1b5fd0>", line 11 .getOrCreate() ^ IndentationError

+ Section

```
from pyspark.sql import SparkSession
import urllib.request

# Initialize Spark session
spark = SparkSession.builder \
    .appName("Word Count from Online Dataset") \
    .master("local[*]") \
    .getOrCreate()

# Online URL for sample text file
file_url = "https://raw.githubusercontent.com/dwyl/english-words/master/words.txt"

# Download the text file to local storage
file_path = "words.txt"
urllib.request.urlretrieve(file_url, file_path)

# Load the text file into an RDD
rdd = spark.sparkContext.textFile(file_path)

# Split each line into words and flatten the list of lists
words_rdd = rdd.flatMap(lambda line: line.split())

# Count the words (count the occurrences of each word)
word_counts = words_rdd.count()

# Print the result
print(f"Total number of words in the file: {word_counts}")

# Stop the Spark session
spark.stop()
```

```
Total number of words in the file: 466549
```

```
!pip install -q pyspark
!pip install -q opencv-python
from pyspark.sql import SparkSession
import cv2
from google.colab.patches import cv2_imshow

# Initialize Spark session
spark = SparkSession.builder.appName("VideoAnalysis").config("spark.driver.memory", "4

# Video URL (replace with the actual URL)
video_url = "https://www.learningcontainer.com/wp-content/uploads/2020/05/sample-mp4-f

# Initialize video capture from the URL
video_capture = cv2.VideoCapture(video_url)

# Check if video file opened successfully
if not video_capture.isOpened():
    print("Error: Could not open video.")
else:
    # Instead of collecting all frames at once, we'll process them in batches.
    def process_video_in_batches():
        batch_size = 10  # Number of frames to process in each batch
        batch_frames = []

        while True:
            ret, frame = video_capture.read()
            if not ret:
                break  # End of video

            batch_frames.append(frame)

            # When batch reaches batch_size, process it
            if len(batch_frames) == batch_size:
                # Parallelize frames in the batch
                rdd_frames = spark.sparkContext.parallelize(batch_frames)

                # Process the frames using PySpark
                processed_frames = rdd_frames.map(process_frame).collect()

                # Display processed frames (edges)
                for edge_frame in processed_frames:
                    cv2_imshow(edge_frame)

                # Reset the batch after processing
                batch_frames = []

        # Process any remaining frames that didn't form a full batch
        if batch_frames:
            rdd_frames = spark.sparkContext.parallelize(batch_frames)
            processed_frames = rdd_frames.map(process_frame).collect()
            for edge_frame in processed_frames:
                cv2_imshow(edge_frame)

    # Define a simple frame processing function
    def process_frame(frame):
        # Convert to grayscale
        gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        # Apply edge detection
        edges = cv2.Canny(gray_frame, 100, 200)
        return edges

    # Process video in batches
    process_video_in_batches()

# Release the video capture
video_capture.release()

# Stop the Spark session
spark.stop()
```