**Project Overview**

This project involves collecting, processing, and analyzing course data to create personalized recommendation systems. The process begins with web scraping to gather course information and user comments. The data is then stored in a structured format and analyzed through Exploratory Data Analysis (EDA) to uncover insights. Using machine learning models, including Collaborative Filtering and Content-Based Filtering, courses are recommended based on user behavior and course content. Sentiment analysis enhances recommendations by factoring in the emotional sentiment of user comments. The goal is to provide personalized, data-driven insights for course selection and engagement.

#### **Step 1: Data Collection (Web Scraping)**

**Phase 1: Initial Scraping from Coursera**

* We began by scraping data from Coursera, focusing initially on extracting information displayed on course cards.
* The scraping process was divided into two phases:
  1. **Card Information Extraction:** Collected basic course details such as title, partner, rating, reviews, description, and course link.
  2. **Detailed Information Extraction:** Using the URLs collected in the first phase, we fetched additional details like the number of enrollments.
* However, the scraped data from Coursera proved insufficient for the project’s goals. The final output included columns:
  1. Title, Partner, Rating, Reviews, Description, Course Link, Enrollment.
* The limited scope of this dataset led us to explore alternative platforms.

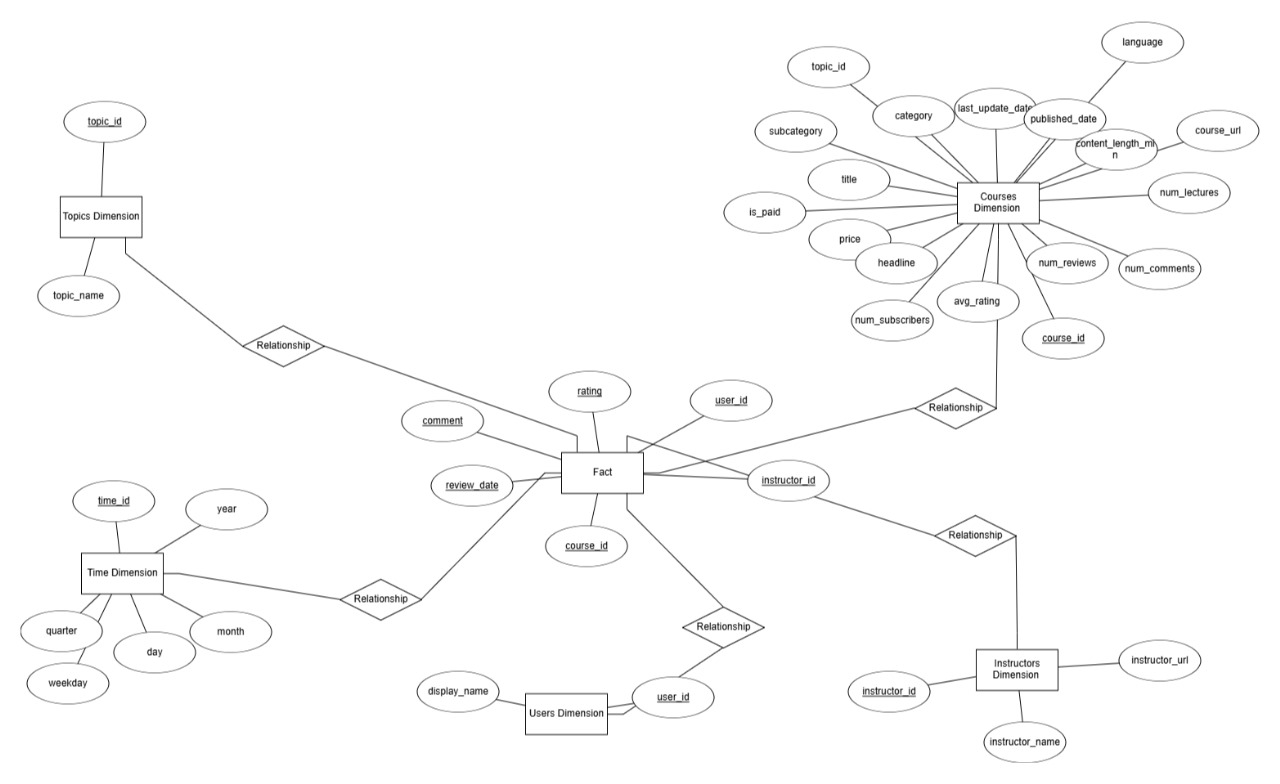
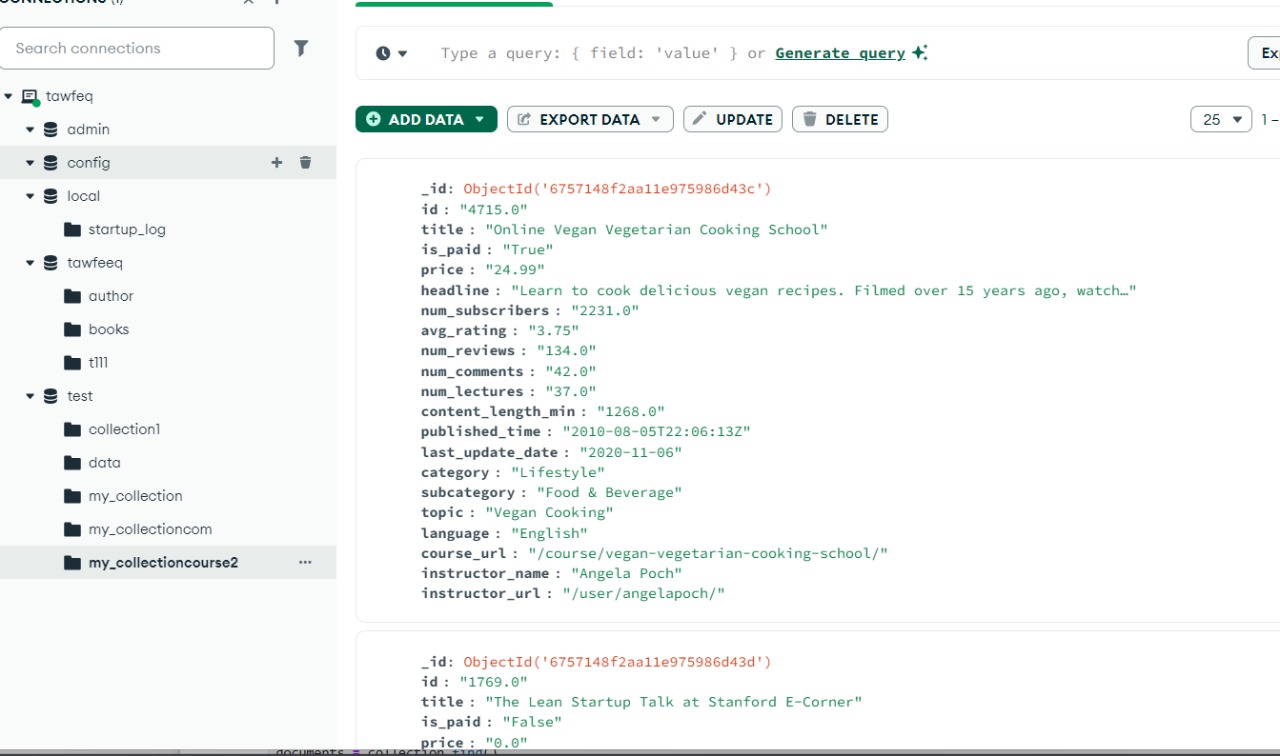
**Phase 2: Transition to Udemy**

* We redirected our efforts to Udemy, which offered more comprehensive and detailed course information suitable for our project targets and models.
* From Udemy, we successfully extracted two datasets:
  1. **Course Information (Course\_info.csv)**
     + Columns: id, title, is\_paid, price, headline, num\_subscribers, avg\_rating, num\_reviews, num\_comments, num\_lectures, content\_length\_min, published\_time, last\_update\_date, category, subcategory, topic, language, course\_url, instructor\_name, instructor\_url.
  2. **Comments (Comments.csv)**
     + Columns: id, course\_id, rate, date, display\_name, comments.

**Libraries Used**

* **Libraries:**
  + **Selenium:** For automated browser interaction and dynamic data extraction.
  + **BeautifulSoup:** For parsing and extracting static HTML content.
  + **Requests:** For making HTTP requests to fetch web pages.
  + **Threading:** To improve scraping efficiency by enabling concurrent data extraction.

**Step 2: Data Storage and Design**

* **ERD**:
* **Store it in Mongodb**
* **We also used SQL server database for storing data and apply complex joins**
* **Libraries used : MongoDB - SQLite**

**Step 3: Data Preprocessing and EDA**

**1-Course Dataset - EDA**

## **Price distribution:** Udemy Courses Price and Category-Wise Distribution

### Price Distribution

# The majority of Udemy courses are priced between 0and200. Among the courses:

* **Free Courses**: 21,738 courses (approximately 10% of total courses).
* **Paid Courses**: 187,996 courses (approximately 89.64% of total courses).

For paid courses:

* Nearly **20%** of the courses are sold at **$19.99, and about 8.34% of the courses are sold at $199.99.**

### Visualizations

#### Share of Free and Paid Courses

The count plot below visualizes the distribution of free and paid courses, showing a significant majority of courses are in the paid category.

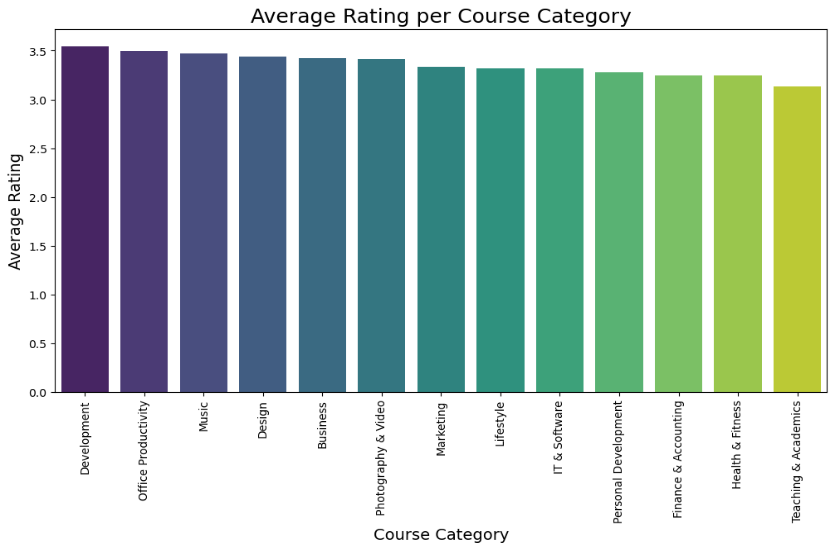
#### Category-Wise Price Distribution

The boxplot displays the price distribution of courses across different categories, providing insights into pricing trends and variations for each category.

**Course Rating in Udemy:**

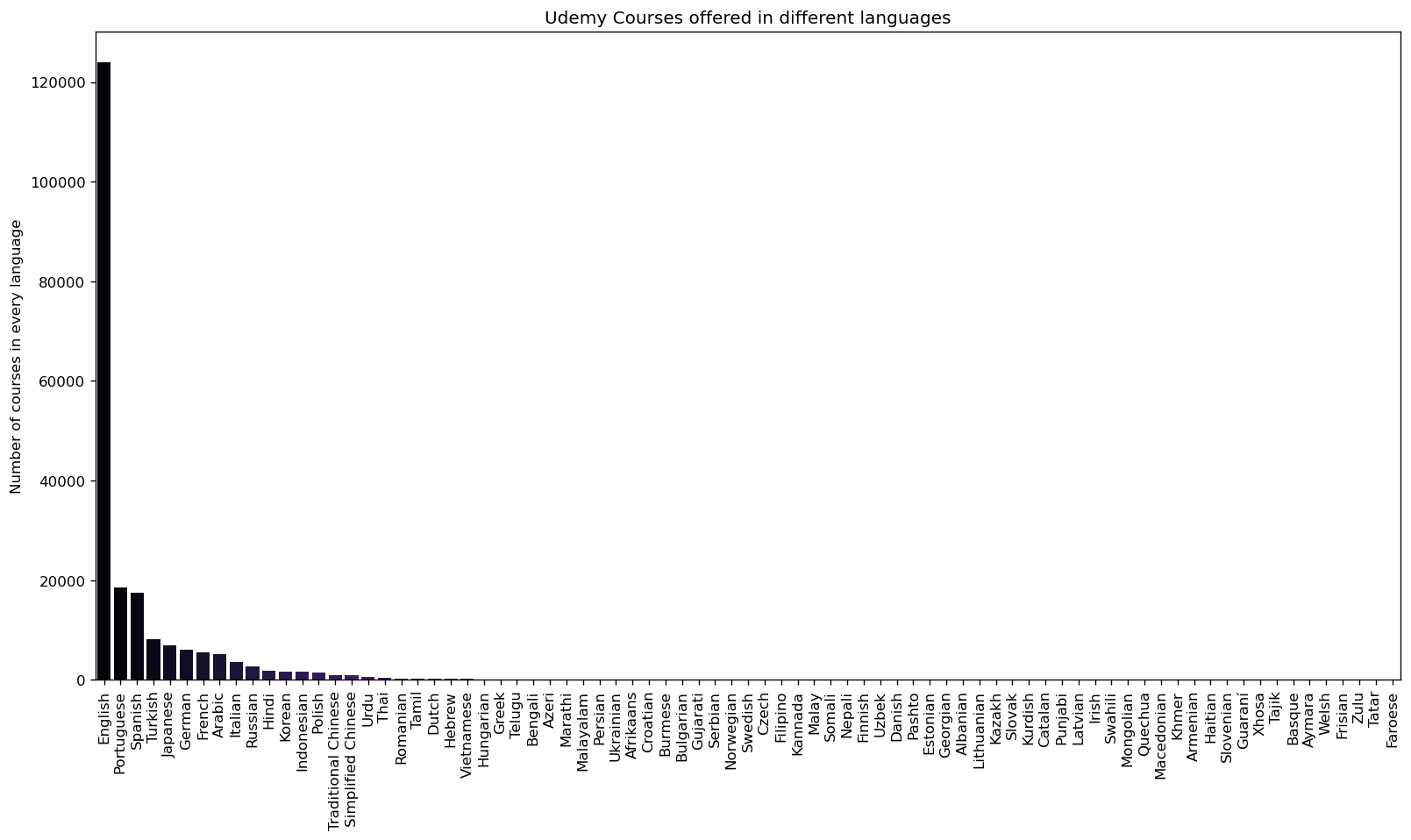
Most Udemy Courses have +4 ratings and the average rating for each category is approximately close (3.0-3.5)

**A graph with numbers and a bar

Description automatically generated**

**Course Languages:**

A screenshot of a computer

Description automatically generated****

Language Distribution of Udemy Courses

Based on the course dataset, Udemy offers courses in 79 languages. The top three languages are:

* English: 59% of the courses
* Portuguese: 8.8% of the courses
* Spanish: 8.3% of the courses

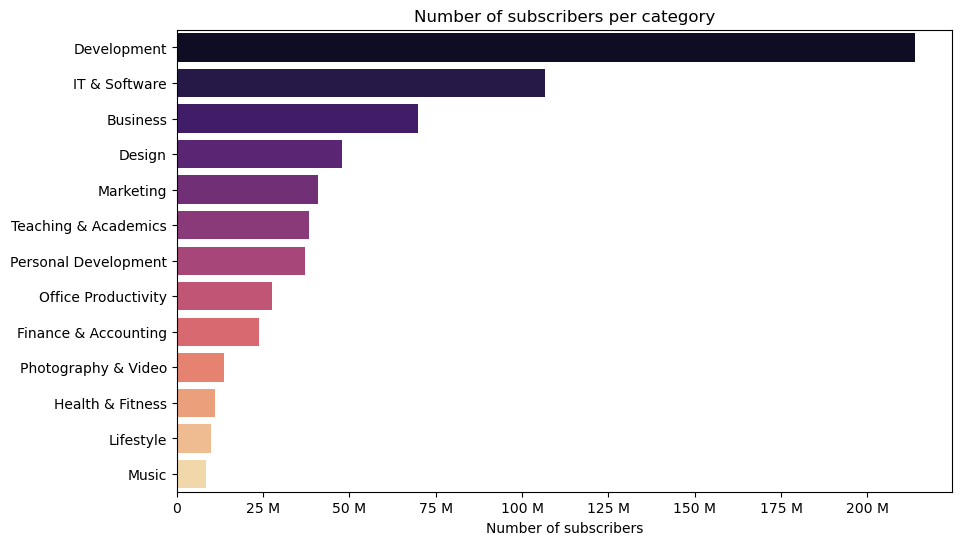
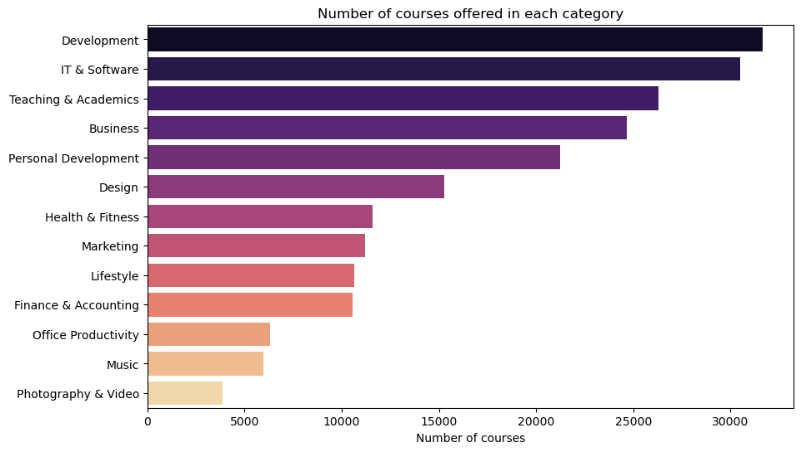
The top 15 languages in which the courses are offered are visualized in the pie chart below.

**Courses under different categories:**

**Udemy Course Categories and Subcategories**

Udemy courses are organized into 13 main categories, which are further divided into 130 subcategories. Additionally, there are 3,818 unique topics under which various courses are offered. Below are the key insights from the analysis:

* **Top Categories by Number of Courses**:
  + **Development**: 31,643 courses
  + **IT & Software**: 30,479 courses
  + **Teaching & Academics**: 26,293 courses
* **Top Categories by Number of Subscribers**:
  + **Development**: ~213 Million subscribers
  + **IT & Software**: ~106 Million subscribers
  + **Business**: ~70 Million subscribers

****

**Subcategory Level Data Visualization:**

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

**Insights on Udemy Course Categories and Subcategories**

* The **Development** category has the highest number of subscribers, with approximately **213 Million** subscribers.
  + Within the **Development** category:
    - **Web Development**: 76.6 Million subscribers
    - **Programming Languages**: 58.5 Million subscribers
* The **Music** category has the least number of subscribers, with around **8.5 Million** subscribers.
  + Within the **Music** category:
    - **Instruments** subcategory has the highest number of subscribers, with about **3.9 Million**.

**Visualizations:**

* **Sunburst charts**: These display the number of courses and subscribers at each subcategory level for different categories, providing a detailed view of subscriber distribution across subcategories.

## **Instructor Earnings:**

## **Insights on Udemy Instructors' Income**

There are **72,731 instructors** on the Udemy platform. The total income of the instructors is calculated by multiplying the **price** and **num\_subscribers** columns. This provides an estimate of their earnings from course sales.

Note: The earnings calculation does not take into account any discounts or coupons that may have been offered, as such data is not available.

**2-Comments Dataset – EDA:**

**Comments Analysis on Udemy Courses**

**Total Comments Made Every Year**

The bar plot below displays the total number of comments posted each year since Udemy's inception in 2010.

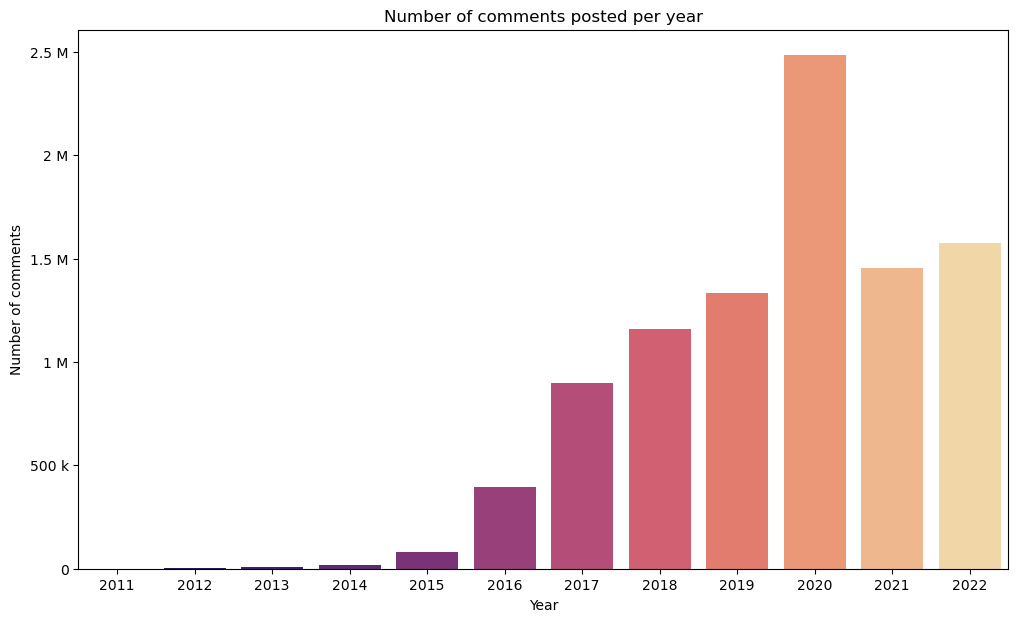
* **Initial Years (2010-2016)**: Less than 500,000 comments were made during this period.
* **Growth Period (2017-2020)**: The number of comments increased significantly, reaching a peak of nearly 2.5 million in 2020.

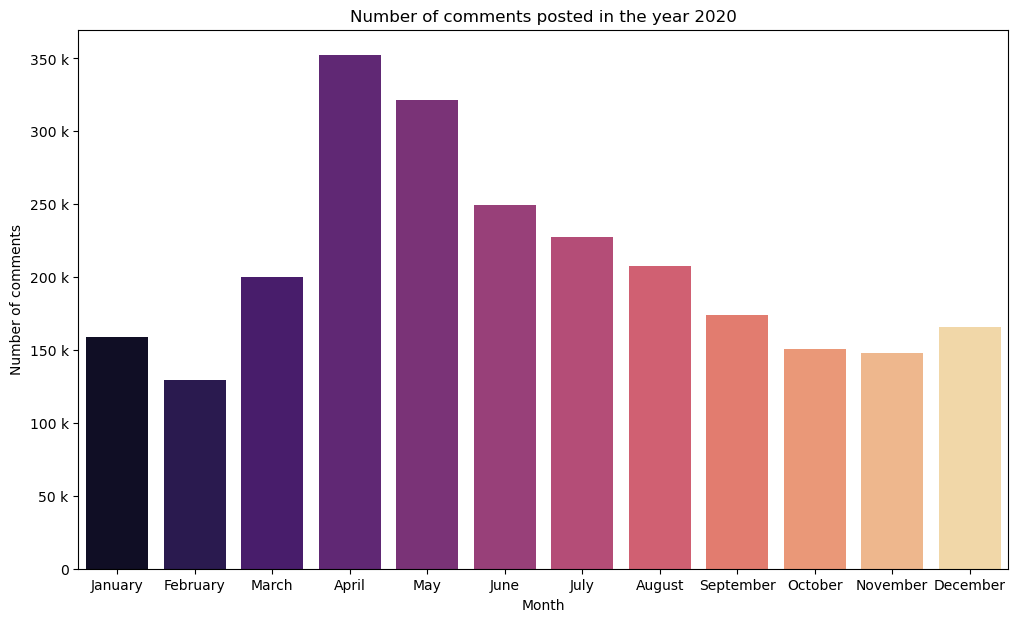
**Comments Posted Per Month in 2020**

The monthly trend of comments posted during 2020 was analyzed to understand user engagement throughout the year. A detailed visualization is shown below, highlighting variations in activity over the months.

**Visualizations**

**Comments Made Every Year**

****

****

**Conclusion**

The two datasets related to Udemy courses and comments were analyzed using data analysis tools. Relationships between features were explored, and new columns were created from existing features. Below is the summary of the findings:

* **Course Pricing**: Nearly 20% of the courses are priced at $19.99. About 10% of the total courses are free of charge.
* **Languages Offered**: Courses are offered in 79 languages, with 59% of the courses taught in English.
* **Category Insights**: The 'Development' category has the highest number of courses and subscribers.
* **Sentiment Analysis**: From the comments dataset:
  + 91% of the comments are positive.
  + 5% of the comments are negative.
  + The remaining comments are neutral.
* **Yearly Activity**: The year 2020 recorded the highest number of comments posted on Udemy.

**Step 4: Recommendation System**

 **Data Preparation:**

* Loaded Comments.csv and Course\_info.csv.
* Filtered courses to include only Development, IT & Software, and Business.
* Merged comments with filtered courses.
* Removed users with ≤5 ratings and courses with ≤10 ratings.

 **Feature Engineering:**

* Calculated course-level metrics: average rating, number of subscribers, and weighted scores (combining average rating, log-transformed subscriber count, and mean rating).
* Selected top 1000 courses based on weighted scores.

 **Collaborative Filtering:**

* Created a User-Item matrix from filtered data (users as rows, courses as columns, and ratings as values).
* Reduced dimensionality using SVD (50 components).
* Computed user similarity using cosine similarity.
* Recommended courses for a user based on ratings from similar users.

 **Content-Based Filtering:**

* Aggregated comments by course and performed TF-IDF vectorization (limited to 5000 features).
* Calculated content similarity between courses using cosine similarity.
* Recommended similar courses based on the content of their comments.

 **Sentiment Analysis:**

* Applied nltk’s Sentiment Intensity Analyzer to comments.
* Added sentiment scores (neg, neu, pos, compound) to each course.
* Identified top positive and negative courses based on average sentiment scores.
* Visualized sentiment distribution.

**Libraries Used :**

1. **Data Manipulation**:
   * pandas, numpy.
2. **Machine Learning**:
   * sklearn (for TF-IDF, TruncatedSVD, cosine similarity).
3. **Text and Sentiment Analysis**:
   * nltk (for sentiment analysis using SentimentIntensityAnalyzer).
4. **Visualization**:
   * seaborn, matplotlib.pyplot.
5. **Sparse Matrix Operations**:
   * scipy.sparse.

**Step 5: Machine Learning Models**

--**A SQLite join to generate a table that links comments to courses. Specifically**, I established a foreign key relationship by using the primary key from the courses table as the reference for the comments table. This allows each comment to be associated with a specific course. Here's how I approached it in detail:

1. Courses Table: The courses table contains information about each course, with a primary key that uniquely identifies each course.

2. Comments Table: The comments table contains comments made by users, and each comment is linked to a specific course.

3. Foreign Key Relationship: In the comments table, I used the primary key from the courses table as a foreign key to establish the connection between a comment and a specific course.

By performing this join, I was able to create a relationship between the two tables, enabling me to retrieve comments associated with their respective courses

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**--I utilized several libraries to build and evaluate machine learning models. Specifically**:

• NumPy and Pandas were used for data manipulation and analysis.

• Matplotlib and Seaborn helped in visualizing data and model results.

• Scikit-learn (sklearn) was essential for implementing various machine learning algorithms, including classification and regression models such as MLPRegressor, MLPClassifier, DecisionTreeClassifier, DecisionTreeRegressor, LinearRegression, Lasso, Ridge, ElasticNet, KNeighborsClassifier, KNeighborsRegressor, RandomForestClassifier, RandomForestRegressor, GradientBoostingClassifier, SVC, SVR, and LogisticRegression.

• I also used GridSearchCV, RandomizedSearchCV, and cross\_val\_score from sklearn.model\_selection for model tuning and cross-validation.

• For scaling and preprocessing, I employed StandardScaler, MinMaxScaler, LabelEncoder, and PolynomialFeatures.

**-- addressed outliers in the dataset by identifying and handling them appropriately**. Various techniques, such as statistical methods (e.g., Z-scores and IQR) or transformation techniques (e.g., winsorization), were applied to detect and manage outliers, ensuring they did not skew the results of the machine learning models. This step is crucial for improving model accuracy and stability.

**-- I handled categorical variables by encoding** them into a format suitable for machine learning models. This was achieved using techniques such as Label Encoding to convert categorical labels into numerical values, and One-Hot Encoding to create binary columns for each category. These encoding methods helped the models process categorical data effectively, ensuring better performance during training and prediction.

-- **processes a dataset to summarize the sentiment of comments for different courses and stores the results in a CSV file. Here's a step-by-step breakdown:**

1. Group by Multiple Columns: The data is grouped by various course-related features like id, is\_paid, price, num\_subscribers, avg\_rating, etc. The target variable for sentiment, comment\_sentiment, is included in the grouping.

2. Calculate Positive Sentiment Percentage: For each group, the code calculates the percentage of comments with a positive sentiment (where comment\_sentiment == 1) by applying a lambda function.

3. Create Course Sentiment Label: Based on the positive sentiment percentage, a new column course\_sentiment is created. If the positive sentiment is greater than 50%, the course is labeled as having positive sentiment (1), otherwise negative sentiment (0).

4. Remove Temporary Columns: The intermediate positive\_percentage column is dropped after it is no longer needed.

5. Output and Save: The course\_sentiment distribution is printed, and the final results are saved in a CSV file named Aggregated\_Course\_Sentiment.csv.

-- **used the following regression models to predict continuous variables and analyze relationships between the input features and target variable:**

1. Linear Regression: The simplest regression model, which assumes a linear relationship between the dependent and independent variables.

2. Polynomial Regression: This extends linear regression by fitting a polynomial function to the data, useful for modeling non-linear relationships between variables.

3. Ridge Regression: A variant of linear regression that uses L2 regularization to prevent overfitting by shrinking less important feature coefficients.

4. Polynomial Ridge Regression: A combination of polynomial regression and ridge regression. It captures non-linear relationships while also preventing overfitting using L2 regularization.

5. Lasso Regression: Similar to Ridge regression but uses L1 regularization, which can force some feature coefficients to zero, helping with feature selection.

6. K-Nearest Neighbors Regression (KNN): A non-parametric regression technique where predictions are made by averaging the target variable values of the nearest neighbors.

7. Grid Search for KNN: I used Grid Search to optimize the hyperparameters for the KNN model, improving the model’s performance by selecting the best values for parameters like the number of neighbors.

8. Decision Tree Regressor: A non-linear model that splits the data into subsets based on feature values, producing a tree-like structure where each leaf node represents a predicted value.

9. Multilayer Perceptron (MLP) Regressor: A type of neural network that uses multiple layers of neurons to model complex non-linear relationships between input features and the target variable.

10. Random Forest Regression: An ensemble of decision trees that aggregates predictions from multiple trees, reducing overfitting and improving generalization.

-- **used the following classification models to predict categorical outcomes based on input features:**

1. Logistic Regression: A fundamental classification algorithm used to model the probability of a binary outcome. It’s suitable for predicting binary classes (0 or 1) based on input variables.

2. K-Nearest Neighbors Classifier (KNN): A non-parametric classification model where predictions are made by finding the majority class among the k nearest data points. It is simple but effective for many classification tasks.

3. Grid Search for KNN: I used Grid Search to optimize the hyperparameters of the KNN model, such as the number of neighbors, to improve classification accuracy by selecting the best combination of parameters.

4. Decision Tree Classifier: A tree-based model that splits the data into subsets based on feature values, creating a tree structure where each leaf node represents a predicted class. It is easy to interpret and can handle non-linear relationships.

5. Grid Search for Decision Tree Classifier: I applied Grid Search to tune the hyperparameters of the Decision Tree model, such as the maximum depth and minimum samples per leaf, to optimize its performance and prevent overfitting.

6. Random Forest Classifier: An ensemble method that combines multiple decision trees to improve classification accuracy and reduce overfitting. It aggregates predictions from various trees to make a final decision.

7. Artificial Neural Network (MLP Classifier): A neural network model that consists of multiple layers of neurons, capable of learning complex patterns in the data. It is useful for non-linear classification tasks and can adapt to various types of data.

8. Gradient Boosting Classifier: A boosting ensemble method that combines multiple weak learners (typically decision trees) to create a strong predictive model. It builds trees sequentially, with each tree correcting errors made by the previous ones