

## ASSIGNMENT-2

### SMARTBRIDGE EXTERNSHIP MODERN APPLICATION DEVELOPMENT

(JAVA SPRING BOOT)

Name: SOMURAJ.K

Reg\_No: 20MIS0043

Phone\_No: 9361926901

Campus: VIT Vellore

COMMANDS IN MYSQL AND MANGODB

# SCREENSHOTS OF MYSQL QUERIES

## 1.CREATE TABLE

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree with 'college' expanded, showing 'javaspringboot' and 'student' tables. The 'student' table is selected, showing its columns: 'stud\_id' (int PK), 'stud\_name' (varchar(45)), 'stud\_age' (varchar(45)), 'stud\_mark' (varchar(45)), and 'stud\_Academicyear' (varchar(45)). The main editor shows the SQL query to create the 'student' table:

```
1 CREATE TABLE student(  
2     stud_id INT,  
3     stud_name VARCHAR(20),  
4     stud_age VARCHAR(10),  
5     stud_joinyear DATE,  
6     Stud_closeyear DATE  
7 );
```

The 'Output' tab at the bottom shows the execution results:

#	Time	Action	Message
1	19:53:41	CREATE TABLE student( stud_id INT, stud_name VARCHAR(20), stud_age VARCHAR(10), ...	0 row(s) affected

## 2.INSERT VALUES

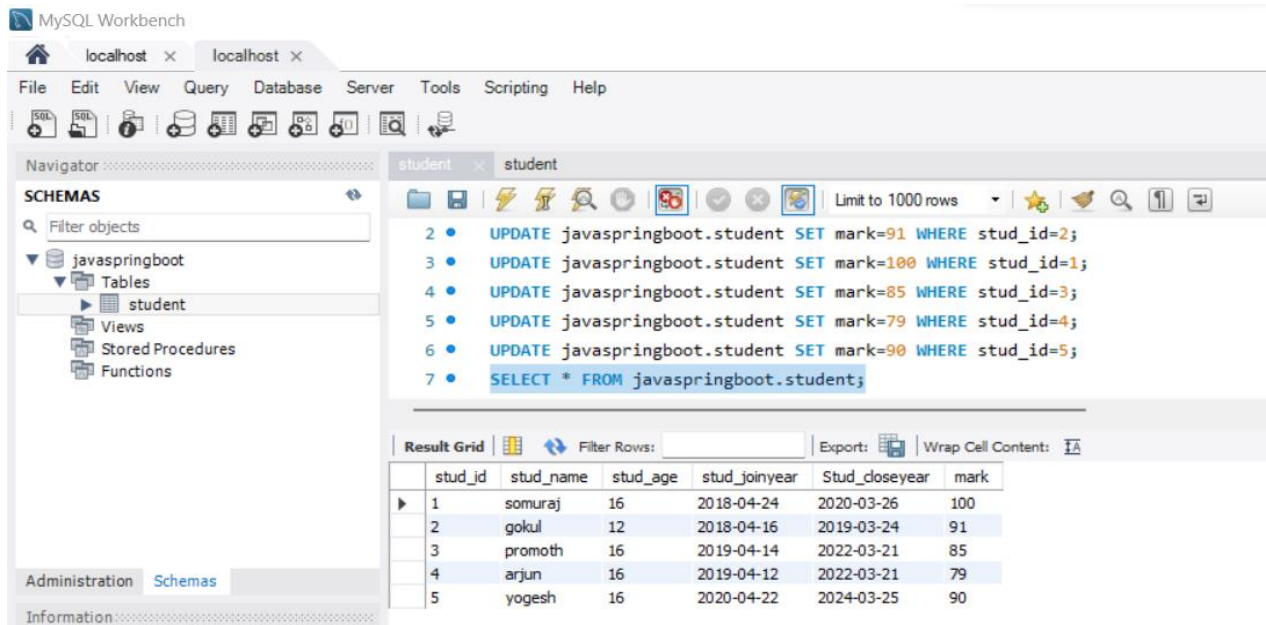
The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree with 'javaspringboot' expanded, showing 'student' table. The main editor shows the SQL query to insert five rows into the 'student' table:

```
1 INSERT INTO javaspringboot.student values(1,"somuraj","16","2018-04-24","2020-03-26");  
2 INSERT INTO javaspringboot.student values(2,"gokul","12","2018-04-16","2019-03-24");  
3 INSERT INTO javaspringboot.student values(3,"promoth","16","2019-04-14","2022-03-21");  
4 INSERT INTO javaspringboot.student values(4,"arjun","16","2019-04-12","2022-03-21");  
5 INSERT INTO javaspringboot.student values(5,"yogesh","16","2020-04-22","2024-03-25");
```

The 'Result Grid' at the bottom shows the data inserted:

stud_id	stud_name	stud_age	stud_joinyear	Stud_closeyear
1	somuraj	16	2018-04-24	2020-03-26
2	gokul	12	2018-04-16	2019-03-24
3	promoth	16	2019-04-14	2022-03-21
4	arjun	16	2019-04-12	2022-03-21
5	yogesh	16	2020-04-22	2024-03-25

### 3.ADDING NEW COLUMN

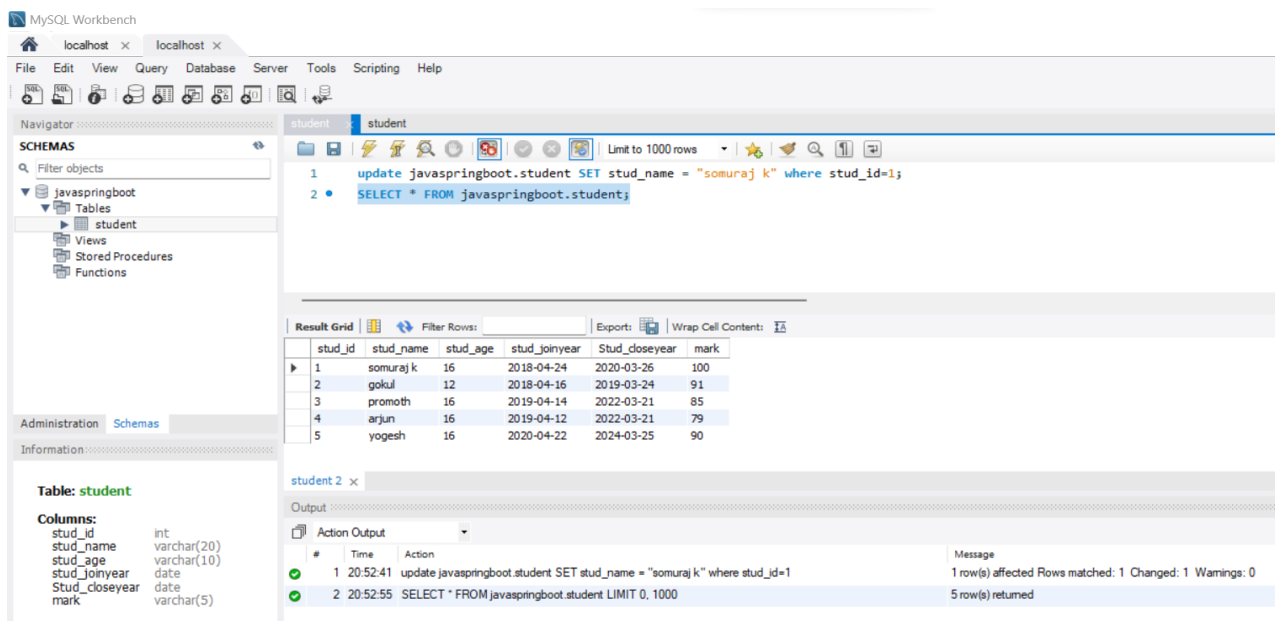


The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'javaspringboot' selected, containing 'Tables', 'Views', 'Stored Procedures', and 'Functions'. The 'student' table is selected under 'Tables'. The main editor shows a SQL query with seven statements: six UPDATE statements to set marks for specific students and one SELECT statement to view the results. The 'Result Grid' at the bottom displays the data after the updates.

```
2 • UPDATE javaspringboot.student SET mark=91 WHERE stud_id=2;
3 • UPDATE javaspringboot.student SET mark=100 WHERE stud_id=1;
4 • UPDATE javaspringboot.student SET mark=85 WHERE stud_id=3;
5 • UPDATE javaspringboot.student SET mark=79 WHERE stud_id=4;
6 • UPDATE javaspringboot.student SET mark=90 WHERE stud_id=5;
7 • SELECT * FROM javaspringboot.student;
```

stud_id	stud_name	stud_age	stud_joinyear	Stud_closeyear	mark
1	somuraj	16	2018-04-24	2020-03-26	100
2	gokul	12	2018-04-16	2019-03-24	91
3	promoth	16	2019-04-14	2022-03-21	85
4	arjun	16	2019-04-12	2022-03-21	79
5	yogesh	16	2020-04-22	2024-03-25	90

### 4.UPDATE THE VALUES



The screenshot shows the MySQL Workbench interface. The left sidebar is the same as in the previous screenshot. The main editor shows a SQL query with two statements: an UPDATE statement to change the name of the student with ID 1 to 'somuraj k' and a SELECT statement to view the results. The 'Result Grid' displays the data after the update. Below the result grid, the 'student 2' tab is active, showing the 'Output' section with the execution details of the two statements.

```
1 • update javaspringboot.student SET stud_name = "somuraj k" where stud_id=1;
2 • SELECT * FROM javaspringboot.student;
```

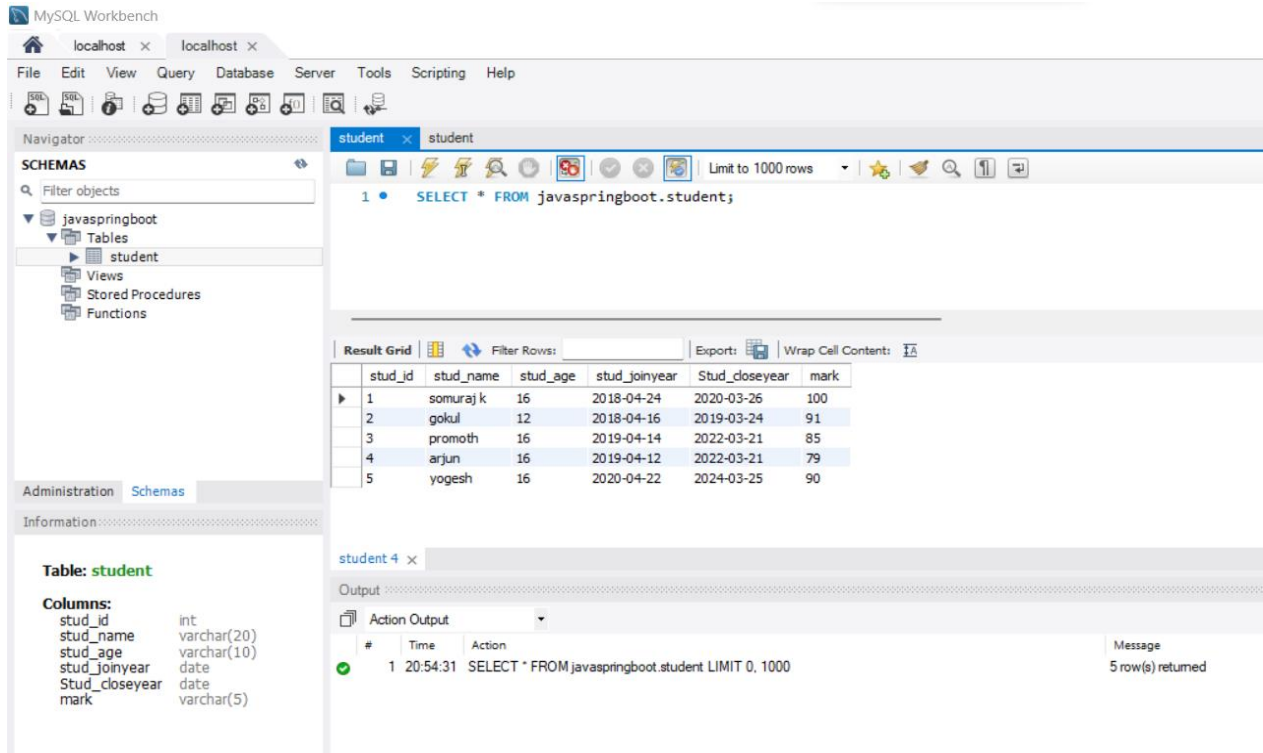
stud_id	stud_name	stud_age	stud_joinyear	Stud_closeyear	mark
1	somuraj k	16	2018-04-24	2020-03-26	100
2	gokul	12	2018-04-16	2019-03-24	91
3	promoth	16	2019-04-14	2022-03-21	85
4	arjun	16	2019-04-12	2022-03-21	79
5	yogesh	16	2020-04-22	2024-03-25	90

**student 2**

Output

#	Time	Action	Message
✓ 1	20:52:41	update javaspringboot.student SET stud_name = "somuraj k" where stud_id=1	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
✓ 2	20:52:55	SELECT * FROM javaspringboot.student LIMIT 0, 1000	5 row(s) returned

## 5.SELECT THE TABLE



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'javaspringboot' expanded, showing 'Tables' and 'student'. The main editor shows a query: `SELECT * FROM javaspringboot.student;`. The 'Result Grid' displays 5 rows of data:

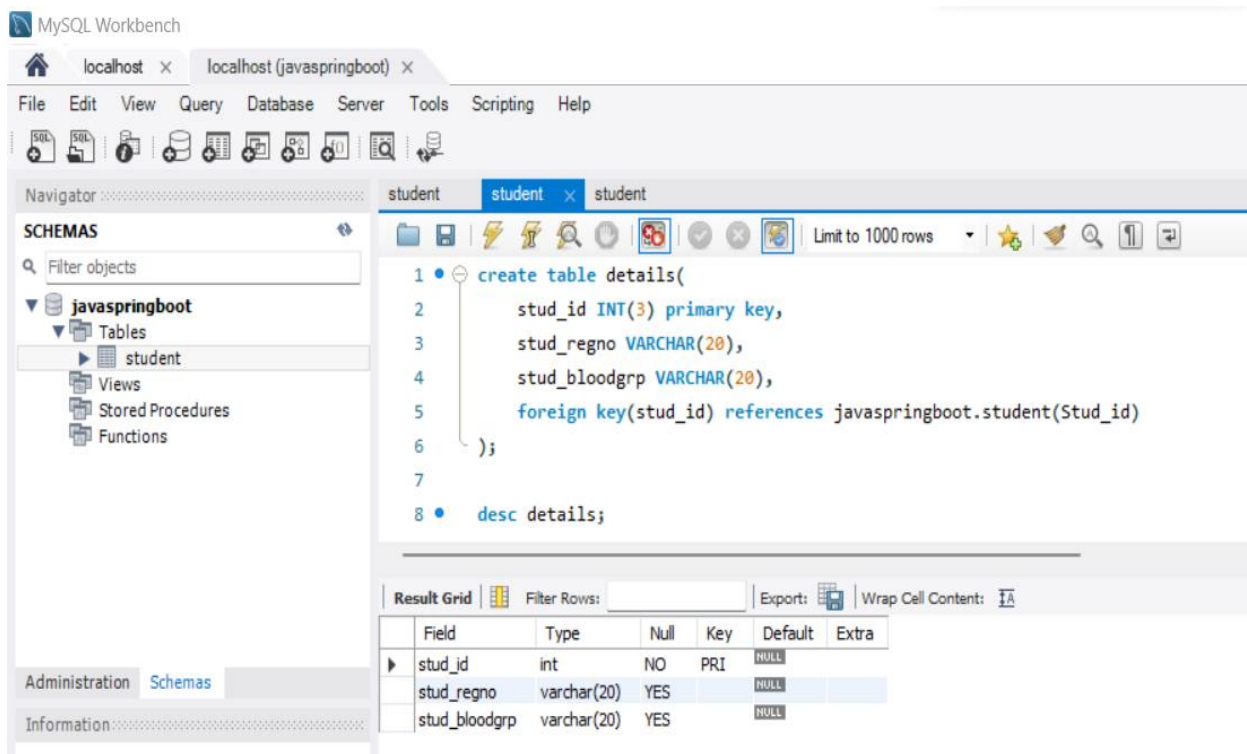
	stud_id	stud_name	stud_age	stud_joinyear	Stud_closeyear	mark
1		somuraj k	16	2018-04-24	2020-03-26	100
2		gokul	12	2018-04-16	2019-03-24	91
3		promoth	16	2019-04-14	2022-03-21	85
4		arjun	16	2019-04-12	2022-03-21	79
5		yogesh	16	2020-04-22	2024-03-25	90

The 'Information' tab shows the table structure for 'student':

Column	Type
stud_id	int
stud_name	varchar(20)
stud_age	varchar(10)
stud_joinyear	date
Stud_closeyear	date
mark	varchar(5)

The 'Output' tab shows the execution message: 'SELECT \* FROM javaspringboot.student LIMIT 0, 1000' and '5 row(s) returned'.

## 6.FOREIGN KEY



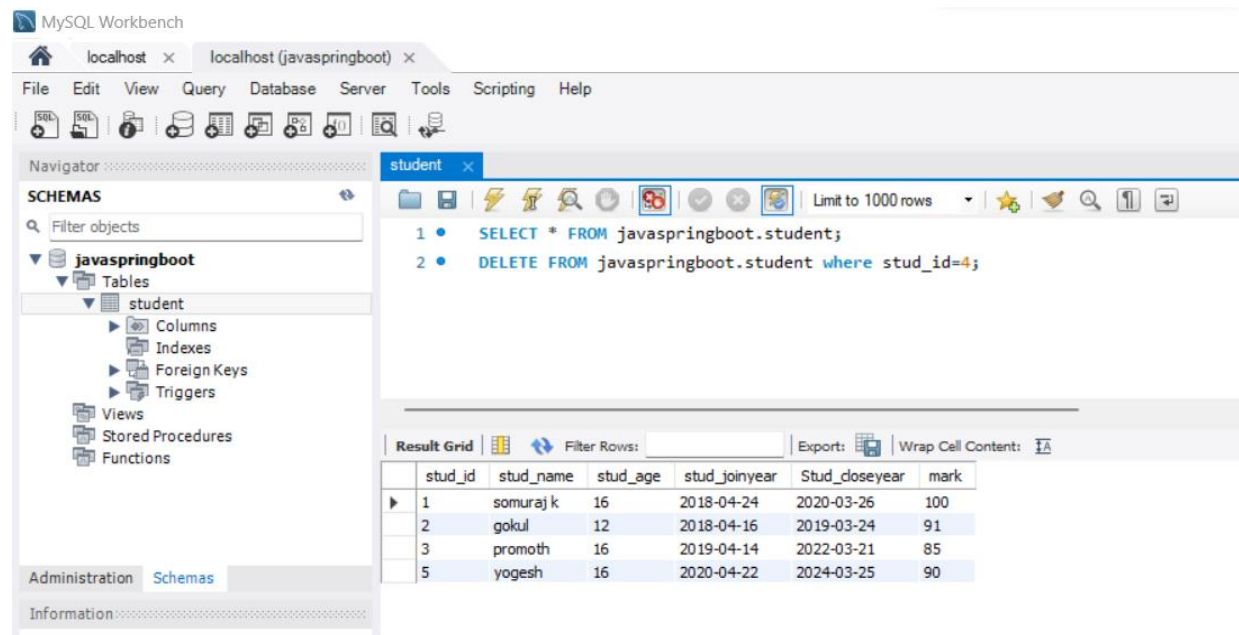
The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'javaspringboot' expanded, showing 'Tables' and 'student'. The main editor shows a query to create a new table 'details' with a foreign key:

```
1 • create table details(  
2     stud_id INT(3) primary key,  
3     stud_regno VARCHAR(20),  
4     stud_bloodgrp VARCHAR(20),  
5     foreign key(stud_id) references javaspringboot.student(Stud_id)  
6 );  
7  
8 • desc details;
```

The 'Result Grid' displays the table structure for 'details':

Field	Type	Null	Key	Default	Extra
stud_id	int	NO	PRI	NULL	
stud_regno	varchar(20)	YES		NULL	
stud_bloodgrp	varchar(20)	YES		NULL	

## 7.DELETE



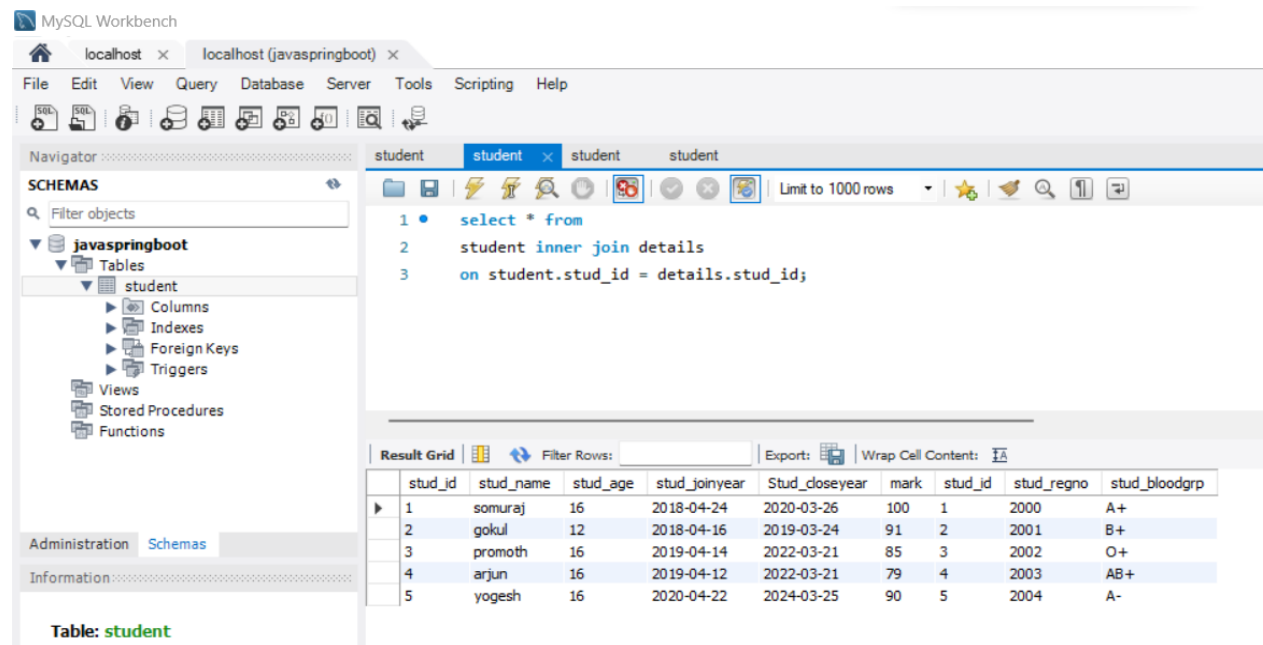
The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'javaspringboot' selected, containing a table named 'student'. The main query editor shows two SQL statements:

```
1 • SELECT * FROM javaspringboot.student;  
2 • DELETE FROM javaspringboot.student where stud_id=4;
```

The 'Result Grid' below the query editor displays the data after the deletion:

	stud_id	stud_name	stud_age	stud_joinyear	Stud_closeyear	mark
▶	1	somuraj k	16	2018-04-24	2020-03-26	100
	2	gokul	12	2018-04-16	2019-03-24	91
	3	promoth	16	2019-04-14	2022-03-21	85
	5	yogesh	16	2020-04-22	2024-03-25	90

## 8.INNER JOIN



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'javaspringboot' selected, containing a table named 'student'. The main query editor shows three SQL statements:

```
1 • select * from  
2 student inner join details  
3 on student.stud_id = details.stud_id;
```

The 'Result Grid' below the query editor displays the data after the inner join:

	stud_id	stud_name	stud_age	stud_joinyear	Stud_closeyear	mark	stud_id	stud_regno	stud_bloodgrp
▶	1	somuraj	16	2018-04-24	2020-03-26	100	1	2000	A+
	2	gokul	12	2018-04-16	2019-03-24	91	2	2001	B+
	3	promoth	16	2019-04-14	2022-03-21	85	3	2002	O+
	4	arjun	16	2019-04-12	2022-03-21	79	4	2003	AB+
	5	yogesh	16	2020-04-22	2024-03-25	90	5	2004	A-

## 9.RIGHT JOIN

MySQL Workbench

localhost x localhost (javaspringboot) x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

javaspringboot

Tables

student

Columns

Indexes

Foreign Keys

Triggers

Views

Stored Procedures

Functions

Administration Schemas

Information

Table: student

student student student student

Limit to 1000 rows

```

1 • select * from
2   student right join details
3   on student.stud_id = details.stud_id;

```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	stud_id	stud_name	stud_age	stud_joinyear	Stud_closeyear	mark	stud_id	stud_regno	stud_bloodgrp
▶	1	somuraj	16	2018-04-24	2020-03-26	100	1	2000	A+
	2	gokul	12	2018-04-16	2019-03-24	91	2	2001	B+
	3	promoth	16	2019-04-14	2022-03-21	85	3	2002	O+
	4	arjun	16	2019-04-12	2022-03-21	79	4	2003	AB+
	5	yogesh	16	2020-04-22	2024-03-25	90	5	2004	A-

## 10.LEFT JOIN

MySQL Workbench

localhost x localhost (javaspringboot) x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

javaspringboot

Tables

student

Columns

Indexes

Foreign Keys

Triggers

Views

Stored Procedures

Functions

Administration Schemas

Information

Table: student

student student student student

Limit to 1000 rows

```

1 • select * from
2   student LEFT join details
3   on student.stud_id = details.stud_id;

```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	stud_id	stud_name	stud_age	stud_joinyear	Stud_closeyear	mark	stud_id	stud_regno	stud_bloodgrp
▶	1	somuraj	16	2018-04-24	2020-03-26	100	1	2000	A+
	2	gokul	12	2018-04-16	2019-03-24	91	2	2001	B+
	3	promoth	16	2019-04-14	2022-03-21	85	3	2002	O+
	4	arjun	16	2019-04-12	2022-03-21	79	4	2003	AB+
	5	yogesh	16	2020-04-22	2024-03-25	90	5	2004	A-

## MONGODB QUERIES



# 1.CREATION AND INSERTION FIELD

The screenshot shows the MongoDB Compass interface for a local instance at localhost:27017. The 'Databases' tab is active, displaying a table for the 'admin' database with columns for 'Storage size' (430 kB), 'Collections' (1), and 'Indexes' (1). Below this, the 'My Queries' tab is selected, showing a list of queries in the 'Student' collection. The queries are as follows:

```
> MONGODB
> db.Student.insertOne({name:"Somuraj k",Reg_no:"20MIS0043"})
< {
  acknowledged: true,
  insertedId: ObjectId("647385dcbda83f6282e31ad")
}
> db.Student.insertOne({name:"gokul",Reg_no:"20BSC0034"})
< {
  acknowledged: true,
  insertedId: ObjectId("6473861bdda83f6282e31ae")
}
> db.Student.insertOne({name:"promoth",Reg_no:"20MIC0012"})
< {
  acknowledged: true,
  insertedId: ObjectId("64738634bda83f6282e31af")
}
> db.Student.insertOne({name:"arjun",Reg_no:"20MIS0013"})
< {
  acknowledged: true,
  insertedId: ObjectId("64738646bda83f6282e31b0")
}
> db.Student.insertOne({name:"yogesh",Reg_no:"20MIS0012"})
< {
  acknowledged: true,
  insertedId: ObjectId("64738662bda83f6282e31b1")
}
Enterprise JAVASPRINGBOOT>
```

The Windows taskbar at the bottom shows the date and time as 10:22 PM on 5/28/2023.

# 2.SELECTION FIELD

The screenshot shows the MongoDB Compass interface for a local instance at localhost:27017. The 'Documents' tab is active, displaying a table for the 'JAVASPRINGBOOT.Student' collection with columns for 'Documents' (5) and 'Indexes' (1). Below this, the 'My Queries' tab is selected, showing a list of queries in the 'Student' collection. The queries are as follows:

```
> MONGODB
> db.student.insertOne({name:"Somuraj k",Reg_no:"20MIS0043"})
< {
  acknowledged: true,
  insertedId: ObjectId("647388c6065a4525773f6c25")
}
> db.student.find({name:"Somuraj k"})
< {
  _id: ObjectId("647388c6065a4525773f6c25"),
  name: 'Somuraj k',
  Reg_no: '20MIS0043'
}
Enterprise test>
```

The Windows taskbar at the bottom shows the date and time as 10:22 PM on 5/28/2023.

### 3.UPDATE FIELD

The screenshot displays the MongoDB Compass interface for the 'JAVASPRINGBOOT.Student' collection. The left sidebar shows the database structure with 'Student' selected. The main panel shows the 'Documents' tab with a filter bar and a table of documents. Below the table, a MONGODB terminal window is open, showing the following commands and results:

```
> db.student.updateOne({'_id': ObjectId('647388c6065a4525773f6c25')}, {$set: {name: 'Arun R'}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.student.findOne((name: 'Arun R'))
< {
  _id: ObjectId('647388c6065a4525773f6c25'),
  name: 'Arun R',
  Reg_no: '20MIS0043'
}
```

### 4.DELETE FIELD

The screenshot displays the MongoDB Compass interface for the 'JAVASPRINGBOOT.Student' collection. The left sidebar shows the database structure with 'Student' selected. The main panel shows the 'Documents' tab with a filter bar and a table of documents. Below the table, a MONGODB terminal window is open, showing the following commands and results:

```
> db.student.deleteOne((name: 'arjun'))
< {
  acknowledged: true,
  deletedCount: 0
}
> db.student.find()
< {
  _id: ObjectId('647388c6065a4525773f6c25'),
  name: 'Arun R',
  Reg_no: '20MIS0043'
}
```