1) As a lifelong baseball enthusiast with a background in data science and mathematics, my love for the game goes far beyond mere fandom—it's deeply woven into my professional ambitions and personal passion. One of my most cherished memories is witnessing the final pitch of the 2011 World Series in person with my biggest role model, my grandfather. That same fall, the movie Moneyball came out, leaving a lasting impression and fueling my journey into baseball analytics. As a naïve 14 year old, I immediately went to the store to buy the Moneyball book, knowing then that my understanding of baseball would be forever changed. Although life's demands as a high school and college student shifted my focus at times, I stayed engaged with analytics trends, never losing sight of my passion for the game's data-driven side.

   Family and societal expectations pushed me toward a "traditional" career, leading me initially to a Mathematics major with plans to become an actuary. Later, I transitioned to Electrical Engineering for broader career options. After completing a Master's degree in Engineering Data Analytics and Statistics and securing a software engineering role at a Fortune 500 company, I found myself feeling unfulfilled in this "traditional" career. Despite my efforts to distract myself with hobbies, friends, and travel, something still felt missing. Over the last few years, however, I've found myself drawn back to baseball analytics—a passion I've rekindled to the point of subtle obsession.

   I've developed custom models to quantify player skills, engineered new stats, and created predictive systems that assess player value for my fantasy league, which has led to success in high-stakes leagues. I spend hours studying work by Eno Sarris on Stuff+ models, Tom Tango's sabermetric foundations, and numerous other articles on FanGraphs.

   Even as analytics have become mainstream—appearing regularly in broadcasts—I still see untapped potential, particularly in biomechanics, where pitcher injury prevention is becoming critical, and in player development. I envision hitting models analogous to Stuff+ for pitchers, such as a "bat path+" model to evaluate raw hit tools or a "hitter eye+" metric that more accurately quantifies plate discipline beyond traditional chase% or z-swing% and chase% differentials. Even with the public availability of Hawk-Eye and Statcast data, we're only beginning to unlock the potential for deeper insights.

   As someone who thrives on discovery, I'm thrilled by the strides being made in baseball analytics and excited by the potential for future advancements. The game's vast, readily available datasets position baseball to lead the way in Big Data, and I can no longer ignore the chance to be part of this analytics evolution. Applying for the Machine Learning Engineer position with the St. Louis Cardinals, I'm bringing not only my love for the game but also my analytical skills and vision for where baseball analytics can go next. I want to help the team succeed while shaping the future of the sport's analytics on a larger scale.

Even during this season, which some may call disappointing by the Cardinals' standards (I think a lot of fans are forgetting we only missed the playoffs by six games, but under the surface it is in fact gloomy with a run differential that ranked 11th in the National League), I attended games weekly, often by myself. I kept score while using the Statcast game tracker to analyze players and stay engaged as an active, analytical consumer of the game. In this role with the Cardinals, I could channel that same dedication and insight to support the team’s goals through my love for analytics and through my longest love, Cardinals baseball.

2) The most comprehensive model I’ve built was a movie recommendation engine, developed as my Master's capstone project. Recommendation engines often use two main techniques: collaborative filtering, which predicts a user's affinity for an item based on the preferences of similar users, and content-based filtering, which recommends items similar to those the user has previously enjoyed. I chose collaborative filtering due to its flexibility and effectiveness, as it allows the model to learn from diverse user preferences and suggest genres the user might not have previously interacted with. In contrast, content-based filtering could limit recommendations to a certain niche, reinforcing existing preferences.

One common challenge with collaborative filtering is the cold start problem: without sufficient interaction data, the model lacks enough information to generate reliable recommendations. To address this, I designed the interface as a web application powered by the Flask framework, where users are prompted to rate 20 randomly selected movies, creating a sufficient profile for recommendation generation. Users could also choose between collaborative filtering algorithms—nearest neighbors or Singular Value Decomposition (SVD)—to provide recommendations.

For validation, the application allows selecting any user from the database to view recommendations across algorithms. This was useful as a sanity check, helping confirm that personalized recommendations varied appropriately across users and algorithms. In terms of algorithmic performance, the nearest neighbors method included hyperparameter tuning on the similarity metric, where both Pearson and cosine similarity measures were tested. Nearest neighbors were highly efficient because they only require calculating similarity for one user at a time, allowing it to run at request time. This efficiency came at the cost of accuracy, as SVD methods demonstrated superior performance on validation data in terms of root mean-squared error (RMSE).

The SVD algorithm decomposes the user-item matrix into three components: the user-latent feature matrix (left singular vectors), a diagonal matrix of singular values representing the importance of each latent feature, and the item-latent feature matrix (right singular vectors). After decomposition, the predicted rating $r_{ui}$ for a user ‘u’ and item ‘i’ is calculated as:

$$r_{ui} = q_i^T p_u$$

where qi is the latent feature vector for item i and pu is the latent feature vector for the user.

As one can imagine, this is computationally expensive, and therefore, in order to improve user experience, the choice was made to launch SVD for the original user-item matrix (prior to the new user's insertion if the choice to get personal recommendations is made) upon starting the engine, in order for quick retrieval of recommendations for the already stored users in the database. On the other hand, this isn't feasible for the new user wanting recommendations, as they obviously need their interactions stored in the matrix before the algorithm can provide them recommendations. At the expense of the long run time in the latter case, the SVD outperformed the other algorithms (Stochastic Gradient Descent and Alternating Least Squares were simple algorithms provided by the surprise library API and used merely as benchmarks to compare the other algorithms against). Other considerations regarding the project were the use of Content based filtering, and a hybrid approach where the recommendations for both content and collaborative filtering were combined into an aggregate score. For the hybrid model my idea was to weigh each score based off the number of interactions the user had, where the larger the total interactions were the more heavily weighted the collaborative filtering score would have been i.e.:

$$Hybrid = \left(1 - e^{-\beta n}\right) * CF + e^{-\beta n} * CB$$

where CF is collaborative filtering score, CB the content based score, n as the number of interactions, and beta as the rate of decay for the content based score importance. While this was considered, the ultimate goal and scope of the project was to provide a demonstratable model that accurately provided recommendations. While this hybrid approach sounds great in theory, the additional time necessary to execute both methods in a capstone presentation setting wasn't feasible. So, bearing in mind user experience, the choice was made to keep it simple, but effective by limiting the scope to the powerful collaborative filtering techniques.

Additional non-technical considerations were the design of the user interface, and the overall experience it provided. The randomized movies prompted to the user seeking recommendations provides a fun and effective method of collecting interactions, as the movies were unpredictable and were diverse in content, spanning across multiple genres and generations, providing a complete profile for the user.

3) I would say my go-to "tool" in my technical toolbox, and the one I take the most pride in, is my raw mathematical ability. With a broad, yet deep understanding of probability, linear algebra, statistics, stochastic processes, optimization, etc., I have a solid foundation to tackle the complexities of machine learning. This knowledge is essential for grasping the underlying principles of the algorithms, helping me not only to select the right algorithms for the specific problem, but also to make informed adjustments that optimize model performance. This expertise also enables me to approach exploratory data analysis effectively, which ultimately guides how I execute the data modeling stage. A strong grasp of the math behind machine learning gives me a reliable basis for building, troubleshooting, and enhancing complex models across various data types.

With this foundation in mathematics and statistics, alongside the programming skills I've developed through my engineering career, I am now focused on expanding my expertise in the latest industry database and data infrastructure tools. My next steps involve advancing my knowledge in cloud computing, data storage systems, and API development for efficient data transfer. Mastering these tools will allow me to bridge the gap between data storage and modeling stages, enabling streamlined, scalable data processing pipelines to support robust machine learning workflows.

4) The inception of the Stuff+ suite of models and all the subsequent research around pitch modelling has inspired much of my recent research to revolve around the pitching side. However, I somewhat jokingly mentioned a "Hitter Eye+" model to better assess true plate discipline talent rather than relying on chase rate, z-swing% and derivatives thereof, so let's go ahead and explore this idea.

   The "Hitter Eye+" metric would aim to quantify a batter's decision-making ability by assessing their swing choices in terms of selectivity, aggression, and the difficulty of pitch recognition. This metric would account for both the count and the expected run value difference from the current to the resulting count, penalizing or rewarding decisions accordingly. I envision categorizing choices into classifications such as "bad takes," "bad swings," and "good decisions," each contributing to a weighted score based on the probability of a strike being called for the pitch location. Unlike broader metrics such as walk rate or existing plate discipline measures, this approach would provide a more granular, process-oriented assessment of a player's plate discipline skills, offering insight into their ability to identify and react to optimal pitches.

   As an extension or possible second iteration, I would incorporate a pitch recognition difficulty metric to be used as another weighting parameter. Starting with available Statcast pitch-level data—including velocity, movement, spin rate, and release point across the pitcher's full arsenal—I would analyze how different pitches compare in terms of profile similarity. Rather than calculating a Stuff+ measure for each pitch, I'd focus on using these features to determine at what point similar pitches become distinguishable (e.g., when a cutter can be differentiated from a slider). By examining this point of recognition, I could create features that estimate the reaction time required for a hitter to make an informed swing decision, effectively capturing the difficulty level of each pitch recognition task.

   Using these newly engineered data points, I would develop a decision score for each swing or take, evaluating whether the action aligned with an "optimal" choice given the count and the pitch's location. Training this model on historical pitch location data to uncover the probability of a called strike for each location, and the resulting change in run value for each outcome would enable it to reward positive outcomes and penalize negative ones. This Hitter Eye+ score would offer a robust measure of a batter's pitch selection skill, highlighting strengths and weaknesses on a pitch-by-pitch basis and giving teams a data-driven tool to identify and develop players with a highly advanced approach at the plate.

In-game, teams could leverage this metric to tailor recommendations and guide hitters on strategies best suited for facing a specific pitcher's arsenal. Developing a Hitter Eye+ metric would provide teams with breakthrough insights into one of the game's most critical yet subtle skills, offering a strategic advantage in player development and game planning.

5) I recently automated an ETL process to build a baseball data pipeline, designed to collect, transform, and load player performance data from multiple sources into a centralized database. The need for automation stemmed from the requirement to maintain a real-time data feed that could manage large volumes of data and handle updates efficiently. Without automation, the frequent data retrieval and transformation tasks would have been too time-consuming and prone to errors, making it difficult to ensure consistency in data processing.

   To develop this pipeline, I evaluated various tools, including Airflow for workflow orchestration, AWS Glue for managed ETL, and dbt (Data Build Tool) for transformation management. Ultimately, I selected Airflow for scheduling and orchestrating tasks due to its flexibility and robust support for custom pipelines, as well as Pandas and SQLAlchemy for data extraction and transformation. PostgreSQL was chosen as the data warehouse because it allows for scalable storage and efficient querying of large datasets.

   The pipeline is organized into three main stages: extraction, transformation, and loading. In the extraction stage, Airflow schedules and runs data extractions from sources like Statcast and other open baseball APIs, saving raw data into staging tables. The transformation stage includes cleaning and unifying formats, standardizing metrics, and engineering features such as pitch-level metrics and performance trends. For these tasks, I used Pandas for data manipulation and stored transformations as SQL scripts managed by Airflow. In the loading stage, the transformed data is loaded into PostgreSQL, where it's accessible for further analysis and model inputs.

   For non-technical considerations, my main focus is on designing a user interface that balances the amount of information displayed with functionality. In addition to the prediction models for K-BB% for pitchers and projected wRC+ for hitters that are part of the pipeline, I am exploring the feasibility of integrating VBA-based functionality from my fantasy baseball Excel sheets directly into the database. This involves deciding whether to build functions that can replace the VBA scripts currently used to automate data collection. These functions would enable features such as date ranges, separate tabs for projected fantasy values, date-specific ranges, proposed auction values, cheat sheets for draft season, and trackers for emerging or undervalued players in the market. Another consideration is how to keep these tools separate from machine learning models yet make them easily accessible. This setup would provide a seamless, accessible experience for users while maintaining the independence and scalability of the machine learning models.