# Chapter 03: conception and implementation:

**Dataset**

Our goal is to develop a deep learning model capable of detecting SQL injection attacks apart from other regular queries. In order to train a successful and effective our deep learning model, the dataset must be carefully processed. To achieve this, we needed to construct a Dataset consisting of samples divided into two classes: "malicious queries" and "non-malicious queries". By using this dataset to train our model, it can learn to distinguish between the two classes and accurately identify SQL injection attacks.

During our search for a suitable Datasets, we came across the "SQL Injection Dataset" list on the Kaggle platform [1]. We found a list of datasets that was created by Syed Hussain contain Three (03) versions:

* **SQLi.csv** (723.15 kB) contains **3951** samples with **78%** classified as normal queries and **28%** as malicious queries.
* **SQLiV2.csv** (3.61 MB) contains **33726** samples with **66%** classified as normal queries and **34%** as malicious queries.
* **SQLiV3.csv** (2.32 MB) contains **30873** samples with **62%** classified as normal queries and **37%** as malicious queries and **1%** as other.

At first vision we choose the **SQLiV2** Dataset because of the size of samples in it in comparison to other, we trained our model with that Dataset in first time but the results were not satisfied while predicting normal queries. After analyzing the situation, we found that there was no normal queries in the dataset, only free text in place flagged as normal queries.

Because of the big issue in **SQLiV2.csv**, we tried the **SQLiV3.csv** Dataset**.** After reviewing it, we identified certain deficiencies that need to be cleared using some preprocessing steps like:

* Remove any unnecessary Strings or characters in the sql statements. We found two comas (,,) in the end of the queries so we removed them using a Register-Expression technique using a python script.
* Remove not valid empty columns. We need only two valid columns, the **Statement** and the **Label**, in that Dataset we found two empty and not valid columns that was removed using the Office Excel Software
* Remove empty and free text Rows. We found many empty and free text rows that was removed using the Office Excel Software
* Remove wrong SQL queries. We found not valid SQL statements that was identified and cleared manually

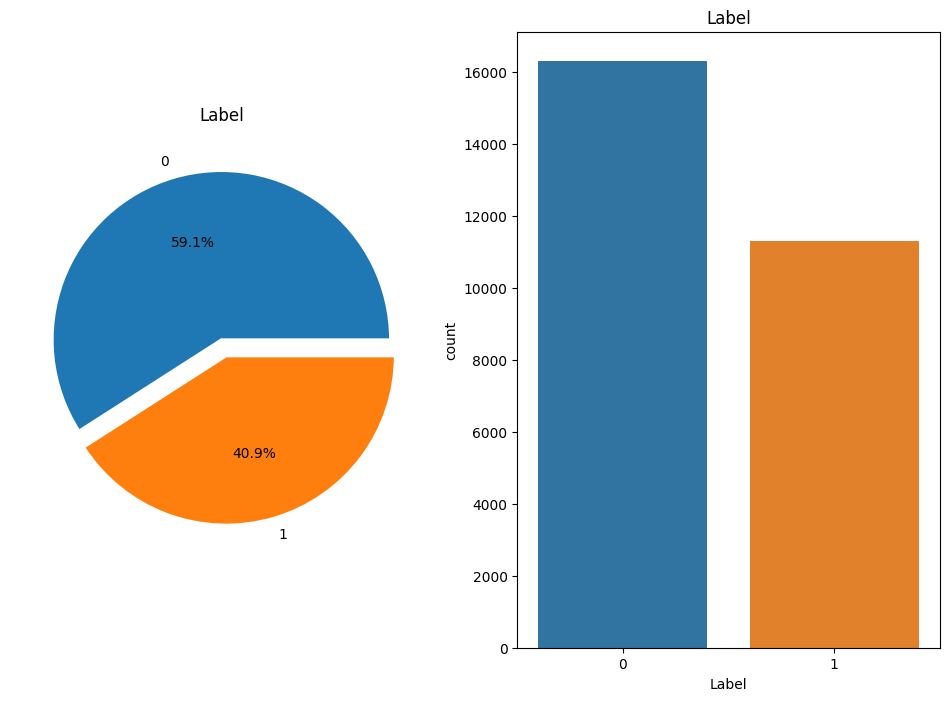
By applying the above-preprocessed steps, we generated a new Dataset with **50%** classified as normal queries and **50%** as malicious queries.

In real life, web applications receives a number of normal queries more than the malicious queries. Basing on this rule, we decided to increase the number of normal queries by generating random SQL statements using an SQL generator that was created with Python.

Our SQL generator use Four (04) separated python files:

* **sql\_SELECT\_generator.py**, generates 2000 random SELECT statements with random data samples.
* **sql\_INSERT\_generator.py**, generates 1000 random INSERT statements with random data samples.
* **sql\_UPDATE\_generator.py,** generates 1000 random UPDATE statements with random data samples.
* **sql\_DELETE\_generator.py,** generates 1000 random DELETE statements with random data samples.

After we added the generated SQL statements to the Dataset, we finally came up with a partitioned version that has **16307** **(59.1%)** "non-malicious queries" against **11298 (40.9%)** "malicious queries".



After clearing the different deficiencies and increased the number of normal queries, we came up with a new valid preprocessed Dataset that has sufficient diversity in both categories, and ready for use in the next step. We ensured that our deep learning model would only train on preprocessed samples that belongs to one of the two classes and could, therefore, learn to differentiate between malicious and non-malicious queries effectively.

[1] [https://www.kaggle.com/datasets/syedsaqlainhussain/sql-injection-dataset](https://www.kaggle.com/datasets/syedsaqlainhussain/sql-injection-dataset?select=sqliv2.csv)