Name-Harshit Selarka
Prn-23070521131
Section-B
Batch-B2

String Functions in SQL*Plus (Oracle) & MySQL

String functions allow you to **manipulate and process text data** in SQL. Below is a detailed comparison of **SQL*Plus (Oracle)** and **MySQL** string functions, including examples.

1. String Functions in SQL*Plus (Oracle)

1.1 CONCAT – String Concatenation

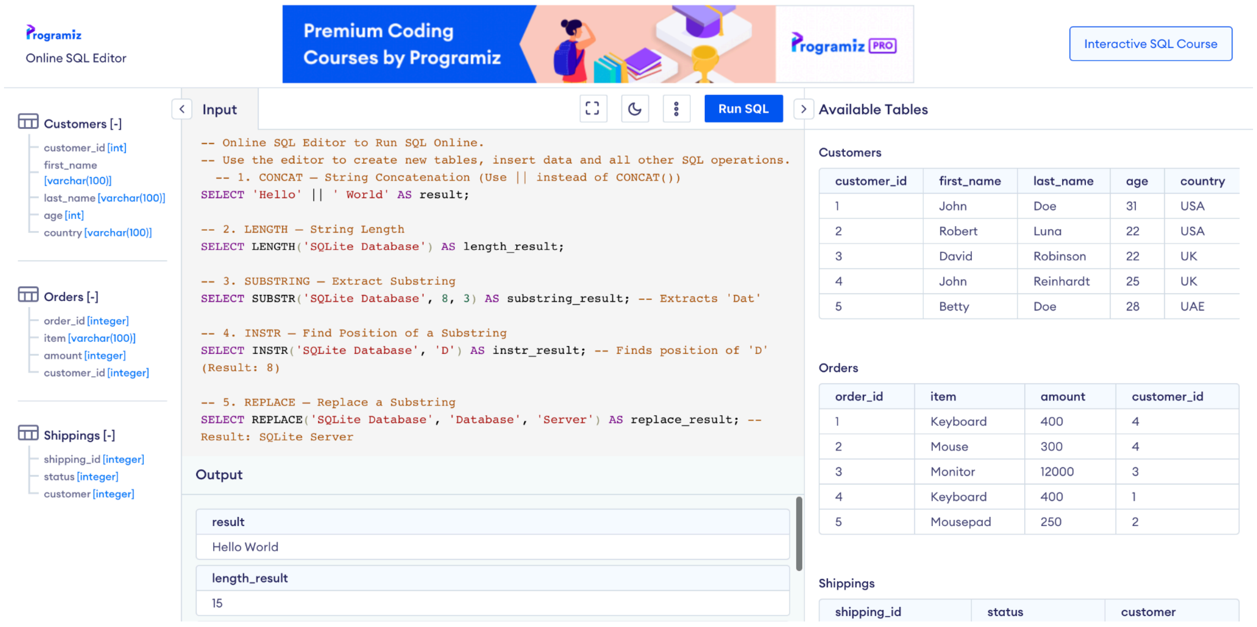
```
SELECT CONCAT('Hello', ' World') FROM dual; -- Result: Hello World
SELECT 'Hello' || ' World' FROM dual; -- Alternative method using ||
```

1.2 LENGTH – String Length

```
SELECT LENGTH('Oracle Database') FROM dual; -- Result: 16
```

1.3 SUBSTR – Extract Substring

```
SELECT SUBSTR('Oracle Database', 8, 3) FROM dual; -- Extracts 'Dat'
(Start from 8, length 3)
```



Customers [-]
- customer_id [int]
- first_name [varchar(100)]
- last_name [varchar(100)]
- age [int]
- country [varchar(100)]

Orders [-]
- order_id [integer]
- item [varchar(100)]
- amount [integer]
- customer_id [integer]

Shippings [-]
- shipping_id [integer]
- status [integer]
- customer [integer]

```
-- Online SQL Editor to Run SQL Online.
-- Use the editor to create new tables, insert data and all other SQL operations.
  -- 1. CONCAT – String Concatenation (Use || instead of CONCAT())
SELECT 'Hello' || ' World' AS result;

-- 2. LENGTH – String Length
SELECT LENGTH('SQLite Database') AS length_result;

-- 3. SUBSTRING – Extract Substring
SELECT SUBSTR('SQLite Database', 8, 3) AS substring_result; -- Extracts 'Dat'

-- 4. INSTR – Find Position of a Substring
SELECT INSTR('SQLite Database', 'D') AS instr_result; -- Finds position of 'D'
(Result: 8)

-- 5. REPLACE – Replace a Substring
SELECT REPLACE('SQLite Database', 'Database', 'Server') AS replace_result; --
Result: SQLite Server
```

```
result
Hello World

length_result
15
```

Customers

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE

Orders

order_id	item	amount	customer_id
1	Keyboard	400	4
2	Mouse	300	4
3	Monitor	12000	3
4	Keyboard	400	1
5	Mousepad	250	2

Shippings

shipping_id	status	customer

1.4 INSTR – Find Position of a Substring

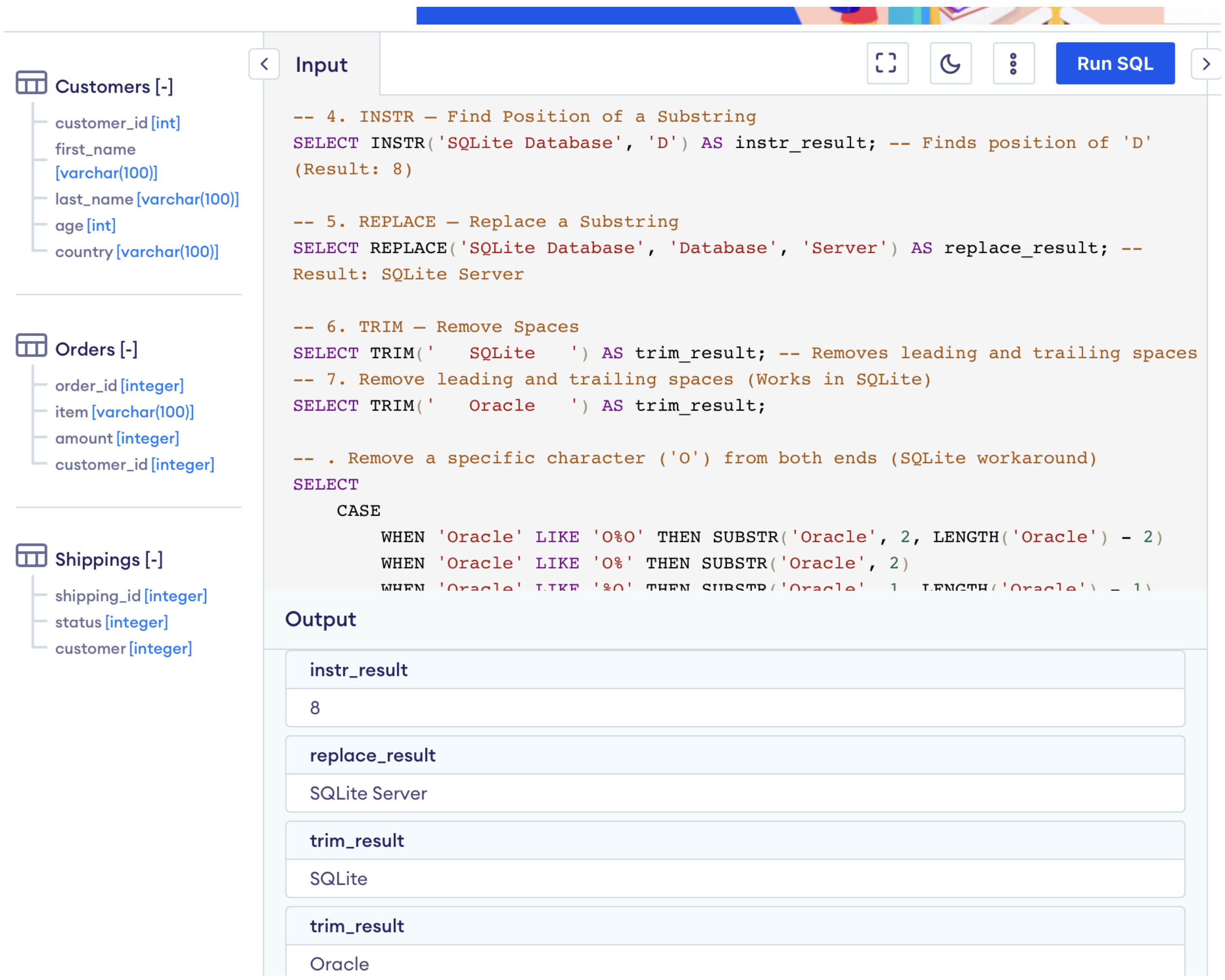
```
SELECT INSTR('Oracle Database', 'D') FROM dual; -- Finds position of
'D' (Result: 8)
```

1.5 REPLACE – Replace a Substring

```
SELECT REPLACE('Oracle Database', 'Database', 'SQL') FROM dual; --
Result: Oracle SQL
```

1.6 TRANSLATE – Replace Multiple Characters

```
SELECT TRANSLATE('123-456-7890', '123', 'XYZ') FROM dual; -- Result:
XYZ-456-7890
```



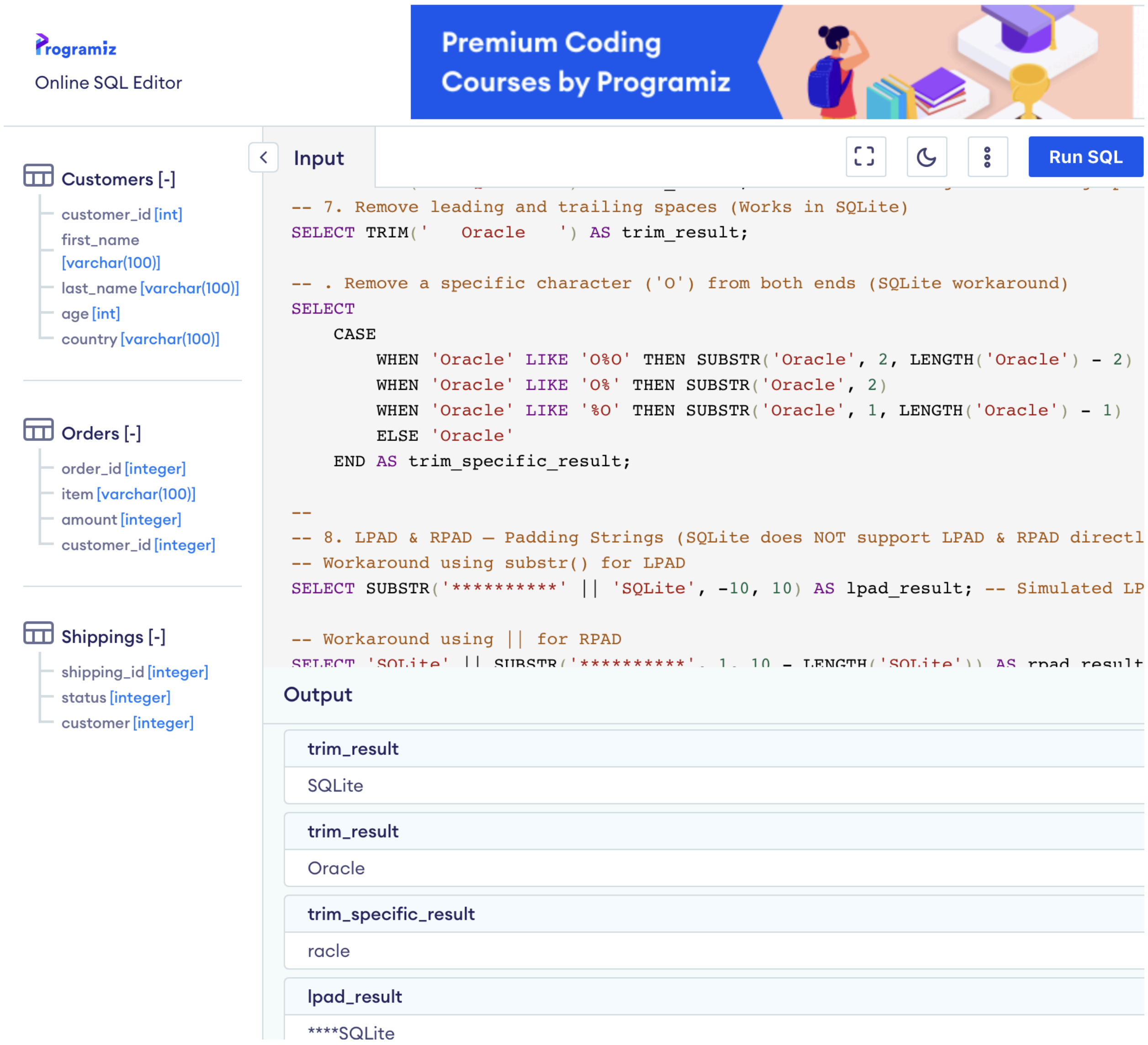
```
Customers [-]
  customer_id [int]
  first_name
  [varchar(100)]
  last_name [varchar(100)]
  age [int]
  country [varchar(100)]

Orders [-]
  order_id [integer]
  item [varchar(100)]
  amount [integer]
  customer_id [integer]

Shippings [-]
  shipping_id [integer]
  status [integer]
  customer [integer]
```

```
-- 4. INSTR – Find Position of a Substring
SELECT INSTR('SQLite Database', 'D') AS instr_result; -- Finds position of 'D'
(Result: 8)

-- 5. REPLACE – Replace a Substring
SELECT REPLACE('SQLite Database', 'Database', 'Server') AS replace_result; --
Result: SQLite Server

-- 6. TRIM – Remove Spaces
SELECT TRIM('   SQLite   ') AS trim_result; -- Removes leading and trailing spaces
-- 7. Remove leading and trailing spaces (Works in SQLite)
SELECT TRIM('   Oracle   ') AS trim_result;

-- . Remove a specific character ('O') from both ends (SQLite workaround)
SELECT
    CASE
        WHEN 'Oracle' LIKE 'O%O' THEN SUBSTR('Oracle', 2, LENGTH('Oracle') - 2)
        WHEN 'Oracle' LIKE 'O%' THEN SUBSTR('Oracle', 2)
        WHEN 'Oracle' LIKE '%O' THEN SUBSTR('Oracle', 1, LENGTH('Oracle') - 1)
```

```
instr_result
8

replace_result
SQLite Server

trim_result
SQLite

trim_result
Oracle
```

1.7 TRIM – Remove Spaces or Characters

```
SELECT TRIM(' Oracle ') FROM dual; -- Removes leading and trailing
spaces
SELECT TRIM('O' FROM 'Oracle') FROM dual; -- Removes 'O' from both
ends
```



Customers [-]
- customer_id [int]
- first_name [varchar(100)]
- last_name [varchar(100)]
- age [int]
- country [varchar(100)]

Orders [-]
- order_id [integer]
- item [varchar(100)]
- amount [integer]
- customer_id [integer]

Shippings [-]
- shipping_id [integer]
- status [integer]
- customer [integer]

```
-- 7. Remove leading and trailing spaces (Works in SQLite)
SELECT TRIM('   Oracle   ') AS trim_result;

-- . Remove a specific character ('O') from both ends (SQLite workaround)
SELECT
    CASE
        WHEN 'Oracle' LIKE 'O%O' THEN SUBSTR('Oracle', 2, LENGTH('Oracle') - 2)
        WHEN 'Oracle' LIKE 'O%' THEN SUBSTR('Oracle', 2)
        WHEN 'Oracle' LIKE '%O' THEN SUBSTR('Oracle', 1, LENGTH('Oracle') - 1)
        ELSE 'Oracle'
    END AS trim_specific_result;

--
-- 8. LPAD & RPAD – Padding Strings (SQLite does NOT support LPAD & RPAD directl
-- Workaround using substr() for LPAD
SELECT SUBSTR('**********' || 'SQLite', -10, 10) AS lpad_result; -- Simulated LP

-- Workaround using || for RPAD
SELECT 'SQLite' || SUBSTR('**********', 1, 10 - LENGTH('SQLite')) AS rpad_result
```

```
trim_result
SQLite

trim_result
Oracle

trim_specific_result
racle

lpad_result
****SQLite
```

1.8 LPAD & RPAD – Padding Strings

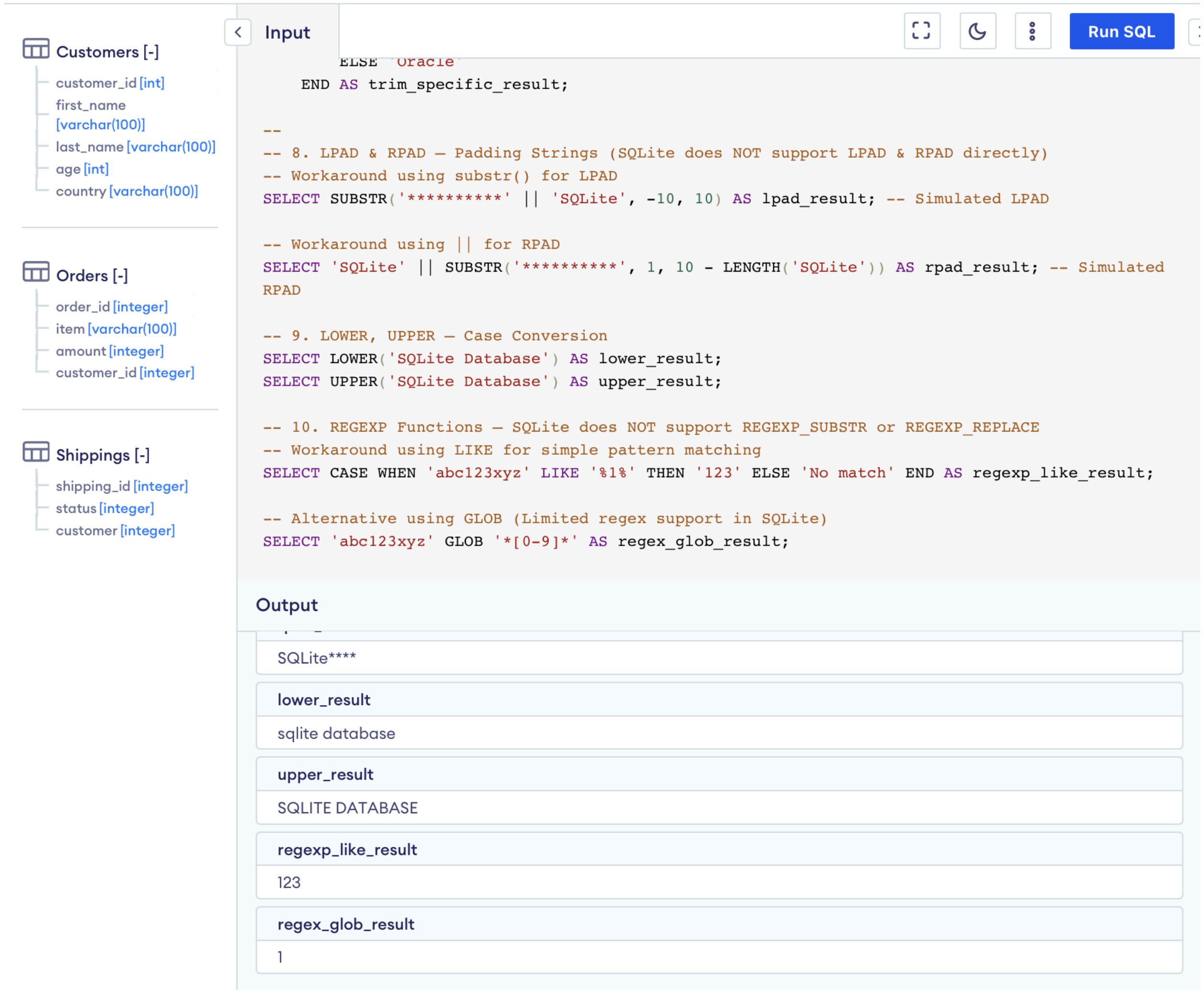
```
SELECT LPAD('Oracle', 10, '*') FROM dual; -- Result: ****Oracle
SELECT RPAD('Oracle', 10, '*') FROM dual; -- Result: Oracle****
```

1.9 LOWER, UPPER, INITCAP – Case Conversion

```
SELECT LOWER('Oracle Database') FROM dual; -- Result: oracle database
SELECT UPPER('Oracle Database') FROM dual; -- Result: ORACLE DATABASE
SELECT INITCAP('oracle database') FROM dual; -- Result: Oracle
Database
```

1.10 REGEXP Functions – Regular Expressions

```
SELECT REGEXP_SUBSTR('A123B456C', '[0-9]+') FROM dual; -- Extracts
first number (Result: 123)
SELECT REGEXP_REPLACE('abc123xyz', '[0-9]', '*') FROM dual; --
Replaces digits with '*' (Result: abc***xyz)
```



```
        ELSE 'Oracle'
    END AS trim_specific_result;

--
-- 8. LPAD & RPAD – Padding Strings (SQLite does NOT support LPAD & RPAD directly)
-- Workaround using substr() for LPAD
SELECT SUBSTR('**********' || 'SQLite', -10, 10) AS lpad_result; -- Simulated LPAD

-- Workaround using || for RPAD
SELECT 'SQLite' || SUBSTR('**********', 1, 10 - LENGTH('SQLite')) AS rpad_result; -- Simulated
RPAD

-- 9. LOWER, UPPER – Case Conversion
SELECT LOWER('SQLite Database') AS lower_result;
SELECT UPPER('SQLite Database') AS upper_result;

-- 10. REGEXP Functions – SQLite does NOT support REGEXP_SUBSTR or REGEXP_REPLACE
-- Workaround using LIKE for simple pattern matching
SELECT CASE WHEN 'abc123xyz' LIKE '%1%' THEN '123' ELSE 'No match' END AS regexp_like_result;

-- Alternative using GLOB (Limited regex support in SQLite)
SELECT 'abc123xyz' GLOB '*[0-9]*' AS regex_glob_result;
```

```
SQLite****

lower_result
sqlite database

upper_result
SQLITE DATABASE

regexp_like_result
123

regex_glob_result
1
```

2. String Functions in MySQL

2.1 CONCAT – String Concatenation

```
SELECT CONCAT('Hello', ' World'); -- Result: Hello World
```

2.2 LENGTH – String Length

```
SELECT LENGTH('MySQL Database'); -- Result: 15
```

2.3 SUBSTRING – Extract Substring

```
SELECT SUBSTRING('MySQL Database', 8, 3); -- Extracts 'Dat' (Start
from 8, length 3)
```

2.4 LOCATE & INSTR – Find Position of a Substring

```
SELECT LOCATE('D', 'MySQL Database'); -- Result: 8
SELECT INSTR('MySQL Database', 'D'); -- Result: 8
```

2.5 REPLACE – Replace a Substring

```
SELECT REPLACE('MySQL Database', 'Database', 'Server'); -- Result:
MySQL Server
```

2.6 TRIM – Remove Spaces or Characters

```
SELECT TRIM(' MySQL '); -- Removes leading and trailing spaces
SELECT TRIM('M' FROM 'MySQL'); -- Removes 'M' from both ends
```

2.7 LPAD & RPAD – Padding Strings

```
SELECT LPAD('MySQL', 10, '*'); -- Result: *****MySQL
SELECT RPAD('MySQL', 10, '*'); -- Result: MySQL*****
```

2.8 LOWER, UPPER – Case Conversion

```
SELECT LOWER('MySQL Database'); -- Result: my database
SELECT UPPER('MySQL Database'); -- Result: MYSQL DATABASE
```

2.9 REGEXP Functions – Regular Expressions

```
SELECT REGEXP_SUBSTR('abc123xyz', '[0-9]+'); -- Extracts first number
(Result: 123)
SELECT REGEXP_REPLACE('abc123xyz', '[0-9]', '*'); -- Replaces digits
with '*' (Result: abc***xyz)
```

3. Key Differences Between SQL*Plus (Oracle) and MySQL String Functions

Function	Oracle (SQL*Plus)	MySQL
Concatenation	CONCAT(str1, str2) or `'	

Substring	`SUBSTR(str, start, length)`	`SUBSTRING(str, start, length)`
Find Position	`INSTR(str, substring)`	`LOCATE(substring, str)` or `INSTR(str, substring)`
Replace Substring	`REPLACE(str, old, new)`	`REPLACE(str, old, new)`
Trim Spaces	`TRIM(str)`	`TRIM(str)`
Padding	`LPAD(str, length, pad_char)`, `RPAD(str, length, pad_char)`	`LPAD(str, length, pad_char)`, `RPAD(str, length, pad_char)`
Case Conversion	`UPPER(str)`, `LOWER(str)`, `INITCAP(str)`	`UPPER(str)`, `LOWER(str)`
Regular Expressions	`REGEXP_SUBSTR()`, `REGEXP_REPLACE()`	`REGEXP_SUBSTR()`, `REGEXP_REPLACE()`

4. Special Notes

- Oracle has **`INITCAP()`**, which capitalizes the first letter of each word, whereas MySQL does **not**.
- **`CONCAT()`** in Oracle only takes **two** arguments, while in MySQL it can take **multiple**.
- Regular expressions (`REGEXP_...`) are available in both, but Oracle has more advanced capabilities.