

# H2 Database Phase 2

Akash Desai  
ad3059@rit.edu

Deepam Shah  
ds3689@rit.edu

Nihal Parchand  
np9603@rit.edu

Viraj Chaudhari  
vc6346@rit.edu

## INTRODUCTION:

For implementing an enhanced feature for H2 Database in terms of data storage and indexing we have decided to implement a new data type called `PhoneNumber`. Phone number is an important piece of information which is present in almost every database. For our project, we are considering valid US phone numbers which follow the given constraints:

1. The phone number should have a length 10/12/13.

2. The user input should follow one of the following patterns:

2345678901

234-567-8901

(234)5678901

(234)567-8901

3. Then we compare this number with a list built with all the valid area codes as implemented by the North American Numbering Plan (NANP). It compares the first three digits from the user input and then compares it with the list entries and then returns if the number is a valid number or not. It also checks for the first digit of the number. If it is 0 or 1 then it is an invalid US number, otherwise it returns true.

## IMPLEMENTATION:

H2 allows users to create their own data types by implementing the minimal required `CustomDataTypesHandler` API. To use this feature, the `CustomPhoneDataTypeHandler.java` file should implement all the abstract methods defined in the `CustomDataTypesHandler` interface.

The following changes have been made to the following files:

CustomPhoneDataTypeHandler.java - This class implements all the methods from the CustomDataTypesHandler interface.

ErrorCode.java - We added another unique custom error code for throwing exceptions when the user enters an invalid phone number by incrementing the last error code value in the original errorcode file.

```
/**
 * The error with code <code>90149</code> is thrown when trying to enter an invalid phone number
 */
public static final int INVALID_PHONE_NUMBER = 90150;
```

Phone.java - The main function that we created was the validatePhoneNumber function which takes the phone number from the user as a string and then checks if it is a valid number or not. We stored all the valid US area codes in a list of strings.

```
String[] areaCodes = {"201", "202", "203", "204", "205", "206", "207", "208", "209", "210", "212", "213", "214", "215", "216",
    "217", "218", "219", "223", "224", "225", "226", "228", "229", "231", "234", "239", "240", "242", "246", "248", "250",
    "251", "252", "253", "254", "256", "260", "262", "264", "267", "268", "269", "270", "276", "281", "284", "289", "301",
    "302", "303", "304", "305", "306", "307", "308", "309", "310", "312", "313", "314", "315", "316", "317", "318", "319",
    "320", "321", "323", "325", "330", "334", "336", "337", "339", "340", "345", "347", "351", "352", "360", "361", "386",
    "401", "402", "403", "404", "405", "406", "407", "408", "409", "410", "412", "413", "414", "415", "416", "417", "418",
    "419", "423", "424", "425", "430", "432", "434", "435", "438", "440", "441", "443", "450", "456", "469", "473", "478",
    "479", "480", "484", "500", "501", "502", "503", "504", "505", "506", "507", "508", "509", "510", "512", "513", "514",
    "515", "516", "517", "518", "519", "520", "530", "540", "541", "551", "559", "561", "562", "563", "567", "570", "571",
    "573", "574", "580", "585", "586", "600", "601", "602", "603", "604", "605", "606", "607", "608", "609", "610", "612",
    "613", "614", "615", "616", "617", "618", "619", "620", "623", "626", "630", "631", "636", "641", "646", "647", "649",
    "650", "651", "660", "661", "662", "664", "670", "671", "678", "682", "684", "700", "701", "702", "703", "704", "705",
    "706", "707", "708", "709", "710", "712", "713", "714", "715", "716", "717", "718", "719", "720", "724", "727", "731",
    "732", "734", "740", "754", "757", "758", "760", "762", "763", "765", "767", "769", "770", "772", "773", "774", "775",
    "778", "780", "781", "784", "785", "786", "787", "800", "801", "802", "803", "804", "805", "806", "807", "808", "809",
    "810", "812", "813", "814", "815", "816", "817", "818", "819", "828", "829", "830", "831", "832", "843", "845", "847",
    "848", "850", "856", "857", "858", "859", "860", "862", "863", "864", "865", "866", "867", "868", "869", "870", "876",
    "877", "878", "888", "900", "901", "902", "903", "904", "905", "906", "907", "908", "909", "910", "912", "913", "914",
    "915", "916", "917", "918", "919", "920", "925", "928", "931", "936", "937", "939", "940", "941", "947", "949", "951",
    "952", "954", "956", "970", "971", "972", "973", "978", "979", "980", "985", "989"};
```

Along with this, we also used regex patterns to eliminate invalid numbers and restricted the valid inputs to the above mentioned patterns. Another condition that we added to this is that the first number can not be a 0 or 1.

SysProperties.java - The CustomPhoneDataTypeHandler class name that is used must be the same on client and server to work correctly. This function provides support for user defined custom data types. The initial value is set to null if there are no custom data types used.

```
public static final String CUSTOM_DATA_TYPES_HANDLER =
    Utils.getProperty("h2.customDataTypesHandler", "org.h2.api.CustomPhoneDataTypeHandler");
```

Value.java - This is the base class file which contains all the data types, it's conversion methods and comparison methods supported by the H2 database. For adding the custom data type Phone, the following code is used.

```
/**
 * The value type for PHONE values.
 */
public static final int PHONE = 42;

/**
 * The number of value types.
 */
public static final int TYPE_COUNT = PHONE + 1;
```

We modified the switch case for additional condition for Phone datatype.

```
case PHONE:
    return 53 000;
```

ValuePhone.java - This class overrides the methods from the Value class which is the base class for all data types.

\_messages\_en.prop - We added a line in the original error code file.

“90150=Invalid US phone number or phone number format “ to display the custom error message for invalid phone numbers.

## DEMO:

### 1. Creating a table with custom Phone Number data type

Create a table STUDENT with attributes ID INT primary key, name VARCHAR(15) and phoneNumber PHONE.

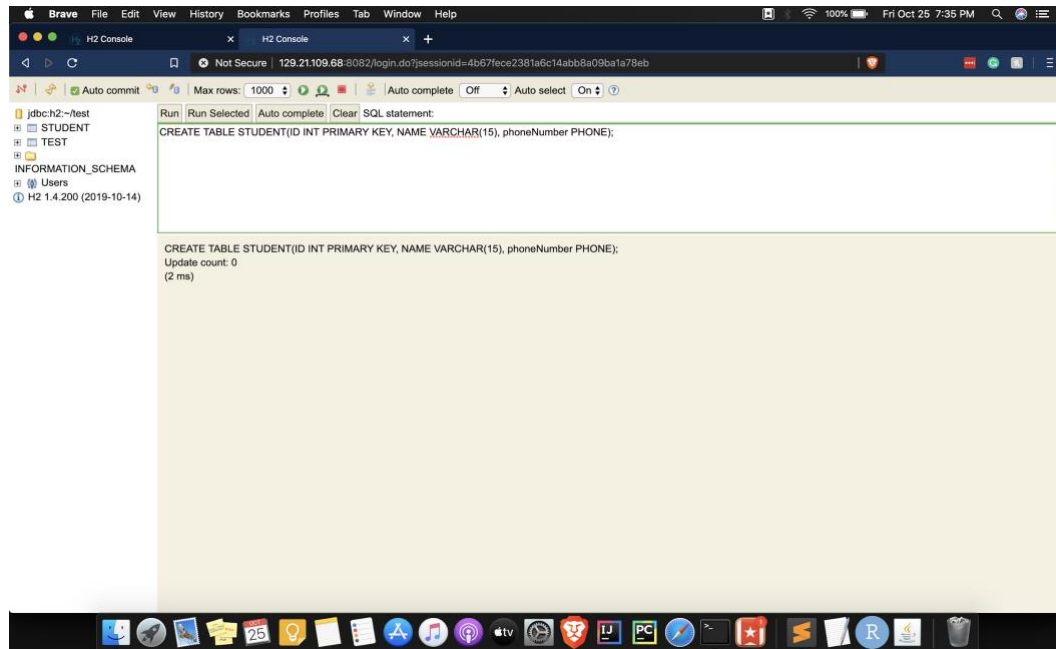
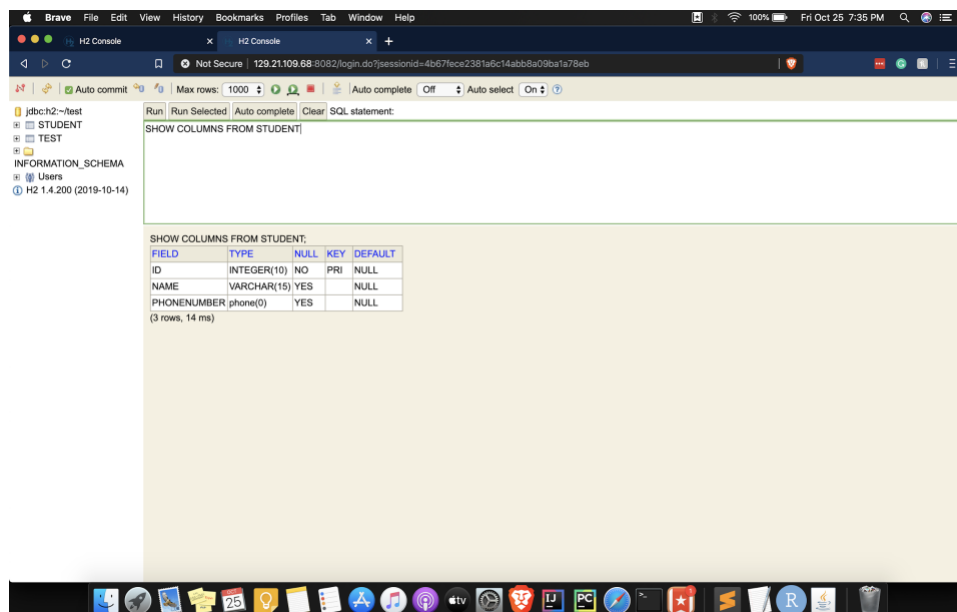
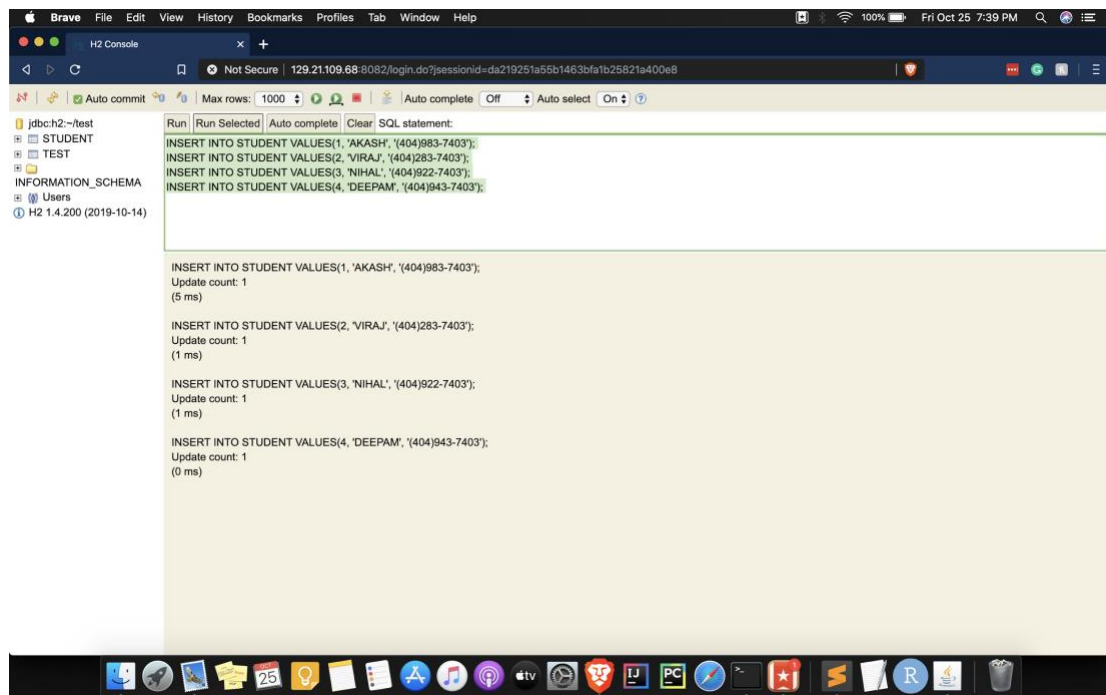


Table STUDENT is successfully created with attributes ID INT primary key, name VARCHAR(15) and phoneNumber PHONE.

### 2. Displaying the column information for the created table



### 3. Inserting new records in the user info table



The screenshot shows the H2 Console interface in a Brave browser. The left sidebar displays the database schema with 'STUDENT' selected. The main area contains four INSERT statements, each followed by its execution status and update count.

```
Run Run Selected Auto complete Clear SQL statement:
INSERT INTO STUDENT VALUES(1, 'AKASH', '(404)983-7403');
INSERT INTO STUDENT VALUES(2, 'VIRAJ', '(404)283-7403');
INSERT INTO STUDENT VALUES(3, 'NIHAL', '(404)922-7403');
INSERT INTO STUDENT VALUES(4, 'DEEPAM', '(404)943-7403');

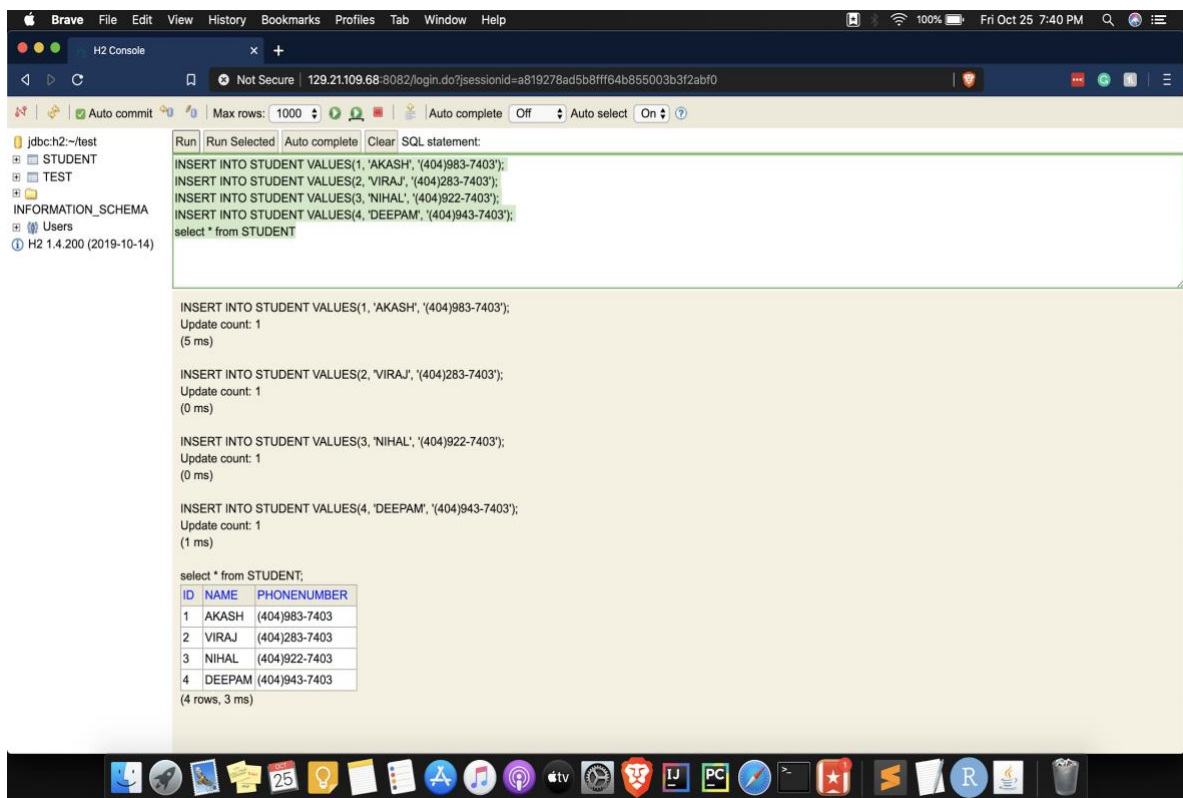
INSERT INTO STUDENT VALUES(1, 'AKASH', '(404)983-7403');
Update count: 1
(5 ms)

INSERT INTO STUDENT VALUES(2, 'VIRAJ', '(404)283-7403');
Update count: 1
(1 ms)

INSERT INTO STUDENT VALUES(3, 'NIHAL', '(404)922-7403');
Update count: 1
(1 ms)

INSERT INTO STUDENT VALUES(4, 'DEEPAM', '(404)943-7403');
Update count: 1
(0 ms)
```

### 4. Displaying records of the user info table



The screenshot shows the H2 Console interface with the same four INSERT statements as in the previous image, followed by a SELECT statement. The results of the SELECT statement are displayed in a table format.

```
Run Run Selected Auto complete Clear SQL statement:
INSERT INTO STUDENT VALUES(1, 'AKASH', '(404)983-7403');
INSERT INTO STUDENT VALUES(2, 'VIRAJ', '(404)283-7403');
INSERT INTO STUDENT VALUES(3, 'NIHAL', '(404)922-7403');
INSERT INTO STUDENT VALUES(4, 'DEEPAM', '(404)943-7403');
select * from STUDENT

INSERT INTO STUDENT VALUES(1, 'AKASH', '(404)983-7403');
Update count: 1
(5 ms)

INSERT INTO STUDENT VALUES(2, 'VIRAJ', '(404)283-7403');
Update count: 1
(0 ms)

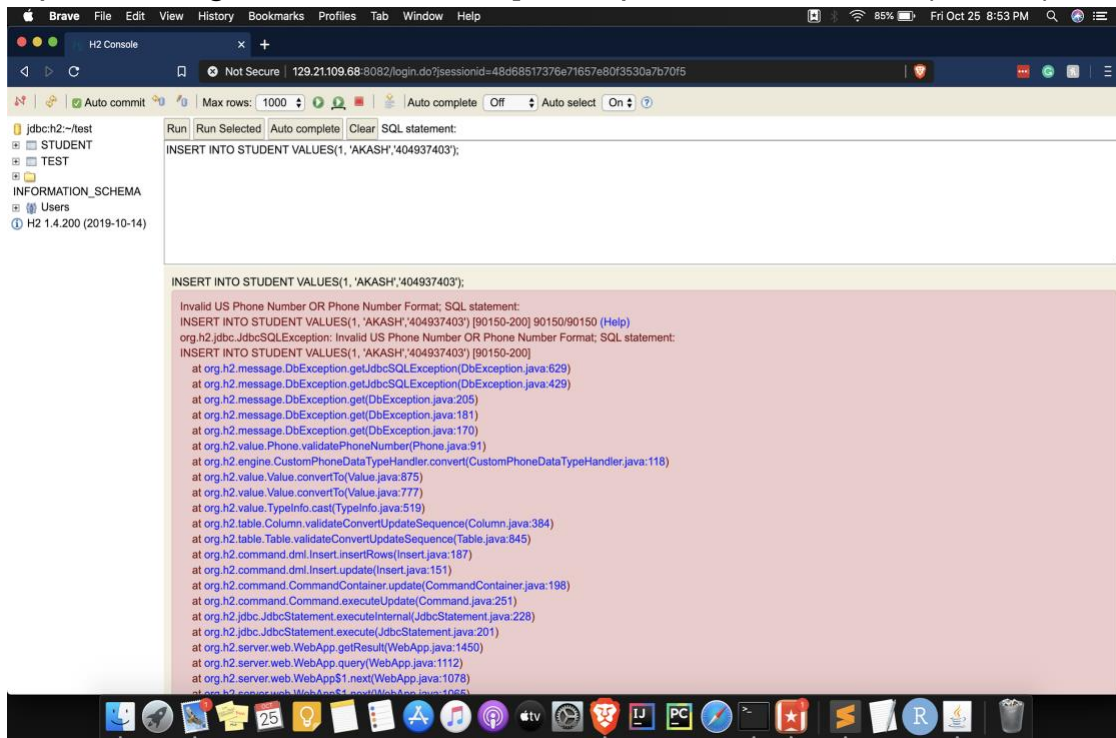
INSERT INTO STUDENT VALUES(3, 'NIHAL', '(404)922-7403');
Update count: 1
(0 ms)

INSERT INTO STUDENT VALUES(4, 'DEEPAM', '(404)943-7403');
Update count: 1
(1 ms)

select * from STUDENT;
ID NAME PHONENUMBER
1 AKASH (404)983-7403
2 VIRAJ (404)283-7403
3 NIHAL (404)922-7403
4 DEEPAM (404)943-7403
(4 rows, 3 ms)
```

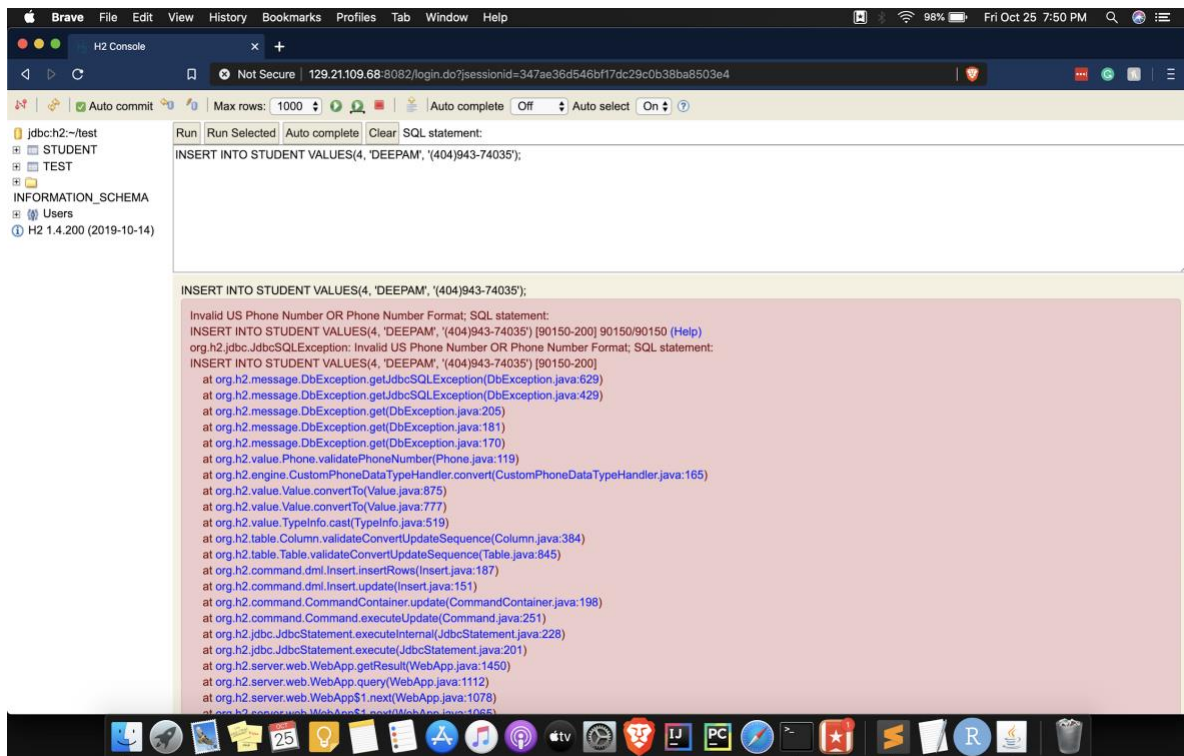
## 5. Some examples of invalid inputs

### a. Input of length less than 10 [no separators used “ ( “ , “ ) ” , “ - ” ]



The screenshot shows the H2 Console interface in a Brave browser. The SQL statement entered is `INSERT INTO STUDENT VALUES(1, 'AKASH','404937403');`. The console displays a detailed error message: `Invalid US Phone Number OR Phone Number Format; SQL statement: INSERT INTO STUDENT VALUES(1, 'AKASH','404937403') [90150-200] 90150/90150 (Help)`. The error is followed by a stack trace starting with `org.h2.jdbc.JdbcSQLException: Invalid US Phone Number OR Phone Number Format; SQL statement: INSERT INTO STUDENT VALUES(1, 'AKASH','404937403') [90150-200]`. The stack trace includes various internal H2 database classes like `org.h2.message.DbException`, `org.h2.value.Phone`, and `org.h2.engine.CustomPhoneDataHandler`. The browser's address bar shows the URL `129.21.109.68:8082/login.do?jsessionid=48d68517376e71657e80f3530a7b70f5`.

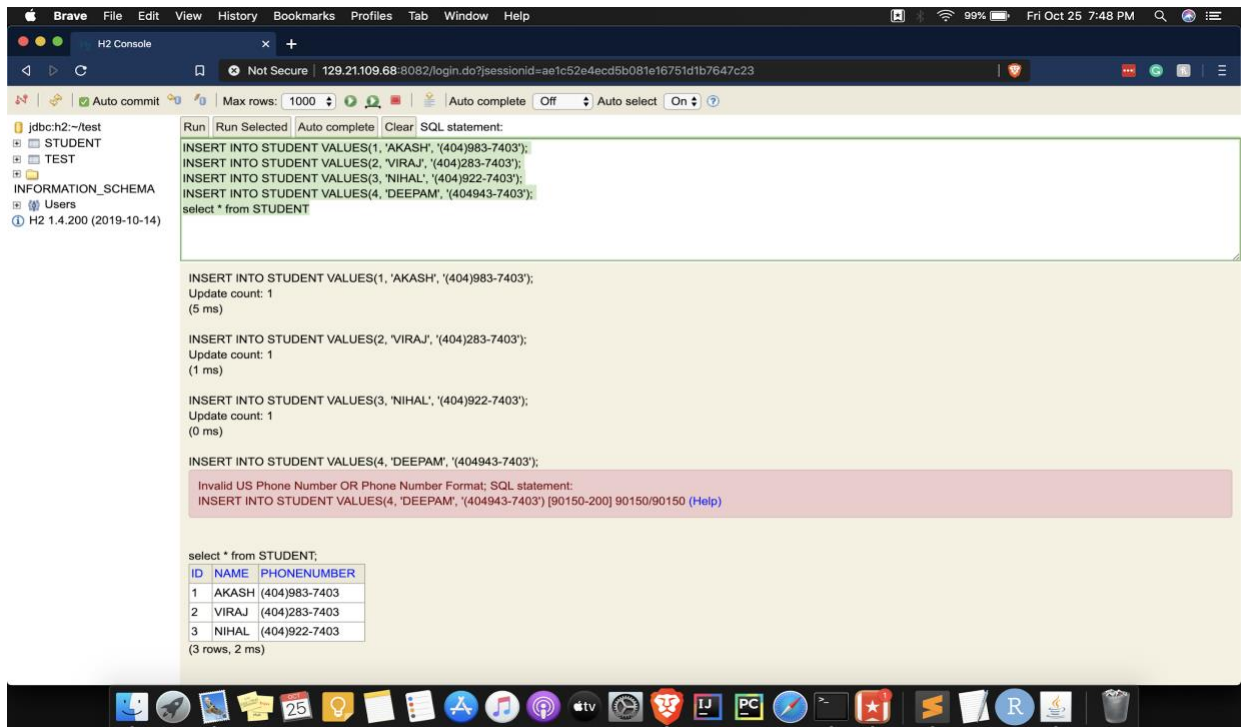
### b. Input of length greater than 13 [separators used “ ( “ , “ ) ” , “ - ” ]



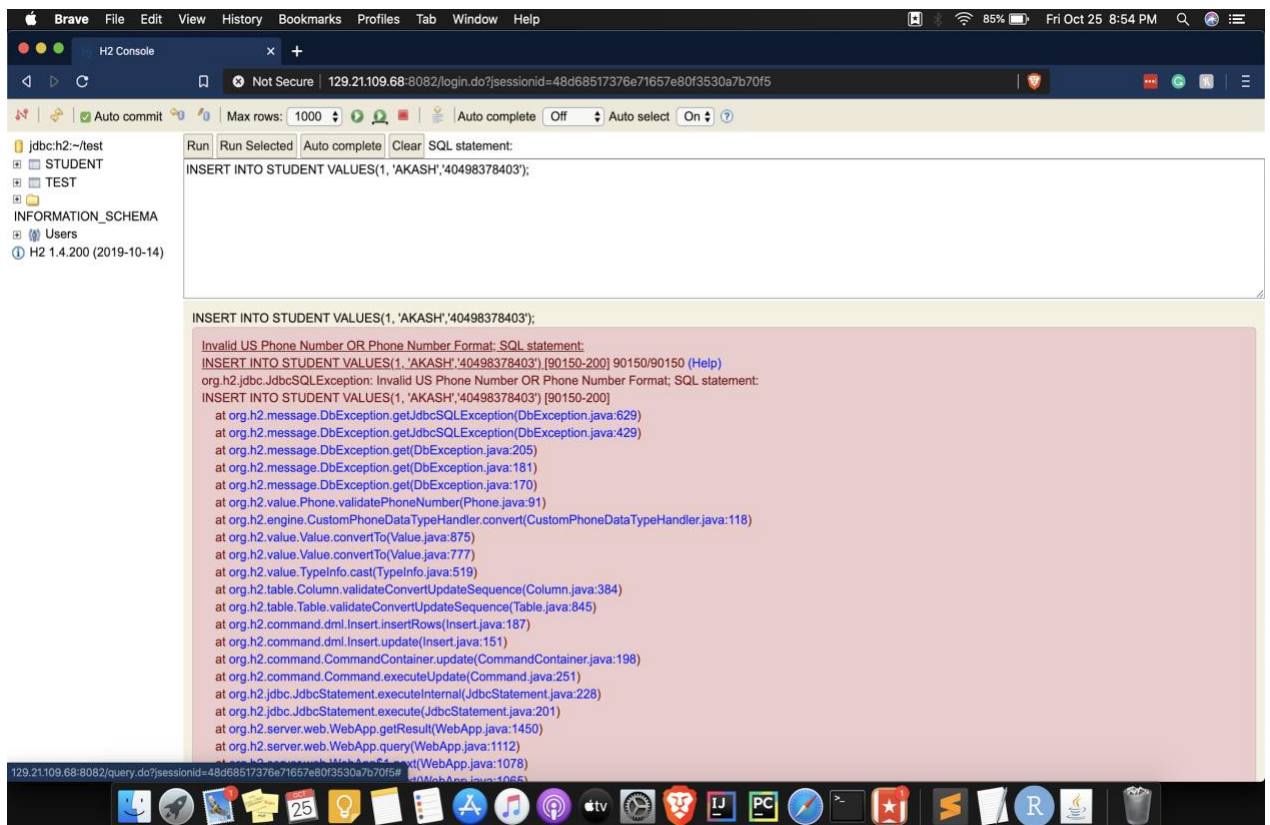
The screenshot shows the H2 Console interface in a Brave browser. The SQL statement entered is `INSERT INTO STUDENT VALUES(4, 'DEEPAM', '(404)943-74035');`. The console displays a detailed error message: `Invalid US Phone Number OR Phone Number Format; SQL statement: INSERT INTO STUDENT VALUES(4, 'DEEPAM', '(404)943-74035') [90150-200] 90150/90150 (Help)`. The error is followed by a stack trace starting with `org.h2.jdbc.JdbcSQLException: Invalid US Phone Number OR Phone Number Format; SQL statement: INSERT INTO STUDENT VALUES(4, 'DEEPAM', '(404)943-74035') [90150-200]`. The stack trace includes various internal H2 database classes like `org.h2.message.DbException`, `org.h2.value.Phone`, and `org.h2.engine.CustomPhoneDataHandler`. The browser's address bar shows the URL `129.21.109.68:8082/login.do?jsessionid=347ae36d546bf17dc29c0b38ba8503e4`.



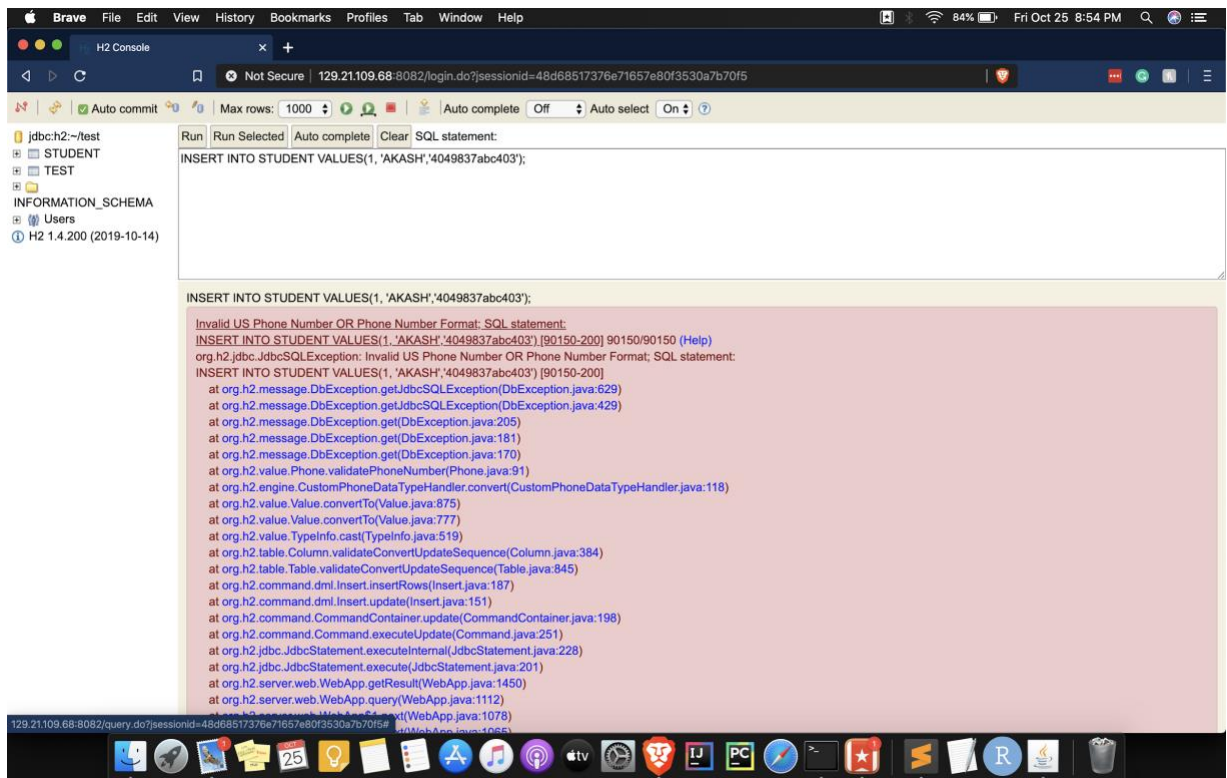
- c. Also handles cases when inserting multiple records with valid and invalid records



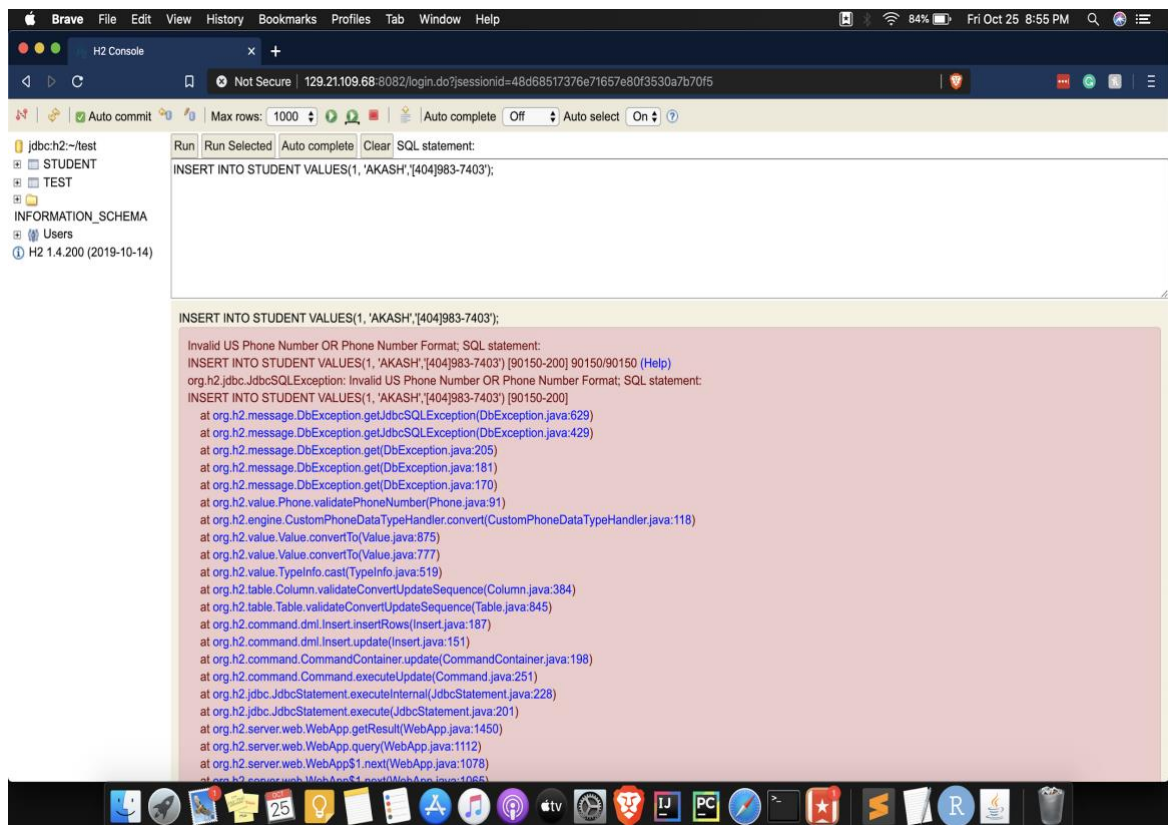
- d. Input of length 11



## e. Input containing characters

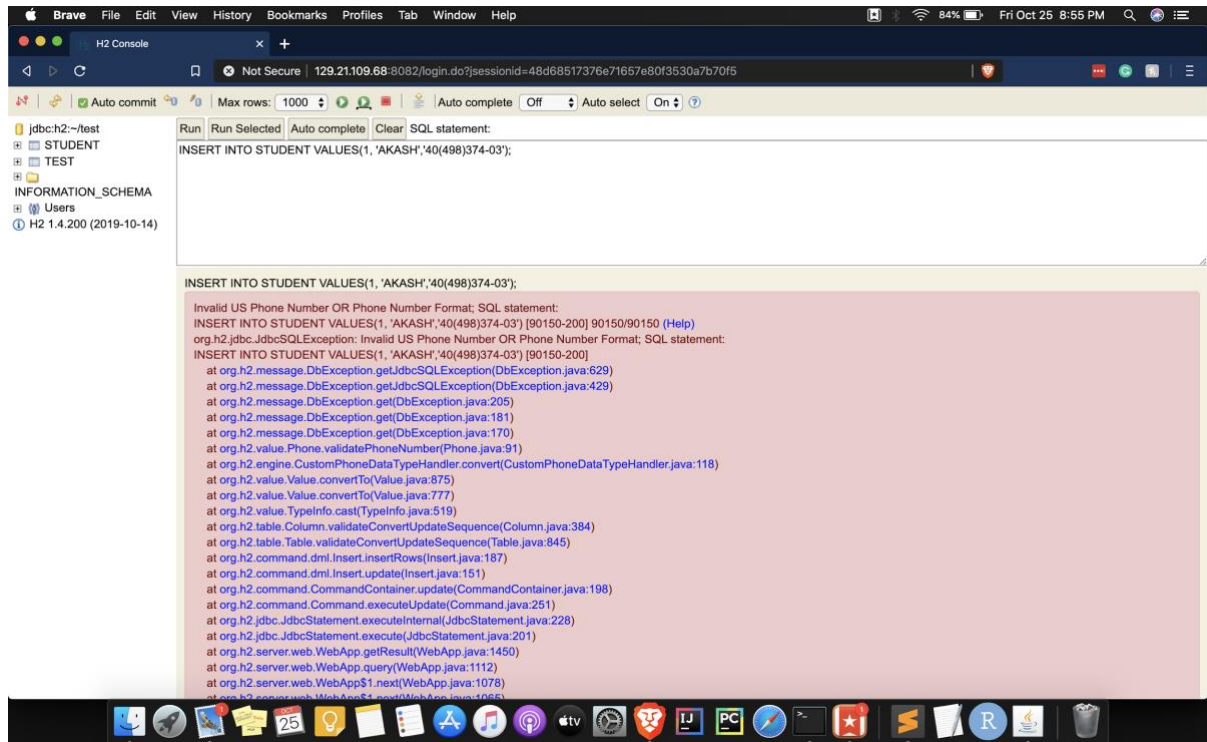


## f. Input containing special characters except “ ( “ , “ ) ” , “ - ”

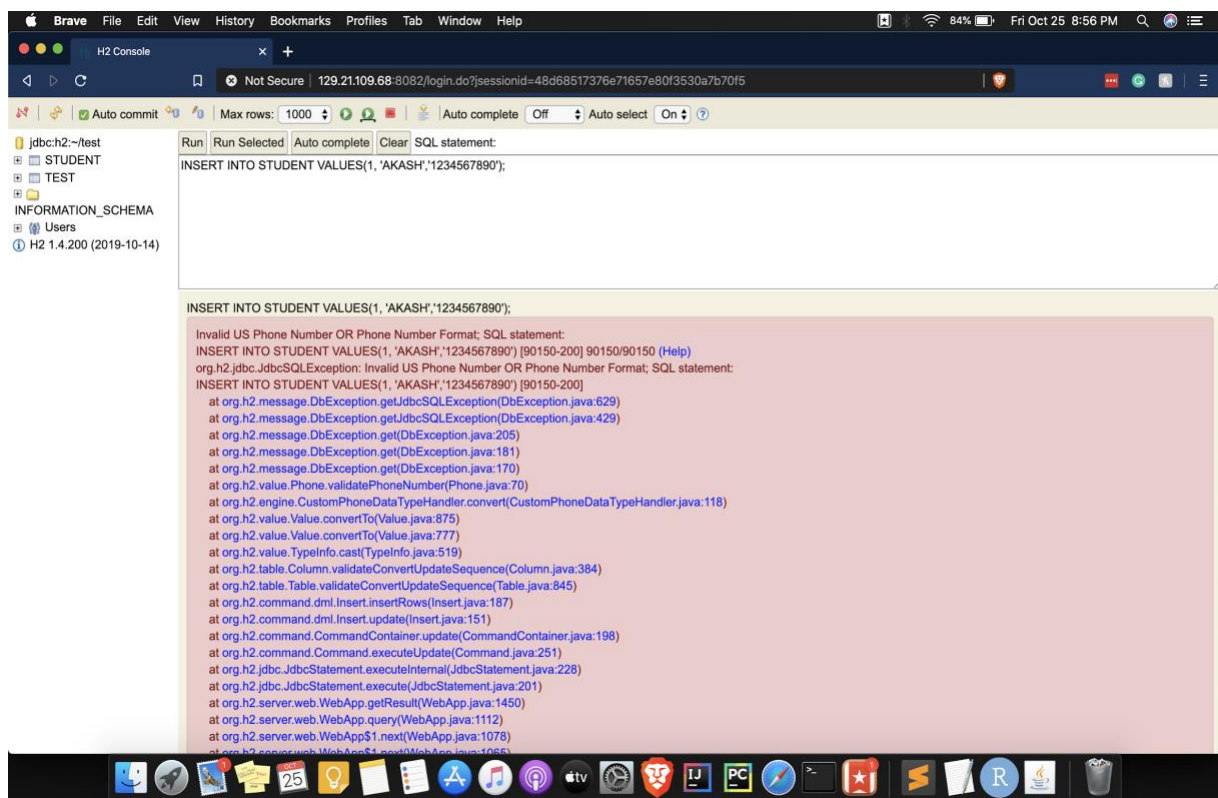




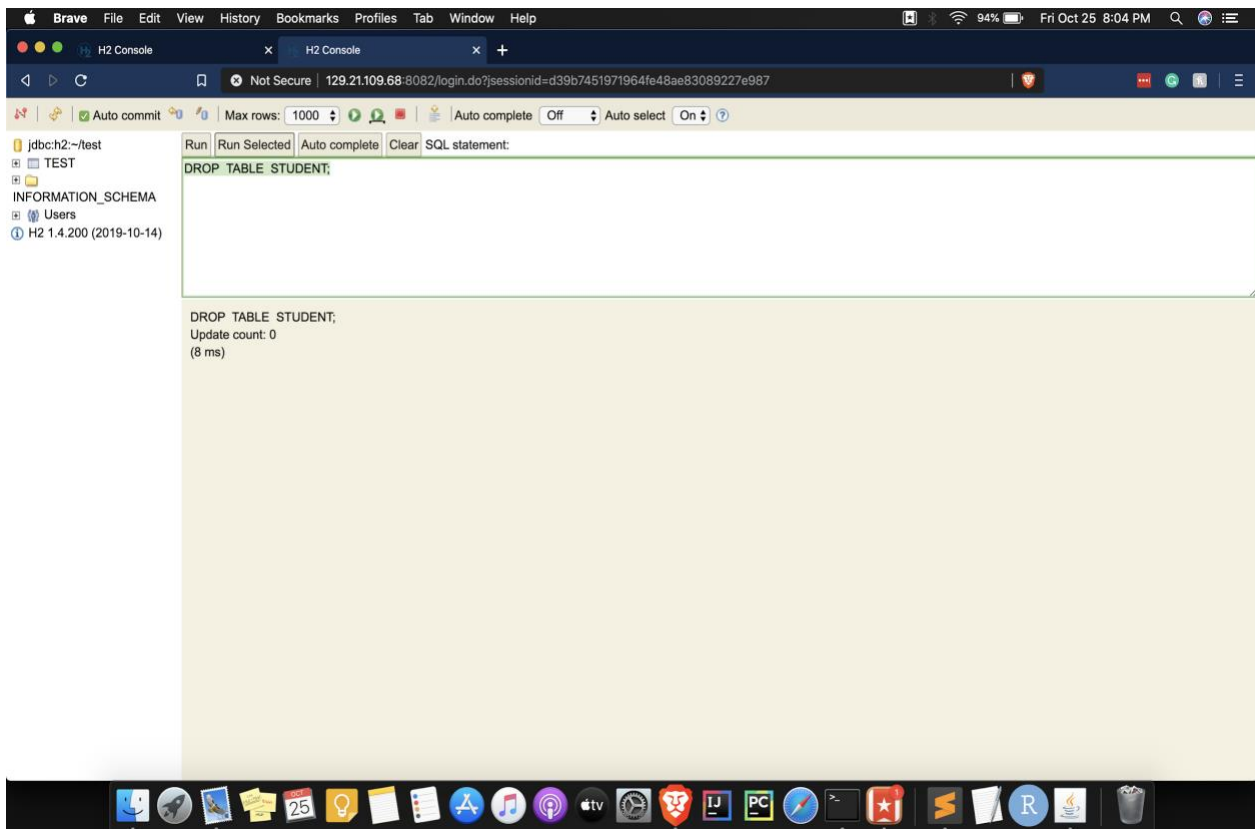
g. Input containing special characters “ ( , “ ) ” , “ - ” at wrong places



h. First digit 1



## 6. Deleting entire table



## REFERENCES:

1. <https://github.com/h2database/h2database>
2. [http://www.h2database.com/html/advanced.html#custom\\_data\\_types\\_handler\\_api](http://www.h2database.com/html/advanced.html#custom_data_types_handler_api)