

SMARTBRIDGE EXTERNSHIP

Modern Application Development Java Spring Boot

Assignment 2

Database Management System Queries

NAME: ARAVINDKRISHNA R

REG. No.: 20BCE0074

DATE: 28-05-2023

1. create, update, delete commands in mysql

Create

Creating table and inserting values,

The screenshot shows a MySQL IDE with a SQL editor containing the following code:

```
1 CREATE TABLE Student (  
2   id INT PRIMARY KEY NOT NULL,  
3   name VARCHAR(50) NOT NULL,  
4   age INT  
5 );  
6
```

Below the editor is an 'Output' pane with a dropdown menu set to 'Action Output'. It displays the execution of the CREATE TABLE command:

#	Time	Action	Message
1	15:32:45	CREATE TABLE Student (id INT PRIMARY KEY NOT NULL, name VARCHAR(50) NOT NULL, age INT)	0 row(s) affected

The screenshot shows the same MySQL IDE with the following SQL code in the editor:

```
1 INSERT INTO Student (id, name, age)  
2 VALUES  
3 (1, 'Aravind', 20),  
4 (2, 'Krishna', 22),  
5 (3, 'Raj', 19);  
6 select * from javaspring.Student  
7
```

Below the editor is a 'Result Grid' showing the data inserted into the 'Student' table:

	id	name	age
1	1	Aravind	20
2	2	Krishna	22
3	3	Raj	19
*	NULL	NULL	NULL

Below the result grid is an 'Output' pane with a dropdown menu set to 'Action Output'. It displays the execution of the INSERT INTO and SELECT commands:

#	Time	Action	Message
1	15:32:45	CREATE TABLE Student (id INT PRIMARY KEY NOT NULL, name VARCHAR(50) NOT NULL, age INT)	0 row(s) affected
2	15:41:56	INSERT INTO Student (id, name, age) VALUES (1, 'Aravind', 20), (2, 'Krishna', 22), (3, 'Raj', 19)	3 row(s) affected Records
3	15:42:53	select * from javaspring.Student LIMIT 0, 1000	3 row(s) returned

Update

Updating the values

```
1 • UPDATE javaspring.Student
2   SET age = 21
3   WHERE id = 2;
4
5 • Select * from javaspring.Student
```

Automatic context help is disabled for the current caret position

id	name	age
1	Aravind	20
2	Krishna	21
3	Raj	19
NULL	NULL	NULL

Student 2

Apply Revert Context Help Snippets

Output

Action Output

#	Time	Action	Message
✓ 1	15:32:45	CREATE TABLE Student (id INT PRIMARY KEY NOT NULL, name VARCHAR(50) NOT NULL, age INT)	0 row(s) affected
✓ 2	15:41:56	INSERT INTO Student (id, name, age) VALUES (1, 'Aravind', 20), (2, 'Krishna', 22), (3, 'Raj', 19)	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0
✓ 3	15:42:53	select * from javaspring.Student LIMIT 0, 1000	3 row(s) returned
✓ 4	15:46:55	UPDATE javaspring.Student SET age = 21 WHERE id = 2	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
✓ 5	15:48:51	Select * from javaspring.Student LIMIT 0, 1000	3 row(s) returned

Delete

Deleting the values and the table

```
1 • DELETE FROM Student
2   WHERE id = 3;
3
4 • Select * from javaspring.Student
```

Automatic context help is disabled for the current caret position

id	name	age
1	Aravind	20
2	Krishna	21
NULL	NULL	NULL

Student 3

Apply Revert Context Help Snippets

Output

Action Output

#	Time	Action	Message
✓ 2	15:41:56	INSERT INTO Student (id, name, age) VALUES (1, 'Aravind', 20), (2, 'Krishna', 22), (3, 'Raj', 19)	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0
✓ 3	15:42:53	select * from javaspring.Student LIMIT 0, 1000	3 row(s) returned
✓ 4	15:46:55	UPDATE javaspring.Student SET age = 21 WHERE id = 2	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
✓ 5	15:48:51	Select * from javaspring.Student LIMIT 0, 1000	3 row(s) returned
✓ 6	15:52:20	DELETE FROM Student WHERE id = 3	1 row(s) affected
✓ 7	15:52:24	Select * from javaspring.Student LIMIT 0, 1000	2 row(s) returned

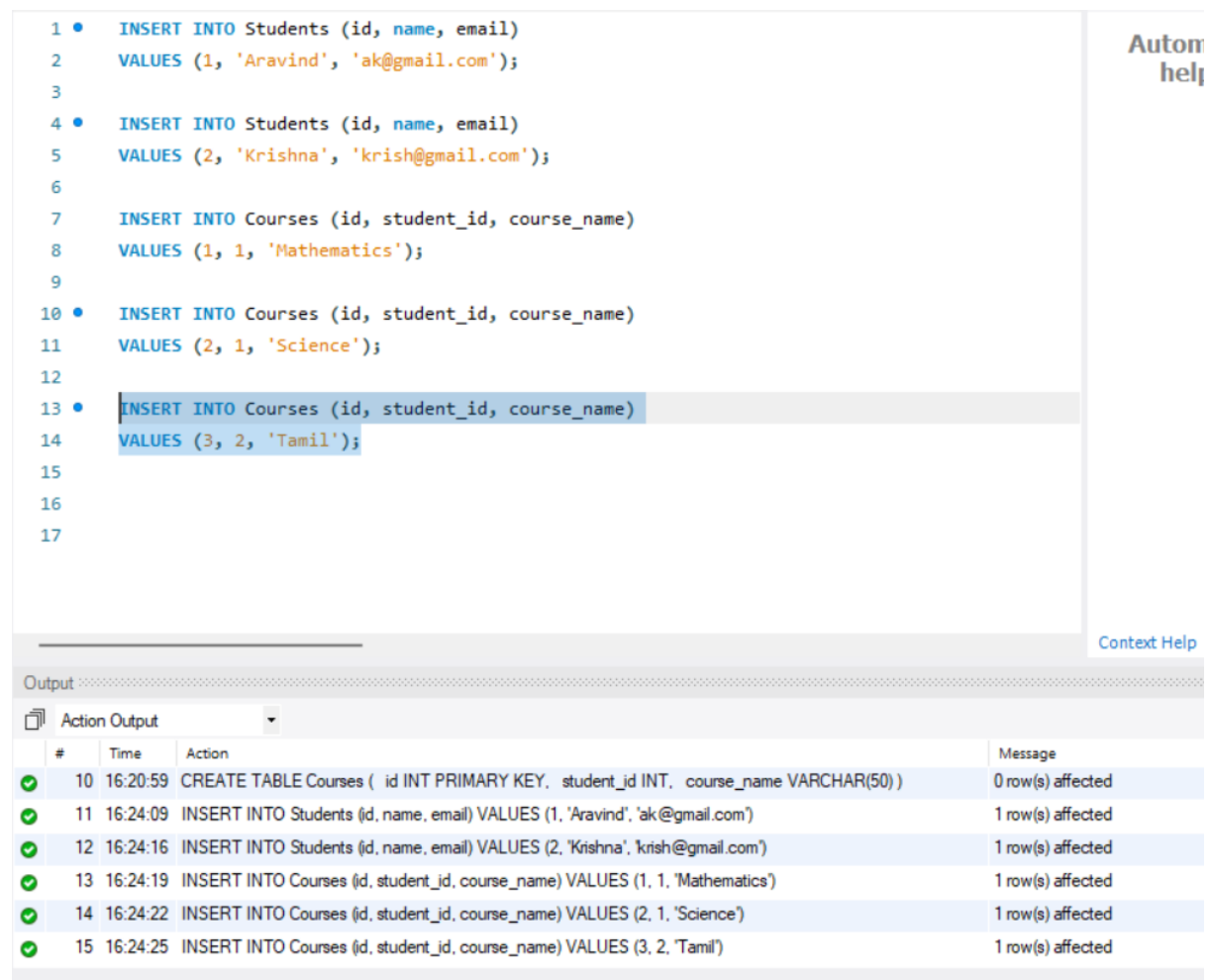
2. create tables and perform joins in mysql

2 tables created



```
1 • CREATE TABLE Students (  
2     id INT PRIMARY KEY,  
3     name VARCHAR(50),  
4     email VARCHAR(50)  
5 );  
6  
7 • CREATE TABLE Courses (  
8     id INT PRIMARY KEY,  
9     student_id INT,  
10    course_name VARCHAR(50)  
11 );  
12  
13
```

Inserting values



```
1 • INSERT INTO Students (id, name, email)  
2   VALUES (1, 'Aravind', 'ak@gmail.com');  
3  
4 • INSERT INTO Students (id, name, email)  
5   VALUES (2, 'Krishna', 'krish@gmail.com');  
6  
7   INSERT INTO Courses (id, student_id, course_name)  
8   VALUES (1, 1, 'Mathematics');  
9  
10 • INSERT INTO Courses (id, student_id, course_name)  
11   VALUES (2, 1, 'Science');  
12  
13 • INSERT INTO Courses (id, student_id, course_name)  
14   VALUES (3, 2, 'Tamil');  
15  
16  
17
```

Autom help

Context Help

Output

Action Output

#	Time	Action	Message
✓ 10	16:20:59	CREATE TABLE Courses (id INT PRIMARY KEY, student_id INT, course_name VARCHAR(50))	0 row(s) affected
✓ 11	16:24:09	INSERT INTO Students (id, name, email) VALUES (1, 'Aravind', 'ak@gmail.com')	1 row(s) affected
✓ 12	16:24:16	INSERT INTO Students (id, name, email) VALUES (2, 'Krishna', 'krish@gmail.com')	1 row(s) affected
✓ 13	16:24:19	INSERT INTO Courses (id, student_id, course_name) VALUES (1, 1, 'Mathematics')	1 row(s) affected
✓ 14	16:24:22	INSERT INTO Courses (id, student_id, course_name) VALUES (2, 1, 'Science')	1 row(s) affected
✓ 15	16:24:25	INSERT INTO Courses (id, student_id, course_name) VALUES (3, 2, 'Tamil')	1 row(s) affected

```
INSERT INTO Students (id, name, email)
VALUES (3, 'Raj', 'Raj@gmail.com');
```

Result

Students

Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:	Result Grid
	id	name	email		
▶	1	Aravind	ak@gmail.com		
	2	Krishna	krish@gmail.com		
	3	Raj	Raj@gmail.com		
*	NULL	NULL	NULL		

Courses

Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:	Result Grid
	id	student_id	course_name		
▶	1	1	Mathematics		
	2	1	Science		
	3	2	Tamil		
*	NULL	NULL	NULL		

Joins

Inner Join

SQL File 4* x

Limit to 1000 rows

1 •

SELECT *

2

FROM Students

3

INNER JOIN Courses ON Students.id=Courses.student_id;

4

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

▶

1

Aravind

ak@gmail.com

1

1

Mathematics

1

Aravind

ak@gmail.com

2

1

Science

2

Krishna

krish@gmail.com

3

2

Tamil

Result Grid

Form Editor

Left Join

The screenshot shows a SQL IDE interface. The query editor contains the following SQL code:

```
1 • SELECT *
2 FROM Students
3 Left JOIN Courses ON Students.id=Courses.student_id;
4
```

The result grid displays the following data:

	id	name	email	id	student_id	course_name
▶	1	Aravind	ak@gmail.com	2	1	Science
	1	Aravind	ak@gmail.com	1	1	Mathematics
	2	Krishna	krish@gmail.com	3	2	Tamil
	3	Raj	Raj@gmail.com	NULL	NULL	NULL

The interface includes a toolbar with icons for file operations, a 'Limit to 1000 rows' dropdown, and buttons for 'Result Grid' and 'Form Editor'.

Right Join

The screenshot shows a SQL IDE interface. The query editor contains the following SQL code:

```
1 • SELECT *
2 FROM Students
3 Right JOIN Courses ON Students.id=Courses.student_id;
4
```

The result grid displays the following data:

	id	name	email	id	student_id	course_name
▶	1	Aravind	ak@gmail.com	1	1	Mathematics
	1	Aravind	ak@gmail.com	2	1	Science
	2	Krishna	krish@gmail.com	3	2	Tamil

The interface includes a toolbar with icons for file operations, a 'Limit to 1000 rows' dropdown, and buttons for 'Result Grid' and 'Form Editor'.

Full Join

as give in git material

The screenshot shows a SQL editor window titled "SQL File 4*" with a toolbar and a "Limit to 1000 rows" dropdown. The SQL query is as follows:

```
1 SELECT *
2 FROM Students
3 Full JOIN Courses
4
```

Below the query, the "Result Grid" is displayed with the following data:

	id	name	email	id	student_id	course_name
▶	3	Raj	Raj@gmail.com	1	1	Mathematics
	2	Krishna	krish@gmail.com	1	1	Mathematics
	1	Aravind	ak@gmail.com	1	1	Mathematics
	3	Raj	Raj@gmail.com	2	1	Science
	2	Krishna	krish@gmail.com	2	1	Science
	1	Aravind	ak@gmail.com	2	1	Science
	3	Raj	Raj@gmail.com	3	2	Tamil
	2	Krishna	krish@gmail.com	3	2	Tamil
	1	Aravind	ak@gmail.com	3	2	Tamil

On the right side of the interface, there are buttons for "Result Grid", "Form Editor", and "Field Types".

another one (normal union method)

The screenshot shows a SQL editor window titled "SQL File 4*" with a toolbar and a "Limit to 1000 rows" dropdown. The SQL query is as follows:

```
1 SELECT Students.id, Students.name, Students.email, Courses.course_name
2 FROM Students
3 LEFT JOIN Courses ON Students.id = Courses.student_id
4 UNION
5 SELECT Courses.id, Students.name, Students.email, Courses.course_name
6 FROM Courses
7 RIGHT JOIN Students ON Courses.student_id = Students.id;
```

Below the query, the "Result Grid" is displayed with the following data:

	id	name	email	course_name
▶	1	Aravind	ak@gmail.com	Science
	1	Aravind	ak@gmail.com	Mathematics
	2	Krishna	krish@gmail.com	Tamil
	3	Raj	Raj@gmail.com	NULL
	2	Aravind	ak@gmail.com	Science
	3	Krishna	krish@gmail.com	Tamil
	NULL	Raj	Raj@gmail.com	NULL

3. create, update, delete commands in mongo

Create

Create Database

Database Name

School

Collection Name

Students

☐ Time-Series

Time-series collections efficiently store sequences of measurements over a period of time. [Learn More](#)

➤ Additional preferences (e.g. Custom collation, Capped, Clustered collections)

Cancel

Create Database

School database and Student collection is created,

```
> show dbs
< School      8.00 KiB
  admin      40.00 KiB
  config     72.00 KiB
  local      72.00 KiB
  sample     8.00 KiB
test> |
```

selecting that School database,

```
> use School
< switched to db School
School>
```

Creating one more collection,

```
> db.createCollection("Teachers")
< { ok: 1 }
School>
```


Inserting data into Students, we can also insert in teachers
(Creating documents) using insertOne,

```
> db.Students.insertOne({name: "Aravind", age: 20})
< {
  acknowledged: true,
  insertedId: ObjectId("6473098eaf43f055a81c9f67")
}
> db.Students.find()
< {
  _id: ObjectId("6473098eaf43f055a81c9f67"),
  name: 'Aravind',
  age: 20
}
School > |
```

we can also use insertMany

```
> db.Students.insertMany([{name: "Krishna", age: 21},{name: "Raj", age: 25}])
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("64730bacaf43f055a81c9f68"),
    '1': ObjectId("64730bacaf43f055a81c9f69")
  }
}
> db.Students.find()
< {
  _id: ObjectId("6473098eaf43f055a81c9f67"),
  name: 'Aravind',
  age: 20
}
{
  _id: ObjectId("64730bacaf43f055a81c9f68"),
  name: 'Krishna',
  age: 21
}
{
  _id: ObjectId("64730bacaf43f055a81c9f69"),
  name: 'Raj',
  age: 25
}
School > |
```

Update - Updating the values using updateOne,

```
> db.Students.updateOne({name:"Aravind"}, {$set: {age: 19}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
> db.Students.find()
< {
  _id: ObjectId("6473098eaf43f055a81c9f67"),
  name: 'Aravind',
  age: 19
}
{
  _id: ObjectId("64730bacaf43f055a81c9f68"),
  name: 'Krishna',
  age: 21
}
{
  _id: ObjectId("64730bacaf43f055a81c9f69"),
  name: 'Raj',
  age: 25
}
School >
```

Updating the values using updateMany,

```
> db.Students.updateMany({age: {$gt:20}}, {$set: {age: 26}})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
> db.Students.find()
< {
  _id: ObjectId("6473098eaf43f055a81c9f67"),
  name: 'Aravind',
  age: 19
}
{
  _id: ObjectId("64730bacaf43f055a81c9f68"),
  name: 'Krishna',
  age: 26
}
{
  _id: ObjectId("64730bacaf43f055a81c9f69"),
  name: 'Raj',
  age: 26
}
School > |
```

Delete

deleting the values using deleteOne,

```
> db.Students.deleteOne({name: "Krishna"})
< {
  acknowledged: true,
  deletedCount: 1
}
> db.Students.find()
< {
  _id: ObjectId("6473098eaf43f055a81c9f67"),
  name: 'Aravind',
  age: 19
}
{
  _id: ObjectId("64730bacaf43f055a81c9f69"),
  name: 'Raj',
  age: 26
}
School >
```

deleting the values using deleteMany,

```
> db.Students.deleteMany({age: {$lt: 30}})
< {
  acknowledged: true,
  deletedCount: 2
}
> db.Students.find()
<
School > |
```

deleting the collection and the database,

```
> db.Students.drop()
< true
```

```
> db.Students.drop()
< true
> db.dropDatabase()
< { ok: 1, dropped: 'School' }
```