**Cricket Knowledge**

- Self-assessment: You can provide your self-assessment rating from 1 to 5 based on your knowledge of cricket.

python

cricket\_knowledge = 3 # Your self-assessment rating (1 to 5)

**Question 1: Data Ingest Process**

python

# Import necessary libraries

import sqlite3

import requests

import json

import pandas as pd

# Create an SQLite database and tables

conn = sqlite3.connect("cricket.db")

cursor = conn.cursor()

# Create tables

cursor.execute("""

CREATE TABLE IF NOT EXISTS match\_results (

match\_id INTEGER PRIMARY KEY,

team1 TEXT,

team2 TEXT,

result TEXT,

year INTEGER,

gender TEXT

)

""")

cursor.execute("""

CREATE TABLE IF NOT EXISTS ball\_by\_ball (

match\_id INTEGER,

delivery INTEGER,

runs INTEGER,

PRIMARY KEY (match\_id, delivery)

)

""")

# Load and insert match results data

with open("match\_results.json", "r") as file:

match\_results\_data = json.load(file)

for match\_result in match\_results\_data:

cursor.execute("""

INSERT INTO match\_results (match\_id, team1, team2, result, year, gender)

VALUES (?, ?, ?, ?, ?, ?)

""", (match\_result["match\_id"], match\_result["team1"], match\_result["team2"], match\_result["result"], match\_result["year"], match\_result["gender"]))

# Load and insert ball-by-ball data

with open("ball\_by\_ball.json", "r") as file:

ball\_by\_ball\_data = json.load(file)

for ball\_data in ball\_by\_ball\_data:

cursor.execute("""

INSERT INTO ball\_by\_ball (match\_id, delivery, runs)

VALUES (?, ?, ?)

""", (ball\_data["match\_id"], ball\_data["delivery"], ball\_data["runs"]))

# Commit changes and close the database

conn.commit()

conn.close()

print("Data ingested into the database.")

**Question 2: SQL Queries**

**a. The win records (percentage win and total wins) for each team by year and gender, excluding ties, matches with no result, and matches decided by the DLS method.**

sql

-- Query 2a: Win Records

SELECT

year,

gender,

team1 AS team,

SUM(CASE WHEN result LIKE '%won' THEN 1 ELSE 0 END) AS total\_wins,

ROUND(AVG(CASE WHEN result LIKE '%won' THEN 1 ELSE 0 END), 2) \* 100 AS win\_percentage

FROM

match\_results

WHERE

result NOT LIKE 'tie'

AND result NOT LIKE 'no result'

GROUP BY

year, gender, team;

**b. Which male and female teams had the highest win percentages in 2019?**

sql

-- Query 2b: Highest Win Percentages in 2019

WITH WinRecords AS (

SELECT

year,

gender,

team1 AS team,

SUM(CASE WHEN result LIKE '%won' THEN 1 ELSE 0 END) AS total\_wins,

ROUND(AVG(CASE WHEN result LIKE '%won' THEN 1 ELSE 0 END), 2) \* 100 AS win\_percentage

FROM

match\_results

WHERE

result NOT LIKE 'tie'

AND result NOT LIKE 'no result'

GROUP BY

year, gender, team

)

SELECT

year,

gender,

team,

MAX(win\_percentage) AS max\_win\_percentage

FROM

WinRecords

WHERE

year = 2019

GROUP BY

year, gender;

**c. Which players had the highest strike rate as batsmen in 2019?**

sql

-- Query 2c: Highest Strike Rate Batsmen in 2019

WITH PlayerStats AS (

SELECT

M.match\_id,

M.year,

M.gender,

M.team1 AS team,

B.runs,

B.delivery

FROM

match\_results M

JOIN

ball\_by\_ball B

ON

M.match\_id = B.match\_id

WHERE

result NOT LIKE 'tie'

AND result NOT LIKE 'no result'

AND year = 2019

)

SELECT

year,

gender,

team,

SUM(runs) AS total\_runs,

COUNT(delivery) AS total\_deliveries,

ROUND(SUM(runs) \* 1.0 / COUNT(delivery), 2) AS strike\_rate

FROM

PlayerStats

GROUP BY

year, gender, team

HAVING

COUNT(delivery) >= 100

ORDER BY

strike\_rate DESC;

**Question 3: Incremental Loading**

To address the concept of incremental loading of new match data on a go-forward basis, we need to modify the data ingest process and database management. Below is an explanation of how the solution would change:

**1. Change Data Ingest Process:**

- Create a mechanism to identify and download new match data from cricsheet.org on a regular basis. This can be done through APIs, web scraping, or any other suitable method.

- Develop a script that compares the new data with existing data in the database to identify new matches or updates to existing matches.

**2. Database Modifications:**

- Include additional columns in the match results and ball-by-ball tables to track the date or timestamp of when the data was ingested.

- Modify the tables to handle updates, especially for ball-by-ball data that might change due to corrections or updates.

**3. Scheduled Process:**

- Implement a scheduled job that periodically checks for new data, such as daily, weekly, or monthly, depending on the data source's update frequency.

**4. Data Deduplication:**

- When new data is ingested, it should be compared with existing data to avoid duplicates. This involves identifying matches that are already in the database and updating them if necessary.

**5. Audit Trail:**

- Keep a record of when and where new data was ingested, including any updates or corrections made to existing data.

**6. Error Handling:**

- Implement error handling and notifications to alert administrators in case of issues during the data ingestion process.

**7. Data Archiving:**

- Consider archiving older data if needed to manage the database size.

In summary, for incremental loading, the data engineering solution would require setting up a process to regularly fetch and integrate new data while handling potential updates and avoiding data duplication. This ensures that the database remains up-to-date and accurate over time.

**Question 4: Learning New Techniques**

This question is about sharing an example from your experience when you learned a new technique, method, or tool during a project or analysis. Here's a model response:

**Example:**

During a recent data engineering project, I encountered a scenario where the data pipeline required real-time data streaming and processing. The traditional batch processing approach wouldn't suffice due to the need for low-latency data analysis. After some research, I discovered Apache Kafka and its stream processing capabilities.

**What Inspired Me to Learn:**

The project's requirements for real-time data analysis inspired me to explore Apache Kafka. The need for processing large volumes of data as it arrived in real-time was a crucial aspect of the project, and Kafka was known for its ability to handle such use cases effectively.

**How I Learned:**

I started by reading documentation and tutorials to understand the fundamental concepts of Kafka. I set up a local Kafka cluster and experimented with producing and consuming messages. To implement this new technique, I had to modify the existing data pipeline significantly.

I also utilized online courses and joined the Kafka community forums to learn from experienced Kafka users. This collaborative learning approach allowed me to gain insights into best practices and overcome challenges.

**Application in the Project:**

Implementing Apache Kafka into the data pipeline had a transformative impact on the project. Real-time data ingestion and processing became much more efficient. We were able to meet the low-latency data analysis requirements and provide immediate insights to end-users. This significantly improved the project's success and user satisfaction.

In conclusion, learning about Apache Kafka and incorporating it into the project enhanced my data engineering skill set and demonstrated the adaptability and innovation required to solve complex real-world data challenges. This experience emphasized the importance of staying up-to-date with emerging technologies and being open to learning new techniques to achieve project goals effectively.