USER GUIDE

Here's a simple user guide using some examples. Please see the images below to learn how to execute commands.

- **CREATE**
  - *create table dogs (id **int**, age **smallint**, name **text**);*
- **INSERT**
  - *insert into dogs (id, age, name) values (2, 5, "rover");*
  - *insert into dogs (id, age, name) values (6, 9, "pug");*
- **SELECT**
  - *select * from dogs;*

```
davisql> create table dogs ( id int, age smallint, name text );
CASE: CREATE
STUB: Calling your method to create a table
Parsing the string:"create table dogs ( id int, age smallint, name text )"
davisql> insert into dogs ( id, age, name ) values (2, 5, "rover");
CASE: INSERT
INSERT METHOD
Parsing the string:"insert into dogs ( id, age, name ) values (2, 5, "rover")"
davisql> insert into dogs ( id, age, name ) values (6, 9, "pug");
CASE: INSERT
INSERT METHOD
Parsing the string:"insert into dogs ( id, age, name ) values (6, 9, "pug")"
davisql> select * from dogs;
CASE: SELECT
STUB: This is the parseQuery method
        Parsing the string:"select * from dogs"
-----------------------
id   |age   |name      |
-----------------------
2    |5     |"rover"   |
6    |9     |"pug"     |
davisql>
```

- **PRIMARY KEY**
  - *create table dogs (id int* **primary key**, *age smallint, name text);*

(by default it assumes first column as primary key.

Here is an example where if you try inserting the same key again you get Duplicate key error.

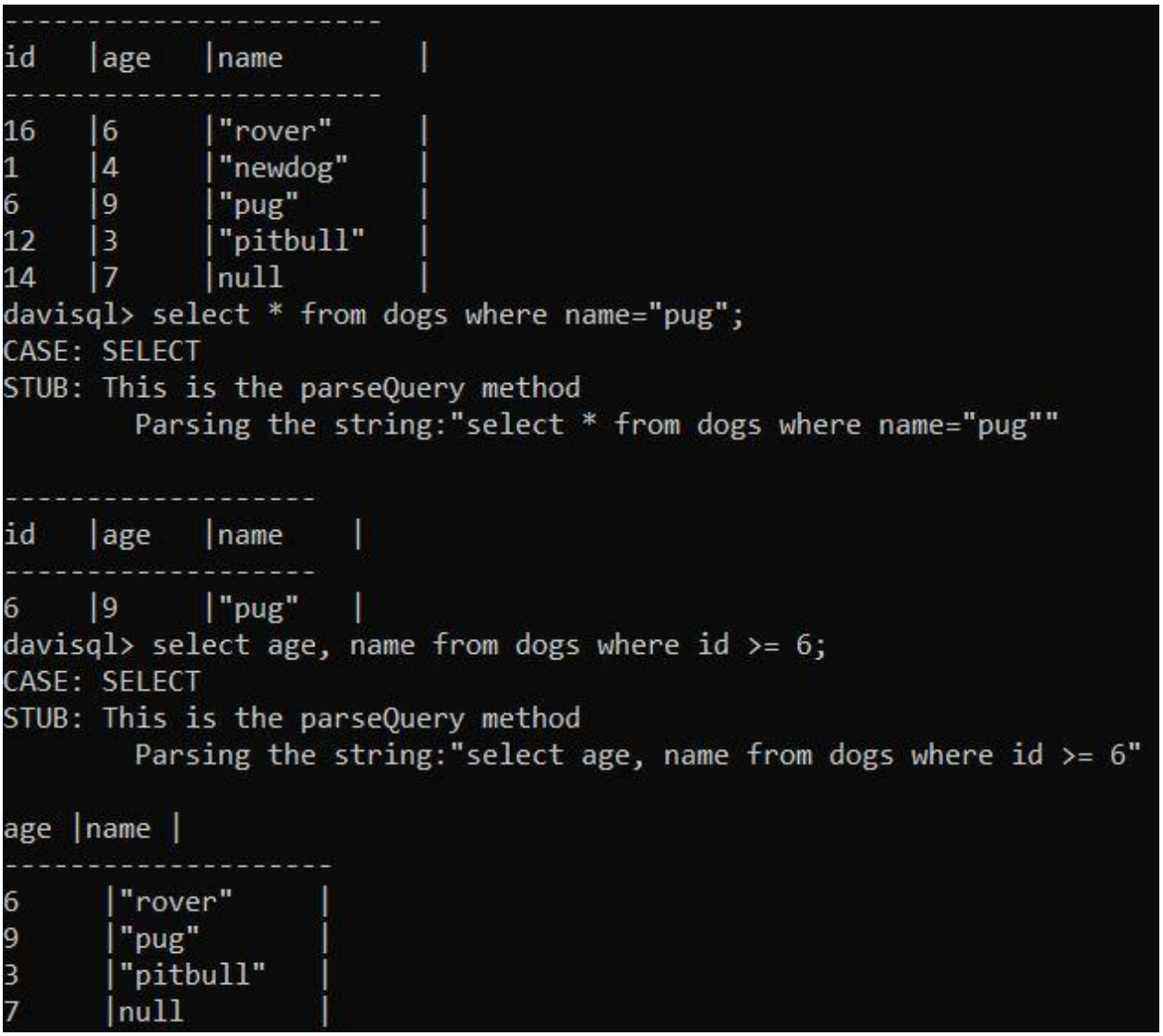
```
davisql> create table dogs ( id int primary key, name text );
CASE: CREATE
STUB: Calling your method to create a table
Parsing the string:"create table dogs ( id int primary key, name text )"
davisql> insert into dogs ( id, name ) values ( 3, "asd" );
CASE: INSERT
INSERT METHOD
Parsing the string:"insert into dogs ( id, name ) values ( 3, "asd" )"
davisql> insert into dogs ( id, name ) values ( 3, "ddd");
CASE: INSERT
INSERT METHOD
Parsing the string:"insert into dogs ( id, name ) values ( 3, "ddd")"
@@@Error:: Duplicate key found
davisql>
```

- **SELECT specifics**

SELECT is supported with selecting specific columns, for example in this case:

select age, name from dogs; //will display just age and name columns.

- *select * from dogs where name=”pug”;*
- *select* **age, name** *from dogs where id >= 6;*



- **SELECT with binary comparison operators**

In where clause the "=" operator works for all columns, however other operators like >, >=, <, <= works only on first column since it is the primary key.

- *select * from dogs where* **id >= 12**;

```
davisql> select * from dogs;
CASE: SELECT
STUB: This is the parseQuery method
        Parsing the string:"select * from dogs"
--------------------------
id   |age   |name        |
--------------------------
6    |9     |"pug"       |
12   |3     |"pitbull"   |
15   |4     |"rover"     |
davisql> select * from dogs where id >= 12;
CASE: SELECT
STUB: This is the parseQuery method
        Parsing the string:"select * from dogs where id >= 12"
--------------------------
id   |age   |name        |
--------------------------
12   |3     |"pitbull"   |
15   |4     |"rover"     |
davisql>
```

- **SHOW tables and DROP Tables example**

➢ *show table*s;

➢ *drop table cats*; //NOTE: you must have executed create table cats (id int) for example.

```
davisql> show tables;
CASE: SHOW
SHOW METHOD
Parsing the string:"show tables"

table_name|
--------------------
davisbase_tables    |
dogs                |
cats                |

davisql> drop table cats;
CASE: DROP
DROP METHOD
Parsing the string:"drop table cats"
davisql> show tables;
CASE: SHOW
SHOW METHOD
Parsing the string:"show tables"

table_name|
--------------------
davisbase_tables    |
dogs                |
```

- **Handling NULL Values**

Initially dogs didn't have NOT NULL check on name

- insert into dogs (id, age, name) values (14, 7, null);

```
davisql> insert into dogs ( id, age, name ) values ( 14, 7, null );
CASE: INSERT
INSERT METHOD
Parsing the string:"insert into dogs ( id, age, name ) values ( 14, 7, null )"
davisql> select * from dogs;
CASE: SELECT
STUB: This is the parseQuery method
        Parsing the string:"select * from dogs"

----------------------
id   |age   |name        |
----------------------
16   |6     |"rover"     |
1    |4     |"newdog"    |
6    |9     |"pug"       |
12   |3     |"pitbull"   |
14   |7     |null        |
```

If dogs had been CREATEd with NOT NULL constraint as shown belore then INSERTing NULL would throw *"@@@Cant enter null in given column as per schema*"

```
davisql> create table dogs ( id int, age smallint, name text not null );
CASE: CREATE
STUB: Calling your method to create a table
Parsing the string:"create table dogs ( id int, age smallint, name text not null )"
davisql> insert into dogs ( id, age, name ) values ( 2, 4, null );
CASE: INSERT
INSERT METHOD
Parsing the string:"insert into dogs ( id, age, name ) values ( 2, 4, null )"
@@@Cant enter null in given column as per schema
```