Answer 0

- ➔ I would rate my knowledge of cricket as a 5 out of 5. I have a comprehensive understanding of the sport, regularly watch matches, and am familiar with various teams, players, and rules. My knowledge extends beyond basic awareness, and I am well-equipped to engage in discussions or tasks related to cricket.

Answer 3:

- ➔ To achieve incrementally loading of data instead of batch backfilling we would need to make several changes to our existing code/architecture. The system needs to shift from bulk processing past data to promptly handling new cricket match information in real-time or near-real-time as it emerges. This needs a streaming mechanism for real-time data capture and ingestion.
- ➔ Some key points we should focus on to achieve this.
  - ➢ We would need to introduce the new technologies like Apache Kafka and Apache spark to facilitate real-time ingestion of match data.
  - ➢ Modify the existing data processing pipeline to handle incremental updates rather than batch processing. This may involve adapting the data transformation and loading steps to work with smaller portions of data and updating the relevant components accordingly.
  - ➢ Update database schema and indexing and If needed we can move to azure datalake storage (ADLS) where we can store data in delta format which is very useful for incremental real-time data loading and processing. This ensures efficient retrieval and updating of data as new match information arrives.
  - ➢ Enhance monitoring capabilities for real-time data ingestion. In this case we can use Azure Data Factory(ADF) or Azure Databriks to monitor the workflows. Also we can implement alerting system using same
  - ➢ We can set auto scaling of workers in Spark system so that if load increases it will increase the number of workers. Hence our system is capable of handling varying loads of incoming data.
  - ➢ Ensure the integrity and reliability of the data by implementing techniques that manage concurrent updates effectively. Employ transactional processes or idempotent methods to maintain consistent and accurate data, especially in situations where multiple updates may occur simultaneously.
  - ➢ Address challenges related to synchronizing historical data when transitioning from batch backfilling to incremental loading. Maintain a cohesive and current dataset by resolving any potential issues that may arise during the transition, ensuring that historical records remain accurate and synchronized with the incremental updates.

Answer 4:

- ➔ I encountered a significant learning opportunity when our client decided to migrate from Azure to GCP, and they entrusted our team, including myself, with this critical task. Having had no prior experience with GCP, I transparently communicated this to the client. Their trust in our capabilities inspired me to proactively embark on learning GCP concurrently with the migration.
- ➔ Throughout the process, I focused on mastering key tools such as BigQuery, Cloud Composer, Apache Airflow, and Data Proc. This not only involved theoretical understanding but also hands-on application in our project. For instance, we leveraged Apache Airflow to orchestrate workflows and execute queries in BigQuery. Our data, stored and processed in BigQuery, was seamlessly downstreamed to other business units for consumption.
- ➔ The successful migration of the Campaign master from Azure to GCP stands as a testament to the effectiveness of my learning approach and its direct application in achieving project objectives. This experience not only broadened my skill set but also highlighted the importance of adaptability and continuous learning in dynamic project environments.

Answer 5:

- ➔ In a crucial project, I was tasked with developing code to handle malformed records across 49 different entities, where these records failed data quality constraints. Testing posed a significant challenge, as the available data in dev and UAT environments was insufficient, and testing in production was not an option.
- ➔ Upon deploying the code to production, it unexpectedly got stuck during execution for every entity, leading to considerable frustration and time consumption in identifying the root cause without the ability to test on actual data. In collaboration with my team lead, we decided to temporarily pause the workflow and devised a solution.
- ➔ To ensure minimal impact on production data, I proposed copying the malformed data to a separate location for thorough testing. After obtaining approval from both my team lead and the Data team director, I proceeded to replicate a subset of records from the source to this isolated location. This allowed me to meticulously test and debug the code, identifying and addressing the issues with precision.
- ➔ Once the bugs were fixed and the code enhancements were validated, I initiated a new pull request with the changes. The revised code was then successfully deployed, resolving the initial issue. While this approach was time-consuming, especially across all 49 entities, the satisfaction of overcoming the challenge and ensuring the integrity of the data used by the ML and Data Sciences teams for model building made it a rewarding experience.