UNIT–5 : Frameworks, Visualization and R Programming (PYQ Solution)

AKTU · Data Analytics (BADS601) · Solutions by **Chirag Jain**, Ycotes

SECTION A Answers (≈70 Words Each)

Q1. What is the basic description of a Box Plot in R? (2022)

A box plot (box-and-whisker plot) shows the five-number summary of a variable: minimum, first quartile (Q1), median, third quartile (Q3) and maximum. The box spans Q1 to Q3 (the IQR) with a line at the median; whiskers extend to the most extreme values within 1.5 × IQR and points beyond are drawn as outliers. In R it is drawn with `boxplot(mpg ~ cyl, data = mtcars, main = "Mileage")`.

Q2. List two Data Visualization Tools. / Write names of two visualization tools. (2022, 2023)

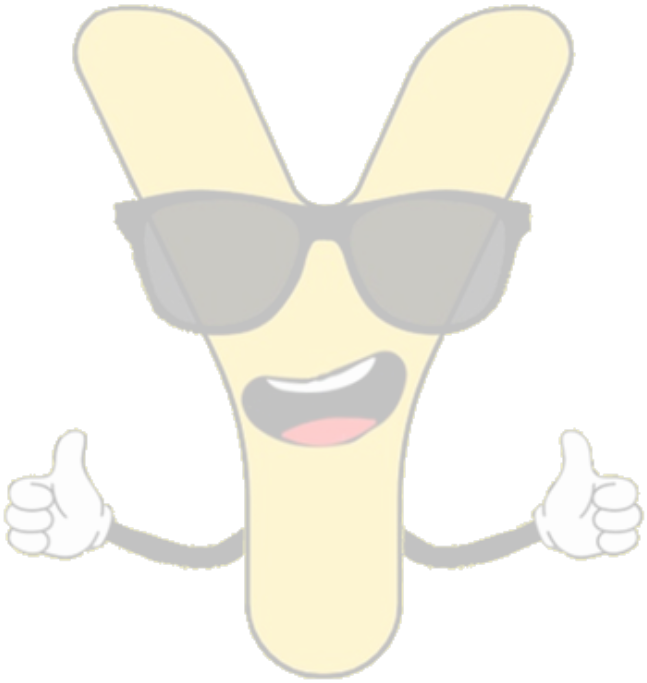
Two popular data visualisation tools are Tableau and Microsoft Power BI. Others are Qlik Sense, Google Data Studio (Looker Studio), D3.js, Plotly, and the R package ggplot2. These tools turn data into interactive charts, maps and dashboards that make patterns easy to see.

Q3. Differentiate between Pig and SQL. (2023)

Apache Pig uses Pig Latin, a procedural data-flow language where the user writes a step-by-step script that is compiled into MapReduce jobs; it handles structured and unstructured data and has a flexible, optional schema. SQL is a declarative language in which the user states *what* result is wanted, and the engine decides how; it works on structured data with a fixed schema in an RDBMS and supports transactions and indexes. Pig is for ETL on big data, SQL for querying relational tables.

Q4. Write a short note on R graphical user interface. (2023)

R is a language for statistics and graphics, and its basic interface is a command-line console. A graphical user interface (GUI) makes it easier to use. The standard R GUI offers a console, script editor and menus; RStudio is the most popular IDE with panes for source code, console, environment and plots/help; R Commander (Rcmdr) gives point-and-click menus for statistics and graphs; and Deducer, Jamovi and Tinn-R are other options. A GUI reduces typing and helps beginners.



SECTION B Answers (≈250 Words Each)

Q1. Differentiate between NoSQL database and a Relational database. Identify when one should use a NoSQL database instead of a relational database with a suitable example. / What makes NoSQL databases different from RDBMS? / How is RDBMS different from NoSQL? (Jan 2025, 2022, 2023)

A **Relational database (RDBMS)** stores data in tables of rows and columns with a fixed schema and uses **SQL**; examples are MySQL, Oracle and PostgreSQL. A **NoSQL ("not only SQL") database** stores data in flexible, non-tabular models and is designed to scale across many servers; examples are MongoDB, Cassandra, Redis and Neo4j.

Types of NoSQL databases

- **Document store:** JSON-like documents (MongoDB, CouchDB).
- **Key-value store:** key → value pairs (Redis, DynamoDB).
- **Column-family store:** wide columns (Cassandra, HBase).
- **Graph database:** nodes and edges (Neo4j).

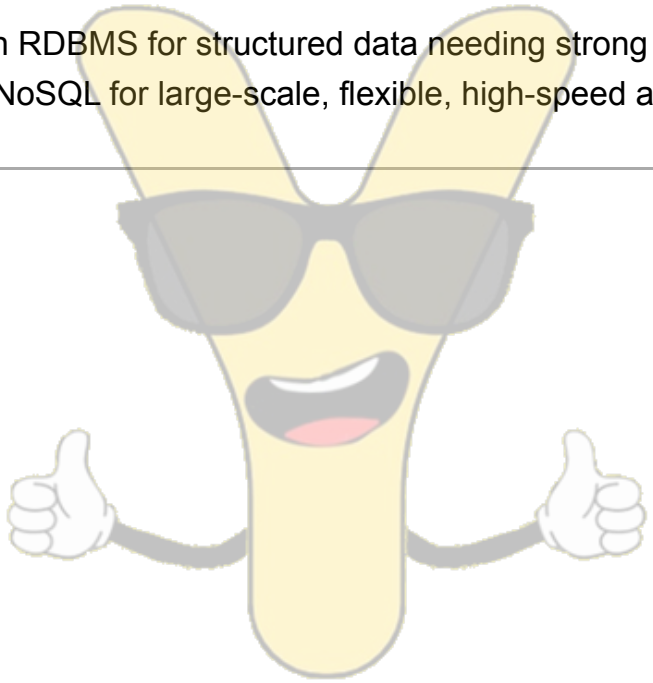
Basis	RDBMS	NoSQL
Data model	Tables, rows, columns	Document, key-value, column, graph
Schema	Fixed, predefined (schema-on-write)	Dynamic, flexible schema
Query language	SQL	Database-specific APIs, no standard language
Scaling	Vertical (bigger server)	Horizontal (add commodity nodes, sharding)
Consistency	**ACID** transactions	**BASE** (eventual consistency), CAP trade-off
Relationships	Joins and foreign keys	Few or no joins; data embedded
Data types	Mainly structured	Structured, semi- and unstructured
Best for	Complex queries, transactions	Big data, high speed, changing data
Examples	MySQL, Oracle, SQL Server	MongoDB, Cassandra, Redis, Neo4j

When to use NoSQL instead of RDBMS

- Data is **huge in volume** and growing, and must be spread over many servers.
- The data is **semi-structured or unstructured**, or the structure changes frequently.
- **High read/write speed and availability** matter more than strict consistency.
- Needs are simple key look-ups or relationship traversals, not complex joins.
- Rapid development with changing requirements.

Example: a social-media or e-commerce site storing millions of user profiles, posts and product catalogues with different attributes. A document database like **MongoDB** stores each profile as a JSON document, adds new fields without altering a schema, and scales out easily. By contrast, a **banking system** needs ACID transactions and is better with an RDBMS.

Conclusion: use an RDBMS for structured data needing strong consistency and complex joins, and NoSQL for large-scale, flexible, high-speed applications.



SECTION C Answers (≈500 Words Each)

Q1. Brief about the main components of MapReduce. (Jan 2025)

MapReduce is a programming model and framework for processing very large data sets in parallel on a Hadoop cluster. A job is divided into small tasks that run close to the data, and results are combined. Its working is based on key-value pairs: input (k1, v1) → Map → list(k2, v2) → Shuffle & sort → Reduce → list(k3, v3).



MapReduce data flow

Main components

1. **Input format and splits:** the input file in HDFS is divided into **input splits** (usually one HDFS block, 128 MB). A **RecordReader** converts each split into (key, value) records.
2. **Mapper:** processes each record and emits intermediate (key, value) pairs. One map task runs per split, in parallel, on the node that stores it (**data locality**).
3. **Combiner (optional):** a local "mini-reducer" that merges map output on the same node, reducing network traffic.
4. **Partitioner:** decides which reducer receives each key, normally hash(key) mod number of reducers, so that all values of one key go to the same reducer.
5. **Shuffle and sort:** the intermediate data is transferred to reducers, sorted and grouped by key, giving (key, list of values).
6. **Reducer:** takes each key with its values and aggregates them (sum, count, max) to produce the final output.
7. **Output format and RecordWriter:** write the reducer results to HDFS.

Execution framework (daemons)

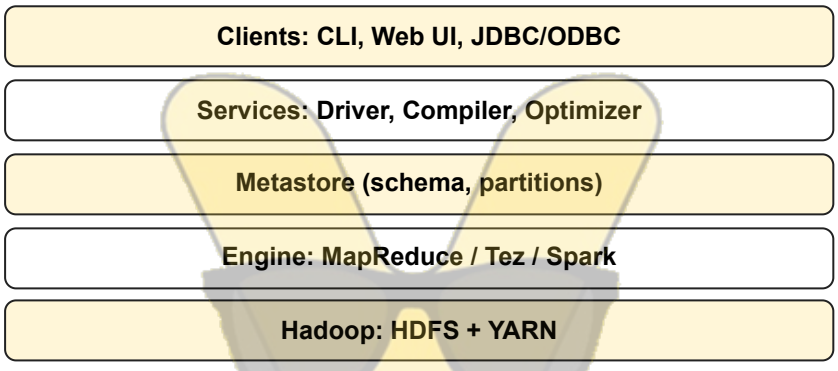
- **JobTracker / ResourceManager:** accepts the job, schedules tasks and monitors them.
- **TaskTracker / NodeManager:** runs the map and reduce tasks on each worker node and reports progress.
- Failed tasks are re-run on another node (fault tolerance).

Example – word count: Map emits (word, 1) for every word; shuffle groups all 1s for the same word; Reduce adds them to give (word, total).

Conclusion: MapReduce splits work into map and reduce phases supported by combiner, partitioner and shuffle components, giving scalable and fault-tolerant big-data processing.

Q2. Draw the architecture of HIVE with its features. / Draw and discuss the architecture of Hive in detail. (Jan 2025, 2023)

Apache Hive is a data-warehouse system built on top of Hadoop that lets users query and analyse large data stored in HDFS using **HiveQL**, an SQL-like language. HiveQL queries are converted into MapReduce (or Tez/Spark) jobs. It was developed at Facebook and is used for batch analytics, not for real-time transactions.



Hive architecture

Components of the architecture

1. **Hive clients:** the **CLI/Beeline**, web interface, and **JDBC/ODBC/Thrift** applications through which users submit HiveQL.
2. **Hive Server (HiveServer2):** accepts client requests and passes them to the driver; supports many concurrent clients.
3. **Driver:** receives the query, creates a session, and manages the life cycle of query compilation, optimisation and execution.
4. **Compiler:** parses the HiveQL, checks semantics using metastore information, and produces a logical plan (a DAG of stages).
5. **Optimizer:** improves the plan, e.g. pushes filters down, joins efficiently, prunes partitions.
6. **Execution engine:** runs the plan as MapReduce, Tez or Spark jobs and sends results back.
7. **Metastore:** a central database (Derby, MySQL) that stores metadata – table definitions, columns, data types, partitions and HDFS locations.
8. **HDFS and YARN:** store the real data and schedule cluster resources.

Query flow: client → driver → compiler (asks metastore) → optimizer → execution engine → Hadoop (MapReduce on HDFS) → results back to client.

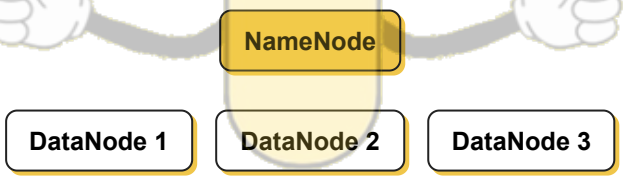
Features of Hive

- **SQL-like language (HiveQL)** – easy for database users.
- **Scalable** for petabytes using Hadoop's distributed storage.
- Supports **tables, partitions and buckets** for faster queries.
- **Schema-on-read** – schema applied when data is read; handles structured and semi-structured data.
- **User-defined functions (UDFs)** and many file formats (text, ORC, Parquet, Avro).
- **Fault tolerant** and open source; integrates with BI tools.
- Limitation: high latency, so not suitable for OLTP or row-level updates.

Conclusion: Hive hides MapReduce complexity behind a familiar SQL interface, with the metastore, driver, compiler and execution engine working together to run queries over data in HDFS.

Q3. Explain the working of Hadoop Distributed File System. (2022)

HDFS is the storage layer of Hadoop. It is a **distributed, fault-tolerant file system** designed to store very large files (GBs to PBs) across clusters of cheap commodity machines, following a **write-once, read-many** model with streaming access.



HDFS master-slave architecture

Architecture (master–slave)

- **NameNode (master):** stores the **metadata** – directory tree, file names, permissions, and the mapping of each file's blocks to DataNodes. It keeps the namespace in memory (FsImage and EditLog on disk).
- **DataNodes (slaves):** store the actual **blocks** of data and serve read/write requests. They send **heartbeats** (every 3 seconds) and **block reports** to the NameNode.
- **Secondary NameNode / Standby NameNode:** periodically merges the FsImage and EditLog (checkpoint); the standby gives high availability.

- **Block:** files are split into large blocks (default **128 MB**), each replicated, by default, **3 times** on different nodes and racks.

Write operation

1. The client asks the NameNode to create a file.
2. The NameNode checks permissions and returns the DataNodes for the first block.
3. The client streams the data to the first DataNode, which forwards it to the second and then the third (**replication pipeline**).
4. Acknowledgements travel back; the client repeats for each block and then tells the NameNode the file is complete.

Read operation

1. The client asks the NameNode for the block locations of the file.
2. The NameNode returns the nearest DataNodes for each block.
3. The client reads blocks **directly from the DataNodes** and joins them.

Fault tolerance

- If a DataNode stops sending heartbeats, the NameNode marks it dead and **re-replicates** its blocks onto other nodes.
- **Rack awareness** places replicas on different racks to survive rack failure.
- Checksums detect corrupted blocks, which are replaced from another replica.

Features: scalability, high throughput, data locality, low cost, reliability through replication.

Conclusion: HDFS splits files into replicated blocks stored on DataNodes, while the NameNode manages the metadata, giving reliable, scalable storage for big data.

Q4. List and explain five R functions used in Descriptive Statistics. (2022)

Descriptive statistics summarise the main features of a data set. R provides built-in functions for this. Let `x <- c(12, 15, 11, 18, 14, 15, 20)`.

1. **`mean(x)` – average.** Sum divided by the number of values. `mean(x)` gives 15. Option `na.rm = TRUE` ignores missing values. Measures the central tendency.
2. **`median(x)` – middle value.** Value at the centre of the sorted data; here sorted 11, 12, 14, 15, 15, 18, 20 → **15**. It is not affected by outliers. (R has no built-in

function for the statistical mode; `table()` is used.)

3. **`sd(x)` and `var(x)` – spread.** `var(x)` gives the sample variance and `sd(x)` its square root, the **standard deviation**; they show how far values lie from the mean. Here `sd(x)` ≈ 3.08.
4. **`range(x)`, `min(x)`, `max(x)` – extent.** `range(x)` returns the minimum and maximum (11 20); the **IQR** is given by `IQR(x)`. Used to see the spread and to spot extremes.
5. **`quantile(x)` – percentiles.** `quantile(x)` returns 0%, 25%, 50%, 75% and 100% points (11, 13, 15, 16.5, 20); `quantile(x, 0.9)` gives the 90th percentile. It is the base of box plots.

Bonus – `summary(x)` prints min, 1st quartile, median, mean, 3rd quartile and max in one call, and **`table(x)`** gives frequency counts, and `cor(x, y)` gives the correlation.

Function	Describes
mean()	Central value
median()	Middle value
sd(), var()	Dispersion
range(), IQR()	Spread
quantile()	Position / percentiles

Example output: `summary(x)` → Min 11, 1st Qu. 13, Median 15, Mean 15, 3rd Qu. 16.5, Max 20.

Conclusion: these functions give quick numerical summaries – centre, spread and position – and are the first step of any data analysis in R.

Q5. Explain Apache Hadoop, KNIME & Open Refine in detail. (2023)

(a) Apache Hadoop. An open-source framework for **distributed storage and processing of big data** on clusters of commodity hardware, created by Doug Cutting.

- **HDFS:** distributed file system, with a NameNode and DataNodes, replicating 128 MB blocks.
- **MapReduce:** parallel programming model with Map and Reduce phases.
- **YARN:** resource manager and job scheduler.
- **Hadoop Common:** shared libraries.

- **Ecosystem:** Hive (SQL), Pig (scripts), HBase (NoSQL), Sqoop (import/export), Flume, Spark, Oozie, ZooKeeper.
- **Features:** scalable, fault tolerant, cost effective, flexible for any data type, data locality.
- **Uses:** log analysis, recommendation engines, search indexing, fraud detection.

(b) KNIME (Konstanz Information Miner). An open-source, **visual (drag-and-drop) data analytics platform**.

- Users build **workflows** by connecting **nodes** (data input, cleaning, transformation, modelling, visualisation) with lines, with no or little coding.
- Supports machine learning (classification, clustering, regression), text mining, and integrates with R, Python, Weka, Spark and Hadoop, databases and cloud.
- Allows data blending from files, databases, and web services; gives interactive views and reports.
- Used for data preparation, ETL, predictive modelling and teaching.

(c) OpenRefine. (formerly Google Refine) An open-source desktop tool for **cleaning and transforming messy data**.

- Loads CSV, Excel, JSON, XML; works in a browser interface on local data.
- **Facets and filters** explore the data; **clustering** finds and merges variations of the same value ("N.Y.", "New York"); **split/merge columns**, remove duplicates, and transform with the **GREL** expression language.
- Can **reconcile** data with external databases such as Wikidata and add columns from web services; keeps an undo/redo history and exports the cleaned data.

Tool	Main purpose	Interface	Strength
Hadoop	Storage and batch processing of big data	Command line, APIs	Scalability, fault tolerance
KNIME	Workflow-based analytics and ML	Drag-and-drop GUI	No-code modelling
OpenRefine	Data cleaning	Browser GUI	Clustering and transforming messy data



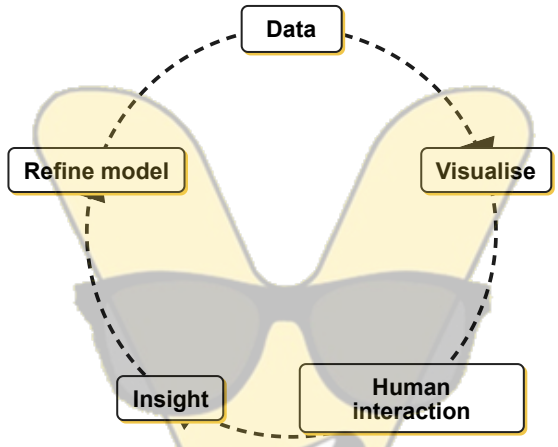
Complementary roles of the three tools

Conclusion: Hadoop provides the big-data platform, KNIME offers visual analytics workflows and OpenRefine prepares clean data, and together they cover storage,

preparation and analysis.

Q6. What are the approaches to integrate the human in data exploration process to realize different types of approaches to visual data mining? (2023)

Visual data mining combines the computer's processing power with the human's ability to see patterns, so that the analyst takes an active part in exploring data. The "visual analytics" loop is: data → visualisation → user interaction → hypothesis → refined model/visualisation.



Human in the visual exploration loop

Ways of integrating the human (Keim's classification by when the human is involved)

1. **Visualisation of the data mining results (mine first, then visualise).** An algorithm (clustering, association rules) is run automatically, and the user explores the *results* visually, e.g. the dendrogram of clusters. The human only interprets.
2. **Visualisation before data mining (visualise first, then mine).** The raw data is shown (scatter plots, parallel coordinates), the user spots interesting regions or selects features, and the mining algorithm is then run on the chosen subset.
3. **Tight integration (visualisation and mining together).** The user interacts with the visualisation while the algorithm runs: changes parameters, selects points, and the model is recomputed and redrawn at once. This is the **visual analytics** approach and gives the strongest human involvement.

4. **Visual data exploration without algorithms.** Following Shneiderman's mantra "**overview first, zoom and filter, then details on demand**", the user browses the data by direct manipulation.

Interaction techniques that bring the human in

- **Dynamic projection** (changing views of high-dimensional data) and **interactive filtering** (querying and selecting).
- **Zooming** and **distortion** (focus + context, fisheye).
- **Linking and brushing:** selecting items in one view highlights them in all other views.
- **Details on demand** and annotation of findings.

Types of visualisation techniques used

Technique	Example
Geometric	Scatter-plot matrices, parallel coordinates
Pixel-oriented	Recursive pattern, circle segments
Icon-based	Chernoff faces, star glyphs
Hierarchical	Treemaps, cone trees
Graph-based	Network and node-link diagrams

Benefits: exploits human perception and domain knowledge, works well when the goal is not exactly known, finds unexpected patterns and outliers, and increases trust in results.

Conclusion: the human can be placed after, before, or inside the mining process, and tight integration with interactive visual tools gives the most effective visual data mining.
