✅ *Question 1: What are the 5 V's of Big Data? Why are they important?*

Answer:
The 5 V's define the challenges and opportunities in Big Data systems:

- ***Volume** – Refers to the massive amount of data generated (e.g., Facebook: 4+ PB/day).*
  Systems must scale horizontally to handle this scale.

- ***Velocity** – The speed at which data flows in (e.g., IoT sensors, trading platforms).*
  Necessitates real-time processing with tools like Kafka, Flink.

- ***Variety** – Includes structured, semi-structured, and unstructured data.*
  Systems must ingest and process diverse formats like JSON, images, logs.

- ***Veracity** – Data quality and accuracy.*
  Requires validation, cleaning, and lineage to ensure trustworthy insights.

- ***Value** – The insights and business impact derived from data.*
  Justifies investment in big data platforms; it's about extracting ROI from data.

👉 *These characteristics shape how systems are architected—from storage to processing and analytics.*

✅ *Question 2: Why do traditional databases struggle with Big Data workloads?*

Answer:
Traditional RDBMSs are built for structured data and vertical scaling. They struggle with:

- ***Scalability**: Can't easily scale horizontally across commodity hardware.*

- ***Flexibility**: Rigid schemas don't support semi/unstructured data well.*

- ***Performance**: Bottlenecks occur under high write/read loads and complex joins.*

- ***Cost**: Scaling with proprietary solutions can be expensive.*

👉 *Big Data systems (like Hadoop, NoSQL, Spark) are designed to overcome these limitations using distributed architectures and flexible data models.*

✅ *Question 3: Compare HDFS and S3. In what scenarios would you choose one over the other?*

Answer:
HDFS *(Hadoop Distributed File System):*

- *Tightly coupled with Hadoop ecosystem*
- *Optimized for high-throughput, batch workloads*
- *On-premises or private cloud*
- *Manual scaling and management*

S3 *(Amazon Simple Storage Service):*

- *Cloud-native, fully managed, highly durable*
- *Supports decoupled compute (Athena, EMR, Redshift Spectrum)*
- *Auto-scales and cost-effective for elastic workloads*
- *Serverless and suitable for modern data lake architectures*

*👉 Choose **HDFS** for controlled, on-prem clusters with traditional Hadoop jobs.*
*👉 Choose **S3** for scalable, cloud-native pipelines needing flexibility, reliability, and lower ops overhead.*

✅ *Question 4: What types of workloads qualify as Big Data problems?*

Answer:
Workloads that involve:

- ***High Volume**: Terabytes–petabytes of data (e.g., application logs, CCTV footage)*
- ***High Velocity**: Continuous data inflow (e.g., sensor streams, financial transactions)*
- ***High Variety**: Heterogeneous data sources (e.g., web logs, social media, audio)*

***Examples**:*

- *Clickstream analysis*

- *Fraud detection*
- *Predictive maintenance with IoT data*
- *Training ML models on massive datasets*
- *Real-time bidding in ad platforms*

👉 When traditional systems break down due to size, speed, or complexity—it's a Big Data problem.

✅ *Question 5: What is the difference between batch and stream processing? Which would you use for fraud detection?*

Answer:
Batch Processing

- *Processes data in chunks (e.g., nightly reports)*
- *High throughput, but higher latency*
- *Examples: Hadoop MapReduce, Spark batch*
- *Use for model training, historical analysis, data warehousing*

Stream Processing

- *Processes events as they arrive (low-latency)*
- *Real-time use cases: alerts, monitoring, dashboards*
- *Examples: Apache Flink, Kafka Streams*

Fraud Detection Use Case
Stream processing is ideal because immediate detection is crucial.
E.g., Kafka ingests transactions → Flink applies detection logic → Alerts raised in real time.
Batch may still be used to train models with historical fraud data.

✅ *Question 6: What is Delta Lake and how does it improve upon traditional data lakes?*

Answer:
Delta Lake is an open-source storage layer built on top of data lakes (e.g., S3, HDFS) that brings:

- ***ACID Transactions** – Ensures reliability for concurrent reads/writes*
- ***Schema Enforcement** – Avoids corrupt or invalid data*
- ***Time Travel** – Enables rollback and historical data access*
- ***Performance Optimizations** – Through caching, indexing, and compaction*

Traditional data lakes lack consistency and governance. Delta Lake fills this gap, making lakes production-ready for analytics and ML.

✅ *Question 7: How would you design a system to process terabytes of log data daily?*

Answer:
A scalable, fault-tolerant pipeline could look like:

- ***Ingestion**: Kafka or AWS Kinesis to ingest logs in real time*
- ***Storage**: Store raw logs in S3 or Delta Lake for cost-effective, scalable storage*
- ***Processing**:*
    - *Use Spark (batch) for ETL and aggregation*
    - *Use Flink (stream) for real-time alerting or filtering*
- ***Querying**: Presto, Athena, or Redshift Spectrum for ad-hoc analytics*
- ***Visualization**: Tools like Grafana, Superset, or Tableau for insights*

👉 *Design should ensure horizontal scalability, data durability, and support both real-time and batch workflows.*

✅ *Question 8: What storage and processing frameworks would you use and why?*

Answer:
Storage

- ***S3/Delta Lake****: Scalable, low-cost, supports schema enforcement and time travel*
- ***HDFS****: Suitable for on-prem, tightly coupled Hadoop workloads*
- ***NoSQL (e.g., Cassandra, MongoDB)****: For high-velocity, semi-structured data with low-latency reads*

Processing

- ***Apache Spark****: For large-scale batch jobs, ETL, machine learning*
- ***Apache Flink / Kafka Streams****: For real-time processing with low latency*
- ***Presto/Trino****: For federated, interactive SQL queries on large data*

👉 *Tool choice depends on latency requirements, data structure, scale, and whether the workload is batch, stream, or hybrid.*