**Data Engineering Assignment**

Write a paragraph about yourself, your hobbies and your major achievements in your career or study so far.

I graduated university with a 1st class degree in Creative Technology, a course and dissertation which focused on creating software tools and interfaces for creative users. I achieved a record-breaking score in my final year which was a major achievement, with a project which allowed users of an audio art installation to interact with sound in the room from six purpose built front-ends, ranging from simple mouse/keyboard input, to a bespoke motion tracking system I had developed myself.

In my career I’ve worked at a range of technology companies from web development to automated communication systems and as a data analyst for an investment company. In my last 2 jobs (spanning 6 years) I have risen quickly up the technology ladder and am now the Head of Business Intelligence where 6 years ago I was a lowly Product Support Engineer. In my work life I am proudest of the relationships I’ve built with people in every team of the companies I have worked at, and believe wholeheartedly that by understanding and showing compassion for the pain-points of colleagues, we can provide technology solutions with the most valuable impact.

My hobbies include food, restaurants, and the occasional video game, but all that is dominated by books, books, books! Perhaps my favourite time of day is the hour before work… with a latte and my latest fantasy saga or anti-trope speculative science-fiction novel on my lap. I live in West London with my girlfriend and dog, who we even named after a fantasy character. His name is Albus.

At Cult Wines we’ve embarked on, and almost completed, and incredibly ambitious technology rollout since 2020. Starting from a tiny technology footprint (a single SQL Server running on premise in our Hammersmith office), we have now developed a huge suite a

complex cloud-based data-driven platforms. I have been integrally involved in designing and architecting bespoke analytics front-end for internal management of investment positions, an inventory management system backed by blockchain, and will shortly be

launching a brand new B2C trading platform. As the first Data Engineer hired into the team, I have been intimately involved throughout the planning of that entire rollout, wearing many hats along the way… from engineer to business analyst to product manager.

**Part A**

Consider Bookly, an internet company that owns a marketplace for selling books, a blog and other complementary online products with an overlapping user base. Each product is backed by its own independent Ruby web server and relational SQL database.

Bookly has approx. 100k books and 1M users.

**Question 2**

The Marketing Team at Bookly wants to merge and analyse all the data that is being collected by the different products in order to extract useful business insights of various kinds. Examples of such analytics include (but are not restricted to):

• List all-time top-rated books and trending ones.

• Measure user sign-up rate over certain periods (weekly, quarterly, etc).

• Show the total number of real-time (current) page views for any given book description page (product page).

*Design a conceptual data pipeline to drive and aggregate data from all the different sources to ultimately be accessible by a user-friendly data exploration/dashboarding tool of your choice. Feel free to pick any technology available (e.g., open source, cloud providers, etc.).   
  
Describe the different components of the architecture, tools involved and compare possible approaches.*

Design Concept:

The first consideration when designing an architecture where your data-sources are disparate is architecture concept. Where in the past we might have moved towards a unified system and have all our data-sources migrated / replicated to a centralised system, more modern data architectures commonly follow the concept of either data fabric or data mesh.

A data mesh focuses on domain-oriented design, in which each of the system is designed and managed separately with focus on the requirements of that domain. Commonly this results in larger data structures being broken into micro-service style components along lines of data domain. In the case of Bookly, you might separate the databases of each relational databases into services controlling CRM domain, sales domain, product domain and so on. In this way the architecture is designed with data gravity to the development user in mind, i.e., by separating sales domain per server into a service designed specifically for that purpose, we are prioritising the requirements of users who need to work with/on that data, over the analytical/transactional requirements of the companywide data architecture. Once our domain-oriented data-sources are prepared, they will be utilised in a centralised analytical environment (see below) to be served used in data pipelines and so on.

A data fabric shifts the requirement for unifying our disparate data into the centralised analytical environment itself. Whereas a mesh will prepare each data-source internally before connecting through the environment, data fabric allows each source to be rename (within reason) untouched. Each source is pulled into the analytical environment as is, and their relations and necessary transformations managed there. This architecture moves our data gravity closer to the end-user, i.e., we are shift the control of unifying our data-sources closer to the front-office, as opposed to prioritising back-office requirements.

The choice between these 2 approaches is broad and depends largely on the priorities and future intentions of the system. Fabric has the advantage of allowing much more flexibility in the sources we might choose to bring into our environment. In future we might have something as simple as a CSV file which we also wish to query in our analytics. A data fabric architecture will allow that without minimal development required… simply provide the CSV (via a cloud storage capability like Blob Storage or S3), connect to the file from your environment and it is ready to be used in pipelines / front-office. A data mesh requires more preparation of the new data-source before it can be used, but in doing so ensures that data governance and lineage remain consistent and manageable once that preparation is complete.

It is of course also possible to mix these concepts depending on the source of your data.

For Bookly, given that we currently have 3 separate Ruby Web Servers each with its own relational SQL database, but without any truly disparate datatypes / formats to be considered, I would suggest a data mesh design for this architecture. The 3 data-sources are running on separate instances of the same technology, and likely contain similar data domains. It is therefore worth the upfront work to prepare the management of those domains properly at the source and gaining the data governance/lineage advantages inherent in that, before bringing each source into our unified data architecture.

NB. Lineage / Governance: there are plenty of advantages to ensuring proper data lineage and governance. In Europe the most obvious of these are requirements of GDPR around personal identifiable information. Using data-mesh in conjunction with a cataloguing tool such as Purview, we ensure that all PII is instantly tagged and tracked through all downstream pipelines and front-office document stores. This lineage cataloguing alone will be sufficient to avoid any GDPR concerns raised, and will make the task of deleting customer information an easy one.

Unified Architecture Technology:

With our data sources prepared, we can now connect them via our chosen architecture. There are many technologies available for this in the market the biggest players being Azure Synapse (formerly Data Factory), Databricks, GCP Big Query and AWS Redshift. The choice between those technologies will largely depend on the current technology stack of Bookly, and the skillsets of their data engineering team. Since no information about that stack/team is provided here, I would choose Azure Synapse in this case. Azure has a strong affiliation with Azure SQL Database, excellent integration with Jupyter Notebooks for more advanced data-science analytics, Logic Apps for API connectivity, and native key-vault management for security.

Unified Architecture Identity Management:

Assuming our data-sources are now prepared and connected to Synapse (or alternative), the first step in our data pipelining is to ensure unique identification across the systems. Bookly have a relatively large and over-lapping customer base, but there is no indication that those overlapping users can be uniquely identified across systems, i.e., the same customer might well exist in all 3 data-sources, with separate and unrelated Ids. How do we ensure that we’re able to match a customer in one data-source with the same customer in another?

Relating them using some field other than Id – for example name or email address – is obviously flawed. We therefore need to transactional service at the point of customer creation to manage our UUIDs.

I would therefore pitch that a separated customer identification service be created for generating unique identifiers. When a “new customer” transaction is created on any data-source, this service is asked to create, store, and return a UUID that is guaranteed unique across the entire company. We should not generate this UUID within any single data-source. If a single database is elevated to generate UUIDs within its own SQL database, our unique identification service is dependent on the state of a single source.

With this system in place and all existing customers passed through it, we can now be sure that when querying/transforming our 3 data-sources in pipelines, we will have consistent and future-proof identifier on which to join them.

NB. A similar system might also be used for identifying products (books, blog posts etc.) too. The necessity of that requirement would depend on the process through which those products are created. If books, for example, are sourced for outside our environment or by user input then identifiers like title or SKU may not be reliable, and therefore an internal identifier of this sort might be necessary for true data lineage. As a general rule, any Trusted Data Asset on which the operations of the business depend should have a managed and reconciled identification system like the one described.

Middle Office Transformation / Pipeline:

With our data-sources connected and properly relatable, data transformation can now be done dependent on the specific requirement. This might be a broader requirement aimed at creating a data product for general use (e.g., create a large object containing a wide variety of sales data for internal business intelligence), or a specific requirement aimed at providing a single result to a front-end / B2C product.

Another consideration here is performance vs. real-time requirements of the data product. Where a query/pipeline will be slow, but the result need not be real-time, our pipeline might target a data warehouse table downstream from its source of truth for storing pre-processed data for faster delivery. Where the data product must be real-time though, optimisation will need to be handled at the source, or the transformation kept light.

Without schema designs for the data-sources, the specific steps of workflows are not definable here. Generally, there will be collection of main workflows for creating the bulk, generalised schemas that can be used for most front-end tasks. Where specific results or reports are required, specific workflows can be created according to requirement in an ad-hoc manner. On the right-hand side of the work flows we have several options in the proposed architecture.

* We might store pre-processed data in a sink database if the results might be used as a dataset for a different workflow.
* We might output results from pipelines into a dedicated data warehouse for heavy-duty processing requirements; large scale data science / machine learning requirements and so on.
* Where results of the workflow are intended for front-end display, we must then prepare the result for front-office database.

Front Office Document Storage:

The final stage of a production data architecture is front office document storage. Since that topic and design is generally the remit of product / front-end designers rather than data engineers, I will not go into detail here. For this pipeline design we will assume use of a high-performance document store such as Cosmos/Mongo/Dynamo, and that our front ends will retrieve and orchestrate those documents using an APIM native to the technology stack, or alternative such a GraphQL.

Data Pipeline Design Diagram:

*data\_pipeline\_architecture.png*

**Question 3**

The Customer Support team spends considerable time scanning through customer reviews and comments in order to filter out illegitimate ones. Multiple factors can contribute to label comments as authentic or not:

• Comment is made by a registered user vs. anonymous.

• Level of user activity (e.g.: number of past reviews and comments);

• Content of the comment (e.g.: unauthorized advertising)

*To automate the filtering process, design a conceptual, real-time, decision support system that takes data as input (user properties, user actions, content, etc) and automatically labels comments as legitimate/illegitimate for the Customer Support Team to quickly flag and remove the unwanted.*

*Describe the different components of the architecture, tools involved and compare possible approaches.*

Decision Support System:

For analysing the legitimacy of online comments there are a few levels of analysis that we can utilise. Immediate analysis of the commentor compared to our CRM system, on the spot analysis of the content of the comment itself, and comparative analysis of the comment versus those that have been deemed legitimate in the past. Passing through each of these layers, the decision support system would assign a mistrust score, and a threshold set on that score beyond which we flag the comment.

1. Immediate commentor analysis:
   1. Are they registered? Compare the UUID of the user to your CRM source of truth.
   2. Unregistered users get a strongly weighted mistrust score.
2. Single comment content analysis:
   1. Links.
      1. It can generally be said that links posted in comments are likely posted by SEO bots or for affiliate marketing advertisement.
      2. In the case of specific customer domain like Bookly (instead of a very broad social media platform where legitimate comments might be about anything), it is likely far more efficient to maintain a whitelist of websites to which we *allow* linking rather than a blacklist of websites from which we do not (for example whitelist book review sites & stores).
      3. Comments containing links that do not pass the whitelist are given a strongly weighted mistrust score.
   2. Structure Analysis:
      1. Simple text engines can complete analysis of word-count, paragraph count and punctuation use.
      2. Comment with word/paragraph/punctuation that drops below a set threshold given a lightly weighted mistrust score.
      3. Above those thresholds no decisions about legitimacy can be made from the structure of a single comment in isolation, but results for comments deemed legitimate can be stored for cross-comment analysis later (see below)
   3. Topic Analysis:
      1. Data science tooling and / or ML can be leveraged to isolate key words from the body of the text to give a weighted score to what the comment is most likely referring.
      2. Clearly nefarious and hate-speech topics immediately assigned a full mistrust score. Trigger auto-removal?
      3. Topics which fall outside of the general domain of platform (books and related topics), but are not deemed clearly hateful, are given a lightly weighted mistrust score.
      4. Topic analysis scores for comments deemed legitimate stored for cross-comment analysis (see below)
   4. Sentiment analysis:
      1. Data science tooling and / or ML can be leveraged to isolate key words from the body of the text to give a measure of positive/negative content of the comment.
      2. Although no decisions about legitimacy or weighted scoring can be made from the sentiment of a single comment in isolation (it is legitimate to leave a negative comment about a book of course), results of sentiment analysis for comments deemed legitimate stored for cross-comment analysis later (see below)
3. Cross-comment Comparison:
   1. Having passed our full comment history and all new comments through Single Comment Analysis above and storing the results, we now have a model for analysing the structure, topics and sentiments of comments that have been deemed legitimate.
   2. All comments can have their own scores in each of those areas compared to our legitimate comment analysis results, and mistrust weighted according to their disparity from those results. Where for example a legitimate comment contains on average > 100 words, 2 paragraphs and a generally positive sentiment on the topic of books, we can score new comments with very few words, no line-breaking, and negative sentiment about another topic, to most likely be provided by bots and/or forum trolling.

Using a decision support system liked the one described above provides multiple levels of authenticity to new comments, and a system for flagging comments to moderators. It allows those moderators to set thresholds at various levels on each analysis for tuning the decision support system, and by always storing those results for comments, provides a system that will be become more and more accurate with time.

Tools / Technologies:

Of course, there are a wide range of machine platform technologies that a company might employ, but in the case of Bookly I’m not sure I would take that route. The cost of ML platforms and the data science expertise that’s needed to set that up and maintain it is tricky to justify for a company and data size of Bookly. A one million user base is a good size for any company, but the throughput of comments through this decision support system won’t be high enough to justify a full ML solution for this requirement.

Instead, I would deploy a separate server specifically for this analysis, and replicate comment content and other required data (CRM) to that server so that this computation won’t impact performance of the main system. Once we have the data there, that data source should be used as source for Python (or similar) functions since full-text processing capabilities are much better there than in SQL or workflow/ETL steps. The end goals are:

* Store analysis results of legitimate comments for later cross-comment comparison.
* Pass flag results (legitimate/illegitimate) to either A) auto-removal transaction in main system middle office, or B) manual moderation front-end.

Decision Support Design Diagram:

*decision\_support\_system\_design.png*

**Part B**

The marketing Team at Bookly introduced A/B Testing on their blog - each blog post will show a given registration popup from a set of pre-configured popups. The raw dataset in [SEE EMAIL] offers a table with various content properties and the conversion rates for each combination of blog post and registration popup.  
  
**Question 4**

*What is the major contributor to user registrations? In other words, what is the most relevant factor that contributes the most to convert user views into user registrations?*

*Describe in detail all the steps you take to perform the analysis, provide code snippets, relevant data transformations and results.*

Methodology:

Considering the dataset is mostly text content etc., we must first transform it into an analytical dataset from which insights can be derived.

I therefore imported the CSV into SQL Server Management Studio for initial processing. The dataset being largely text content, analytics would be more efficiently processed using Python / R, which have better functionality for free text search, but being more comfortable with SQL than alternatives, I will be doing this analysis in SQL. Upon import to table, I also assigned a identity to each row, to allow proper joining of sub-queries later in the process.

By separating each field by whitespace and simple grouping of results I identified keywords which appear most frequently in each column. For example, the words “Free” and “Download” appear often in popup\_header column. In this way I identified binary metrics that could be applied to each column, converting the content to a numerical table against with analysis could be performed. I ran that query and inserted results to analytical table (*analytical\_dataset.sql*), in a similar manner as you might within a data warehouse.

I then ran a series of experimental queries against the dataset to deuce which of the binary metrics retrieved from the dataset has most impact on registrations. I have considered inclusive (positive) metrics for results only – that is to say, a keyword or phrase appearing in a column will be considered for my analysis; a keyword or phrase not appearing in a column will not.

Of all keywords and phrases found in each column, the inclusive metric that has the most impact on registrations, is the use of the word “Free” in the popup\_header column. **77% of all 47432 registrations came from popups with the keyword “Free” in the header.**

We might also say that volume of views is the major contributor of registrations, since 40% of all registrations come from popups with 10000 views or more. Since that is a mathematically predictable outcome though, I’ve not defined that as the major contributor here.

*analytics.sql*

**Question 5**

*Explore the dataset. What other insights can be extracted?*

*Describe in detail all the steps you take to perform the analysis, provide code snippets, relevant data transformations and results.*

Other insights:

* There is a strong correlation between popup views and registrations, as you would expect. 40% of all registrations come from the top 53 popups (of 2063 popups with content – 2%) sorted by views.
* 48% of data rows in the dataset have NULL content across the board. This most likely indicates an issue with storing/CRUD of information into the database, or that Bookly are presenting almost half of their customer with empty popups, which would clearly be a bug. In either case, steps should be taken to resolve.
* Of the remaining popups, 75% receive 0 registrations. Marketing departments might chose raising this as a KPI/OKR for the future, since this shows considerable opportunity for improvement even where the above “bug” is excluded.
* Of popup subjects, “Perfecting your Craft” is most successful. Subjects including that phrase resulted in 48% of registrations.
* Of popup subjects, “From our authors” is the least successful in driving registrations, with only 56 registrations coming from these subjects.
* “Book Marketing” is also highly unsuccessful. Subjects including that phrase resulted in only 4% of registrations.
* There is no strong correlation between popup name and registrations. The most successful popup name is “Character Profile Checklist 4” with 7007 registrations, but in general there is a healthy spread of registrations across different popup name groups.