
| GETDATE() | Same as NOW() in some DBs |
| ABS() | Returns absolute value |

**Summary Table**

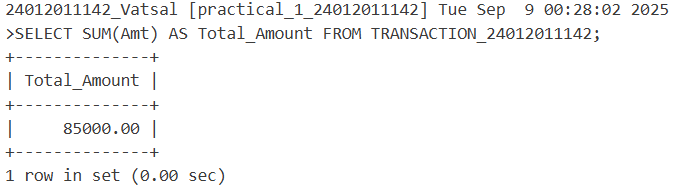
|  |  |  |
| --- | --- | --- |
| **Concept** | **Used For** | **Example** |
| GROUP BY | Grouping data & aggregation | GROUP BY Department |
| COUNT() | Count rows | COUNT(\*) |
| SUM() | Sum of values | SUM(Salary) |
| AVG() | Average of values | AVG(Marks) |
| UPPER() | Convert to uppercase | UPPER(Name) |
| ROUND() | Round numbers | ROUND(Salary, 2) |
| LENGTH() | Get string length | LENGTH(Name) |
| NOW() | Current date and time | NOW() |

**Create TABLE “TRANSACTION\_yorEnNo” as given below.**

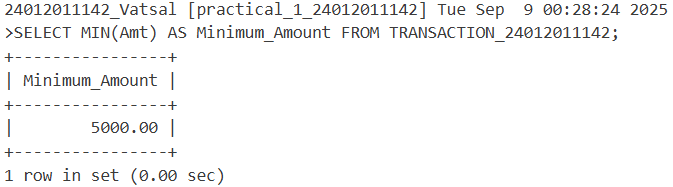
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Acc\_no** | **Tr\_date** | **Amt** | **Type\_of\_tr** | **Mode\_of\_pay** |
| A001 | 3-may-21 | 10000 | D | Cash |
| A002 | 5-july-21 | 5000 | W | Cheque |
| A003 | 12-Aug-21 | 25000 | D | Cheque |
| A004 | 15-may-21 | 30000 | D | Cheque |
| A005 | 22-oct-21 | 15000 | W | Cash |

**Perform given queries:**

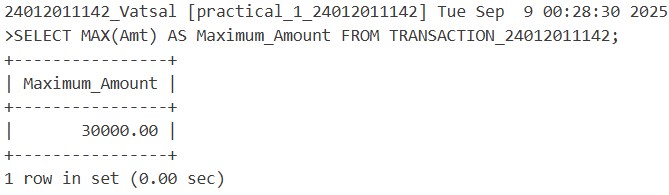
1. **Find the total transaction amount of account holder from transaction table.**

****

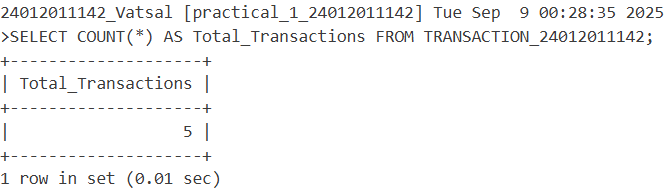
1. **Find minimum amount of transaction table.**

****

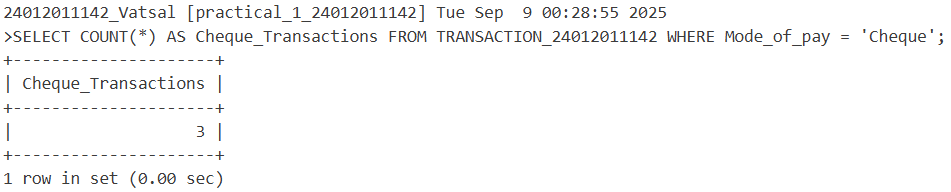
1. **Find maximum amount of transaction table.**

****

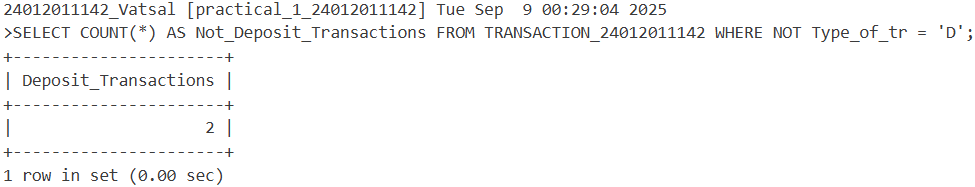
1. **Count the total account holders from transaction table.**

****

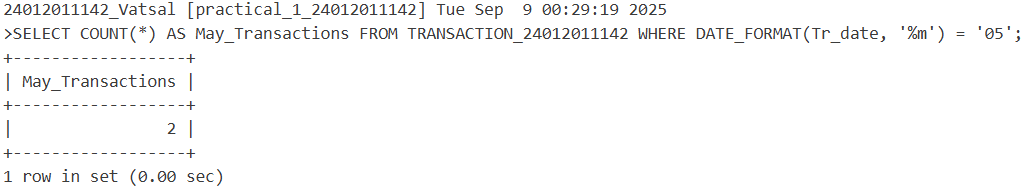
1. **Count only those records whose made of payment is ‘Cheque’.**

****

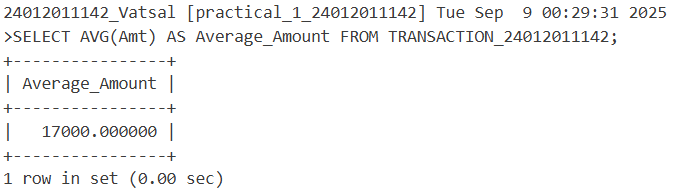
1. **Count only those records whose Type\_of\_tr is not ‘D’.**

****

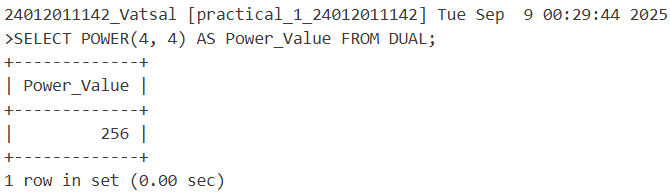
1. **Count only those records whose transaction made in the month of ‘may’.**

****

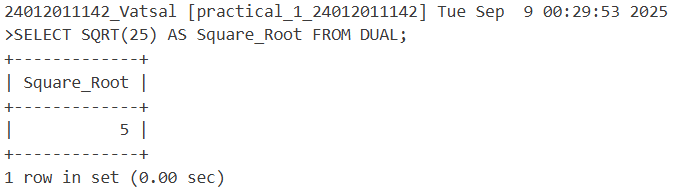
1. **Find the average value of transaction.**

****

1. **Display the result of 4 rest to 4 (use power function).**

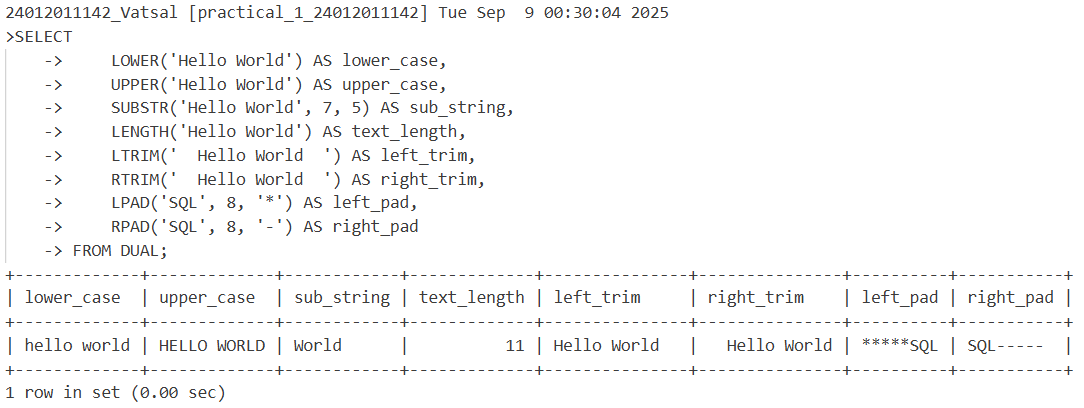
****

1. **Find the square root of 25 (use sqrt function).**

****

1. **Write the query for the following inbuilt Function.**

**LOWER, INITCAP, UPPER, SUBSTR, LENGTH, LTRIM, RTRIM, LPAD, RPAD.**

****