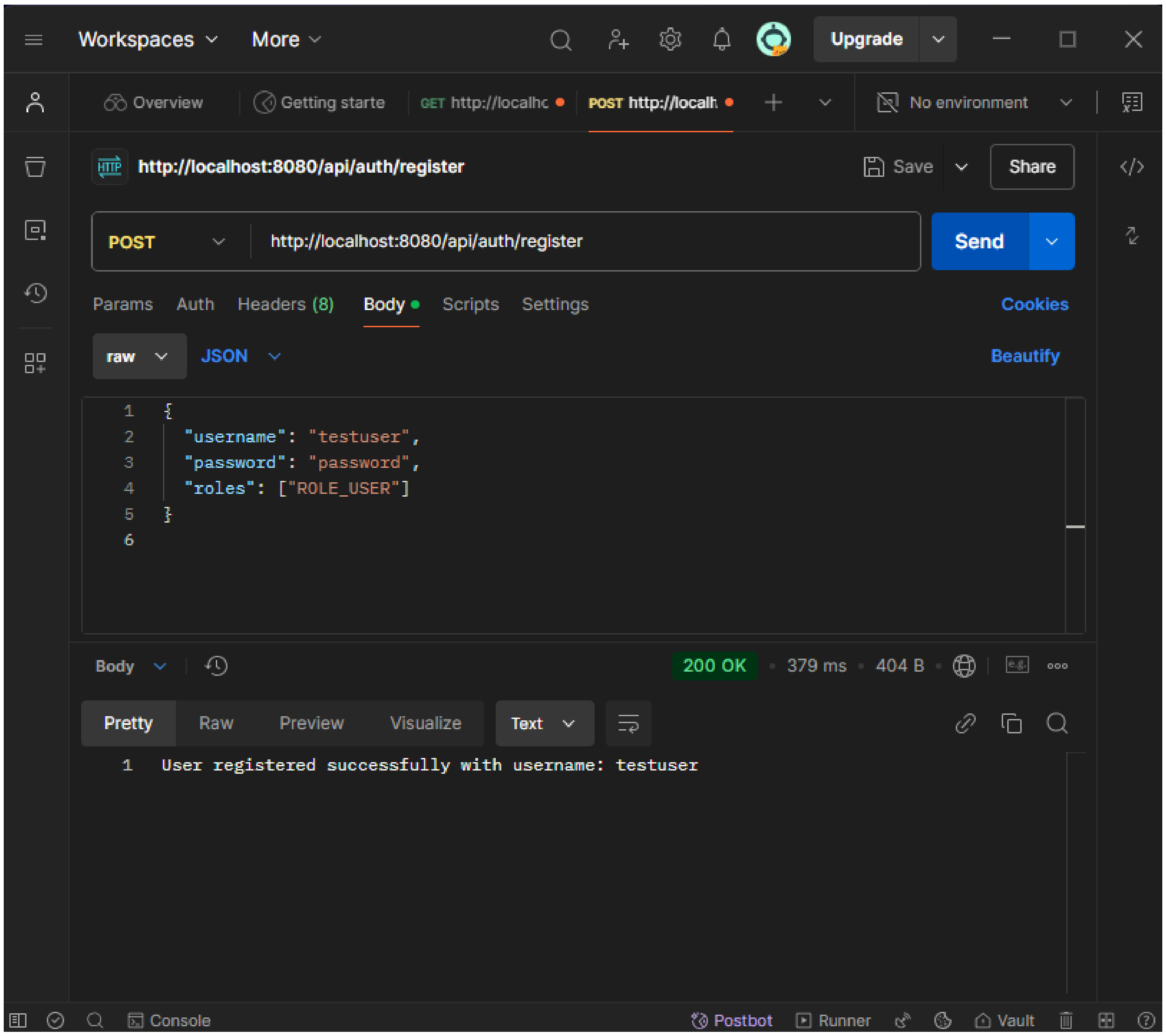
VRV Security's Backend Developer Intern Assignment

User Registration (POST /api/auth/register):

- The user was successfully created in the database.
- The response was positive confirming the registration.



POST http://localhost:8080/api/auth/register

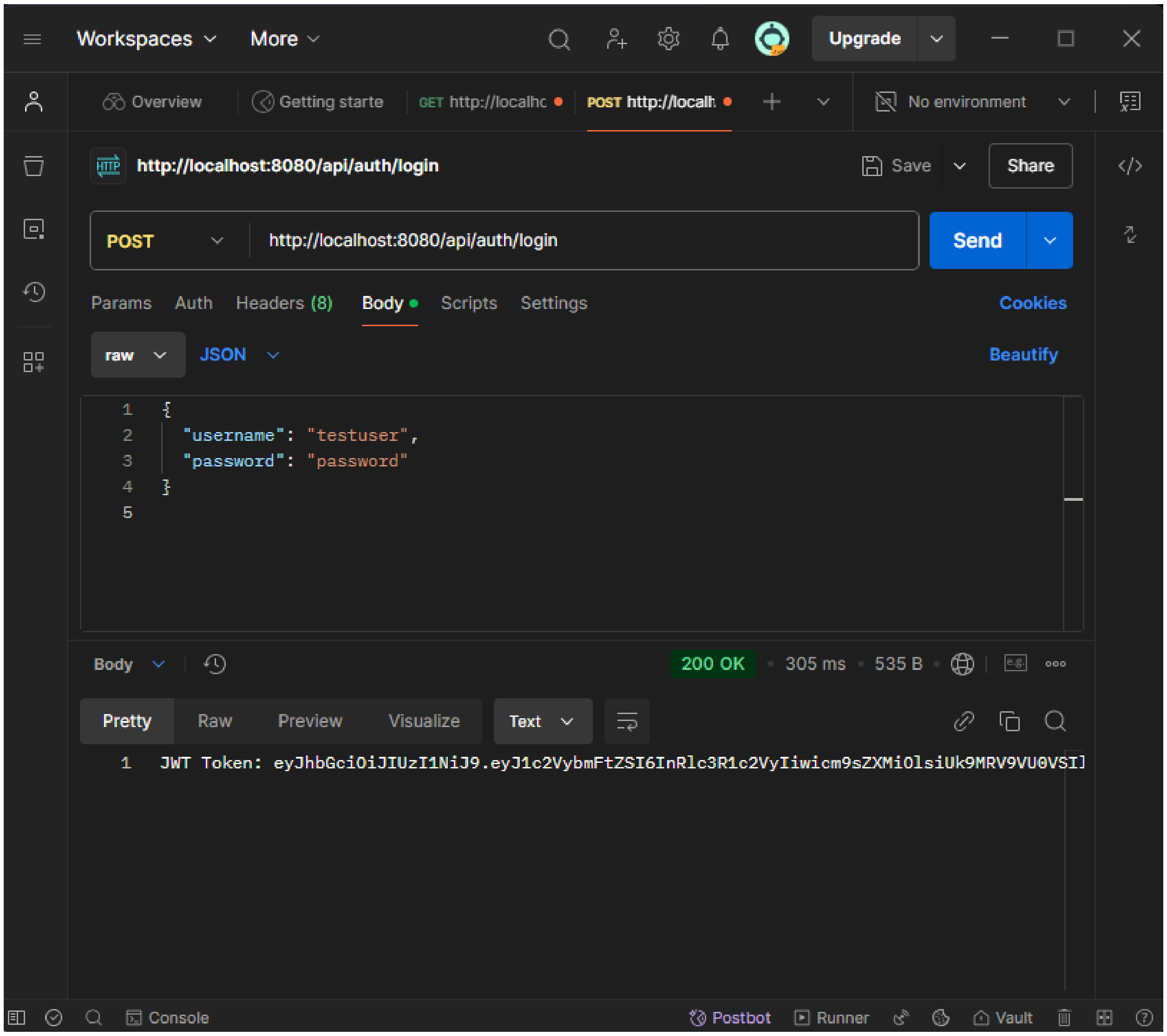
```
{
  "username": "testuser",
  "password": "password",
  "roles": ["ROLE_USER"]
}
```

200 OK · 379 ms · 404 B

```
User registered successfully with username: testuser
```

User Login (POST /api/auth/login):

- You received a valid JWT token in response.



http://localhost:8080/api/auth/login

POST http://localhost:8080/api/auth/login

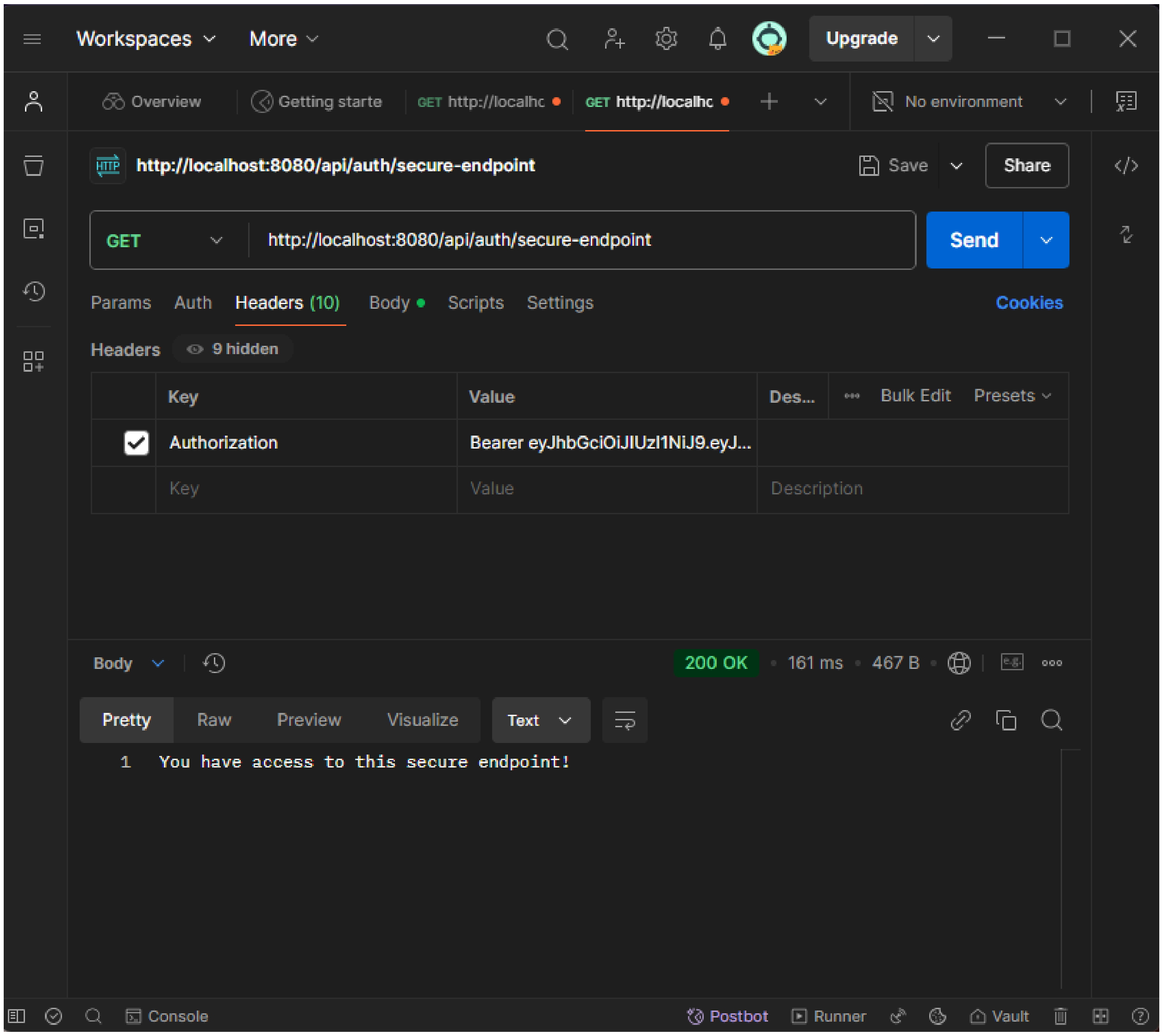
```json
{
  "username": "testuser",
  "password": "password"
}
```

200 OK • 305 ms • 535 B

```
JWT Token: eyJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6InRlc3R1c2VyIiwicm9sZXMiOlsiUk9MRV9VU0VSIl
```

Access Secure Endpoint (GET /api/auth/secure-endpoint):

- The secure endpoint was successfully accessed using the token with the correct Authorization header.



http://localhost:8080/api/auth/secure-endpoint

GET http://localhost:8080/api/auth/secure-endpoint

Headers

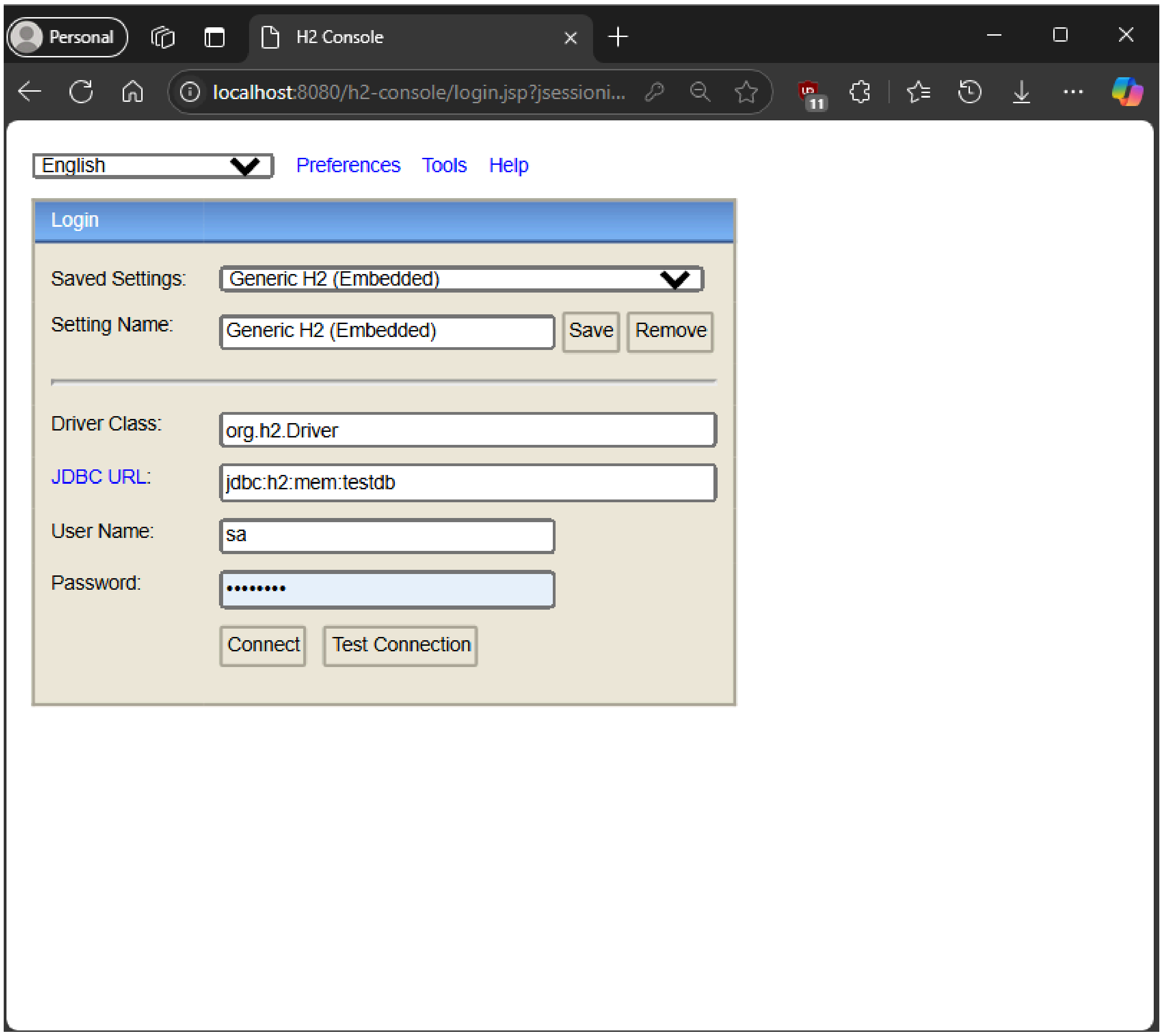
	Key	Value	Des...
✓	Authorization	Bearer eyJhbGciOiJIUzI1NiJ9.eyJ...	
	Key	Value	Description

200 OK • 161 ms • 467 B

```
You have access to this secure endpoint!
```

H2 Database:

- The user and roles are correctly stored in the database.
- You are able to view data from USER and user_roles tables.



Saved Settings: Generic H2 (Embedded)

Setting Name: Generic H2 (Embedded)

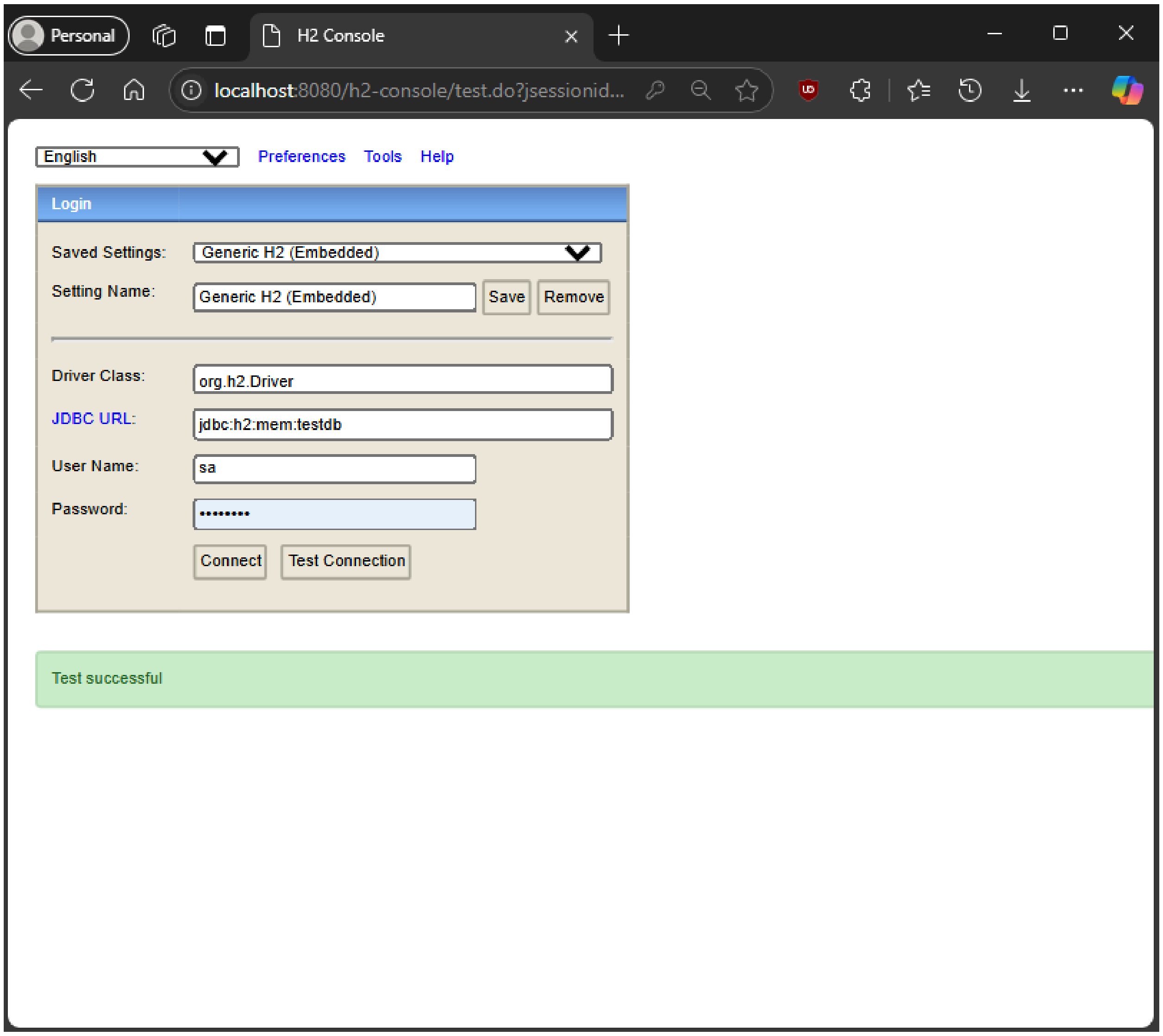
Driver Class: org.h2.Driver

JDBC URL: jdbc:h2:mem:testdb

User Name: sa

Password: ••••••••

Test connection :test successful

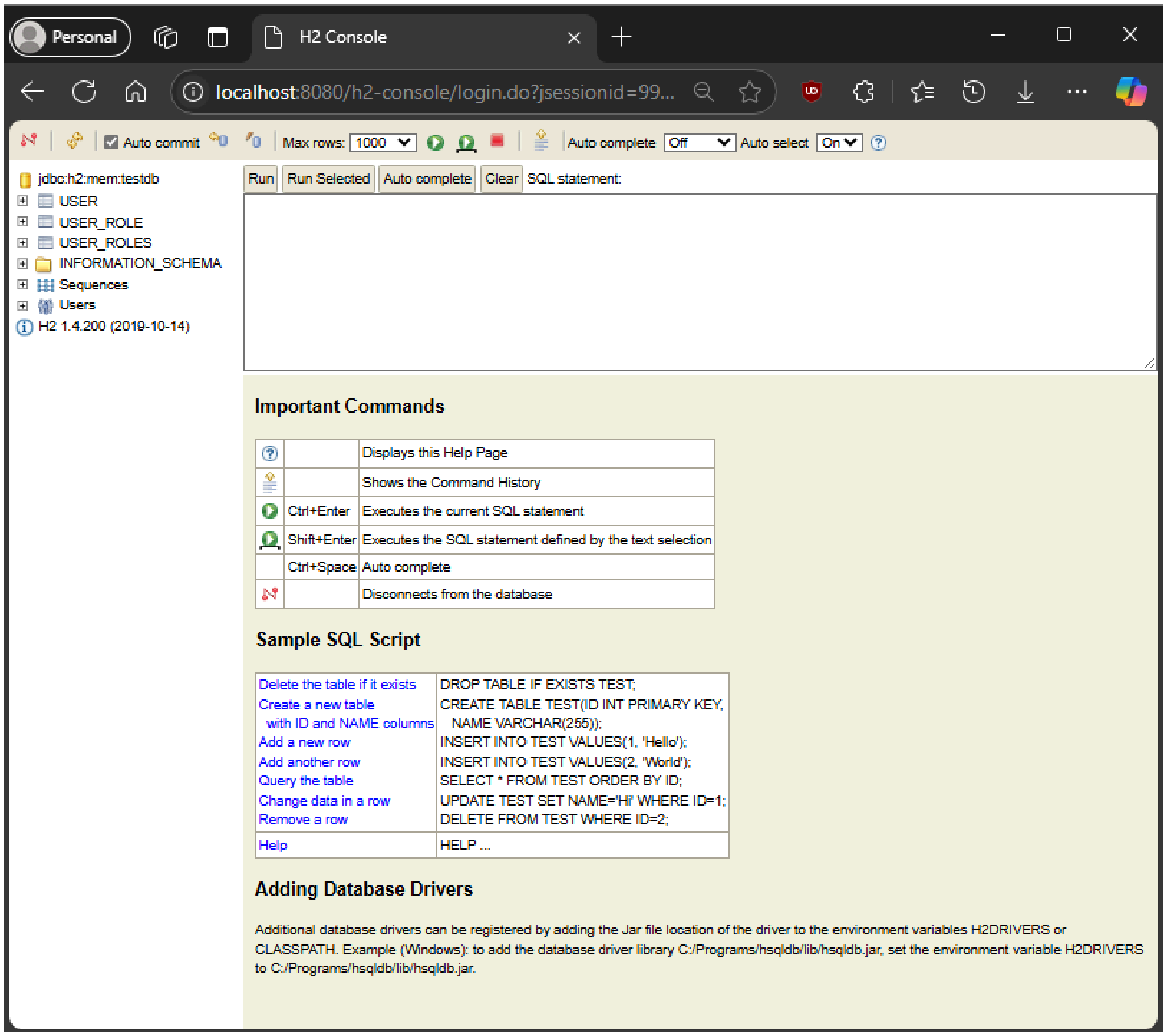


Saved Settings: Generic H2 (Embedded)
Setting Name: Generic H2 (Embedded)

Driver Class: org.h2.Driver
JDBC URL: jdbc:h2:mem:testdb
User Name: sa
Password: ••••••••

Test successful

After clicking connect we can now see a database of h2 console



Important Commands

		Displays this Help Page
		Shows the Command History
	Ctrl+Enter	Executes the current SQL statement
	Shift+Enter	Executes the SQL statement defined by the text selection
	Ctrl+Space	Auto complete
		Disconnects from the database

Sample SQL Script

Delete the table if it exists
Create a new table
with ID and NAME columns
Add a new row
Add another row
Query the table
Change data in a row
Remove a row
	DROP TABLE IF EXISTS TEST;
	CREATE TABLE TEST(ID INT PRIMARY KEY,
	NAME VARCHAR(255));
	INSERT INTO TEST VALUES(1, 'Hello');
	INSERT INTO TEST VALUES(2, 'World');
	SELECT * FROM TEST ORDER BY ID;
	UPDATE TEST SET NAME='Hi' WHERE ID=1;
	DELETE FROM TEST WHERE ID=2;

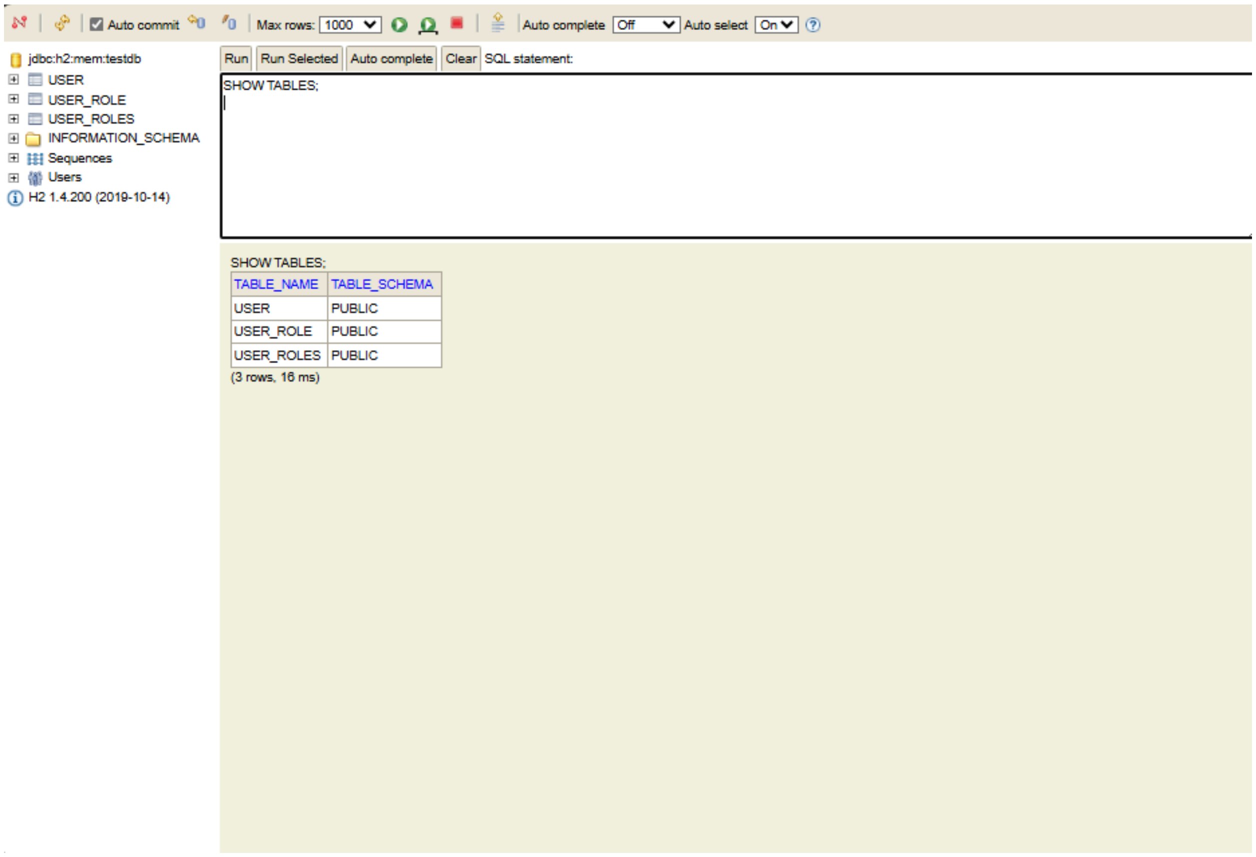
Help	HELP ...

Adding Database Drivers

Additional database drivers can be registered by adding the Jar file location of the driver to the environment variables H2DRIVERS or CLASSPATH. Example (Windows): to add the database driver library C:/Programs/hsqldb/lib/hsqldb.jar, set the environment variable H2DRIVERS to C:/Programs/hsqldb/lib/hsqldb.jar.

View Databases: Once connected, you can query your existing tables using SQL commands such as:

```
SHOW TABLES;
```



```
SHOW TABLES;
```

SHOW TABLES;

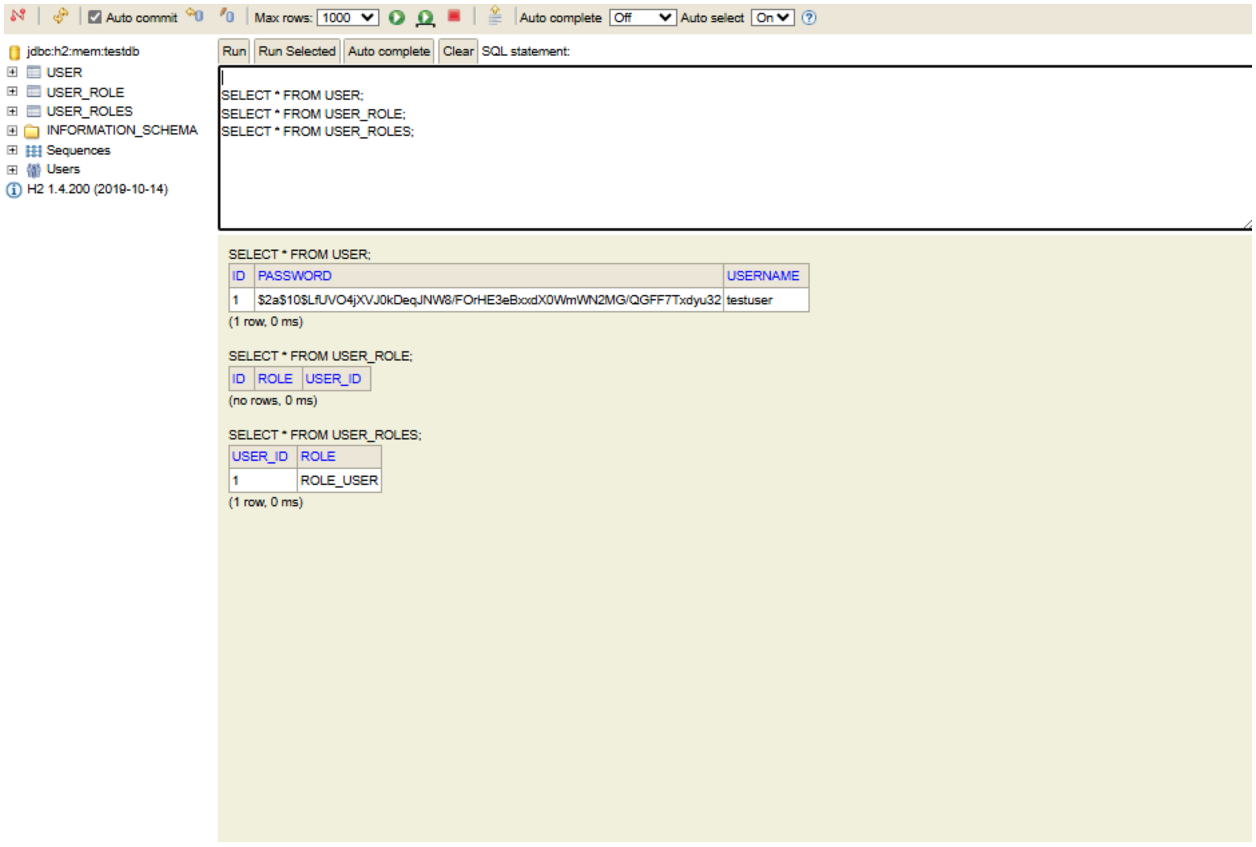
TABLE_NAME	TABLE_SCHEMA
USER	PUBLIC
USER_ROLE	PUBLIC
USER_ROLES	PUBLIC

(3 rows, 16 ms)

1. View Data in the USER, USER_ROLE and USER_ROLES Table:

To see the data stored in the tables, you can run the following SQL query:

```
SELECT * FROM USER;
SELECT * FROM USER_ROLE;
SELECT * FROM USER_ROLES;
```



```
SELECT * FROM USER;
SELECT * FROM USER_ROLE;
SELECT * FROM USER_ROLES;
```

SELECT * FROM USER;

ID	PASSWORD	USERNAME
1	$2a$10$LfUVO4jXVJ0kDeqJNW8/FOrHE3eBxxdX0WmWN2MG/QGFF7Txdyu32	testuser

(1 row, 0 ms)

SELECT * FROM USER_ROLE;

ID	ROLE	USER_ID

(no rows, 0 ms)

SELECT * FROM USER_ROLES;

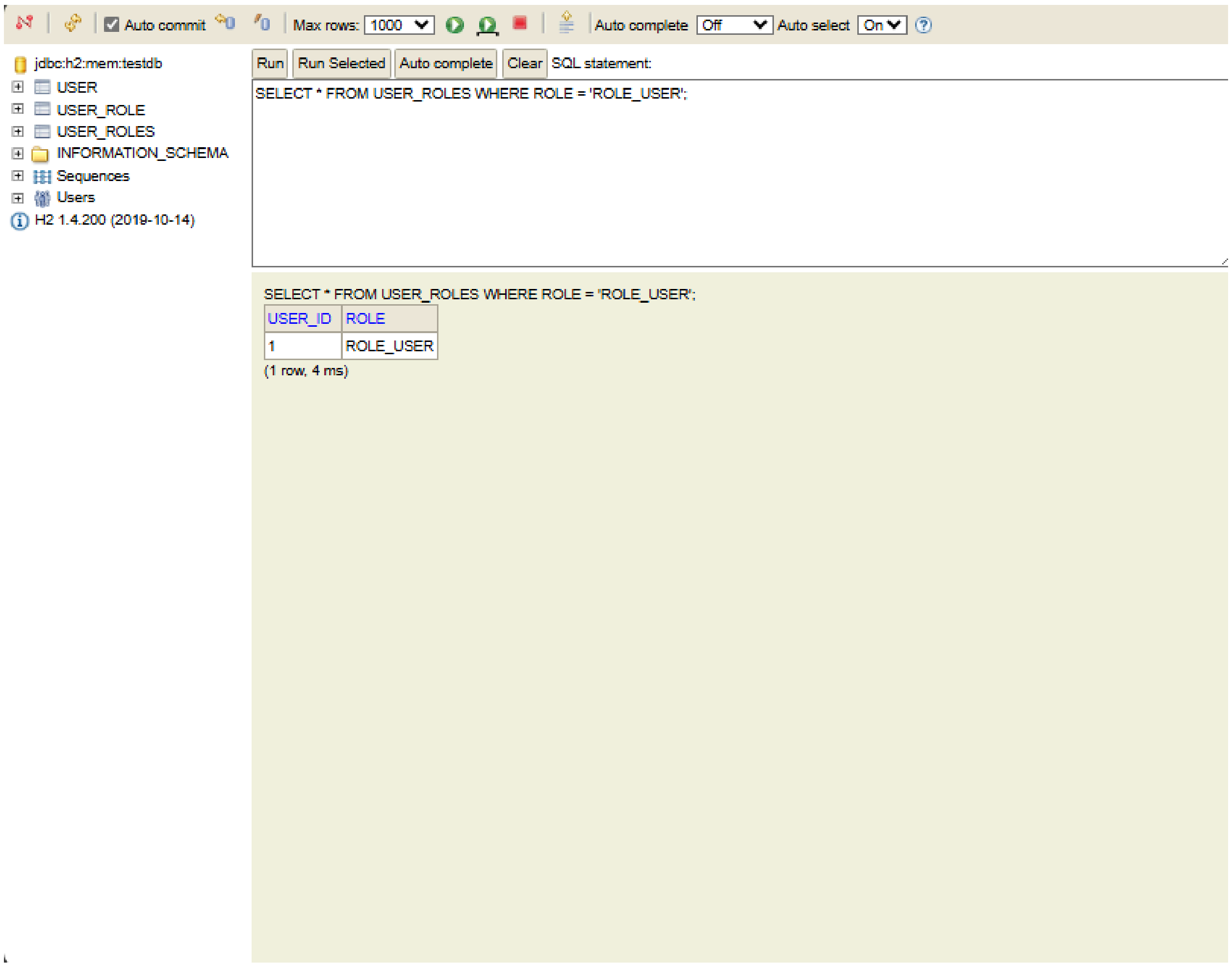
USER_ID	ROLE
1	ROLE_USER

(1 row, 0 ms)

4. Check for Specific Data (Optional):

If you'd like to see specific information, for example, the users with a particular role, you could query:

SELECT * FROM USER_ROLES WHERE ROLE = 'ROLE_USER';



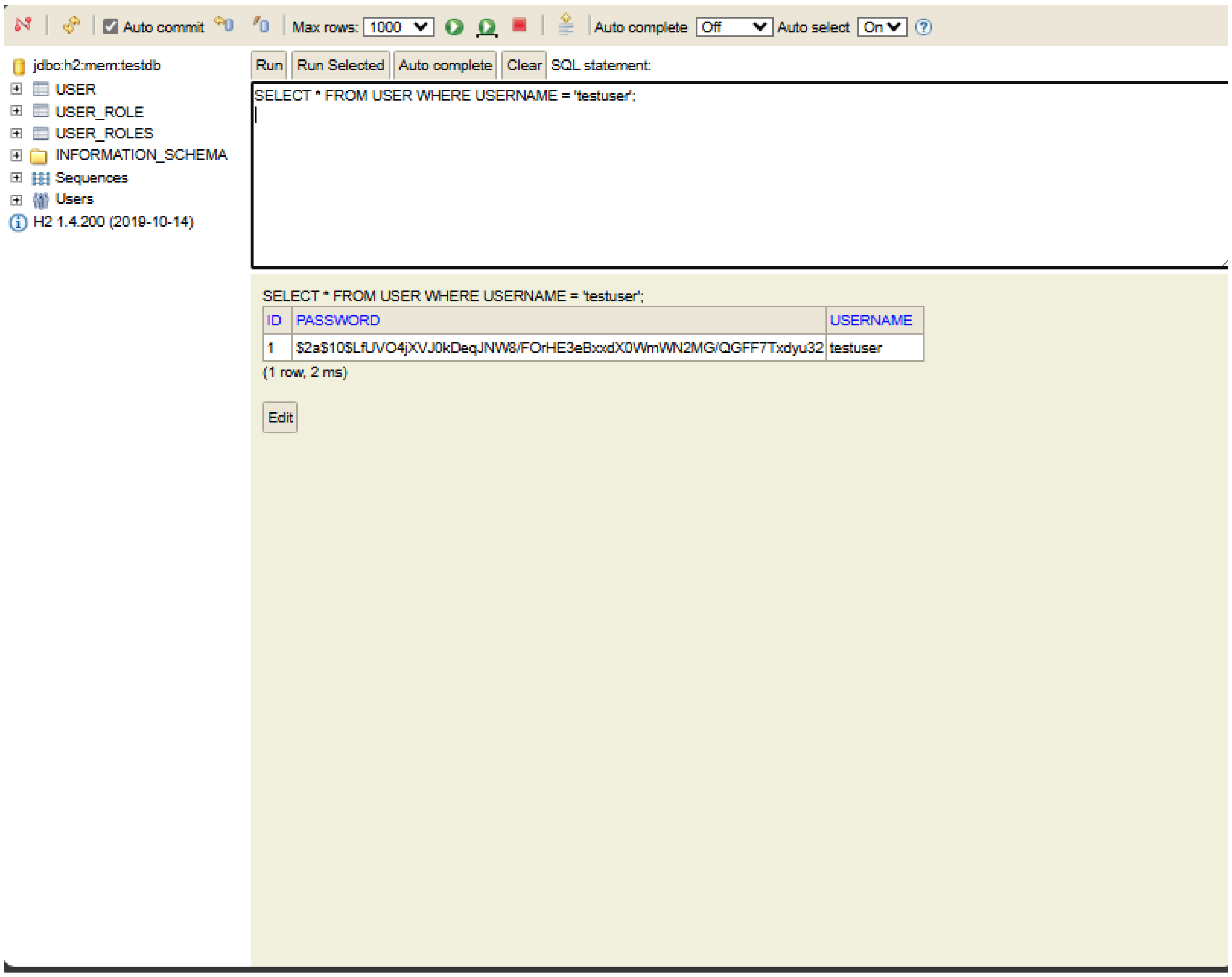
```
SELECT * FROM USER_ROLES WHERE ROLE = 'ROLE_USER';
```

SELECT * FROM USER_ROLES WHERE ROLE = 'ROLE_USER';

USER_ID	ROLE
1	ROLE_USER

(1 row, 4 ms)

Or, to check a user's details:
SELECT * FROM USER WHERE USERNAME = 'testuser';



```
SELECT * FROM USER WHERE USERNAME = 'testuser';
```

SELECT * FROM USER WHERE USERNAME = 'testuser';

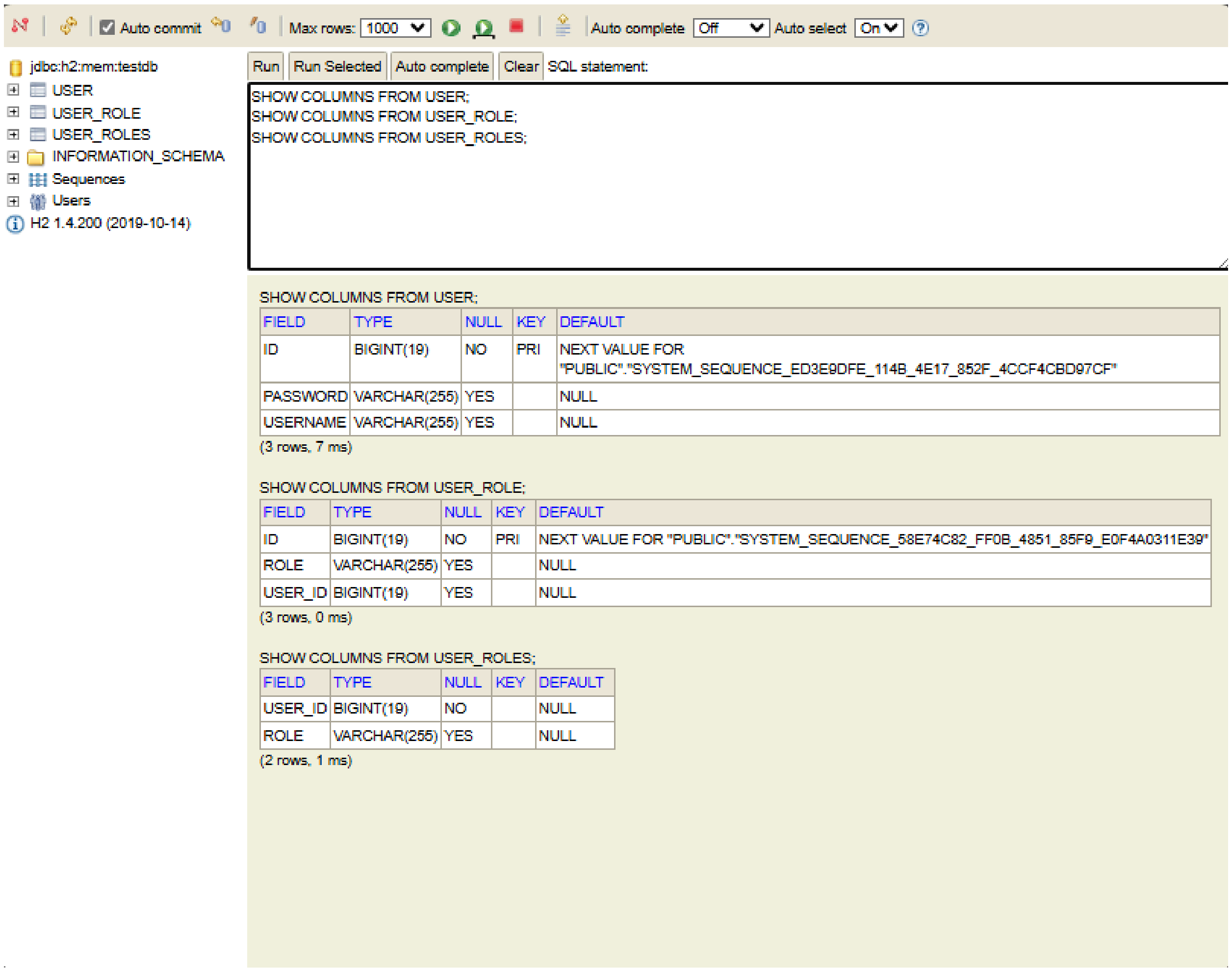
ID	PASSWORD	USERNAME
1	$2a$10$LfUVO4jXVJ0kDeqJNW8/FOrHE3eBxxdX0WmWN2MG/QGFF7Txdyu32	testuser

(1 row, 2 ms)

5. Additional Information (Optional):

If you would like to understand the table schema or the structure of the tables (columns, types, etc.), you can describe the tables:

```
SHOW COLUMNS FROM USER;
SHOW COLUMNS FROM USER_ROLE;
SHOW COLUMNS FROM USER_ROLES;
```



```
SHOW COLUMNS FROM USER;
SHOW COLUMNS FROM USER_ROLE;
SHOW COLUMNS FROM USER_ROLES;
```

SHOW COLUMNS FROM USER;

FIELD	TYPE	NULL	KEY	DEFAULT
ID	BIGINT(19)	NO	PRI	NEXT VALUE FOR "PUBLIC"."SYSTEM_SEQUENCE_ED3E9DFE_114B_4E17_852F_4CCF4CBD97CF"
PASSWORD	VARCHAR(255)	YES		NULL
USERNAME	VARCHAR(255)	YES		NULL

(3 rows, 7 ms)

SHOW COLUMNS FROM USER_ROLE;

FIELD	TYPE	NULL	KEY	DEFAULT
ID	BIGINT(19)	NO	PRI	NEXT VALUE FOR "PUBLIC"."SYSTEM_SEQUENCE_58E74C82_FF0B_4851_85F9_E0F4A0311E39"
ROLE	VARCHAR(255)	YES		NULL
USER_ID	BIGINT(19)	YES		NULL

(3 rows, 0 ms)

SHOW COLUMNS FROM USER_ROLES;

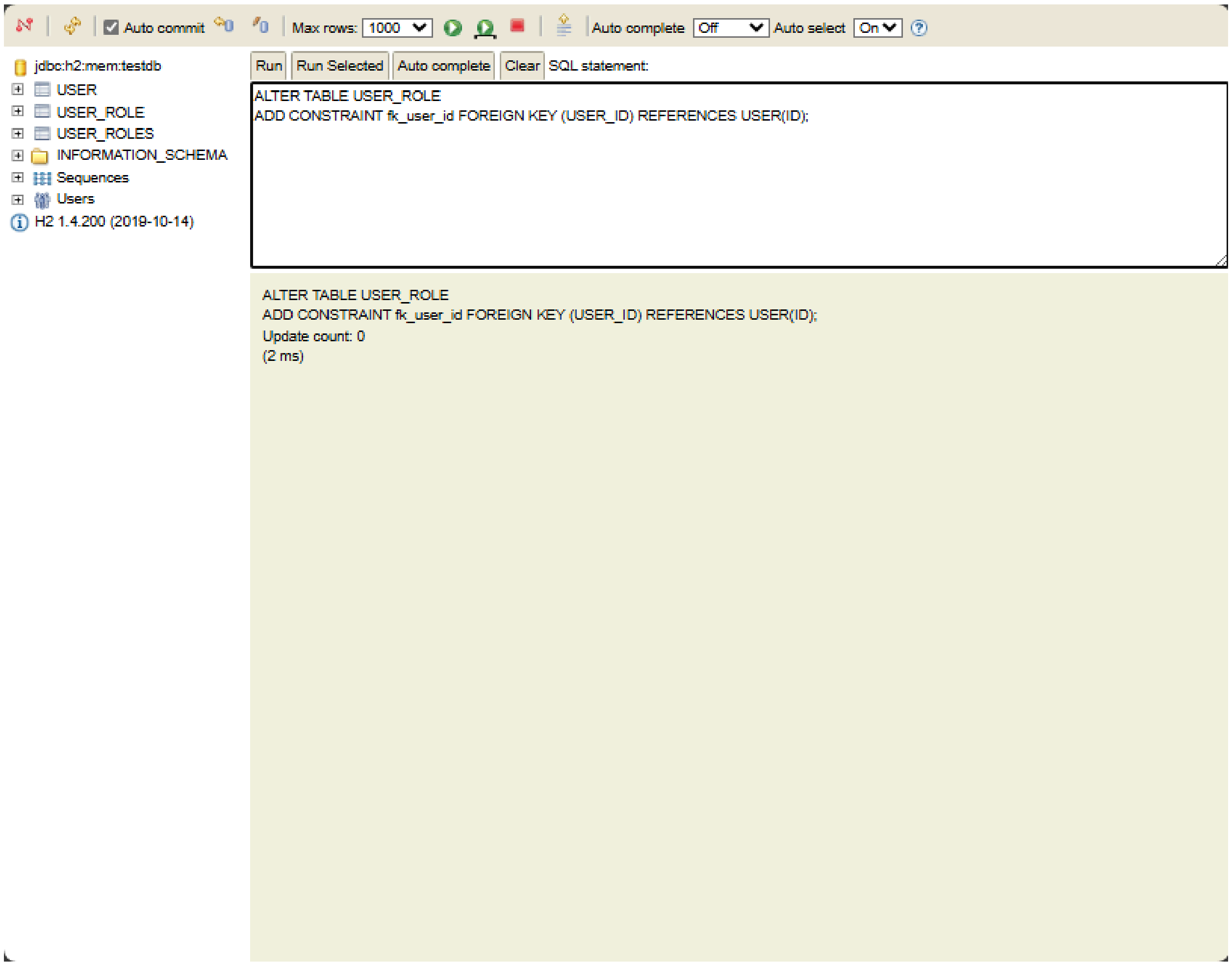
FIELD	TYPE	NULL	KEY	DEFAULT
USER_ID	BIGINT(19)	NO		NULL
ROLE	VARCHAR(255)	YES		NULL

(2 rows, 1 ms)

ou can add a foreign key constraint like this:

1. **To add a foreign key constraint for USER_ID in USER_ROLE**:

```
ALTER TABLE USER_ROLE
ADD CONSTRAINT fk_user_id FOREIGN KEY (USER_ID) REFERENCES USER(ID);
```



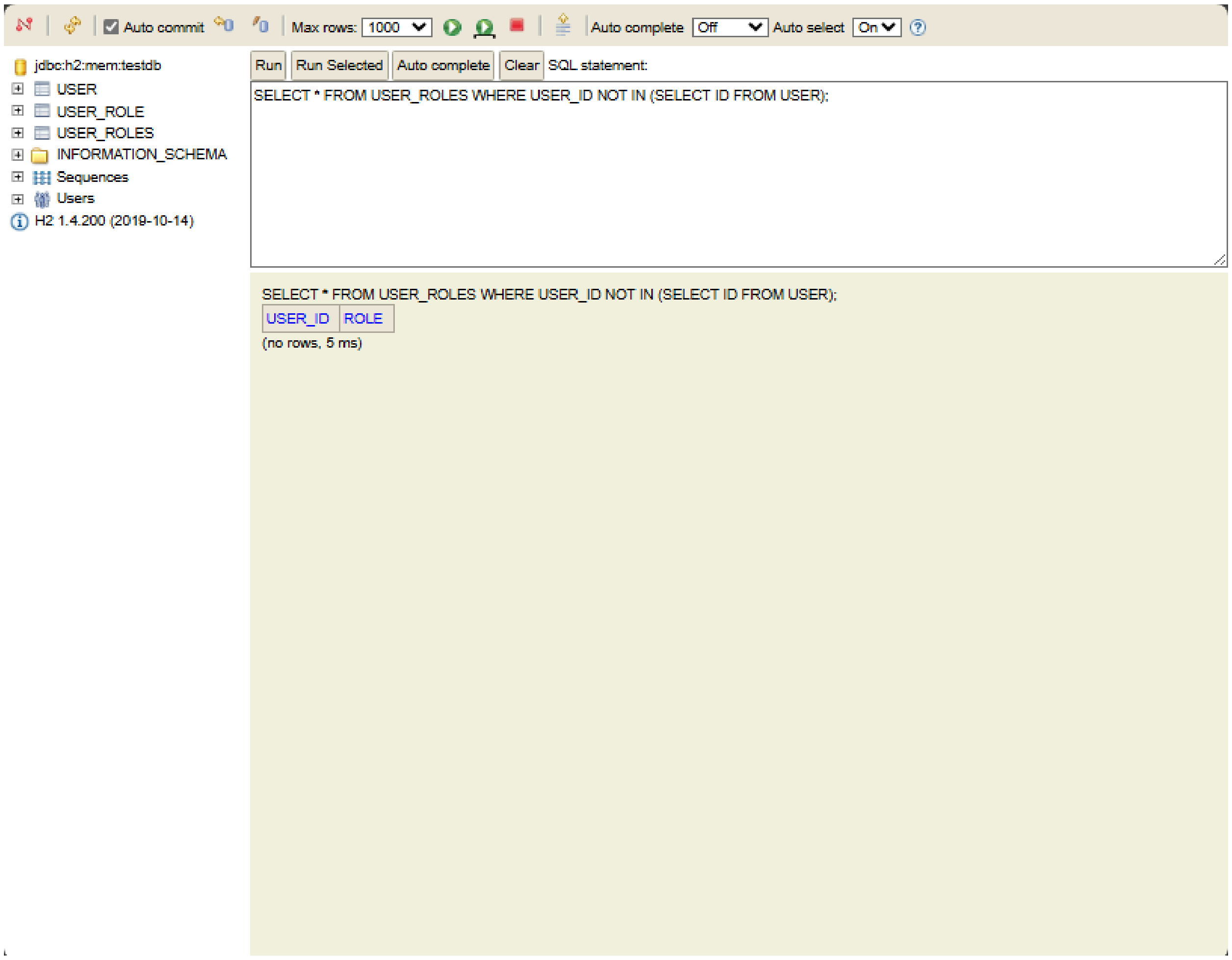
```
ALTER TABLE USER_ROLE
ADD CONSTRAINT fk_user_id FOREIGN KEY (USER_ID) REFERENCES USER(ID);
```

```
ALTER TABLE USER_ROLE
ADD CONSTRAINT fk_user_id FOREIGN KEY (USER_ID) REFERENCES USER(ID);
Update count: 0
(2 ms)
```

Check for Orphaned Data

If you have inserted data into the USER_ROLES table without corresponding data in the USER table (or vice versa), the foreign key constraint will fail. You should verify if any orphaned records exist in the USER_ROLES table:

SELECT * FROM USER_ROLES WHERE USER_ID NOT IN (SELECT ID FROM USER);

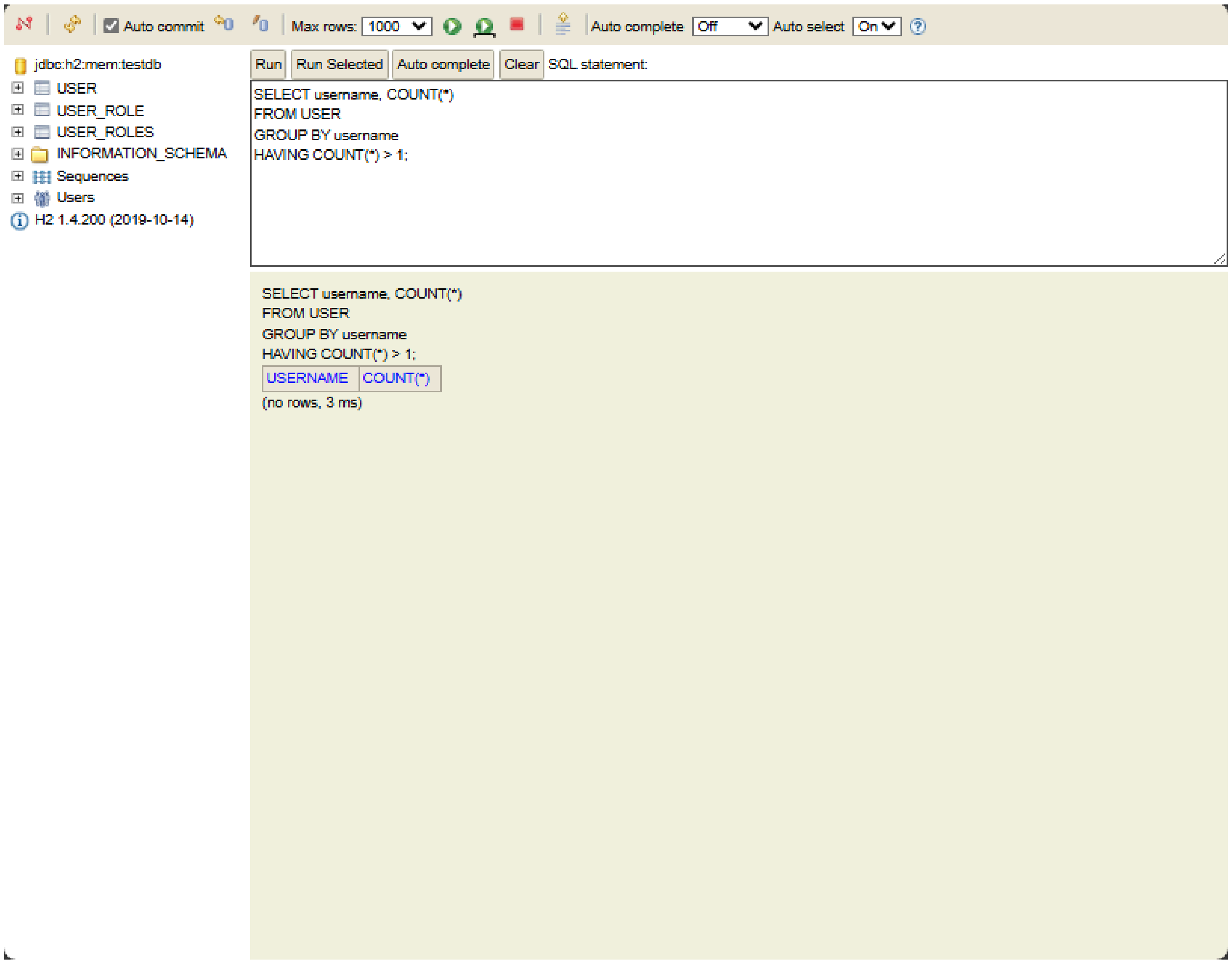


```
SELECT * FROM USER_ROLES WHERE USER_ID NOT IN (SELECT ID FROM USER);
```

```
SELECT * FROM USER_ROLES WHERE USER_ID NOT IN (SELECT ID FROM USER);
USER_ID  ROLE
(no rows, 5 ms)
```

Check for Duplicate Records: You should also ensure that no duplicate user records exist in the USER table.

SELECT username, COUNT(*)
FROM USER
GROUP BY username
HAVING COUNT(*) > 1;



```
SELECT username, COUNT(*)
FROM USER
GROUP BY username
HAVING COUNT(*) > 1;
```

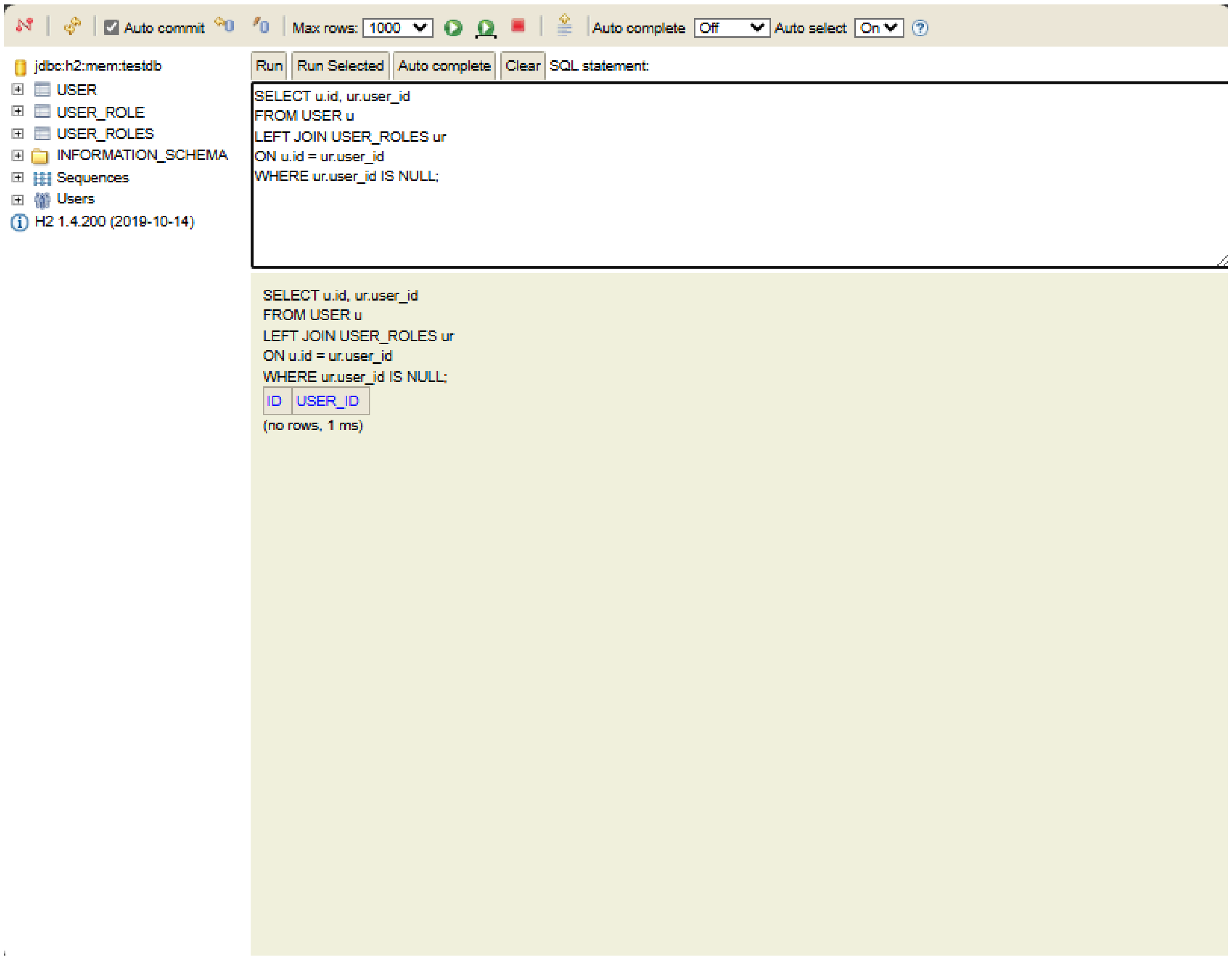
```
SELECT username, COUNT(*)
FROM USER
GROUP BY username
HAVING COUNT(*) > 1;
```

USERNAME	COUNT(*)

```
(no rows, 3 ms)
```

Validate Relationships: Check that all relationships between tables (e.g., USER and USER_ROLES) are properly set with no mismatches.

SELECT u.id, ur.user_id
FROM USER u
LEFT JOIN USER_ROLES ur
ON u.id = ur.user_id
WHERE ur.user_id IS NULL;



```
SELECT u.id, ur.user_id
FROM USER u
LEFT JOIN USER_ROLES ur
ON u.id = ur.user_id
WHERE ur.user_id IS NULL;
```

```
SELECT u.id, ur.user_id
FROM USER u
LEFT JOIN USER_ROLES ur
ON u.id = ur.user_id
WHERE ur.user_id IS NULL;
```

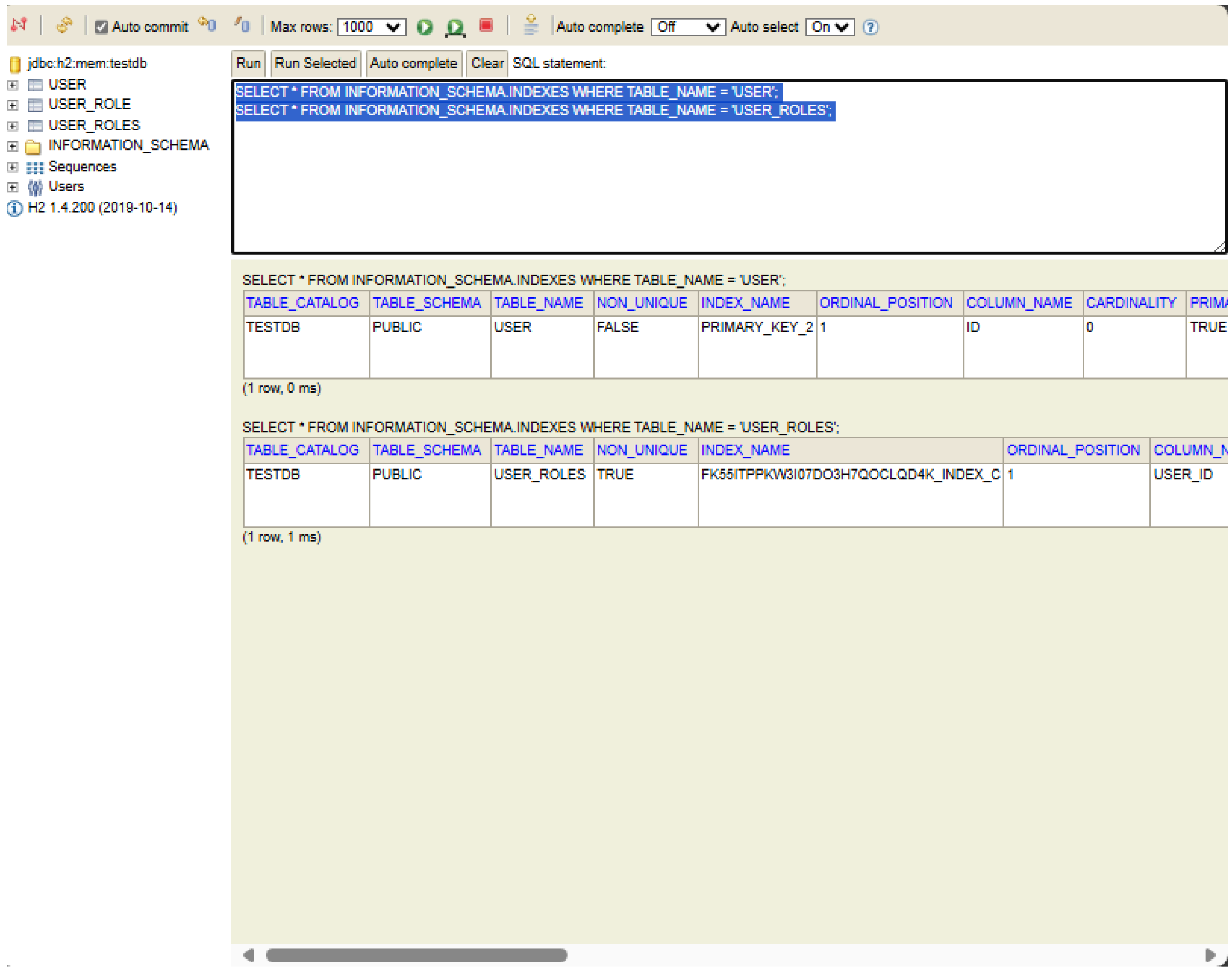
ID	USER_ID

(no rows, 1 ms)

Checking Existing Indexes

The error SHOW INDEXES FROM USER; occurs because the SHOW INDEXES statement isn't supported in H2 SQL. Instead, you can use the following query to list the indexes on the table in H2:

SELECT * FROM INFORMATION_SCHEMA.INDEXES WHERE TABLE_NAME = 'USER';
SELECT * FROM INFORMATION_SCHEMA.INDEXES WHERE TABLE_NAME = 'USER_ROLES';
This query will return information about all the indexes present on the USER and USER_ROLES tables.



Auto commit ✓ Max rows: 1000 Auto complete: Off Auto select: On

Run | Run Selected | Auto complete | Clear | SQL statement:

jdbc:h2:mem:testdb
- USER
- USER_ROLE
- USER_ROLES
- INFORMATION_SCHEMA
- Sequences
- Users

H2 1.4.200 (2019-10-14)

```
SELECT * FROM INFORMATION_SCHEMA.INDEXES WHERE TABLE_NAME = 'USER';
SELECT * FROM INFORMATION_SCHEMA.INDEXES WHERE TABLE_NAME = 'USER_ROLES';
```

SELECT * FROM INFORMATION_SCHEMA.INDEXES WHERE TABLE_NAME = 'USER';

TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	NON_UNIQUE	INDEX_NAME	ORDINAL_POSITION	COLUMN_NAME	CARDINALITY	PRIMA
TESTDB	PUBLIC	USER	FALSE	PRIMARY_KEY_2	1	ID	0	TRUE

(1 row, 0 ms)

SELECT * FROM INFORMATION_SCHEMA.INDEXES WHERE TABLE_NAME = 'USER_ROLES';

TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	NON_UNIQUE	INDEX_NAME	ORDINAL_POSITION	COLUMN_N
TESTDB	PUBLIC	USER_ROLES	TRUE	FK55ITPPKW3I07DO3H7QOCLQD4K_INDEX_C	1	USER_ID

(1 row, 1 ms)

SELECT * FROM INFORMATION_SCHEMA.INDEXES WHERE TABLE_NAME = 'USER';

CATA	SCHE	NAM	IQU	NAMI	L_POS	N_NA	ALIT	Y_KI	TYPE_	RAT	TYPI	_DES		COND	KS			YPE	AINT_	CLASS	Y
				RY_K					**RY KE**							**PRIMA** **C"."PRI** **ON** **C"."USE**			**RAINT**	**vstore.db.M** **x**	

(1 row, 0 ms)

SELECT * FROM INFORMATION_SCHEMA.INDEXES WHERE TABLE_NAME = 'USER_ROLES';

CATA	SCHI	NAN	IQU	NAME	L_PO	N_N	ALI	Y_K	TYPE	RAT	TYP	_DE		CON	KS			YPE	AINT_N	CLASS	Y
		OLI		**PKW3I07D** **4K_INDEX_**		**D**										**INDEX** **C"."FK55ITPI** **QOCLQD4K_** **C"."USER_RC** **D")**			**PKW3I0** **QD4K**	**vstore.db** **ndex**	

(1 row, 1 ms)