Analyzing the Online Course Learning Platforms -Analytical Performance using Databricks Platform Delta Lake for Learners

* **Databricks**: is a unified analytics platform that combines big data processing and machine learning capabilities. It is built on top of Apache Spark and provides a collaborative environment for data scientists, data engineers, and business analysts to work together on large-scale data processing and analytics. Databricks simplifies the process of managing and analyzing data by offering an integrated workspace with tools for data exploration, visualization, and model development. It is widely used for processing and deriving insights from large datasets in a scalable and collaborative manner.
* **Online Course Education:** It involves learning through digital platforms, enabling students to access educational content, lectures, and assessments over the internet. It provides flexibility in scheduling, allowing individuals to pursue education from anywhere with an internet connection. Online courses cover a wide range of subjects and are often self-paced, catering to diverse learning preferences.
* **Course Data Analysis:** A course in data analysis equips students with the skills to examine and interpret complex datasets. Participants learn statistical methods, programming languages like Python or R, and data visualization techniques to extract meaningful insights from raw information.
* **Student Outcomes:** refer to the observable and measurable results of a student’s educational experience, reflecting their achievements, skills, and knowledge acquired during a specific period. These outcomes assess the effectiveness of educational programs in preparing students for future challenges and success. Examples include academic performance, critical thinking abilities, and career readiness.
* **Online Course Comparison Techniques:** using Databricks involve leveraging the platform’s advanced analytics capabilities to assess and contrast various online courses. Databricks facilitates efficient data processing, exploration, and visualization, enabling users to analyze course attributes such as content quality, engagement metrics, and student outcomes for comprehensive comparisons. Its integrated features streamline the evaluation process, making it a powerful tool for educators and learners seeking data-driven insights into online learning experiences.

PERFORMANCE EVALUATION AND PREDICTION

## Bronze Layer

The provided code snippet serves as a crucial component of a research paper’s data preparation process, particularly when dealing with external data sources and credentials. It outlines the steps involved in setting up data access credentials, extracting pertinent information, and preparing the data for analysis. Below, we break down the code into explanatory paragraphs for inclusion in a research paper:

**Data Access and Preparation**

In many research papers, access to external data sources is essential for conducting thorough and data-driven analyses. This code snippet represents the initial stages of data access and preparation.

Its primary objective is to configure and extract credentials necessary for accessing external data, which is common in research articles that rely on data from various sources.

**Creating Data Directories**

The code starts by creating a directory structure using the ‘dbutils.fs.mkdirs‘ command. This directory structure, located at "/FileStore/shared uploads/ms1485@srmist.edu.in/kaggle," serves as a storage location for relevant data and configuration files. In a research context, this step is akin to setting up a dedicated workspace for data-related activities, ensuring proper organization and accessibility.

**Accessing Credentials**

The pivotal part of this code is the extraction of credentials required for data access. It reads a JSON file located at "/FileStore/shared [uploads/ms1485@srmist.edu.in/](mailto:uploads/ms1485@srmist.edu.in/) kaggle/kaggle.json" using the ‘spark.read.format(’json’)‘ command.

This JSON file typically contains sensitive information such as usernames and access keys necessary for authenticating and accessing external data sources. In research papers, this step is vital for transparency and reproducibility, as it clearly outlines how external data is accessed and used.

**Displaying Credentials**

The subsequent step involves displaying the contents of the JSON file using the ‘display(spark credentials df)‘ command. This serves as a transparency measure, allowing researchers to verify the correctness and security of the credentials being used in the analysis. It is important in research articles to provide a clear view of the data access process for the benefit of readers and peer reviewers.

**Extracting and Storing Credentials**

Finally, the code extracts specific credentials from the loaded JSON file. The Figure [3](#_bookmark16) retrieve the username and access key from the JSON file and store them as variables. These credentials are essential for subsequent data retrieval and should be stored securely. %sh pip install kaggle, is used to install the ”kaggle” Python package using the pip package manager. After installing the Kaggle, the following steps should follow [[7](#_bookmark34)].

The Figure [3](#_bookmark16) explains how Databricks, managing and securely storing credentials can be crucial for accessing external data sources or services

**Kaggle Authentication**

In many research papers within the field of data science and machine learning, access to external data sources and platforms is paramount. Kaggle is a prominent platform that hosts a wealth of datasets and data science competitions [[8](#_bookmark35)]. This code snippet serves as a fundamental step in the research process, enabling researchers to authenticate themselves on Kaggle programmatically and access the platform’s resources.

**Setting Environment Variables**

The code begins by setting environment variables using the ‘os.environ‘ dictionary. Specifically, it sets the ‘KAGGLE USERNAME‘ and ‘KAGGLE KEY‘ environment variables to the provided ‘user name‘ and ‘key‘ parameters, respectively. These variables are used by the Kaggle API client to identify and authenticate the user. In a research context, this step ensures that user credentials are securely managed and passed to the Kaggle API.

**Importing Kaggle API Client**

Next, the code imports the necessary modules from the Kaggle API library, specifically the ‘KaggleApi‘ class from ‘kaggle.api.kaggle api extended‘. This class provides the functionality required to interact with Kaggle programmatically, allowing users to access datasets, submit competition entries, and more. Importing this library is a crucial step in the research process as it establishes the connection to Kaggle.

**Authentication**

The subsequent part of the code initiates the Kaggle API client by creating an instance of the ‘KaggleApi‘ class, denoted as ‘api‘. It then calls the ‘authenticate()‘ method on this instance. This method uses the environment variables set earlier (‘KAGGLE USERNAME‘ and ‘KAGGLE KEY‘) to authenticate the user with Kaggle’s servers. Successful authentication is essential for gaining access to Kaggle’s extensive resources.

**Data Acquisition from Kaggle**

In data-driven research papers, acquiring relevant datasets is often a critical step in the research process. This code snippet exemplifies the initial stages of data acquisition from Kaggle, a well-known platform for datasets and data science competitions. The overarching goal is to facilitate seamless access to external data that can be used for analysis and experimentation.

**Authentication for Kaggle**

The code begins by invoking the ‘autunticate kaggle(user name,key)‘ function, which is assumed to have been defined elsewhere in the research environment. This function serves the purpose of authenticating the user with their Kaggle credentials, allowing them to access Kaggle datasets programmatically. Authentication is an essential step as it ensures that the user has the necessary permissions to download datasets from Kaggle.

**Downloading the Dataset**

The subsequent part of the code initiates the download of a specific dataset from Kaggle using the ‘kaggle datasets download‘ command. In this case, the dataset titled **”2020-udemycourses-dataset”** is being retrieved. Data acquisition from external sources is a common practice in research, especially when dealing with real-world data. Researchers must specify the dataset they intend to use and its source.

**Creating a Storage Directory**

To manage the downloaded dataset, the code creates a directory within the Databricks File System (DBFS) using the ‘dbutils.fs.mkdirs‘ command. The directory, named ’udemy mount,’ is established within the DBFS to ensure that the dataset is stored in a designated and organized location. Proper data organization is crucial for maintaining data integrity and facilitating subsequent analysis.



**Figure (2)** Cluster Attach Configuration



**Figure (3)** Extracting and Storing the user credentials

**Moving the Downloaded Dataset**

The code then uses the ‘dbutils.fs.mv‘ command to move the downloaded dataset from its original location (retrieved from Kaggle) to the ’udemy mount’ directory in the DBFS. This step ensures that the dataset is securely stored and easily accessible within the research environment. Moving and organizing data is a fundamental aspect of data management in research, as it ensures that data is stored in a structured and accessible manner.

**Defining File Paths**

The final part of the code defines file paths that researchers can use to reference the downloaded dataset. The ‘kaggle path‘ variable represents the path to the dataset within the original Kaggle environment, while the ‘file path‘ variable represents the path to the dataset within the DBFS. These file paths are important for researchers to locate and access the dataset for subsequent analysis and experimentation. In conclusion, the Python code exemplifies the essential steps involved in acquiring, organizing, and setting up data from an external source, specifically Kaggle, within a research environment.

**Dataset Extraction**

Data acquisition often involves downloading datasets in compressed formats like ZIP to reduce file size for efficient storage and transfer. In research papers, handling these compressed archives is a common requirement. This code snippet represents a crucial phase in the data preparation process, where researchers work to extract, organize, and store dataset contents.

**Extracting the ZIP Archive**

The code begins by using the ‘zipfile.ZipFile‘ module to open the ZIP archive located at ’/databricks/driver/2020-udemy-courses-dataset.zip’. The ‘with‘ statement ensures proper resource management and context handling. Inside the ‘with‘ block, the ‘zp.extractall()‘ method is employed to extract all contents of the ZIP file into a directory denoted by the ‘/‘ format string. This directory corresponds to the ’kaggle path’ variable, which is a critical element in organizing and referencing the dataset.

**Moving Extracted Files**

Following the extraction of dataset contents, the code utilizes the ‘dbutils.fs.mv()‘ command to move the extracted files from their original location (defined as ’file://’) to a specified storage location represented by the ’file path’ variable. This step ensures that the dataset is securely stored within the research environment for subsequent analysis and experimentation. Proper file management is essential to maintain data integrity and facilitate access.

**Removing the Original ZIP Archive**

Finally, the code includes the ‘dbutils.fs.rm()‘ command, which is used to remove the original ZIP archive file, ’/databricks/driver/ipl-2017.zip’. This step is important as it helps declutter the research environment and ensures that unnecessary files are not retained. Data management in research involves maintaining a clean and organized workspace. In conclusion, the Python code illustrates the critical steps involved in handling a downloaded ZIP archive of a dataset within the context of a research paper. The beautifulsoup4,” is a fundamental instruction within a research paper’s methodology section, specifically in the context of web scraping [16](#_bookmark17) or data extraction. It outlines the process of installing the ”beautifulsoup4” Python library using the pip package manager. It focuses on the initial stages of data loading and organization, particularly when working with multiple CSV files that represent various categories of data [[7](#_bookmark34)].

**Data Loading and Organization**

In data-driven research, the acquisition and organization of data are foundational steps that significantly impact the quality and efficiency of subsequent analyses. This code snippet showcases the process of loading multiple CSV files that correspond to different categories of data. Such organization is crucial in research papers that involve the analysis of diverse datasets, each representing a distinct domain or topic.

**Listing Available Data Files**

The code begins by displaying a list of available data files in a specified directory using the ‘dbutils.fs.ls(’dbfs:/FileStore/udemy mount/’)‘ command. This listing provides transparency and a clear view of the datasets that will be utilized in the research. It is essential in research papers to document the source and availability of data for the benefit of readers and peer reviewers.

**Loading CSV Data**

The subsequent part of the code involves loading CSV data files into the research environment using Apache Spark’s capabilities. Specifically, it uses the ‘spark.read.option(’header’, True).csv‘ method to read each CSV file. The datasets cover various domains, including technology, business [[1](#_bookmark27)], finance, office productivity, lifestyle, music, design, marketing, and photography. This step is crucial as it brings raw data into a structured format that can be analyzed. It ensures that data is ready for manipulation and exploration.

**Data Categorization**

The code assigns each loaded dataset to a respective variable with a meaningful name, such as ‘df tech raw‘, ‘df business raw‘, ‘df finance raw‘, and so on. This categorization is essential for maintaining data integrity and clarity within the research environment. Researchers can easily reference and analyze specific datasets based on their research objectives and domain focus. It highlights the importance of transparency in data source documentation and proper categorization for effective data management.

**Data Categorization**

In data-driven research, particularly when dealing with multiple datasets from different domains or categories, it is essential to categorize and label the data appropriately. This code snippet addresses this requirement by adding a new categorical column named ’category’ to each dataset. This additional column helps researchers identify and differentiate datasets based on their respective domains.

**Assigning Categories to Datasets**

The code sequentially processes each dataset, using the ‘withColumn‘ method to add a new column named ’category’ to each DataFrame. The ‘lit‘ function is employed to assign a specific category label to each dataset. Categories such as ’tech,’ ’business,’ ’finance,’ ’office,’ ’lifestyle,’ ’music,’ ’design,’ ’marketing,’ and ’photography’ are assigned to the corresponding datasets based on their domain or subject matter. This step ensures that each dataset is associated with its appropriate category, making it easier for researchers to perform targeted analyses.

**Facilitating Targeted Analyses**

The ’category’ column added to each DataFrame enables researchers to perform targeted analyses within specific domains or categories. For instance, if a research question pertains to the ’finance’ domain, researchers can filter the dataset with the ’finance’ category, focusing their analysis on relevant data. This functionality simplifies the research process and allows for more precise investigations. The data pre-processing steps related to categorizing datasets by domain or category. It underscores the importance of data organization and interpretability in data-driven research.

**Data Integration and Consolidation**

In research papers involving diverse datasets from various domains or categories, it is essential to consolidate the data for comprehensive analysis. This code snippet showcases the process of unifying multiple DataFrames representing categories such as ’business,’ ’finance,’ ’office productivity,’ ’lifestyle,’ ’music,’ ’design,’ ’marketing,’ ’photography,’ and ’tech’ into a single DataFrame named ’src final df.’ The ’union’ operation is used iteratively to combine these DataFrames, creating a unified dataset that encompasses data from all subject areas.

**Ensuring Data Integrity**

The ’src final df.count()’ command is employed to verify the total number of records in the consolidated DataFrame. Ensuring data integrity and validating the correctness of the consolidation process are critical steps in research data preparation. Researchers need to confirm that no records have been lost or duplicated during the union operation.

**Visualization for Data Inspection**

The ’display(src final df)’ command is included to provide researchers with a visual representation of the consolidated dataset. This visualization facilitates data inspection and preliminary exploration. It enables researchers to gain insights into the combined dataset’s structure and content, aiding in subsequent research tasks.

**Data Storage and Accessibility**

To facilitate data storage and accessibility, the code leverages the Databricks File System (DBFS) to remove any existing data at the specified location, ’dbfs:/mini project/DataLake/bronze/udemy1,’ ensuring a clean slate for saving the consolidated dataset. The ’src final df’ is then written to the same location in Parquet format. Additionally, the dataset is registered as a table, ’bronze.udemy1,’ in the Databricks environment [17](#_bookmark18) using the ’saveAsTable’ and ’spark.sql’ commands. This step makes the data readily available for SQL-based querying and analysis [[9](#_bookmark36)].

**Data Retrieval and Structuring**

The code begins by retrieving the previously saved dataset from the ’bronze.udemy1’ table, which represents a consolidated collection of online course data spanning various domains. The retrieved data is then displayed to provide researchers with a visual overview of the dataset’s structure and content, facilitating initial exploration and understanding.

**Data Cleaning and Transformation**

In the pursuit of data quality and consistency, the code proceeds with the removal of any existing data at the location ’dbfs:/mini project/DataLake/silver/udemy.’ This step ensures that the subsequent dataset storage is performed on a clean slate. Following data cleaning, the code performs several critical data transformations. Firstly, it casts the ’Enrollment’ col umn to an integer type, ensuring that enrollment numbers are treated as numerical values for analysis.

Next, the code removes any rows containing missing values (’null’ values) using the ’dropna’ function. This step enhances data quality by eliminating incomplete records [[9](#_bookmark36)].

**Web Scraping for Additional Information**

One of the code’s distinctive features is its integration with web scraping techniques. It utilizes the ’requests’ library and ’BeautifulSoup’ to fetch additional data from the provided URLs. Information such as instructor details, written ratings, total students, latest enrollments, total courses, instructor ratings, and more is extracted from the web pages and appended to the dataset.

**Data Extraction and Parsing / Data Type Conversion and Formatting**

The code then employs user-defined functions (UDFs) to extract and parse specific pieces of information from the extracted web content. These UDFs facilitate the extraction of numeric values and structured data, such as ratings, enrollments, and last modified dates, from unstructured text.To ensure uniformity and compatibility in the dataset, the code further performs data type conversions and formatting. For instance, the ’written rating’ and ’latest enroll’ columns are converted to integers, and the ’last modified’ column is formatted to extract the month and year of the last modification date.

**Schema Specification and Data Storage**

The code concludes by specifying a schema for the dataset based on predefined data types and storing the refined dataset as ’silver.udemy all string’ in Parquet format. This schema specification enhances data consistency and facilitates future data access and analysis.

**Data Filtering**

Lastly, the code filters the dataset to include only records where the ’written course rating’ exceeds a threshold of 4, effectively selecting high-quality courses based on ratings.

## Silver Layer

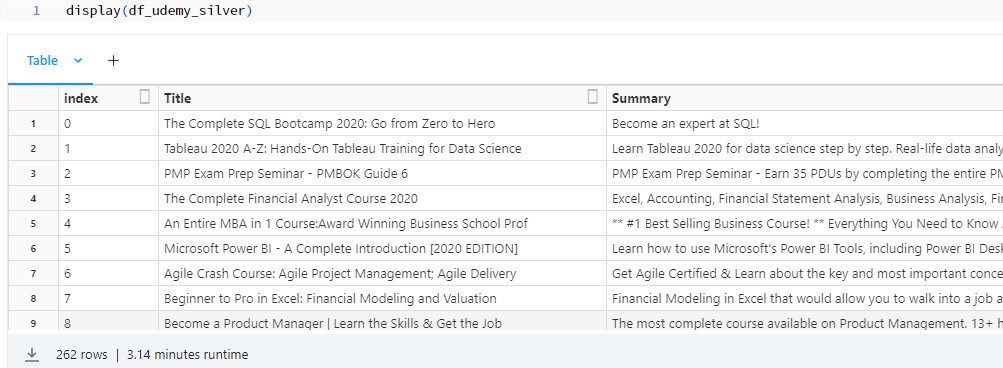
This layer focuses on performing various SQL-based data analysis operations on the ’silver.udemy2’ dataset, which represents refined and enriched online course data. These operations are instrumental in extracting valuable insights and patterns from the dataset, which can be used in the research paper. The key components of the github code are:

**Data Retrieval and Exploration**

The Figure [4](#_bookmark19) display the udemy data, code begins by reading the ’silver.udemy2 into a Spark DataFrame enabling to work with the cleaned and structured dataset. It also uses the ’REFRESH TABLE’ command to ensure the dataset is up to date.

## Analysis 1: Identifying Top-Enrolled Courses

The first SQL query aims to identify the top 10 courses that have experienced the highest increase in enrollments over time. It calculates the increase in enrollments by subtracting the initial enrollment from the latest enrollment for each course and orders the results in descending order. This analysis can help researchers pinpoint courses that have gained significant popularity.



**Figure (4)** Display Udemy Data

## Analysis 2: Enrollment Increase by Category

The second SQL query aggregates the data to calculate the total enrollment increase for each course category. It groups the data by category and computes the sum of enrollment increases. This analysis provides insights into which course categories have witnessed the most substantial growth in enrollments.

## Analysis 3: Top Instructors by Category

The third SQL query identifies the top five instructors within each category based on the latest enrollments. It uses the ’row number()’ function to assign a rank to each instructor’s latest enrollment within their respective categories. This analysis allows researchers to recognize instructors who are particularly popular in specific course categories.

Grouping by Category: The primary operation is performed on the ‘df udemy silver‘ DataFrame. It uses the ‘groupBy‘ method to group the data by the ”category” column. This means that the data will be divided into distinct groups based on the unique values found in the ”category” column.

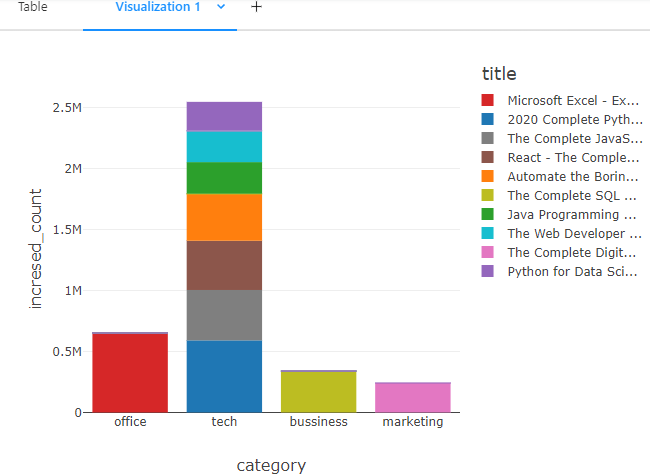
## Analysis 4: Top Instructors by Total Enrollments

The fourth SQL query determines the top instructors in each category based on their total enrollments. It calculates the total enrollment for each instructor within their category and ranks them accordingly. This analysis highlights instructors who have accumulated the highest total enrollments over time.

Aggregation with ‘agg‘: After grouping by category, the ‘agg‘ method is applied to calculate two aggregate values for each category: ‘sum("Enrollment").alias("total enrollments")‘: This calculates the total sum of the ”Enrollment” column for each category and assigns an alias ”total enrollments” to the result. ‘avg("Stars").alias("avg stars")‘: This computes the average of the ”Stars” column for each category and assigns an alias ”avg stars” to the result.

**Data Presentation**

Lastly, the code generates a report DataFrame that includes the titles, categories, and ratings of the top 10 courses based on their ratings. This report provides a concise summary of the highest-rated courses in the dataset. Finally, the ‘display‘ function is called to present the resulting DataFrame ‘report df‘. This DataFrame contains the calculated total enrollments and average star ratings for each category.

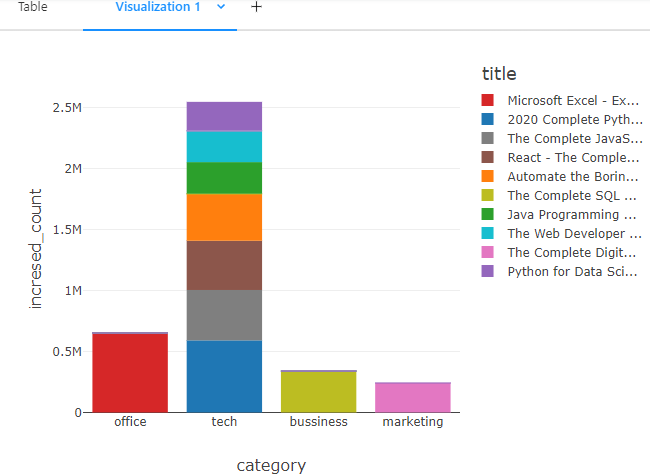


**Figure (5)** Top 5 courses which is enrolled high in few years

RESULTS AND DISCUSSIONS

## Gold Layer

1. ***Top 5 courses which is enrolled high in few years:*** The Figure [5](#_bookmark20) is designed to identify and display the top 10 courses that have experienced the highest increase in enrollments over a certain period, presumably in recent years.
   * Data Selection: The code selects five columns from a table named "silver.udemy2": ”link,” ”title,” ”enrollment,” ”category,” and ”latest enroll.” These columns likely contain information about Udemy courses, including their titles, enrollments, categories, and possibly links to the courses.
   * Calculation of Enrollment Increase: Within a subquery, a new column named "increased count" is created. This column calculates the increase in enrollments for each course. It’s computed by subtracting the ”enrollment” (representing an earlier enrollment count) from "latest enroll" (representing a more recent or current enrollment count). This calculation captures how much the enrollment count has grown for each course.
   * Sorting by Enrollment Increase: The main query orders the results in descending order based on the ”increased count” column. This ensures that the courses with the highest increases in enrollments are listed at the top.
   * Limiting Results: The code includes a ‘LIMIT 10‘ clause, which restricts the output to the top 10 courses with the highest enrollment increases. The code is useful for identifying and showcasing courses on Udemy that have experienced significant growth in enrollments over a specified period.
2. ***Categorywise enrolment report:*** The provided PySpark code is used to generate a category-wise enrollment report from a DataFrame named ‘df udemy silver‘. This code calculates the total enrollments and the average star ratings for courses within each category and then displays the results using the ‘display‘ function. The Figure [6](#_bookmark21) the categorywise enrolment report.
   * Importing Necessary Functions: The code starts by importing the necessary functions



**Figure (6)** Categorywise enrolment report

from the ‘pyspark.sql.functions‘ module. Specifically, it imports the ‘avg‘ function for calculating the average and the ‘sum‘ function for summing up enrollment number [[3](#_bookmark31)]s.

* + Grouping and Aggregation: The ‘groupBy‘ method is applied to the ‘df udemy silver‘ DataFrame, grouping the data by the ”category” column. This groups the courses into categories, allowing for category-wise analysis.

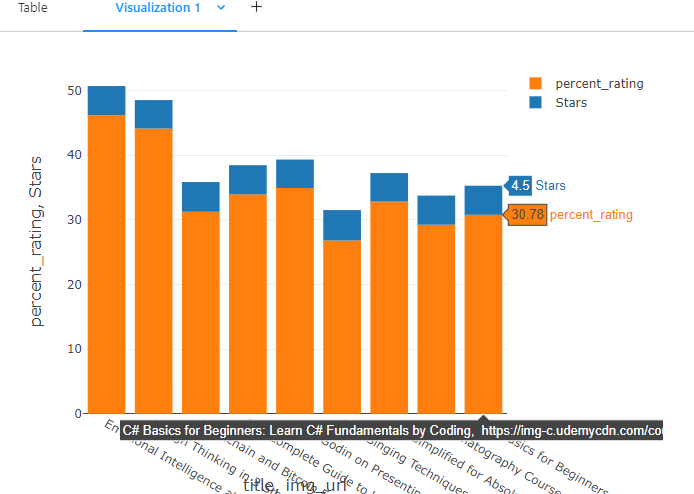
Two aggregation functions are applied to the grouped data: ‘sum("Enrollment").alias("total enrollments")‘: This calculates the total sum of enrollments for all courses within each category and assigns the result to a new column named ”total enrollments.” ‘avg("Stars").alias("avg stars")‘: This calculates the average (mean) star rating for all courses within each category and assigns the result to a new column named ”avg stars.”

* + Displaying the Report: The ‘display‘ function is used to visualize the resulting DataFrame ‘report df‘, which now contains the category-wise total enrollments and average star ratings. It helps analyze and present category-wise statistics for Udemy courses, including the total enrollments and average star ratings within each category.

## Top rated and most percentage user ratings given:

The provided PySpark code is designed to identify and display the top-rated courses within each category, along with the courses that have received the highest percentage of user ratings, from a DataFrame named ‘df udemy silver‘.

* + Data Selection and Calculation: The code starts by selecting relevant columns from the ‘df udemy silver‘ DataFrame. It chooses columns such as ”title,” ”Enrollment,” ”Rating,” ”Stars,” ”category,” and "img url" for further analysis. It then calculates the "percent rating” for each course, which represents the percentage of user ratings relative to the total number of enrollments. This calculation is rounded to two decimal places.
  + Window Specification for Ranking: A window specification named ‘window spec‘ is created using the ‘Window‘ function. It partitions the data by the ”category” column and orders the rows within each category by the ”percent rating” column in descending



**Figure (7)** Top rated and most percentage user ratings given

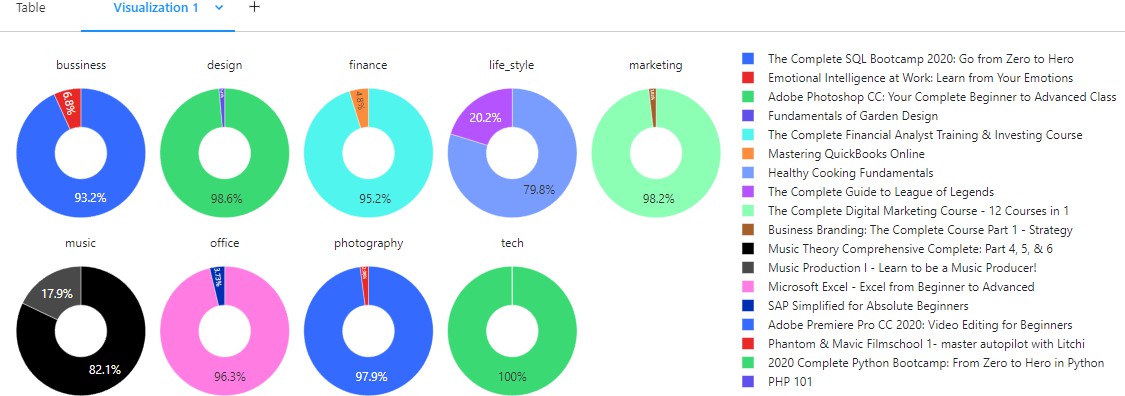
order. This specification is crucial for ranking courses within their respective categories.

* + Ranking Courses: The DataFrame ‘re‘ is augmented with a new column called ”rank” using the ‘row number()‘ function over the defined window specification. This assigns a rank to each course within its category based on the ”percent rating” in descending order.
  + Filtering Top Courses: The ‘df with ranks‘ DataFrame is filtered to include only those rows where the ”rank” is less than or equal to 1. This ensures that only the top-rated courses and courses with the highest percentage of user ratings within each category are retained.
  + Displaying Results: Finally, the‘display‘ function is used to visualize the resulting DataFrame ‘df top 5‘, which contains the top-rated courses and courses with the highest percentage of user ratings within each category. The Figure [7](#_bookmark22) is used to find and display the best-performing courses within each category based on user ratings and the percentage of users who have rated the course. It can be valuable for learners and researchers looking for high-quality courses and assessing user satisfaction on the Udemy platform.

## Course performance top and bottom courses categorywise:

The provided PySpark code is used to identify and display both the top-performing and bottom-performing courses within each category based on enrollment numbers from a DataFrame named ‘silver.udemy2‘. Here’s a brief explanation of the code:

* Data Loading: The code starts by loading data from a table named ”silver.udemy2” using the Spark session. This table likely contains information about Udemy courses.
* Window Specification for Ranking: It defines a window specification ‘w‘ using the ‘Window‘ function. The data is partitioned by the ”category” column, and the rows within each category are ordered by enrollment numbers (”Enrollment”) in descending order. This window specification is essential for ranking courses within their respective categories based on enrollment.
* Ranking Courses: Two new columns are added to the DataFrame ‘df‘ using the ‘withColumn‘ method: ‘enrollment rank desc‘: This column assigns a rank to each course within its category based on descending enrollment numbers. It represents the rank of



**Figure (8)** Course performance top and bottom courses categorywise

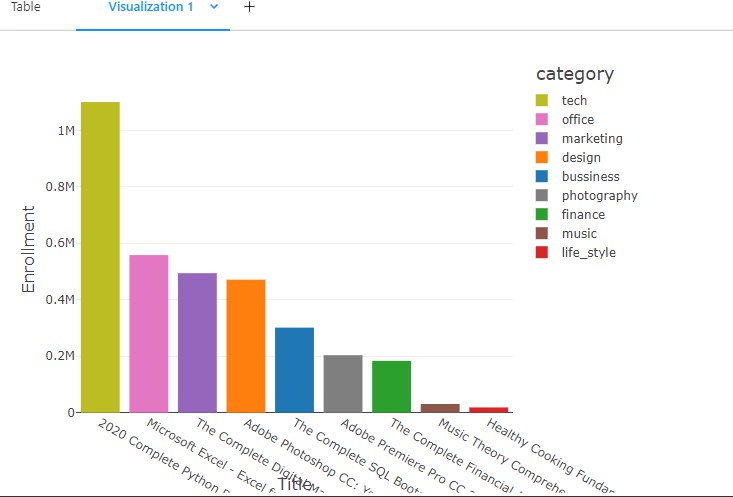
courses with the highest enrollments within each category. ‘enrollment rank asc‘: This column assigns a rank to each course within its category based on ascending enrollment numbers. It represents the rank of courses with the lowest enrollments within each category.

* Filtering Top and Bottom Courses: The DataFrame is then filtered to include only those rows where either ‘enrollment rank desc‘ is less than or equal to 1 (top courses) or ‘enrollment rank asc‘ is less than or equal to 1 (bottom courses). This step effectively selects both the top-performing and bottom-performing courses within each category based on enrollment.
* Sorting by Enrollment: Finally, the DataFrame is sorted by enrollment numbers in descending order (‘df[’Enrollment’].desc()‘). This ensures that the top courses within each category are listed first, followed by the bottom courses.
* Displaying Results: The ‘display‘ function is used to visualize the resulting DataFrame ‘df‘, which contains both the top-performing and bottom-performing courses within each category based on enrollment. The code is useful for identifying and displaying course performance within different categories on the Udemy platform. The Figure [8](#_bookmark23) enables users to see which courses are leading in terms of enrollment within their respective categories and which courses have relatively lower enrollments. Researchers and learners can use this information to understand enrollment trends and preferences in various educational fields on Udemy.

## Improvement percentage:

The provided SQL code [[3](#_bookmark31)] is used to calculate the ”improvement percentage” for courses in a dataset, specifically Udemy courses in the ”silver.udemy2” table. Here’s a brief explanation of the code:

* Subquery with Calculation: The code starts with a subquery. Within the subquery, it selects three columns: ”title,” ”stars,” and "course rating" (aliased as "latest rating"). These columns likely represent information about Udemy courses, including their titles, current star ratings, and some form of ”latest rating.”
* Calculation of Improvement: A new column named ”improvement” is created in the subquery. This column calculates the improvement or change in the course rating. It does this by subtracting the ”stars” (presumably representing an older or previous rating) from the "latest rating" (representing a more recent or current rating).
* Sorting by Improvement: The main query orders the results in ascending order based on the ”improvement” column. This means that courses with the most significant improvement in ratings (those with positive ”improvement”) will be listed first. The Figure [10](#_bookmark25) is used to identify courses that have experienced improvements in their ratings over time. It calculates the difference between the latest rating and an older rating (presumably representing the course’s earlier performance) and sorts the courses in ascending order of this



**Figure (9)** Top5 courses based on enrolment in both categories

A screenshot of a computer

Description automatically generated

**Figure (10)** Improvement Percentage

improvement. The code helps users or analysts identify which courses have shown the most positive changes in their ratings, potentially indicating enhancements or improvements in course quality.

1. ***Giving suggestions based on user interest:*** The provided PySpark code is designed to provide course suggestions to users based on their search interests. It takes user input, searches for courses with titles matching the input string, and recommends the top 5 courses with the highest recent enrollments. Here’s a brief explanation of the code:

* User Input: The code starts by taking user input using the ‘input()‘ function. Users are prompted to enter a search string indicating their interests or preferences for courses they would like to discover on Udemy.
* Filtering by User Interest: After obtaining the user’s input, the code applies a filtering operation on the ‘df udemy silver‘ DataFrame. It uses the ‘lower()‘ function to convert the ”title” column to lowercase and then checks if it contains the user’s search string using the ‘like()‘ function with a wildcard (‘%‘) to match any part of the title.
* Sorting by Enrollment: The filtered DataFrame, ‘result df‘, is then sorted in descending order based on the "latest enroll" column. This likely represents the latest enrollment numbers for each course, with courses having the highest recent enrollments appearing at the top.
* Limiting Results: To provide concise recommendations, the Figure [9](#_bookmark24) explains the results to

A screenshot of a computer

Description automatically generated

**Figure (11)** Giving sugestions based on user interest

the top 5 courses that match the user’s interests and have the highest recent enrollments. This ensures that users receive a focused list of popular courses related to their search.

* Displaying Results: The ‘display‘ function is used to visualize the resulting DataFrame ‘result df‘, which contains the top 5 course recommendations based on the user’s search string and recent enrollments. The code is a simple example of a course recommendation system based on user input and recent enrollment data. The Figure [11](#_bookmark26) helps users discover popular courses that align with their interests, providing a personalized learning experience. Researchers and developers can use this code as a starting point for building more sophisticated recommendation systems in online education platforms like Udemy. Our work has been posted in the github for execution.

[18](#_bookmark29). Download the GitHub repository containing the files named NB INITIALIZE.dbc, NB BRONZE UDEMY.dbc, NB SILVER UDEMY.dbc, NB GOLD UDEMY.dbc files. Afterward, import these files into your Databricks environment and attempt to execute them independently on your own.