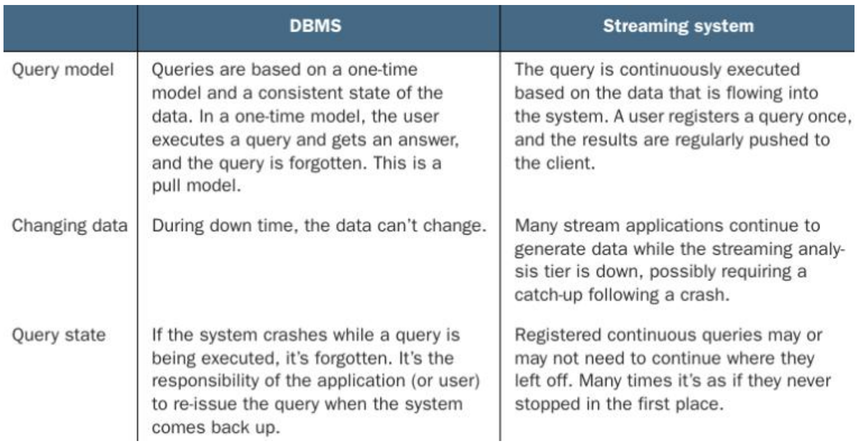
Department of Artificial Intelligence and Machine Learning

Date: 12-01-26	**Test – 3**	**Max. Marks: 10 + 50**
Semester: VII	**UG**	**Duration: 2 Hrs.**
Course Title: Stream Processing and Analytics		**Course Code: AI372TA**

Common to AIML and CSE (Data Science)

PART A

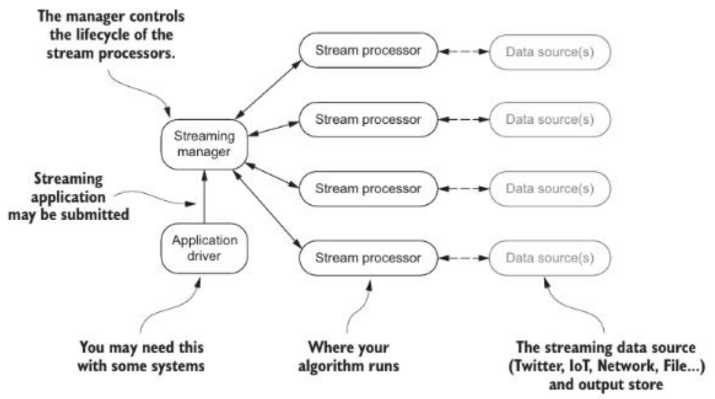
Scheme and Solution

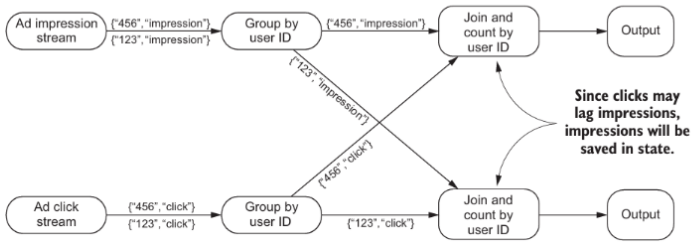
S. No	Questions	M	BT	CO
1	_____________is tuple-at-a-time stream-processing framework designed for real-time processing of data streams, while ________ is a stage-wise stream-processing framework Apache Storm, Apache Samza	2	2	2
2	Identify three key differences between traditional query model and continuous /streaming query model. 	2	2	2
3	What is Concept drift in streaming systems ? give an example. This is a phenomenon that may impact your predictive models. Concept drift may happen over time as your data evolves and various statistical properties of it change. Example: A stock price prediction model trained on pre-pandemic market behavior performs poorly during and after the pandemic because underlying market dynamics and investor behavior have changed.	**2**	**2**	**2**
4	Illustrate tumbling windows in streaming systems and mention their two types. A tumbling window in a streaming system divides the data stream into fixed-size, non-overlapping time intervals, where each event belongs to exactly one window and is processed once. Two types are: count-based and temporal-based	2	2	2

	DBMS	Streaming system
Query model	Queries are based on a one-time model and a consistent state of the data. In a one-time model, the user executes a query and gets an answer, and the query is forgotten. This is a pull model.	The query is continuously executed based on the data that is flowing into the system. A user registers a query once, and the results are regularly pushed to the client.
Changing data	During down time, the data can't change.	Many stream applications continue to generate data while the streaming analysis tier is down, possibly requiring a catch-up following a crash.
Query state	If the system crashes while a query is being executed, it's forgotten. It's the responsibility of the application (or user) to re-issue the query when the system comes back up.	Registered continuous queries may or may not need to continue where they left off. Many times it's as if they never stopped in the first place.

5	Enumerate the techniques used for stream summarization. 1. Random Sampling 2. Counting Distinct elements 3. Frequency 4. membership	2	2	2

PART B

S. No	Questions	M	BT	CO
1a.	Analyze the need for distributed stream processing and explain the key components of a distributed stream-processing architecture using a neat diagram. Need(2M)+diag(2M)+explanation(6M)=10M It may be possible to run an analysis tier on a single computer, but the velocity and volume of the data at some point make this a nonviable option. A Generalized Architecture : Spark, Storm, Flink, and Samza all have the following three common parts: A component that your streaming application is submitted to; this is similar to how Hadoop Map Reduce works. Your application is sent to a node in the cluster that executes your application. Separate nodes in the cluster execute your streaming algorithms. Data sources are the input to the streaming algorithms.  The manager controls the lifecycle of the stream processors. Streaming manager Stream processor Data source(s) Streaming application may be submitted Application driver You may need this with some systems Where your algorithm runs The streaming data source (Twitter, IoT, Network, File...) and output store Figure 4.4 Generic streaming analysis architecture you will find with many products on the market Application driver - With some streaming systems, this will be the client code that defines your streaming programming and communicates with the streaming manager Streaming manager - The streaming manager has the general responsibility of getting your streaming job to the stream processor(s); in some cases it will control or request the resources required by the stream processors.	6	02	02

	Stream processor - the place where your job runs. Although this may take many shapes based on the streaming platform in use, the job remains the same: to execute the job that was submitted. Data source(s) - This represents the input and potentially the output data from your streaming job. With some platforms your job may be able to ingest data from multiple sources in a single job, whereas others may only allow ingestion from a single source.			
2 a	Explain the importance of state management in stream-processing systems using and example.  Ad impression stream {"456","impression"} {"123","impression"} Group by user ID {"456","impression"} Join and count by user ID Output {"123","impression"} {"456","click"} Since clicks may lag impressions, impressions will be saved in state. Ad click stream {"456","click"} {"123","click"} Group by user ID {"123","click"} Join and count by user ID Output **Figure 4.14** **Handling ad impression and ad click streams that use stream state** 2+2=4M State management is essential in stream-processing systems when computations depend on historical events rather than only the current incoming message. Simple stateless processing cannot support operations such as aggregations, joins, or handling delayed events. Without state management, any failure in the streaming job may lead to loss of intermediate results and incorrect outcomes. In an ad-serving application, two separate streams are generated: one for ad impressions and another for ad clicks. Since ad clicks typically occur after impressions, the system must retain impression events in state until the corresponding click events arrive. By maintaining state—keyed by user ID or ad ID—the stream processor can join the impression and click streams and compute meaningful metrics such as click-through counts. Persistent and replicated state storage ensures that this information is not lost during failures and allows accurate stream joins despite event delays, thereby enabling reliable and fault-tolerant real-time analytics.	**4M**	**3**	**3**
2b	Briefly discuss the three ways to write data to long-term storage. 3*2=6M **Direct Writing** **Option A** is the most direct method of getting data to a long term store. Following this pattern, you write each message as it is processed—in essence writing at stream speed.	**6M**	**2**	**2**

	Option B you build up the messages in the analysis tier until either a certain batch size is reached or a certain period of time passes and then write them to long-term storage. **Risk :** If the data store you're writing to can't keep up with the rate at which you're processing data —our stream time—then that may have a negative impact on your ability to process the data in the stream. • If you're processing data at speed, it becomes harder and harder to tune a batch-oriented storage system as your data volume and velocity increases. • Although option B involves build ing up batches of messages before writing them, it is an attempt to optimize for the slow write speed (slow is relative—in this case it's related to stream speed). Indirect Writing • In this case , decouple the storage of the stream-processed data from the long-term store. • write the data out to a message queuing tier as an intermediary. • This helps decouple our streaming analysis tier from the performance of the long-term storage system. • Added Complexity : Batch Loader : The goals of the batch loader are to read batches of mes sages from the message queuing tier and write them to our long-term storage. Benefits • The message queuing tier is designed to handle the speed and volume of a stream, eliminating the risk of not being able to write fast enough. • We can use a component that is dedicated to bulk loading data, ensuring that we can keep our streaming analysis focused on analyzing the stream. Keeping it in Memory Keeping data in-memory can be a way to achieve fast access times, especially for applications that require real-time processing.			
3 a	What are the four possible approaches for handling analyzed or collected data in stream-processing systems? 1.5*4=6M **Analyze and discard the data** **Analyze and push the data back into the streaming platform** **Analyze and store the data for real-time usage** **Analyze and store the data for batch/offline access**	5M	**2**	**2**

3 b	Compare the **State-Machine** and **Rollback Recovery** approaches to fault tolerance in stream-processing systems.	4M	3	2

State-Machine Approach	**Rollback Recovery Approach**
Replicates job across nodes; same input sent to all	Periodically checkpoints state; replays logs on failure
High (k + 1 times)	Lower
Quick failover, minimal disruption	Slower recovery, depends on checkpoint and log replay
Low-latency, real-time applications	Applications tolerant to moderate latency

4	Enumerate the Stream-processing frameworks and explain any two in detail with neat diagram. List(2M) + 4M*2=10M 1. Apache Spark 2. Apache Storm 3. Apache Flink 4. Apache Samza Student need to explain any two in detail	**10**	**2**	**2**
5.	Describe the various stream summarization techniques and illustrate each with an example. 2.5*4=10M Random Sampling (Reservoir Sampling) This technique maintains a fixed-size random sample (reservoir) from a data stream such that every element has an equal probability of being included. Initially, the reservoir is filled with the first k elements. For the nth incoming element, it is included in the reservoir with probability k/n, and if selected, it randomly replaces an existing element. Example: Maintaining a random subset of ad views to answer queries like "How many distinct ads were shown in the last minute?" or sampling packets in a high-speed network intrusion detection system operating at 40 Gbps. Counting Distinct Elements These techniques estimate the number of unique elements in a stream using probabilistic methods. Bit-pattern-based methods: Use leading zero patterns in hashed values to estimate cardinality (e.g., LogLog, HyperLogLog, HyperLogLog++). Order statistics-based methods: Rely on order statistics such as minimum hashed values (e.g., MinCount, Bar-Yossef).	10	2	2

	Example: Estimating the number of distinct users or ads in a massive data stream where storing all elements is infeasible. Frequency Estimation (Count-Min Sketch) Count-Min Sketch uses a two-dimensional array of counters with multiple pairwise-independent hash functions. Each incoming element updates one counter per row. The estimated frequency of an element is the minimum of the corresponding counters, reducing overestimation due to hash collisions. Example: Answering queries like “How many times has stream element X occurred?” or estimating how often a particular ad was viewed. Membership Query (Bloom Filter) A Bloom filter is a probabilistic data structure that tests whether an element is a member of a set. It uses a bit array and multiple hash functions. It may produce false positives but never false negatives. Example: Checking whether a particular ad ID or packet signature has appeared in the stream before, without storing all past elements.			