



**Capstone Summative**

big data with Pyspark

**PySpark Analysis of YouTube Video Dataset**

# Introduction

This project outlines a comprehensive analysis of the YouTube video dataset using PySpark. The dataset, named youtubedata.txt, includes crucial information about various videos such as video ID, uploader, upload date, category, length, views, ratings, comments, and related video IDs.

# Dataset Description

* Video ID (Column 1): Unique identifier for each video.
* Video Uploader (Column 2): The individual or entity responsible for uploading the video.
* Interval between YouTube Establishment and Video Upload (Column 3): Time duration between the establishment of YouTube and the date of video uploading.
* Video Category (Column 4): Categorization of the video content.
* Video Length (Column 5): Duration of the video in minutes or seconds.
* Number of Views (Column 6): Total views received by the video.
* Rating on the Video (Column 7): Rating assigned to the video.
* Number of Ratings (Column 8): The count of ratings given for the video.
* Number of Comments (Column 9): The count of comments posted on the video.
* Related Video IDs (Column 10): IDs of videos related to the uploaded video.

# Analysis and Insights

1. Find the average interval between YouTube establishment and video upload for each video category.

2. Identify the top three video uploaders with the highest average number of views for videos longer than 10 minutes.

3. Calculate the average rating and engagement (comments + ratings) for videos in each category.

4. Find the correlation between video length and the number of views.

5. Identify the most common related video IDs across all videos.

6. Find the average number of comments for videos with specific related video IDs.

7. Investigate whether videos with more comments tend to have higher ratings.

8. Identify the top video uploaders based on the number of views or ratings.

9. Evaluate the overall engagement level on the platform by analyzing the total number of views, ratings, and comments.

Feel free to analyse more insights!!!!

Data Engineering with Azure

**Dataset Attribute Information Attached:**

* first, middle and last names
* gender
* birth date
* Phone
* Salary
* Timestamp

**Problem Statement**: You are working as Azure data engineer & in your project you are getting files we're doing ETL on a file we've received from some customer. That file contains data about people as attribute shared above But, as is unfortunately common in data we get from this customer, the file contains some duplicate records. Worse:

* In Gender column “F” & “M” has been given that needs to replace f-> Female & M-> Male.
* Remove ‘\_’ from column header.
* Rename column phone with cellphone
* For timestamp column cast in timestamp & rename with DateTime

Your job is to remove the duplicate records. The specific requirements of your job are:

* Remove duplicates. It doesn't matter which record you keep; it only matters that you keep one of them.
* Preserve the data format of the columns. For example, if you write the first name column in all lower-case, you haven't met this requirement.
* Write the result as a Parquet file, as designated by destFile.

Azure Task:

Task 0:-> (20 Marks)

Design High level architecture diagram of data processing including all services, setup all azure services with your name. eg: RG\_Deeksha, ADLS\_Deeksha

Task 1:-> 60 marks

1. Setup connectivity with ADLS to ADB (10- marks)
2. Apply the above transformation in problem statement. (40-marks)
3. Move the dataset into Azure Datalake in parquet under refined container(ADLS\_yourname). (10-marks)

Task 2: 20 marks

1. Create a temp view on parquet file to show in SQL. (10 Marks)
2. Create a table on parquet file to show in SQL & perform visulization (10 marks)

**Note: Azure Services should have your name defined use data bricks & pyspark code for all logic**

Submission guidelines:

Take a screenshot of every step clearly in a word document, Mention your name and login ID at the top of the document. Capture step by Step Screenshot for each of the above task, convert Word doc as PDF and upload it on LMS.

EDA

Check the attached .ipynb file for the description of tasks.

SQL

Create the database using the create\_db\_ap.sql file and answers the following questions:

1. Retrieve two columns and a calculated value for a specific invoice.
2. Write a statement without a FROM clause that creates a row with given below columns: Starting Principal Starting principal of $50, 000 Interest 6.5% of the principal Principal Plus interest The principal plus the interest Add the expression used for the first two columnsto calculate the third column.
3. Write a statement that returns one row for each vendor in the Invoices table that contains given columns, vendor\_id, sum of the invoice\_total. Sort the result set in descending sequence by the payment total sum of each vendor.
4. Write a statement that returns one row for each general ledger account number that contains columns account\_description, count of the items in the invoice\_items table that have the same account\_number, sum of the line\_item\_amount columns in the invoice\_line\_items table that have the same account\_number. Return only those rows where the count of line items is greater than 1. Group the result set by the account\_description column and sort the result set in descending sequence by the sum of the line-item amount.
5. Which invoice have a payment total that’s greater than the average payment total for all invoices with a payment total greater than 0? Return the invoice\_number and invoice\_total columns for each invoice and Sort the result by the invoice\_total column in descending order.
6. Write a statement that returns that returns vendor\_id and the largest unpaid invoice for each vendor. Group the result set by the vendor\_id. Write another statement that uses the first SELECT statement that uses the first SELECT statement in its FROM clause. The main query should return a single value that represents the sum of the largest unpaid invoices for each vendor.
7. Write a statement that returns the name, city, and state of each vendor that’s located in a unique city and state. In other words, don’t include vendors that have a city and state in common with another vendor and Sort the results by the vendor\_state and vendor\_city columns.
8. Use a correlated subquery to return one row per vendor, representing the vendor’s oldest invoice (the one with the earliest date). Each row should include these four columns: vendor\_name, invoice\_number, invoice\_date, and invoice\_total. Sort the result by the vendor\_name columns