# SF Crime Statistics with Spark Streaming

## Overview

In this project, you will be provided with a real-world dataset, extracted from Kaggle,

on San Francisco crime incidents, and you will provide statistical analyses of the data using

Apache Spark Structured Streaming. You will draw on the skills and knowledge you've learned in

this course to create a Kafka server to produce data, and ingest data through Spark Structured Streaming.

### Environment

* Spark 2.4.3
* Scala 2.11.x
* Java 1.8.x
* Kafka build with Scala 2.11.x
* Python 3.6.x or 3.7.x

### How to Run?

#### Start Zookeeper and Kafka Server

```

/usr/bin/zookeeper-server-start config/zookeeper.properties

/usr/bin/kafka-server-start config/server.properties

```

#### Run Kafka Producer server

`python kafka\_server.py`

#### Run the kafka Consumer server

`python kafka\_consumer.py`

#### Submit Spark Streaming Job

`spark-submit --packages org.apache.spark:spark-sql-kafka-0-10\_2.11:2.3.4 --master local[\*] data\_stream.py`

### kafka consumer console output

Text

Description automatically generated

### Streaming progress reporter

Shape, rectangle

Description automatically generated

### Output

Text

Description automatically generated

## Step 3

**Question 1**

How did changing values on the SparkSession property parameters affect the throughput and latency of the data?

ANS : It either increased or decreased processedRowsPerSecond

**Question 2**

What were the 2-3 most efficient SparkSession property key/value pairs? Through testing multiple variations on values, how can you tell these were the most optimal?

The options with the most effect was the following, set to the respective values.

ANS : The meassure I used to guage the effect was processedRowsPerSecond and the highest value I observed was 301.5151515151515

spark.sql.shuffle.partitions 10

spark.streaming.kafka.maxRatePerPartition 10

spark.default.parallelism 10000