# ETL Case:

Our learning team would like to consume data from a Learning Management System. They would like to see which users enrolled into which courses.

Using the following LMS documentation <https://canvas.instructure.com/doc/api/>, can you please explain how you could retrieve course data for users?

What are the development steps for the ETL?

What other data could be interesting to retrieve and why?

# Data Visualization Case:

A recruitment leader feels like they could be more efficient in their recruitment process and would like to understand what step to focus-on first. Create a dashboard that would help her answer this question that can be used in the future to optimize their operations.

Dataset: “case recruitment dataset.xlsx”

1. Get developer key setup to configure a public JWK
2. Once the developer key is configured and turned on, your tool can [request an LTI access token using the client\_credentials grant](https://canvas.instructure.com/doc/api/file.oauth_endpoints.html#post-login-oauth2-token).
3. Once you have an access token, you can use it to make LTI service requests. The access token must be included as a Bearer token in the Authorization header:
4. Then attention, the endpoint {{URL}} is different that the user / course ones : https://purl.imsglobal.org/spec/lti-nrps/scope/contextmembership.readonly
5. I can then fetch a few json responses w/ GET: users per course, per group

### **GET /api/lti/groups/:group\_id/names\_and\_roles**

### **GET /api/lti/courses/:course\_id/names\_and\_roles**

1. Authentication:
   * To authenticate your requests to the Canvas API, you can use either an OAuth access token or an API key. Since you will not be acting on behalf of a specific user but performing an extraction, it is recommended to use an API key for simplicity.
2. Obtaining an API key:
   * To obtain an API key, you can navigate to your Canvas account's settings. Typically, you can find this under "Account" or "Profile" settings.
   * Look for the "Approved Integrations" or "Developer Keys" section.
   * Create a new developer key and generate an API key for your extraction purposes. Make sure to note down the generated API key as it will be required for making requests to the API.
3. Making API Requests: testing on postman
   * You can use any programming language or tool that supports making HTTP requests to interact with the Canvas API. Popular choices include Python with libraries like **requests** or tools like Postman.
   * Construct your API requests using the appropriate endpoints and parameters specified in the Canvas API documentation (<https://canvas.instructure.com/doc/api/>).
   * To retrieve course enrollment data for users, you can use the "List enrollments for a user" endpoint (**GET /api/v1/users/:user\_id/enrollments**), where **:user\_id** is the identifier of the user whose enrollments you want to retrieve.
   * Ensure that you include the API key in the request headers for authentication purposes. The exact method for including the API key may depend on the programming language or tool you are using.
4. Extracting Data:
   * Once you make a successful API request, you will receive a response containing the requested data in JSON format.
   * It can be on python with request and json libraries
   * Or in Tableau I can create a custom WDC using JavaScript to define the API endpoint, parameters, and authentication methods.
   * Or specify directly endpoints and params in PowerBI for extraction
   * Extract the relevant information from the response, such as the course IDs and user IDs, as well as any additional details you require.
   * You can store this extracted data in a suitable format for further processing. Common options include saving it to a database, writing to a file (e.g., CSV, JSON), or using an in-memory data structure.
5. Data Transformation and Visualization: load and then transform
   * After extracting the required data, you can perform any necessary transformations to prepare it for visualization.
   * Depending on your preferred visualization tool or framework (e.g., Tableau, Power BI, Python libraries like Matplotlib or Plotly), you can load the extracted data and apply the appropriate transformations.
   * Transform the data into a format suitable for your chosen visualization tool (e.g., data frames, structured queries, or other required input formats).
   * Use the capabilities of your chosen visualization tool to create the desired visual representations of the course enrollment data.
6. Authentication: Obtain an access token or API key to authenticate your requests to the Canvas API. Refer to the Canvas API documentation or contact your Canvas administrator to learn how to generate an access token.
7. API Endpoint: Determine the appropriate API endpoint to retrieve course data for users. According to the Canvas API documentation, you can use the "List enrollments for a user" endpoint (**GET /api/v1/users/:user\_id/enrollments**) to retrieve the enrollments of a specific user.
8. User ID: Identify the ID of the user for whom you want to retrieve course data. You will need the user's ID to make the API request.
9. Make the API Request: Use your preferred programming language or tool to make an HTTP GET request to the API endpoint mentioned above, providing the user ID as a parameter in the URL. Include the necessary headers and authentication information (access token or API key) in your request.
10. Process the Response: Handle the response from the API and extract the relevant course data for the user. The response may contain information about the courses in which the user is enrolled, such as course IDs, names, and other details.
11. Repeat for Multiple Users: If you need to retrieve course data for multiple users, you can loop through the user IDs and make separate API requests for each user.
12. Requirements Gathering: Understand the business requirements and data sources. Identify the data to be extracted, transformed, and loaded, as well as any specific data quality or integration requirements.
13. Data Source Analysis: Analyze the structure, format, and quality of the source data. Identify any data inconsistencies, missing values, or data quality issues that need to be addressed during the transformation process.
14. Extraction: Extract the data from the source systems or files. This may involve querying databases, accessing APIs, reading flat files, or any other method of retrieving data.
15. Data Cleaning and Validation: Cleanse the extracted data by handling missing values, removing duplicates, and performing data validation checks. This step ensures that the data is accurate, consistent, and conforms to the defined rules and standards.
16. Transformation: Apply transformations to the extracted data to convert it into the desired format for analysis or loading into the target system. This may involve data mapping, aggregation, filtering, joining, or any other data manipulation operations.
17. Data Loading: Load the transformed data into the target system, such as a database, data warehouse, or analytics platform. The loading process may include creating tables, defining schemas, and inserting or updating records.
18. Data Quality Assurance: Perform data quality checks and validations on the loaded data to ensure its accuracy and integrity. This step involves verifying data completeness, consistency, and conformity to the expected standards.
19. Error Handling and Logging: Implement mechanisms to handle errors, exceptions, and data inconsistencies encountered during the ETL process. Logging and error tracking help in identifying and resolving issues efficiently.
20. Scheduling and Automation: Set up a scheduling mechanism to run the ETL process at regular intervals or as per the defined schedule. Automation ensures that the data is kept up to date and the ETL pipeline operates reliably.
21. Testing and Validation: Conduct thorough testing to validate the ETL pipeline's functionality, data accuracy, and performance. Perform end-to-end testing, unit testing, and integration testing to identify and rectify any issues.
22. Documentation: Document the ETL process, including data flow diagrams, transformation rules, and technical specifications. Documentation helps in understanding and maintaining the ETL pipeline in the future.
23. Monitoring and Maintenance: Set up monitoring mechanisms to track the performance, data quality, and health of the ETL pipeline. Regular maintenance and updates may be required to accommodate changes in data sources, transformation rules, or target systems.
24. User Demographics: Retrieving demographic information about the users, such as age, gender, location, educational background, or job title, can provide insights into the characteristics of the learner population. This information can be useful for analyzing learning patterns, personalizing course offerings, or identifying target audiences for specific courses.
25. Course Completion Data: Tracking course completion data, including the percentage of users who completed each course, the average completion time, or the completion rate over time, can help evaluate the effectiveness and engagement of the courses. It provides insights into learner behavior, course difficulty, or potential areas for improvement in the course design or content.
26. Assessment Results: Capturing assessment results, such as quizzes, tests, or assignments, allows for analyzing the performance and learning outcomes of the users. It helps identify areas where learners excel or struggle, evaluate the effectiveness of assessments, and make data-driven decisions for course improvements or individualized feedback.
27. User Interactions: Collecting data on user interactions within the learning management system, such as login patterns, course access frequency, time spent on each course page, or discussion forum participation, can provide valuable insights into user engagement and learning behaviors. Analyzing these interactions can help identify active learners, popular course materials, or areas where learners need additional support.
28. Course Feedback and Ratings: Gathering feedback and ratings from users about their learning experiences, course satisfaction, or specific course elements can provide valuable insights into the quality and effectiveness of the courses. It helps identify strengths and weaknesses, areas for improvement, and can guide future course development or modifications.
29. Learning Analytics: Incorporating learning analytics data, such as clickstream data, social network analysis, or sentiment analysis from discussion forums, can provide deeper insights into user behavior, collaboration patterns, or sentiment trends. These advanced analytics techniques help identify hidden patterns, predict learner outcomes, or personalize learning experiences.
30. Instructor Data: Including information about instructors, such as their qualifications, teaching experience, or user ratings, can provide additional context for understanding course quality and learner satisfaction. It allows for analyzing the impact of instructor characteristics on course outcomes or identifying highly effective instructors.