Loading CSV file into any Table:

We can import the data stored into a csv file into a Table inside MySQL with the following command.

```
LOAD data local infile '~/users_500k.csv' INTO TABLE <table_name>;
```

By default, the field delimeter for the CSV file will be a comma. In the case of custom delimiter, it can be done by mentioning in the query as follows:

```
LOAD data local infile '~/users_500k.csv' INTO TABLE <table_name> fields terminated BY '|';
```

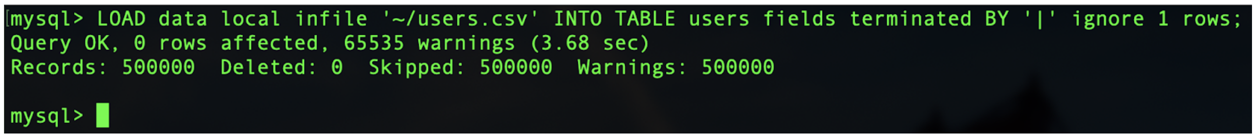
By default, all the entries inside the CSV file will be imported into the MySQL table. Generally, the csv file's first line in the header of the file. So, in such cases, the first row needs to be skipped while importing. The syntax for this is:

```
LOAD data local infile '~/users_500k.csv' INTO TABLE table_name ignore 1 rows;
```

The above 2 conditions can be combined as follows:

```
LOAD data local infile '~/users_500k.csv' INTO TABLE table_name fields
terminated BY '|' ignore 1 rows;
```

Example:



```
mysql> LOAD data local infile '~/users.csv' INTO TABLE users fields terminated BY '|' ignore 1 rows;
Query OK, 0 rows affected, 65535 warnings (3.68 sec)
Records: 500000  Deleted: 0  Skipped: 500000  Warnings: 500000

mysql> 
```

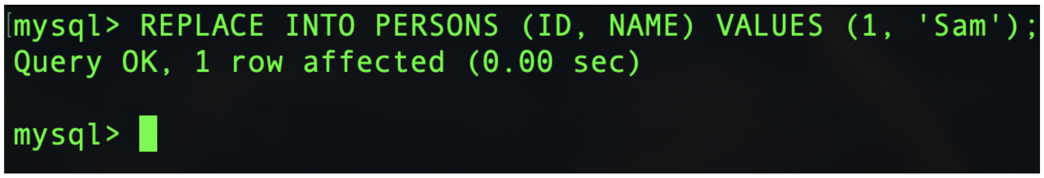
UPSERT Operation:

In some cases, there will be a need to insert a row or update the current row if the primary key and unique constraints are satisfied. These kinds of operations are called UPSERT queries. In MySQL, we can use **"REPLACE INTO"** statements for achieving this.

```
REPLACE [INTO] table_name(column_list) VALUES(value_list);
```

The "**REPLACE INTO"** statement's syntax is nearly identical to the **INSERT INTO** statements, with only the "**INSERT**" keywords being replaced by **"REPLACE"**.

Example:



```
mysql> REPLACE INTO PERSONS (ID, NAME) VALUES (1, 'Sam');
Query OK, 1 row affected (0.00 sec)

mysql> 
```

A **REPLACE INTO** statement deletes the current row before inserting the new row. Thus for a high performant system, it should be replaced by **"INSERT INTO … ON DUPLICATE KEY UPDATE"** queries. This can be done in the following manner.

```
INSERT INTO table_name (column_list) VALUES (value_list) ON DUPLICATE KEY
UPDATE column_name=new_value;
```

Multiple columns can be updated in the UPDATE clause in this query.

Note:
In this project, we are not aiming for a high scale, so using **"INSERT INTO … ON DUPLICATE KEY UPDATE"** is not mandatory. You can read more about this query here: https://dev.mysql.com/doc/refman/5.6/en/insert-on-duplicate.html.

Combining Multiple AND/OR Conditions in WHERE Clause
Multiple AND/OR conditions can be combined in WHERE clause like below.

```
SELECT * FROM users WHERE (gender="All" OR gender="M") AND city="mumbai"
```