



SCHOOL OF COMPUTING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – III – SCSA1603 – BIG DATA ANALYTICS

UNIT III

BIG DATA TOOLS-II

Introduction to Modern databases-No SQL, New SQL, No SQLVs RDBMS databases
Tradeoffs, Working with MongoDB, Data warehouse system for Hadoop.

NO SQL

NoSQL can be defined as a database which is employed for managing the massive collection of unstructured data and when your data is not piled up in a tabular format or relations like that of relational databases. The term NoSQL came from the word non SQL or non-relational. There are a wide variety of existing Relational Databases that have been unsuccessful in solving several complex modern problems such as:

A dynamic change in the nature of data - i.e., nowadays data are in structured, semi-structured, non-structured as well as polymorphic in type. The variety of applications and the type of data feed into them for analysis has now become more diverse and distributed and is approaching cloud-oriented. Also, modern applications and services are serving tens of thousands of users in diverse geo- locations, having diverse time zones. So data integrity needs to be there at all the time.

Data residing in multiple virtual servers and other cloud storage (remote-based) in the cloud infrastructure can be easily analyzed using the NoSQL database management techniques and largely when the data set is in a non-structured manner. So, it can be said that the NoSQL database is intended to overcome the diversity of data, increase performance, modeling of data, scalability, and distribution, which is usually encountered in the Relational Databases.

A NoSQL database includes simplicity of design, simpler horizontal scaling to clusters of machines and finer control over availability. The data structures used by NoSQL databases are different from those used by default in relational databases which makes some operations faster in NoSQL.

When should NoSQL be used:

1. When huge amount of data need to be stored and retrieved.
2. The relationship between the data you store is not that important
3. The data changing over time and is not structured.
4. Support of Constraints and Joins is not required at database level
5. The data is growing continuously and you need to scale the database regular to handle the data.

ADVANTAGES OF NOSQL:

There are many advantages of working with NoSQL databases such as MongoDB and

Big Data Tools-II

Cassandra. The main advantages are high scalability and high availability.

1. High scalability–

NoSQL database use sharding for horizontal scaling. Partitioning of data and placing it on multiple machines in such a way that the order of the data is preserved is sharding. Vertical scaling means adding more resources to the existing machine whereas horizontal scaling means adding more machines to handle the data. Vertical scaling is not that easy to implement but horizontal scaling is easy to implement. Examples of horizontal scaling databases are MongoDB, Cassandra etc. NoSQL can handle huge amount of data because of scalability, as the data grows NoSQL scale itself to handle that data in efficient manner.

2. High availability–

Auto replication feature in NoSQL databases makes it highly available because in case of any failure data replicates itself to the previous consistent state.

DISADVANTAGES OF NOSQL:

NoSQL has the following disadvantages.

1. **Narrow focus** – NoSQL databases have very narrow focus as it is mainly designed for storage but it provides very little functionality. Relational databases are a better choice in the field of Transaction Management than NoSQL.
2. **Open-source**–NoSQL is open-source database. There is no reliable standard for NoSQL yet. In other words two database systems are likely to be unequal.
3. **Management challenge** –The purpose of big data tools is to make management of a large amount of data as simple as possible. But it is not so easy. Data management in NoSQL is much more complex than a relational database. NoSQL, in particular, has a reputation for being challenging to install and even more hectic to manage on a daily basis.
4. **GUI is not available** –GUI mode tools to access the database is not flexibly available in the market.
5. **Backup** –Backup is a great weak point for some NoSQL databases like MongoDB. MongoDB has no approach for the backup of data in a consistent manner.

Big Data Tools-II

6. **Large document size** –Some database systems like MongoDB and CouchDB store data in JSON format. Which means that documents are quite large (BigData, network bandwidth, speed),and having descriptive key names actually hurts, since they increase the documentsize.

TYPES OF NOSQL DATABASE:

Types of NoSQL databases and the name of the databases system that falls in that category are:

1. **MongoDB** falls in the category of NoSQL document baseddatabase.
 2. **Key value store:** Memcached, Redis,Coherence
 3. **Tabular:** Hbase, Big Table,Accumulo
 4. **Document based:** MongoDB, CouchDB,Cloudant
- **Document databases** store data in documents similar to JSON (JavaScript Object Notation) objects. Each document contains pairs of fields and values. The values can typically be a variety of types including things like strings, numbers, booleans, arrays, or objects, and their structures typically align with objects developers are working with in code. Because of their variety of field value types and powerful query languages, document databases are great for a wide variety of use cases and can be used as a general purpose database. They can horizontally scale-out to accomodate large data volumes. MongoDB is consistently ranked as the world's most popular NoSQL database according to DB-engines and is an example of a document database.
 - **Key-value databases** are a simpler type of database where each item contains keys and values. A value can typically only be retrieved by referencing its value, so learning how to query for a specific key-value pair is typically simple. Key-value databases are great for use cases where you need to store large amounts of data but you don't need to perform complex queries to retrieve it. Common use cases include storing user preferences or caching. Redis and DynanoDB are popular key-valuedatabases.
 - **Wide-column stores** store data in tables, rows, and dynamic columns. Wide-column stores provide a lot of flexibility over relational databases because each row is not required to have the same columns. Many consider wide-column stores to be two-dimensional key-value databases. Wide-column stores are great for when you need to store large amounts of data and you can predict what your query patterns will be. Wide-column stores are commonly used for storing Internet of Things data and user profile data. Cassandra and HBase are two of the most popular wide-columnstores.
 - **Graph databases** store data in nodes and edges. Nodes typically store information about people, places, and things while edges store information about the relationships between the nodes. Graph databases excel in use cases where you need

Big Data Tools-II

to traverse relationships to look for patterns such as social networks, fraud detection, and recommendation engines. Neo4j and JanusGraph are examples of graphdatabases.

BENEFITS:

- It allows developers to create large volumes of structured, semi-structured as well as unstructured data for making the application diverse and not restricting its use because of the type of data being used within the application.
- It also allows agile development; rapid iteration along with frequent code pushes, which makes it more popular.
- It can be used with object-oriented programming (OOP), which makes it easy to use with flexibility.
- Data can be stored more efficiently, making it less expensive, providing massive architecture.

NEW SQL:

NewSQL is a set of **new SQL databases** engines with high-performance and scalability. The term **NewSQL** was first used by analyst Matthew Aslett in 2011 in this “NoSQL, **NewSQL** and Beyond” (Aslett, 2011) business analysis report, which discussed the emergence of new **databases** systems

NewSQL databases solve database problems without giving up the advantages of traditional databases. NewSQL relational database management systems provide the same scalable performance of NoSQL systems for OLTP - online transaction processing read-write workloads while maintaining the ACID guarantees of a traditional database system. The data structures used by NoSQL databases differ from those used in relational databases, and that makes some operations faster in NoSQL databases..

Top NewSQL Databases: ClustrixDB, NuoDB, CockroachDB, Pivotal GemFire XD, Altibase, MemSQL, VoltDB, c-treeACE, Percona TokuDB, Apache Trafodion, TIBCO ActiveSpaces, ActorDB are some of the Top NewSQL Databases.

NewSQL databases are relational database systems that combine the OLTP, high performance, and scalability of NoSQL. They maintain the ACID (Atomicity, Consistency, Isolation, and Durability) guarantees of traditional DBMS. ACID transactions ensure complete business processes, concurrent transactions, data survival in case of system failures or errors, and consistency before and after a transaction.

NewSQL databases differ in terms of their internal design, but all of them are RDBMSs that run on SQL. They use SQL to ingest new information, execute many transactions at the same time, and modify the contents of the database. The main categories of NewSQL systems include new architectures, transparent sharding middleware, SQL engines, and Database-as-a-Service (DBaaS).

Big Data Tools-II

Partitioning/Sharding: Almost all NewSQL database management systems scale out by dividing the database into separate subsets known as partitions or shards. The tables are horizontally split into several fragments with boundaries based on column values. Related fragments from different tables are joined to create a partition.

Replication: This feature allows database users to create and maintain copies of a database or a part of a database. Copies of the database are stored in a remote site next to the main site or in a distant site. Users can update the replicas simultaneously or update one node and transfer the resultant state to other replicas.

Secondary Indexes: Secondary indexes allow database users to efficiently access database records by using a different value other than the primary key.

Concurrency Control: This feature addresses the problems that might occur in a multiuser system when many user access or modify data simultaneously. NewSQL systems use this feature to ensure simultaneous transactions while maintaining data integrity.

Crash Recovery: NewSQL databases have a mechanism that enables them to recover data and move to a consistent state when the system crashes.

Some of the benefits include:

Database partitioning reduces the system's communication overhead making it possible to access data with ease.

ACID transactions ensure data integrity even if there is a system failure or error. NewSQL databases can handle complex data. NewSQL systems are highly scalable.

NewSQL is a new approach to relational databases that wants to combine transactional ACID (atomicity, consistency, isolation, durability).



1. Atomicity means that a transaction must exhibit "all or nothing" behavior. Either all of the instructions within the transaction happen, or none of them happen. Atomicity preserves

Big Data Tools-II

the “completeness” of the business process.

2. Consistency refers to the state of the data both before and after the transaction is executed. A transaction maintains the consistency of the state of the data. In other words, after a transaction is run, all data in the database is “correct.”
3. Isolation means that transactions can run at the same time. Any transactions running in parallel have the illusion that there is no concurrency. In other words, it appears that the system is running only a single transaction at a time. No other concurrent transaction has visibility to the uncommitted database modifications made by any other transactions. To achieve isolation, a locking mechanism is required.
4. Durability refers to the impact of an outage or failure on a running transaction. A durable transaction will not impact the state of data if the transaction ends abnormally. The data will survive any failures.

NOSQL VS RDBMS DATABASES TRADEOFFS

RDBMS

- **Stands for Relational Database Management System**
- It is completely a structured way of storing data.
- The amount of data stored in RDBMS depends on physical memory of the system or in other words it is vertically scalable.
- In RDBMS schema represents logical view in which data is organized and tells how the relation are associated.
- It is a mixture of open and closed development models. like oracle, apache and soon.
- RDBMS databases are table based databases This means that SQL databases represent data in form of tables which consists of n number of rows of data
- RDBMS have predefined schemas.
- For defining and manipulating the data RDBMS use structured query language i.e. SQL which is very powerful.
- **RDBMS database examples:** MySql, Oracle, Sqlite, Postgres and MS-SQL.
- RDBMS database is well suited for the complex queries as compared to NoSql.
- If we talk about the type of data then RDBMS are not best fit for hierarchical data storage
- **Scalability:** RDBMS database is vertically scalable so to manage the increasing load by increase in CPU, RAM, SSD on a single server.
- RDBMS is best suited for high transactional based application and its more stable and promise for the atomicity and integrity of the data.
- RDBMS support large scale deployment and get support from the vendors.

Big Data Tools-II

- **Properties:** ACID properties(Atomicity, Consistency, Isolation,Durability).

NOSQL

- **Stands for Not OnlySQL**
- It is completely a unstructured way of storing data.
- While in Nosql there is no limit you can scale it horizontally.
- Work on only open source development models.
- NoSQL databases are document based, key-value pairs, graph databases or wide-column stores.where as NoSQL databases are the collection of key-value pair, documents, graph databases or wide-column stores which do not have standard schema definitions which it needs to adhered to.
- NoSql have dynamic schema with the un structured data.
- It uses UnQL i.e. unstructured query language and focused on collection of documents and vary from database to database.
- **NoSQL database examples:** MongoDB, BigTable, Redis, RavenDb, Cassandra, Hbase, Neo4j and CouchDb
- NoSql is note well suited for complex queries on high level it dosenot have standard interfaces to perform thatqueries.
- NoSql is best bit for hierarchical data storage because it follows the key-value pair way of data similar to JSON. Hbase is the example for thesame.
- **Scalability:** as we know Nosql database is horizontally scalable so to handle the large traffic you can add few servers to supportthat.
- NoSql is still rely on community support and for large scale NoSql deployment only limited experts areavailable.
- **Properties:** Follow Brewers CAP theorem(Consistency, Availability and Partition tolerance).



RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS)

RDBMS Database is a relational database. It is the standard language for relational database management systems. Data is stored in the form of rows and columns in RDBMS. The relations among tables are also stored in the form of the table SQL (Structured query Language) is a programming language used to perform tasks such as update data on a database, or to retrieve data from a database. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, etc.

FEATURES OF RDBMS

1. SQL databases are table based databases
2. Data store in rows and columns
3. Each row contains a unique instance of data for the categories defined by the columns.
4. Provide facility primary key, to uniquely identify the rows

LIMITATIONS FOR SQL DATABASE

Scalability: Users have to scale relational database on powerful servers that are expensive and difficult to handle. To scale relational database it has to be distributed on to multiple servers. Handling tables across different servers is difficult.

Complexity: In SQL server's data has to fit into tables anyhow. If your data doesn't fit into tables, then you need to design your database structure that will be complex and again difficult to handle.

NOSQL

NoSQL commonly referred to as "Not Only SQL". With NoSQL, unstructured, schema less data can be stored in multiple collections and nodes and it does not require fixed table schemas, it supports limited join queries, and we scale it horizontally.

BENEFITS OF NOSQL

Highly And Easily Scalable

Relational database or RDBMS databases are vertically Scalable. When load increases on RDBMS database then we scale database by increasing server hardware power, need to buy expensive and bigger servers and NoSQL databases are designed to expand horizontally and in Horizontal scaling means that you scale by adding more machines into your pool of resources.

Maintaining NoSQL Servers is Less Expensive

Big Data Tools-II

Maintaining high-end RDBMS systems is expensive and need trained manpower for database management but NoSQL databases require less management. it support many Features like automatic repair, easier data distribution, and simpler data models make administration and tuning requirements lesser in NoSQL.

Lesser Server Cost and open-Source

NoSQL databases are cheap and open source. NoSql database implementation is easy and typically uses cheap servers to manage the exploding data and transaction while RDBMS databases are expensive and it uses big servers and storage systems. So the storing and processing data cost per gigabyte in the case of NoSQL can be many times lesser than the cost of RDBMS.

No Schema or Fixed Data model

NoSQL database is schema less so Data can be inserted in a NoSQL database without any predefined schema. So the format or data model can be changed any time, without application disruption.and change management is a big headache in SQL.

Support Integrated Caching

NoSQL database support caching in system memory so it increase data output performance and SQL database where this has to be done using separate infrastructure.

Limitations & disadvantage of NoSQL

1. NoSQL database is Open Source and Open Source at its greatest strength but at the same time its greatest weakness because there are not many defined standards for NoSQL databases, so no two NoSQL databases are equal
2. No Stored Procedures in mongodb (NoSql database).
3. GUI mode tools to access the database is not flexibly available in market
4. too difficult for finding nosql experts because it is latest technology and NoSQL developer are in learning mode

Big Data Tools-II

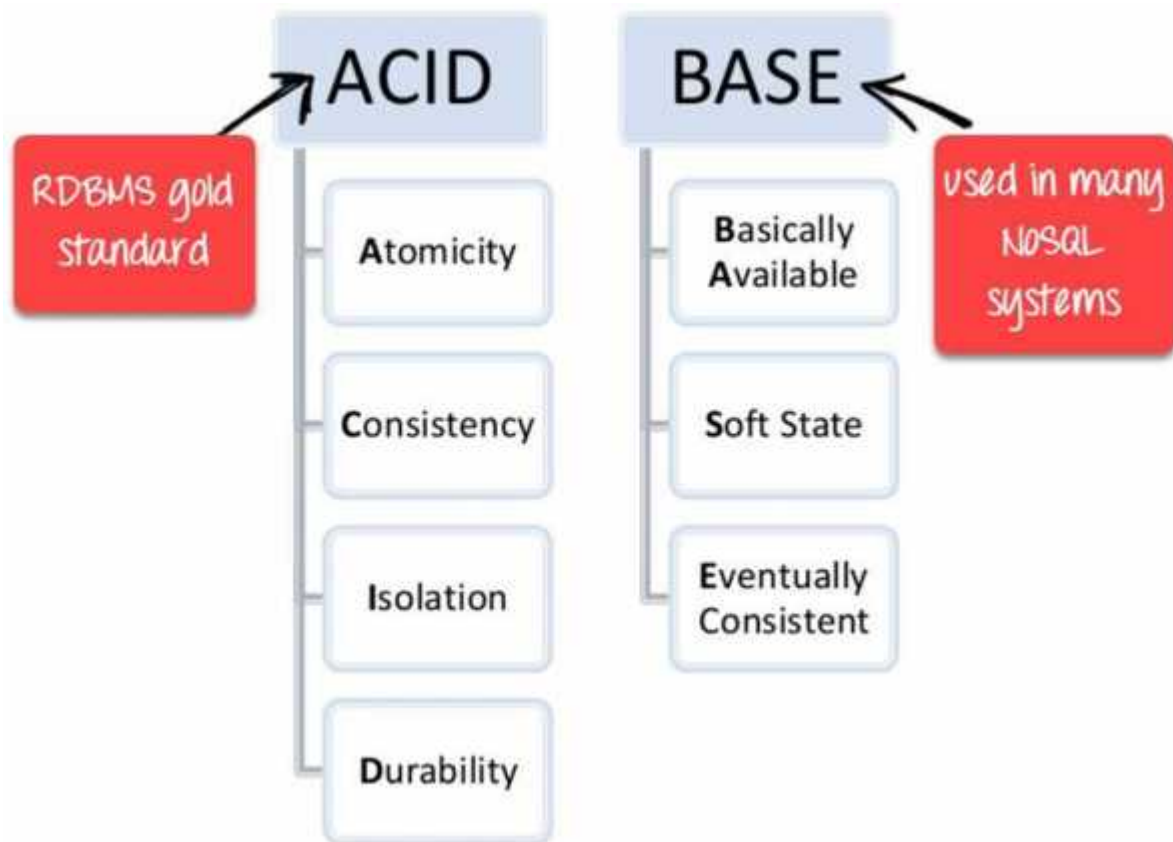
Parameter	SQL	NOSQL
Definition	SQL databases are primarily called RDBMS or Relational Databases	NoSQL databases are primarily called as Non-relational or distributed database
Design for	Traditional RDBMS uses SQL syntax and queries to analyze and get the data for further insights. They are used for OLAP systems.	NoSQL database system consists of various kind of database technologies. These databases were developed in response to the demands presented for the development of the modern application.
Query Language	Structured query language (SQL)	No declarative query language
Type	SQL databases are table based databases	NoSQL databases can be document based, key-value pairs, graph databases
Schema	SQL databases have a predefined schema	NoSQL databases use dynamic schema for unstructured data.
Ability to scale	SQL database is vertically scalable	NoSQL databases are horizontally scalable
Examples	Oracle, Postgres, and MS-SQL.	MongoDB, Redis, , Neo4j, Cassandra, Hbase.
Best suited for	An ideal choice for the complex query intensive environment.	It is not good fit complex queries.
Hierarchical data storage	SQL databases are not suitable for hierarchical data storage.	More suitable for the hierarchical data store as it supports key-value pair method.
Variations	One type with minor variations.	Many different types which include key-value stores, document databases, and graph databases.
Development Year	It was developed in the 1970s to deal with issues with flat file storage	Developed in the late 2000s to overcome issues and limitations of SQL databases.

Big Data Tools-II

Open-source	A mix of open-source like Postgres & MySQL, and commercial like OracleDatabase.	Open-source
Consistency	It should be configured for strong consistency.	It depends on DBMS as some offers strong consistency like MongoDB, whereas others offer only offers eventual consistency, likeCassandra.
Best Used for	RDBMS database is the right option for solving ACID problems.	NoSQL is a best used for solving data availability problems
Importance	It should be used when data validity is super important	Use when it's more important to have fast data than correctdata
Best option	When you need to support dynamic queries	Use when you need to scale based on changing requirements
Hardware	Specialized DB hardware (Oracle Exadata, etc.)	Commodity hardware
Network	Highly available network (Infiniband, Fabric Path,etc.)	Commodity network (Ethernet, etc.)
Storage Type	Highly Available Storage (SAN, RAID, etc.)	Commodity drives storage (standard HDDs, JBOD)
Best features	Cross-platform support, Secure and free	Easy to use, High performance, and Flexible tool.
Top Companies Using	Hootsuite, CircleCI, Gauges	Airbnb, Uber, Kickstarter
Average salary	The average salary for any professional SQL Developeris \$84,328 per year in the U.S.A.	The average salary for "NoSQL developer" ranges fromapproximately \$72,174 per year
ACID vs. BASEModel	ACID(Atomicity, Consistency, Isolation, and Durability) is a standard for RDBMS	Base (Basically Available, Soft state, Eventually Consistent) is a model of many NoSQL systems

Summary:

RDBMS and NoSQL both dbs are great in data management and both are used to keep data storage and retrieval optimized and smooth. It's hard to say which technology is better so developer take decision according requirement and situations



ACID VS BASE

MongoDB

MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

Database

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

Collection

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

Document

A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

The following table shows the relationship of RDBMS terminology with MongoDB.

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key <code>_id</code> provided by mongodb itself)
Database Server and Client	
Mysqld/Oracle	mongod
mysql/sqlplus	mongo

Sample Document

Following example shows the document structure of a blog site, which is simply a comma separated key value pair.

```
{
  _id: ObjectId(7df78ad8902c)
  title: 'MongoDB Overview',
  description: 'MongoDB is no sql database',
  by: 'tutorials point',
  url: 'http://www.tutorialspoint.com',
  tags: ['mongodb', 'database', 'NoSQL'],
  likes: 100,
```

```
comments: [
  {
    user:'user1',
    message: 'My first comment',
    dateCreated: new Date(2011,1,20,2,15),
    like: 0
  },
  {
    user:'user2',
    message: 'My second comments',
    dateCreated: new Date(2011,1,25,7,45),
    like: 5
  }
]
```

_id is a 12 bytes hexadecimal number which assures the uniqueness of every document. You can provide **_id** while inserting the document. If you don't provide then MongoDB provides a unique id for every document. These 12 bytes first 4 bytes for the current timestamp, next 3 bytes for machine id, next 2 bytes for process id of MongoDB server and remaining 3 bytes are simple incremental VALUE.

MongoDB - Advantages

Any relational database has a typical schema design that shows number of tables and the relationship between these tables. While in MongoDB, there is no concept of relationship.

Advantages of MongoDB over RDBMS

- **Schema less** – MongoDB is a document database in which one collection holds different documents. Number of fields, content and size of the document can differ from one document to another.
- Structure of a single object is clear.
- No complex joins.
- Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.
- Tuning.
- **Ease of scale-out** – MongoDB is easy to scale.
- Conversion/mapping of application objects to database objects not needed.
- Uses internal memory for storing the (windowed) working set, enabling faster access of data.

Why Use MongoDB?

- **Document Oriented Storage** – Data is stored in the form of JSON style documents.
- Index on any attribute
- Replication and high availability
- Auto-sharding

- Rich queries
- Fast in-place updates
- Professional support by MongoDB

Where to Use MongoDB?

- Big Data
- Content Management and Delivery
- Mobile and Social Infrastructure
- User Data Management
- Data Hub

MongoDB - Environment

Let us now see how to install MongoDB on Windows.

Install MongoDB On Windows

To install MongoDB on Windows, first download the latest release of MongoDB from <https://www.mongodb.org/downloads>. Make sure you get correct version of MongoDB depending upon your Windows version. To get your Windows version, open command prompt and execute the following command.

```
C:\>wmic os get osarchitecture
OSArchitecture
64-bit
C:\>
```

32-bit versions of MongoDB only support databases smaller than 2GB and suitable only for testing and evaluation purposes.

Now extract your downloaded file to c:\ drive or any other location. Make sure the name of the extracted folder is mongodb-win32-i386-[version] or mongodb-win32-x86_64-[version]. Here [version] is the version of MongoDB download.

Next, open the command prompt and run the following command.

```
C:\>move mongodb-win64-* mongodb
1 dir(s) moved.
C:\>
```

In case you have extracted the MongoDB at different location, then go to that path by using command **cd FOLDER/DIR** and now run the above given process.

MongoDB requires a data folder to store its files. The default location for the MongoDB data directory is c:\data\db. So you need to create this folder using the Command Prompt. Execute the following command sequence.

```
C:\>md data
C:\>md data\db
```

If you have to install the MongoDB at a different location, then you need to specify an alternate path for **\data\db** by setting the path **dbpath** in **mongod.exe**. For the same, issue the following commands.

In the command prompt, navigate to the bin directory present in the MongoDB installation folder. Suppose my installation folder is **D:\set up\mongodb**

```
C:\Users\XYZ>d:
D:\>cd "set up"
D:\set up>cd mongodb
D:\set up\mongodb>cd bin
D:\set up\mongodb\bin>mongod.exe --dbpath "d:\set up\mongodb\data"
```

This will show **waiting for connections** message on the console output, which indicates that the mongod.exe process is running successfully.

Now to run the MongoDB, you need to open another command prompt and issue the following command.

```
D:\set up\mongodb\bin>mongo.exe
MongoDB shell version: 2.4.6
connecting to: test
>db.test.save( { a: 1 } )
>db.test.find()
{ "_id" : ObjectId(5879b0f65a56a454), "a" : 1 }
>
```

This will show that MongoDB is installed and run successfully. Next time when you run MongoDB, you need to issue only commands.

```
D:\set up\mongodb\bin>mongod.exe --dbpath "d:\set up\mongodb\data"
D:\set up\mongodb\bin>mongo.exe
```

Install MongoDB on Ubuntu

Run the following command to import the MongoDB public GPG key –

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10
```

Create a /etc/apt/sources.list.d/mongodb.list file using the following command.

```
echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen'
| sudo tee /etc/apt/sources.list.d/mongodb.list
```

Now issue the following command to update the repository –

```
sudo apt-get update
```

Next install the MongoDB by using the following command –

```
apt-get install mongodb-10gen = 2.2.3
```

In the above installation, 2.2.3 is currently released MongoDB version. Make sure to install the latest version always. Now MongoDB is installed successfully.

Start MongoDB

```
sudo service mongodb start
```

Stop MongoDB

```
sudo service mongodb stop
```

Restart MongoDB

```
sudo service mongodb restart
```

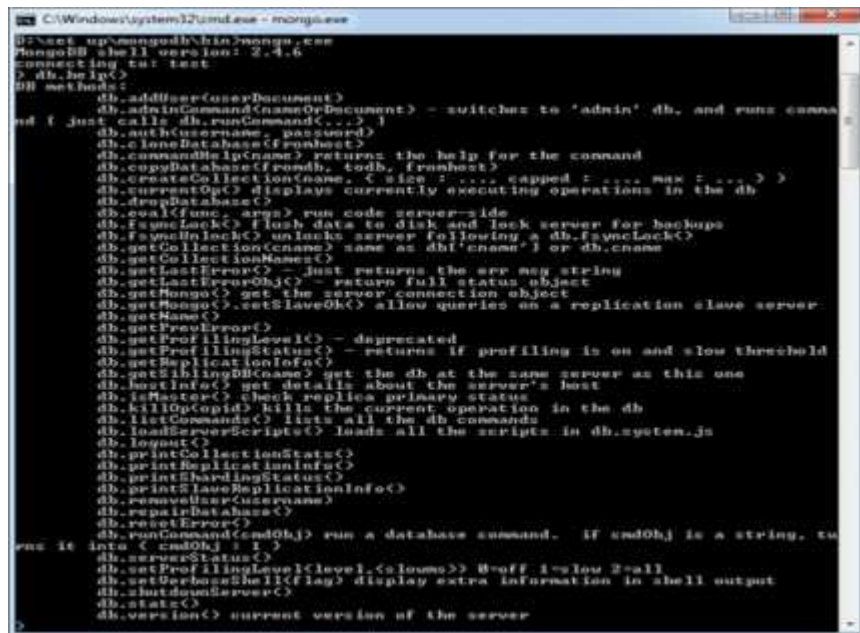
To use MongoDB run the following command.

mongo

This will connect you to running MongoDB instance.

MongoDB Help

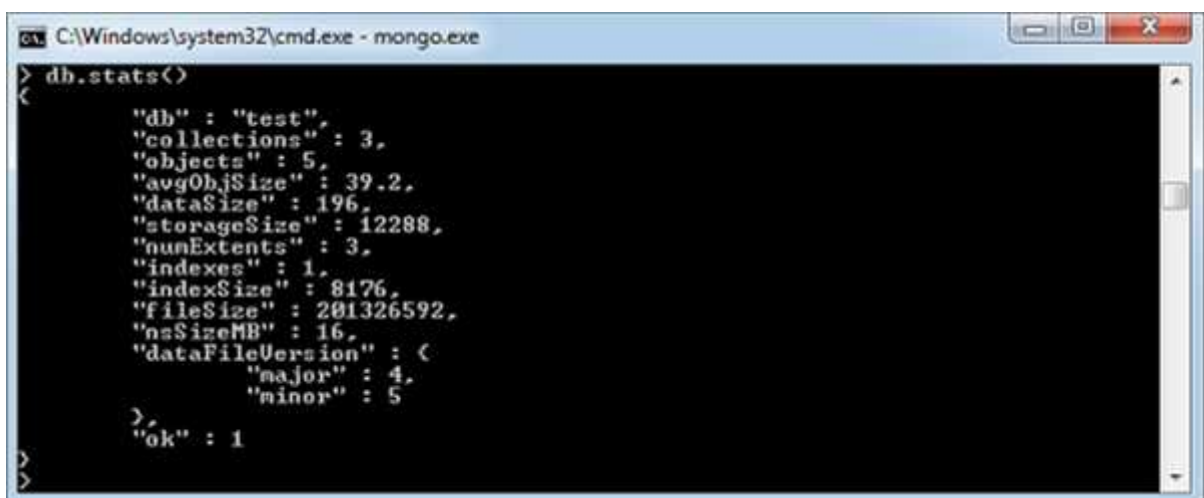
To get a list of commands, type **db.help()** in MongoDB client. This will give you a list of commands as shown in the following screenshot.

A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe - mongo.exe". The prompt shows the user has entered 'mongo' and the shell version is 2.4.6. The user then enters 'db.help()' and the output lists various MongoDB methods such as db.addUser, db.adminCommand, db.auth, db.cloneDatabase, db.command, db.copyDatabase, db.createCollection, db.currentOp, db.dropDatabase, db.eval, db.flush, db.fsync, db.fsyncLock, db.getCollection, db.getLastError, db.getLastErrorObj, db.getMongo, db.getSlave, db.getName, db.getPrevError, db.getProfilingLevel, db.getProfilingStatus, db.getReplicationInfo, db.getSiblingDB, db.isMaster, db.killOp, db.listCommands, db.loadServerScripts, db.logout, db.printCollectionStats, db.printReplicationInfo, db.printSlaveReplicationInfo, db.removeUser, db.repairDatabase, db.resetError, db.runCommand, db.serverStatus, db.setProfilingLevel, db.setVerboseShell, db.shutdownServer, db.stats, and db.version. The output is truncated with '...'.

```
C:\Windows\system32\cmd.exe - mongo.exe
mongo>
MongoDB shell version: 2.4.6
connecting to: test
> db.help()
DB methods:
  db.addUser(username, password) - adds a new user
  db.adminCommand(nameOrDocument) - switch to 'admin' db, and runs command
  db.auth(username, password) - authenticates user
  db.cloneDatabase(fromdb, todb, fromhost) - clones database
  db.command(name, <args>) - returns the help for the command
  db.copyDatabase(fromdb, todb, fromhost) - copies database
  db.createCollection(name, <size>: capped: true, max: 1000000) - creates collection
  db.currentOp() - displays currently executing operations in the db
  db.dropDatabase() - drops the database
  db.eval(func, args) - run code server-side
  db.flush() - flush data to disk and lock server for backups
  db.fsync() - flushes data to disk and locks server
  db.fsyncLock() - locks server following a db.fsync()
  db.getCollection(name) - returns collection object
  db.getCollectionNames() - returns array of collection names
  db.getLastError() - returns the last error message
  db.getLastErrorObj() - returns full error object
  db.getMongo() - get the server connection object
  db.getSlave() - get slave server
  db.getName() - returns the name of the server
  db.getPrevError() - returns the previous error
  db.getProfilingLevel() - deprecated
  db.getProfilingStatus() - returns if profiling is on and slow threshold
  db.getReplicationInfo() - returns replication info
  db.getSiblingDB(name) - get the db at the same server as this one
  db.isMaster() - get details about the server's host
  db.isMaster() - check replica primary status
  db.killOp(opid) - kills the current operation in the db
  db.listCommands() - lists all the db commands
  db.loadServerScripts() - loads all the scripts in db.system.js
  db.logout() - logs out the user
  db.printCollectionStats() - prints collection statistics
  db.printReplicationInfo() - prints replication info
  db.printSlaveReplicationInfo() - prints slave replication info
  db.removeUser(username) - removes user
  db.repairDatabase() - repairs the database
  db.resetError() - resets the error
  db.runCommand(cmdObj) - run a database command. If cmdObj is a string, then
  db.serverStatus() - returns server status
  db.setProfilingLevel(level, <columns>) - set profiling level
  db.setVerboseShell(flag) - display extra information in shell output
  db.shutdownServer() - shuts down the server
  db.stats() - returns database statistics
  db.version() - current version of the server
```

MongoDB Statistics

To get stats about MongoDB server, type the command **db.stats()** in MongoDB client. This will show the database name, number of collection and documents in the database. Output of the command is shown in the following screenshot.

A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe - mongo.exe". The prompt shows the user has entered 'mongo' and the shell version is 2.4.6. The user then enters 'db.stats()' and the output shows statistics for the 'test' database, including the number of collections, objects, data size, storage size, number of extents, indexes, index size, file size, ns size in MB, data file version, and the current version of the server. The output is as follows:

```
C:\Windows\system32\cmd.exe - mongo.exe
mongo>
MongoDB shell version: 2.4.6
connecting to: test
> db.stats()
{
  "db" : "test",
  "collections" : 3,
  "objects" : 5,
  "avgObjSize" : 39.2,
  "dataSize" : 196,
  "storageSize" : 12288,
  "numExtents" : 3,
  "indexes" : 1,
  "indexSize" : 8176,
  "fileSize" : 201326592,
  "nsSizeMB" : 16,
  "dataFileVersion" : {
    "major" : 4,
    "minor" : 5
  },
  "ok" : 1
}
```

MongoDB - Sharding

Sharding is the process of storing data records across multiple machines and it is MongoDB's approach to meeting the demands of data growth. As the size of the data increases, a single machine may not be sufficient to store the data nor provide an acceptable read and write

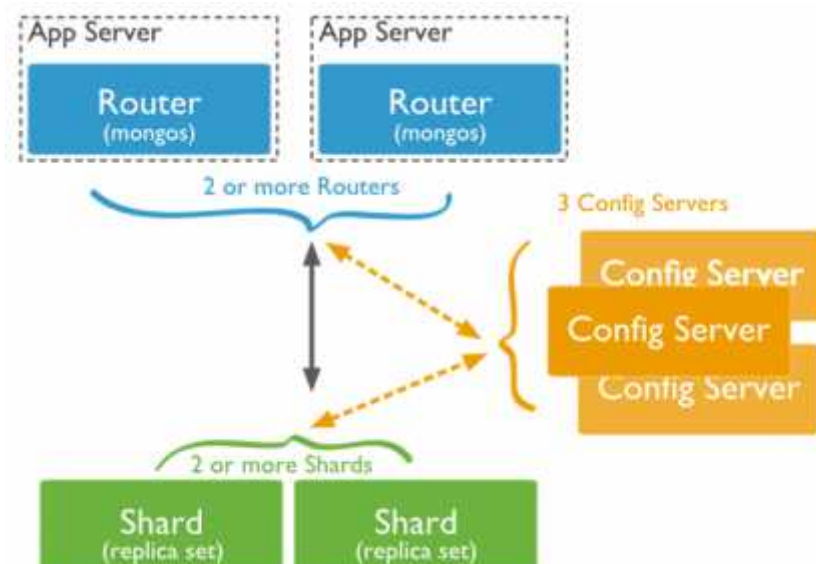
throughput. Sharding solves the problem with horizontal scaling. With sharding, you add more machines to support data growth and the demands of read and write operations.

Why Sharding?

- In replication, all writes go to master node
- Latency sensitive queries still go to master
- Single replica set has limitation of 12 nodes
- Memory can't be large enough when active dataset is big
- Local disk is not big enough
- Vertical scaling is too expensive

Sharding in MongoDB

The following diagram shows the sharding in MongoDB using sharded cluster.



In the following diagram, there are three main components –

- **Shards** – Shards are used to store data. They provide high availability and data consistency. In production environment, each shard is a separate replica set.
- **Config Servers** – Config servers store the cluster's metadata. This data contains a mapping of the cluster's data set to the shards. The query router uses this metadata to target operations to specific shards. In production environment, sharded clusters have exactly 3 config servers.
- **Query Routers** – Query routers are basically mongo instances, interface with client applications and direct operations to the appropriate shard. The query router processes and targets the operations to shards and then returns results to the clients. A sharded cluster can contain more than one query router to divide the client request load. A client sends requests to one query router. Generally, a sharded cluster have many query routers.

Apache Spark – Introduction:

Industries are using Hadoop extensively to analyze their data sets. The reason is that Hadoop framework is based on a simple programming model (MapReduce) and it enables a computing

solution that is scalable, flexible, fault-tolerant and cost effective. Here, the main concern is to maintain speed in processing large datasets in terms of waiting time between queries and waiting time to run the program.

Spark was introduced by Apache Software Foundation for speeding up the Hadoop computational computing software process.

As against a common belief, **Spark is not a modified version of Hadoop** and is not, really, dependent on Hadoop because it has its own cluster management. Hadoop is just one of the ways to implement Spark.

Spark uses Hadoop in two ways – one is **storage** and second is **processing**. Since Spark has its own cluster management computation, it uses Hadoop for storage purpose only.

Apache Spark

Apache Spark is a lightning-fast cluster computing technology, designed for fast computation. It is based on Hadoop MapReduce and it extends the MapReduce model to efficiently use it for more types of computations, which includes interactive queries and stream processing. The main feature of Spark is its **in-memory cluster computing** that increases the processing speed of an application.

Spark is designed to cover a wide range of workloads such as batch applications, iterative algorithms, interactive queries and streaming. Apart from supporting all these workload in a respective system, it reduces the management burden of maintaining separate tools.

Evolution of Apache Spark

Spark is one of Hadoop's sub project developed in 2009 in UC Berkeley's AMPLab by Matei Zaharia. It was Open Sourced in 2010 under a BSD license. It was donated to Apache software foundation in 2013, and now Apache Spark has become a top level Apache project from Feb-2014.

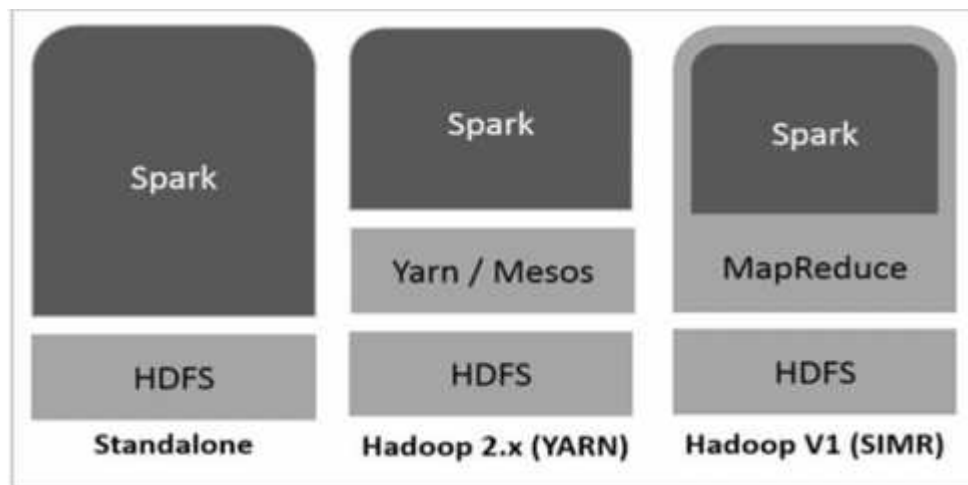
Features of Apache Spark

Apache Spark has following features.

- **Speed** – Spark helps to run an application in Hadoop cluster, up to 100 times faster in memory, and 10 times faster when running on disk. This is possible by reducing number of read/write operations to disk. It stores the intermediate processing data in memory.
- **Supports multiple languages** – Spark provides built-in APIs in Java, Scala, or Python. Therefore, you can write applications in different languages. Spark comes up with 80 high-level operators for interactive querying.
- **Advanced Analytics** – Spark not only supports 'Map' and 'reduce'. It also supports SQL queries, Streaming data, Machine learning (ML), and Graph algorithms.

Spark Built on Hadoop

The following diagram shows three ways of how Spark can be built with Hadoop components.

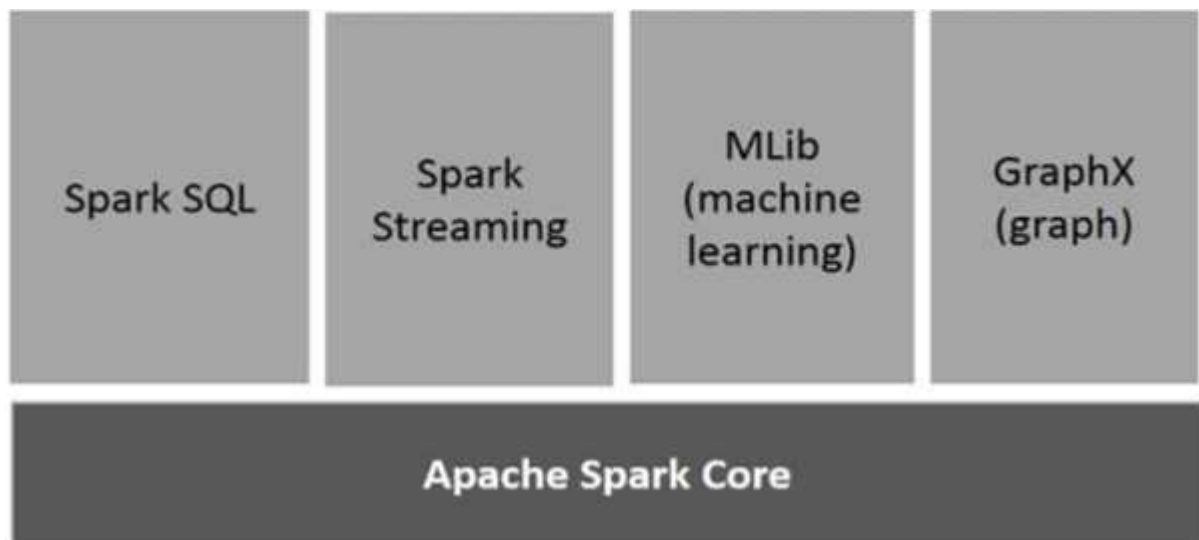


There are three ways of Spark deployment as explained below.

- **Standalone** – Spark Standalone deployment means Spark occupies the place on top of HDFS(Hadoop Distributed File System) and space is allocated for HDFS, explicitly. Here, Spark and MapReduce will run side by side to cover all spark jobs on cluster.
- **Hadoop Yarn** – Hadoop Yarn deployment means, simply, spark runs on Yarn without any pre-installation or root access required. It helps to integrate Spark into Hadoop ecosystem or Hadoop stack. It allows other components to run on top of stack.
- **Spark in MapReduce (SIMR)** – Spark in MapReduce is used to launch spark job in addition to standalone deployment. With SIMR, user can start Spark and uses its shell without any administrative access.

Components of Spark

The following illustration depicts the different components of Spark.



Apache Spark Core

Spark Core is the underlying general execution engine for spark platform that all other functionality is built upon. It provides In-Memory computing and referencing datasets in external storage systems.

Spark SQL

Spark SQL is a component on top of Spark Core that introduces a new data abstraction called SchemaRDD, which provides support for structured and semi-structured data.

Spark Streaming

Spark Streaming leverages Spark Core's fast scheduling capability to perform streaming analytics. It ingests data in mini-batches and performs RDD (Resilient Distributed Datasets) transformations on those mini-batches of data.

MLlib (Machine Learning Library)

MLlib is a distributed machine learning framework above Spark because of the distributed memory-based Spark architecture. It is, according to benchmarks, done by the MLlib developers against the Alternating Least Squares (ALS) implementations. Spark MLlib is nine times as fast as the Hadoop disk-based version of **Apache Mahout** (before Mahout gained a Spark interface).

GraphX

GraphX is a distributed graph-processing framework on top of Spark. It provides an API for expressing graph computation that can model the user-defined graphs by using Pregel abstraction API. It also provides an optimized runtime for this abstraction.

Apache Spark – RDD

Resilient Distributed Datasets

Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark. It is an immutable distributed collection of objects. Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster. RDDs can contain any type of Python, Java, or Scala objects, including user-defined classes.

Formally, an RDD is a read-only, partitioned collection of records. RDDs can be created through deterministic operations on either data on stable storage or other RDDs. RDD is a fault-tolerant collection of elements that can be operated on in parallel.

There are two ways to create RDDs – **parallelizing** an existing collection in your driver program, or **referencing a dataset** in an external storage system, such as a shared file system, HDFS, HBase, or any data source offering a Hadoop Input Format.

Spark makes use of the concept of RDD to achieve faster and efficient MapReduce operations. Let us first discuss how MapReduce operations take place and why they are not so efficient.

Data Sharing is Slow in MapReduce

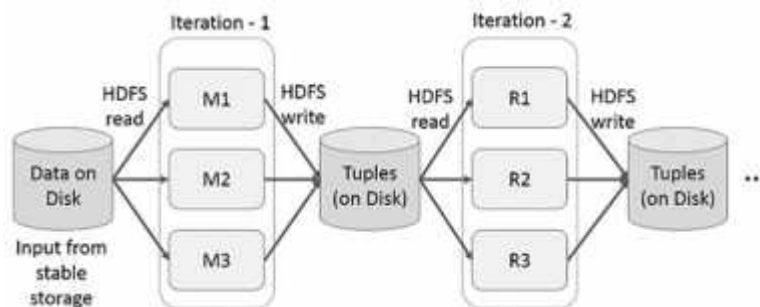
MapReduce is widely adopted for processing and generating large datasets with a parallel, distributed algorithm on a cluster. It allows users to write parallel computations, using a set of high-level operators, without having to worry about work distribution and fault tolerance.

Unfortunately, in most current frameworks, the only way to reuse data between computations (Ex – between two MapReduce jobs) is to write it to an external stable storage system (Ex – HDFS). Although this framework provides numerous abstractions for accessing a cluster's computational resources, users still want more.

Both **Iterative** and **Interactive** applications require faster data sharing across parallel jobs. Data sharing is slow in MapReduce due to **replication**, **serialization**, and **disk IO**. Regarding storage system, most of the Hadoop applications, they spend more than 90% of the time doing HDFS read-write operations.

Iterative Operations on MapReduce

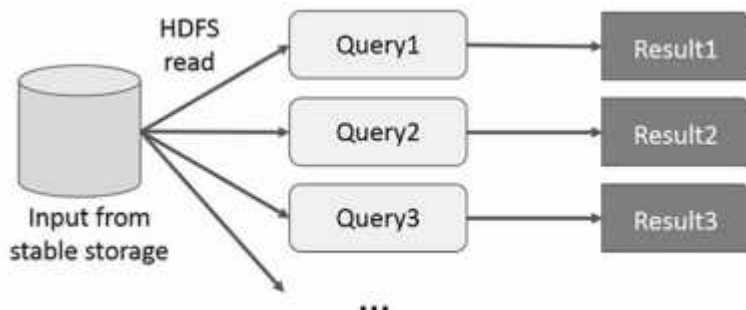
Reuse intermediate results across multiple computations in multi-stage applications. The following illustration explains how the current framework works, while doing the iterative operations on MapReduce. This incurs substantial overheads due to data replication, disk I/O, and serialization, which makes the system slow.



Interactive Operations on MapReduce

User runs ad-hoc queries on the same subset of data. Each query will do the disk I/O on the stable storage, which can dominate application execution time.

The following illustration explains how the current framework works while doing the interactive queries on MapReduce.



Data Sharing using Spark RDD

Data sharing is slow in MapReduce due to **replication**, **serialization**, and **disk IO**. Most of the Hadoop applications, they spend more than 90% of the time doing HDFS read-write operations.

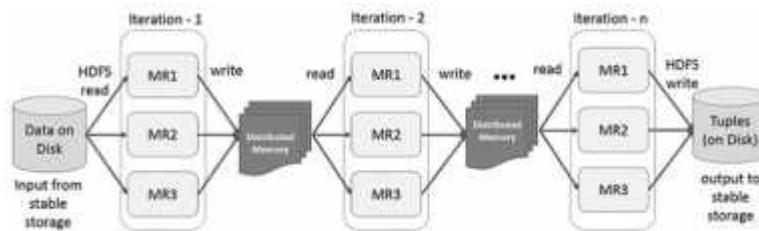
Recognizing this problem, researchers developed a specialized framework called Apache Spark. The key idea of spark is **Resilient Distributed Datasets (RDD)**; it supports in-memory processing computation. This means, it stores the state of memory as an object across the jobs and the object is sharable between those jobs. Data sharing in memory is 10 to 100 times faster than network and Disk.

Let us now try to find out how iterative and interactive operations take place in Spark RDD.

Iterative Operations on Spark RDD

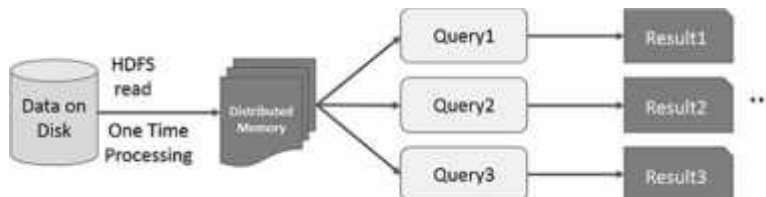
The illustration given below shows the iterative operations on Spark RDD. It will store intermediate results in a distributed memory instead of Stable storage (Disk) and make the system faster.

Note – If the Distributed memory (RAM) is not sufficient to store intermediate results (State of the JOB), then it will store those results on the disk.



Interactive Operations on Spark RDD

This illustration shows interactive operations on Spark RDD. If different queries are run on the same set of data repeatedly, this particular data can be kept in memory for better execution times.



By default, each transformed RDD may be recomputed each time you run an action on it. However, you may also **persist** an RDD in memory, in which case Spark will keep the elements around on the cluster for much faster access, the next time you query it. There is also support for persisting RDDs on disk, or replicated across multiple nodes.

Cassandra - Introduction

Apache Cassandra is a highly scalable, high-performance distributed database designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure. It is a type of NoSQL database. Let us first understand what a NoSQL database does.

NoSQL Database

A NoSQL database (sometimes called as Not Only SQL) is a database that provides a mechanism to store and retrieve data other than the tabular relations used in relational databases. These databases are schema-free, support easy replication, have simple API, eventually consistent, and can handle huge amounts of data.

The primary objective of a NoSQL database is to have

- simplicity of design,
- horizontal scaling, and
- finer control over availability.

NoSql databases use different data structures compared to relational databases. It makes some operations faster in NoSQL. The suitability of a given NoSQL database depends on the problem it must solve.

NoSQL vs. Relational Database

The following table lists the points that differentiate a relational database from a NoSQL database.

Relational Database	NoSql Database
Supports powerful query language.	Supports very simple query language.
It has a fixed schema.	No fixed schema.
Follows ACID (Atomicity, Consistency, Isolation, and Durability).	It is only “eventually consistent”.
Supports transactions.	Does not support transactions.

Besides Cassandra, we have the following NoSQL databases that are quite popular –

- **Apache HBase** – HBase is an open source, non-relational, distributed database modeled after Google’s BigTable and is written in Java. It is developed as a part of Apache Hadoop project and runs on top of HDFS, providing BigTable-like capabilities for Hadoop.
- **MongoDB** – MongoDB is a cross-platform document-oriented database system that avoids using the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas making the integration of data in certain types of applications easier and faster.

What is Apache Cassandra?

Apache Cassandra is an open source, distributed and decentralized/distributed storage system (database), for managing very large amounts of structured data spread out across the world. It provides highly available service with no single point of failure.

Listed below are some of the notable points of Apache Cassandra –

- It is scalable, fault-tolerant, and consistent.
- It is a column-oriented database.
- Its distribution design is based on Amazon's Dynamo and its data model on Google's Bigtable.
- Created at Facebook, it differs sharply from relational database management systems.
- Cassandra implements a Dynamo-style replication model with no single point of failure, but adds a more powerful "column family" data model.
- Cassandra is being used by some of the biggest companies such as Facebook, Twitter, Cisco, Rackspace, ebay, Twitter, Netflix, and more.

Features of Cassandra

Cassandra has become so popular because of its outstanding technical features. Given below are some of the features of Cassandra:

- **Elastic scalability** – Cassandra is highly scalable; it allows to add more hardware to accommodate more customers and more data as per requirement.
- **Always on architecture** – Cassandra has no single point of failure and it is continuously available for business-critical applications that cannot afford a failure.
- **Fast linear-scale performance** – Cassandra is linearly scalable, i.e., it increases your throughput as you increase the number of nodes in the cluster. Therefore it maintains a quick response time.
- **Flexible data storage** – Cassandra accommodates all possible data formats including: structured, semi-structured, and unstructured. It can dynamically accommodate changes to your data structures according to your need.
- **Easy data distribution** – Cassandra provides the flexibility to distribute data where you need by replicating data across multiple data centers.
- **Transaction support** – Cassandra supports properties like Atomicity, Consistency, Isolation, and Durability (ACID).
- **Fast writes** – Cassandra was designed to run on cheap commodity hardware. It performs blazingly fast writes and can store hundreds of terabytes of data, without sacrificing the read efficiency.

History of Cassandra

- Cassandra was developed at Facebook for inbox search.
- It was open-sourced by Facebook in July 2008.
- Cassandra was accepted into Apache Incubator in March 2009.

- It was made an Apache top-level project since February 2010.

Cassandra - Architecture

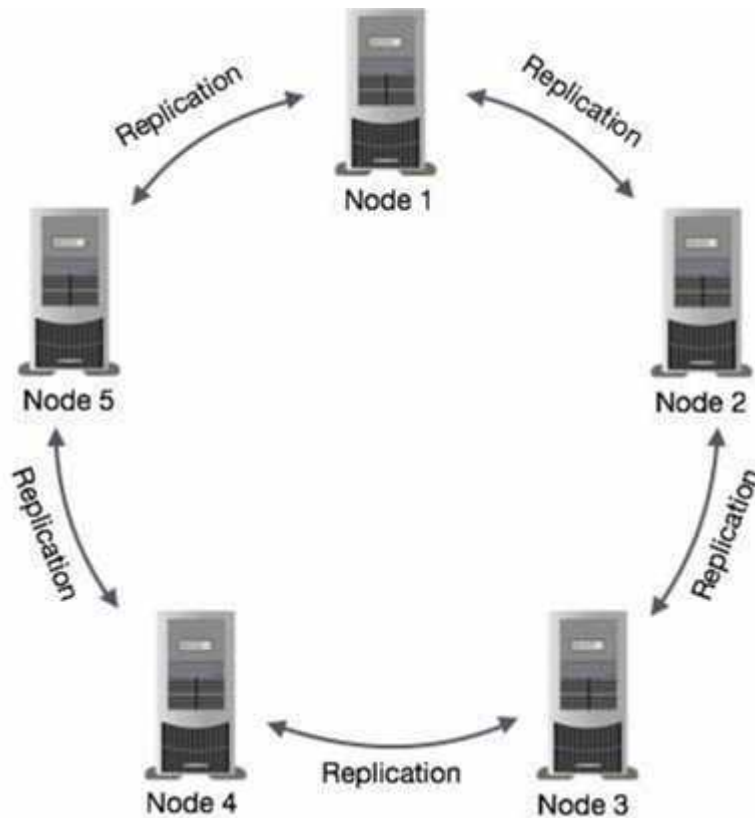
The design goal of Cassandra is to handle big data workloads across multiple nodes without any single point of failure. Cassandra has peer-to-peer distributed system across its nodes, and data is distributed among all the nodes in a cluster.

- All the nodes in a cluster play the same role. Each node is independent and at the same time interconnected to other nodes.
- Each node in a cluster can accept read and write requests, regardless of where the data is actually located in the cluster.
- When a node goes down, read/write requests can be served from other nodes in the network.

Data Replication in Cassandra

In Cassandra, one or more of the nodes in a cluster act as replicas for a given piece of data. If it is detected that some of the nodes responded with an out-of-date value, Cassandra will return the most recent value to the client. After returning the most recent value, Cassandra performs a **read repair** in the background to update the stale values.

The following figure shows a schematic view of how Cassandra uses data replication among the nodes in a cluster to ensure no single point of failure.



Note – Cassandra uses the **Gossip Protocol** in the background to allow the nodes to communicate with each other and detect any faulty nodes in the cluster.

Components of Cassandra

The key components of Cassandra are as follows –

- **Node** – It is the place where data is stored.
- **Data center** – It is a collection of related nodes.
- **Cluster** – A cluster is a component that contains one or more data centers.
- **Commit log** – The commit log is a crash-recovery mechanism in Cassandra. Every write operation is written to the commit log.
- **Mem-table** – A mem-table is a memory-resident data structure. After commit log, the data will be written to the mem-table. Sometimes, for a single-column family, there will be multiple mem-tables.
- **SSTable** – It is a disk file to which the data is flushed from the mem-table when its contents reach a threshold value.
- **Bloom filter** – These are nothing but quick, nondeterministic, algorithms for testing whether an element is a member of a set. It is a special kind of cache. Bloom filters are accessed after every query.

Cassandra Query Language

Users can access Cassandra through its nodes using Cassandra Query Language (CQL). CQL treats the database (**Keyspace**) as a container of tables. Programmers use **cqlsh**: a prompt to work with CQL or separate application language drivers.

Clients approach any of the nodes for their read-write operations. That node (coordinator) plays a proxy between the client and the nodes holding the data.

Write Operations

Every write activity of nodes is captured by the **commit logs** written in the nodes. Later the data will be captured and stored in the **mem-table**. Whenever the mem-table is full, data will be written into the **SSTable** data file. All writes are automatically partitioned and replicated throughout the cluster. Cassandra periodically consolidates the SSTables, discarding unnecessary data.

Read Operations

During read operations, Cassandra gets values from the mem-table and checks the bloom filter to find the appropriate SSTable that holds the required data.

Cassandra - Data Model

The data model of Cassandra is significantly different from what we normally see in an RDBMS. This chapter provides an overview of how Cassandra stores its data.

Cluster

Cassandra database is distributed over several machines that operate together. The outermost container is known as the Cluster. For failure handling, every node contains a replica, and in case of a failure, the replica takes charge. Cassandra arranges the nodes in a cluster, in a ring format, and assigns data to them.

Keyspace

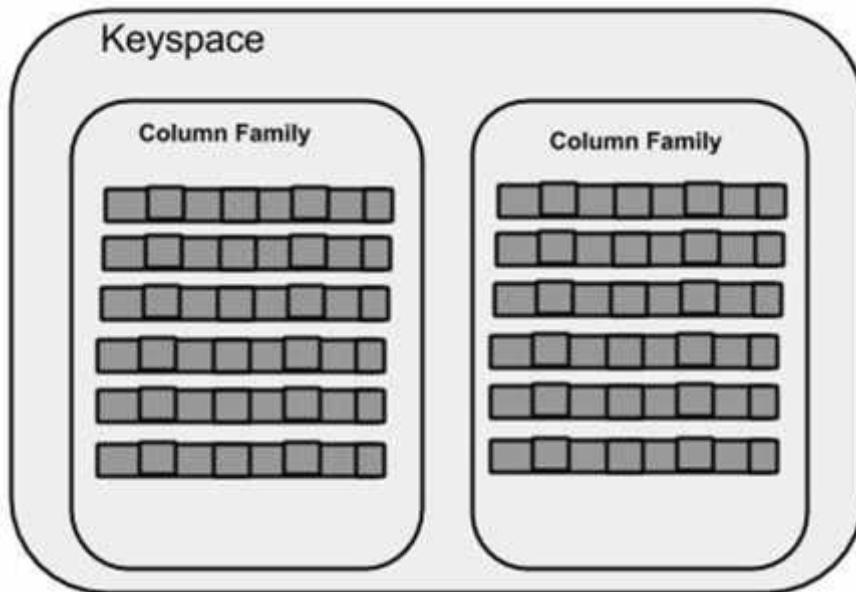
Keyspace is the outermost container for data in Cassandra. The basic attributes of a Keyspace in Cassandra are –

- **Replication factor** – It is the number of machines in the cluster that will receive copies of the same data.
- **Replica placement strategy** – It is nothing but the strategy to place replicas in the ring. We have strategies such as **simple strategy** (rack-aware strategy), **old network topology strategy** (rack-aware strategy), and **network topology strategy** (datacenter-shared strategy).
- **Column families** – Keyspace is a container for a list of one or more column families. A column family, in turn, is a container of a collection of rows. Each row contains ordered columns. Column families represent the structure of your data. Each keyspace has at least one and often many column families.

The syntax of creating a Keyspace is as follows –

```
CREATE KEYSPACE Keyspace name
WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 3};
```

The following illustration shows a schematic view of a Keyspace.



Column Family

A column family is a container for an ordered collection of rows. Each row, in turn, is an ordered collection of columns. The following table lists the points that differentiate a column family from a table of relational databases.

Relational Table	Cassandra column Family
A schema in a relational model is fixed. Once we define certain columns for a table, while inserting data, in every row all the columns must be filled at least with a null value.	In Cassandra, although the column families are defined, the columns are not. You can freely add any column to any column family at any time.
Relational tables define only columns and the user fills in	In Cassandra, a table contains columns, or can be

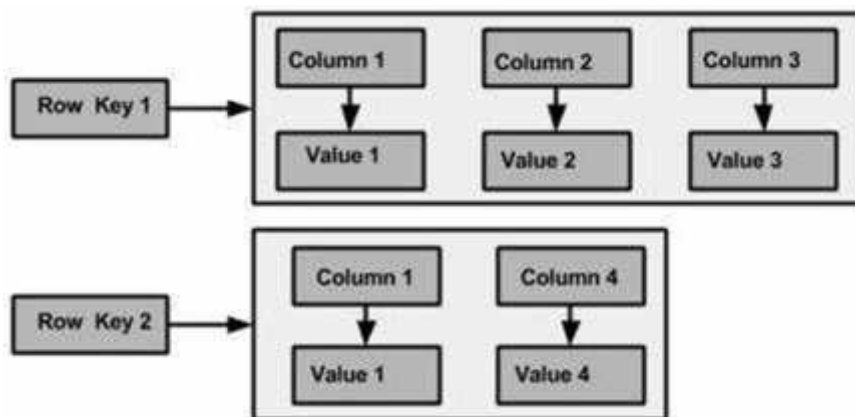
the table with values.	defined as a super column family.
------------------------	-----------------------------------

A Cassandra column family has the following attributes –

- **keys_cached** – It represents the number of locations to keep cached per SSTable.
- **rows_cached** – It represents the number of rows whose entire contents will be cached in memory.
- **preload_row_cache** – It specifies whether you want to pre-populate the row cache.

Note – Unlike relational tables where a column family’s schema is not fixed, Cassandra does not force individual rows to have all the columns.

The following figure shows an example of a Cassandra column family.



Column

A column is the basic data structure of Cassandra with three values, namely key or column name, value, and a time stamp. Given below is the structure of a column.

Column		
name : byte[]	value : byte[]	clock : clock[]

SuperColumn

A super column is a special column, therefore, it is also a key-value pair. But a super column stores a map of sub-columns.

Generally column families are stored on disk in individual files. Therefore, to optimize performance, it is important to keep columns that you are likely to query together in the same column family, and a super column can be helpful here. Given below is the structure of a super column.

Super Column	
name : byte[]	cols : map<byte[], column>

Data Models of Cassandra and RDBMS

The following table lists down the points that differentiate the data model of Cassandra from that of an RDBMS.

RDBMS	Cassandra
RDBMS deals with structured data.	Cassandra deals with unstructured data.
It has a fixed schema.	Cassandra has a flexible schema.
In RDBMS, a table is an array of arrays. (ROW x COLUMN)	In Cassandra, a table is a list of “nested key-value pairs”. (ROW x COLUMN key x COLUMN value)
Database is the outermost container that contains data corresponding to an application.	Keyspace is the outermost container that contains data corresponding to an application.
Tables are the entities of a database.	Tables or column families are the entity of a keyspace.
Row is an individual record in RDBMS.	Row is a unit of replication in Cassandra.
Column represents the attributes of a relation.	Column is a unit of storage in Cassandra.
RDBMS supports the concepts of foreign keys, joins.	Relationships are represented using collections.

Introduction-Importance of Effective data visualization in Big Data

Surely on more than one occasion you have had to face a report, graph etc. without really understanding its content. **Data visualization has become a key** aspect to know how to get the most out of the information, helping to make decisions in a prescriptive way.

Unmanaged data is something unfriendly, which **does not help decision making**. However, when we understand it, it will help us make decisions to improve both our work and the effectiveness we achieve. When data is visualized (instead of looking at it in excel tables) it will change our experience both as a user, collaborator, developer, client or consultant. **Making them understandable and offering us the opportunity to interact.**

The Big Data era

Most of the time when we talk about “data” we usually mean numbers. In the case of Data Mining it is going to involve the **analysis and modeling of data repositories** (i.e. Big Data), involving knowledge areas such as

- Artificial Intelligence.
- Statistics.
- Automatic learning.
- Visualization.

The truth is that nowadays, **we generate a huge amount of data every day**. In 2018 Cisco estimated that during 2019 it would reach 10.4 zb. And considering that each zetabyte corresponds to one trillion gigabytes, that’s a brutal amount of information.

Displaying maps, tables, and even a list of ideas arranged vertically will help **improve your understanding**. That’s why, in the age of gigabytes, information is going to be crucial for citizens and users.

Visualizing data in Big Data

Making the information understandable is crucial. No matter how many figures you have. This requires the use of data analysis software. In this way, we can build dashboards with visual and/or multimedia content from the previously processed data.

Different programming systems such as Phyton, Stata, or RStudio will facilitate the transfer of numerical tables to graphical displays for viewing in reports, publications, and/or presentations. In this way, the data are simplified and instead of showing a table with thousands of cells, we will have a graph with the analysis of those data. In this way, we can analyze all the information in one place.

Nowadays, being able to represent the data in a more understandable way for the user, is something that is done in all sectors, including also politicians, information platforms etc.

Introduction to TABLEAU:

Tableau is a Business Intelligence tool for visually analyzing the data. Users can create and distribute an interactive and shareable dashboard, which depict the trends, variations, and density of the data in the form of graphs and charts. Tableau can connect to files, relational and Big Data sources to acquire and process data. The software allows data blending and real-time collaboration, which makes it very unique. It is used by businesses, academic researchers, and many government organizations for visual data analysis. It is also positioned as a leader Business Intelligence and Analytics Platform in Gartner Magic Quadrant.

TABLEAU – OVERVIEW:

As a leading data visualization tool, Tableau has many desirable and unique features. Its powerful data discovery and exploration application allows you to answer important questions in seconds. You can use Tableau's drag and drop interface to visualize any data, explore different views, and even combine multiple databases easily. It does not require any complex scripting. Anyone who understands the business problems can address it with a visualization of the relevant data. After analysis, sharing with others is as easy as publishing to Tableau Server.

Tableau Features

Tableau provides solutions for all kinds of industries, departments, and data environments. Following are some unique features which enable Tableau to handle diverse scenarios.

- **Speed of Analysis** – As it does not require high level of programming expertise, any user with access to data can start using it to derive value from the data.
- **Self-Reliant** – Tableau does not need a complex software setup. The desktop version which is used by most users is easily installed and contains all the features needed to start and complete data analysis.
- **Visual Discovery** – The user explores and analyzes the data by using visual tools like colors, trend lines, charts, and graphs. There is very little script to be written as nearly everything is done by drag and drop.
- **Blend Diverse Data Sets** – Tableau allows you to blend different relational, semistructured and raw data sources in real time, without expensive up-front integration costs. The users don't need to know the details of how data is stored.
- **Architecture Agnostic** – Tableau works in all kinds of devices where data flows. Hence, the user need not worry about specific hardware or software requirements to use Tableau.
- **Real-Time Collaboration** – Tableau can filter, sort, and discuss data on the fly and embed a live dashboard in portals like SharePoint site or Salesforce. You can save your view of data and allow colleagues to subscribe to your interactive dashboards so they see the very latest data just by refreshing their web browser.
- **Centralized Data** – Tableau server provides a centralized location to manage all of the organization's published data sources. You can delete, change permissions, add tags, and manage schedules in one convenient location. It's easy to schedule extract refreshes and manage them in the data server. Administrators can centrally define a schedule for extracts on the server for both incremental and full refreshes.

TABLEAU - ENVIRONMENT SETUP:

Download Tableau Desktop

The Free Personal Edition of Tableau Desktop can be downloaded from [Tableau Desktop](#). You need to register with your details to be able to download.

After downloading, the installation is a very straightforward process in which you need to accept the license agreement and provide the target folder for installation. The following steps and screenshots describe the entire setup process.

Start the Installation Wizard:

Double-click the **TableauDesktop-64bit-9-2-2.exe**. It will present a screen to allow the installation program to run. Click "Run".



Accept the License Agreement

Read the license agreement and if you agree, choose the "I have read and accept the terms of this license agreement" option. Then, click "Install".



Start Trial

On completion of the installation, the screen prompts you with the option to Start the trial now or later. You may choose to start it now. Also, if you have purchased Tableau then you may enter the License key.



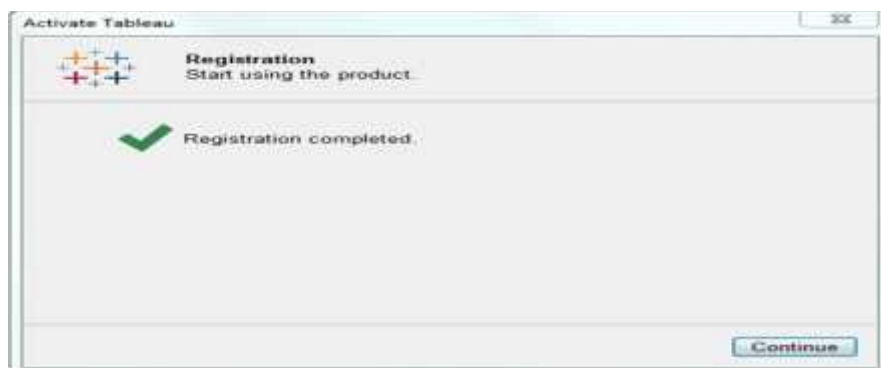
Provide Your Details

Provide your name and organization details. Then, click "Next".

A screenshot of the 'Registration' window. The window has a title bar with 'Activate Tableau' and a close button. Inside, there's a Tableau logo and the text 'Registration' and 'Please complete all fields for the registered user'. The form contains several input fields: 'First Name', 'Last Name', 'Organization', 'Email', 'Phone', 'Job Title', 'City', 'Postal Code', 'Department' (a dropdown menu), 'Country/Region' (a dropdown menu), 'State/Province' (a dropdown menu), and 'Industry' (a dropdown menu). At the bottom right, there is a 'Register' button.

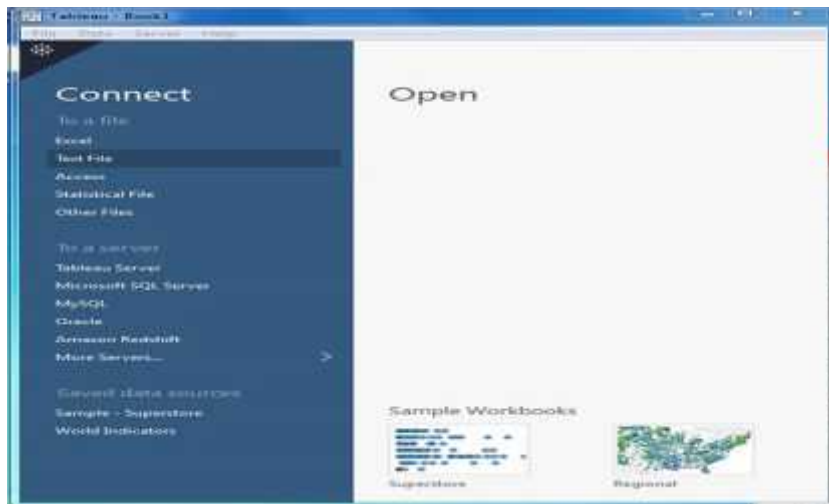
Registration Complete

The registration completion screen appears. Click "Continue".



Verify the Installation

You can verify the installation by going to the Windows start menu. Click the Tableau icon. The following screen appears.



You are now ready to learn Tableau.

TABLEAU - GET STARTED:

In this chapter, you will learn some basic operations in Tableau to get acquainted with its interface. There are three basic steps involved in creating any Tableau data analysis report.

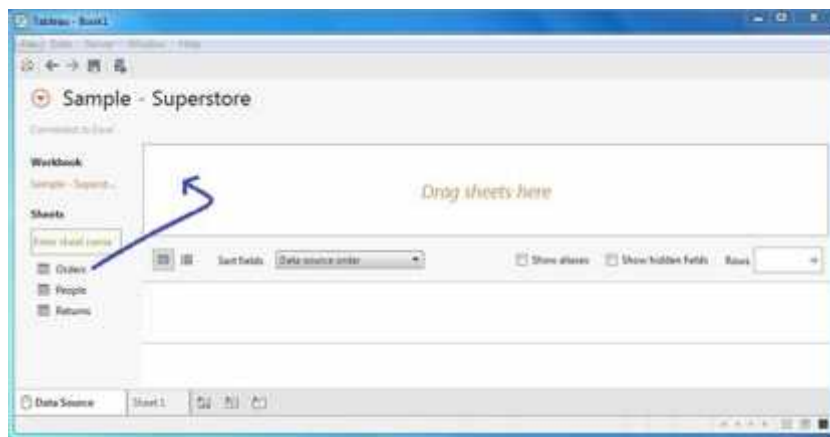
These three steps are –

- **Connect to a data source** – It involves locating the data and using an appropriate type of connection to read the data.
- **Choose dimensions and measures** – This involves selecting the required columns from the source data for analysis.
- **Apply visualization technique** – This involves applying required visualization methods, such as a specific chart or graph type to the data being analyzed.

For convenience, let's use the sample data set that comes with Tableau installation named sample – superstore.xls. Locate the installation folder of Tableau and go to **My Tableau Repository**. Under it, you will find the above file at **Datasources\9.2\en_US-US**.

Connect to a Data Source

On opening Tableau, you will get the start page showing various data sources. Under the header “**Connect**”, you have options to choose a file or server or saved data source. Under Files, choose excel. Then navigate to the file “**Sample – Superstore.xls**” as mentioned above. The excel file has three sheets named Orders, People and Returns. Choose **Orders**.



Choose the Dimensions and Measures

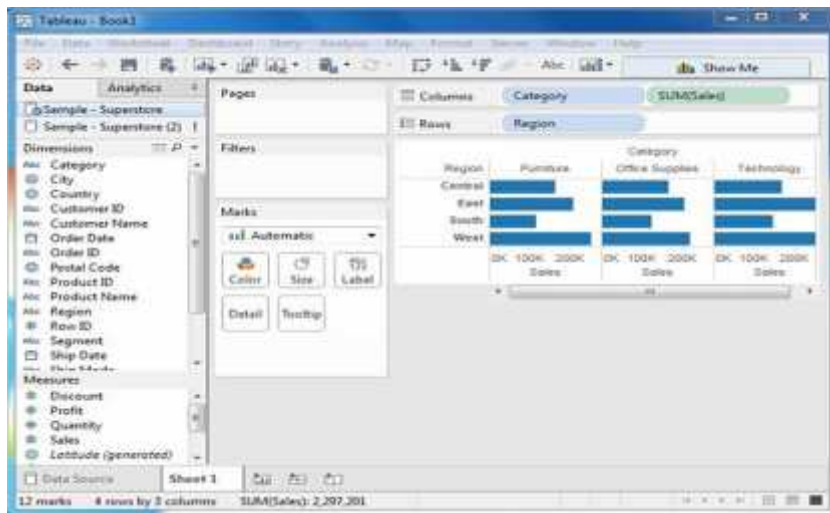
Next, choose the data to be analyzed by deciding on the dimensions and measures. Dimensions are the descriptive data while measures are numeric data. When put together, they help visualize the performance of the dimensional data with respect to the data which are measures. Choose **Category** and **Region** as the dimensions and **Sales** as the measure. Drag and drop them as shown in the following screenshot. The result shows the total sales in each category for each region.



Apply Visualization Technique

In the previous step, you can see that the data is available only as numbers. You have to read and calculate each of the values to judge the performance. However, you can see them as graphs or charts with different colors to make a quicker judgment.

We drag and drop the sum (sales) column from the Marks tab to the Columns shelf. The table showing the numeric values of sales now turns into a bar chart automatically.



You can apply a technique of adding another dimension to the existing data. This will add more colors to the existing bar chart as shown in the following screenshot.

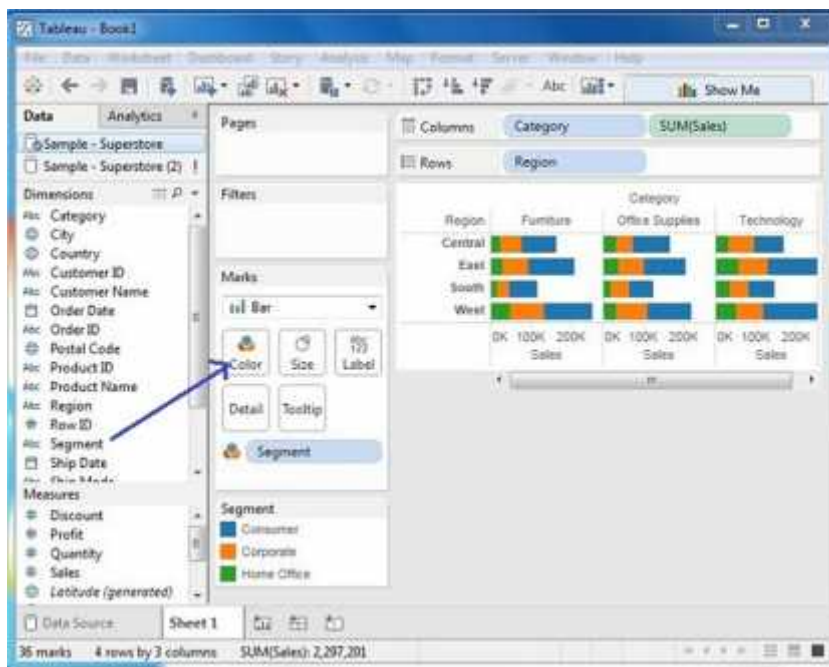
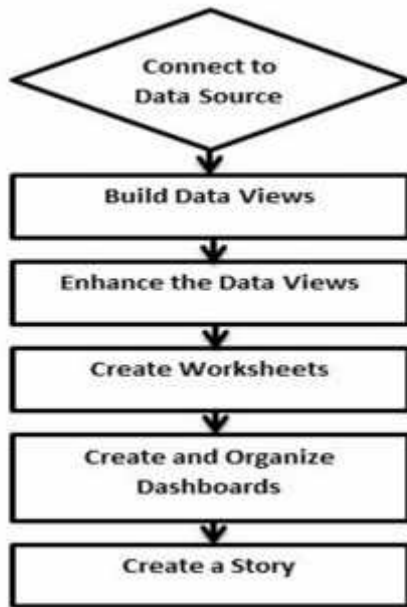


TABLEAU - DESIGN FLOW:

As Tableau helps in analyzing lots of data over diverse time periods, dimensions, and measures, it needs a very meticulous planning to create a good dashboard or story. Hence, it is important to know the approach to design a good dashboard. Like any other field of human endeavor, there are many best practices to be followed to create good worksheets and dashboards.

Though the final outcome expected from a Tableau project is ideally a dashboard with story, there are many intermediate steps which needs to be completed to reach this goal. Following is a flow diagram of design steps that should be ideally followed to create effective dashboards.



Connect to Data Source

Tableau connects to all popular data sources. It has inbuilt connectors which take care of establishing the connection, once the connection parameters are supplied. Be it simple text files, relational sources, SQL sources or cloud data bases, Tableau connects to nearly every data source.

Build Data Views

After connecting to a data source, you get all the column and data available in the Tableau environment. You classify them as dimensions and measures, and create any hierarchy required. Using these you build views, which are traditionally known as Reports. Tableau provides easy drag and drop feature to build views.

Enhance the Views

The views created above needs to be enhanced further by the use of filters, aggregations, labeling of axes, formatting of colors and borders, etc.

Create Worksheets

Create different worksheets to create different views on the same or different data.

Create and Organize Dashboards

Dashboards contain multiple worksheets which are linked. Hence, the action in any of the worksheet can change the result in the dashboard accordingly.

Create a Story

A story is a sheet that contains a sequence of worksheets or dashboards that work together to convey information. You can create stories to show how facts are connected, provide context, demonstrate how decisions relate to outcomes, or simply make a compelling case.

TABLEAU - FILE TYPES:

The result of data analysis in Tableau can be saved in various formats, to be saved and distributed. The various formats are referred as different file types and they are identified by different

extensions. Their formats depend on how they are produced and for what purposes they are used. They are all stored as XML files, which can be opened and edited.

Following table lists the description of each file type and their usage.

File Type	File Extension	Purpose
Tableau Workbook	.twb	It contains information on each sheet and dashboard that is present in a workbook. It has the details of the fields, which are used in each view and the formula applied to the aggregation of the measures. It also has the formatting and styles applied. It contains the data source connection information and any metadata information created for that connection.
Tableau Packaged Workbook	.twbx	This file format contains the details of a workbook as well as the local data that is used in the analysis. Its purpose is to share with other Tableau desktop or Tableau reader users, assuming it does not need data from the server.
Tableau Data Source	.tds	The details of the connection used to create the tableau report are stored in this file. In the connection details, it stores the source type (excel/relational/sap, etc.) as well as the data types of the columns.
Tableau Packaged Data source	.tdsx	This file is similar to the .tds file with the addition of data along with the connection details.
Tableau Data Extract	.tde	This file contains the data used in a .twb file in a highly compressed columnar data format. This helps in storage optimization. It also saves the aggregated calculations that are applied in the analysis. This file should be refreshed to get the updated data from the source.
Tableau Bookmark	.tbn	These files contain a single worksheet that is shared easily to be pasted into other workbooks.
Tableau Preferences	.tps	This file stores the color preference used across all the workbooks. It is mainly used for consistent look and feel across the users.

TABLEAU - DATA TYPES:

As a data analysis tool, Tableau classifies every piece of data into one of the four categories namely - String, Number, Boolean and datetime. Once data is loaded from the source, Tableau automatically assigns the data types. Contrarily, you can also change some of the data types if it satisfies the data conversion rule. The user has to specify the data type for calculated fields.

Following table lists the description of data types supported by Tableau.

Data Type	Description	Example
STRING	Any sequence of zero or more characters. They are enclosed within single quotes. The quote itself can be included in a string by writing it twice.	'Hello' 'Quoted' 'quote'
NUMBER	These are either integers or floating points. It is advised to round the floating point numbers while using them in calculations.	3 142.58
BOOLEAN	They are logical values.	TRUE FALSE
DATE & DATETIME	Tableau recognizes dates in almost all formats. But in case we need to force Tableau to recognize a string as date, then we put a # sign before the data.	"02/01/2015" "#3 March 1982"

TABLEAU – DASHBOARD CREATION:

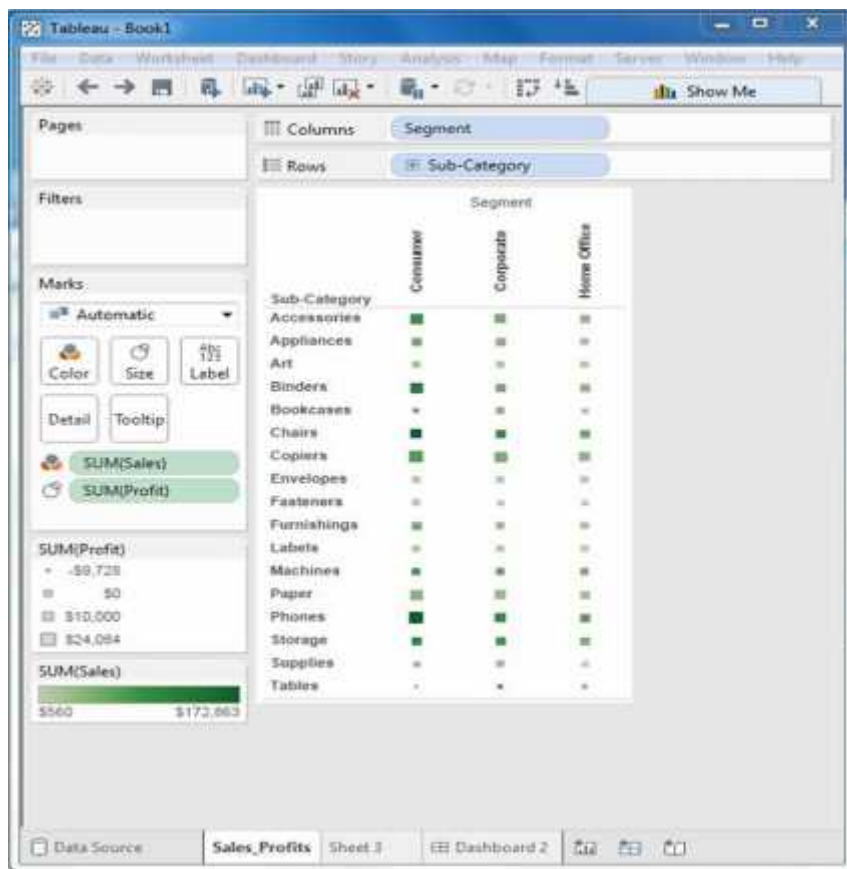
A dashboard is a consolidated display of many worksheets and related information in a single place. It is used to compare and monitor a variety of data simultaneously. The different data views are displayed all at once. Dashboards are shown as tabs at the bottom of the workbook and they usually get updated with the most recent data from the data source. While creating a dashboard, you can add views from any worksheet in the workbook along with many supporting objects such as text areas, web pages, and images.

Each view you add to the dashboard is connected to its corresponding worksheet. So when you modify the worksheet, the dashboard is updated and when you modify the view in the dashboard, the worksheet is updated.

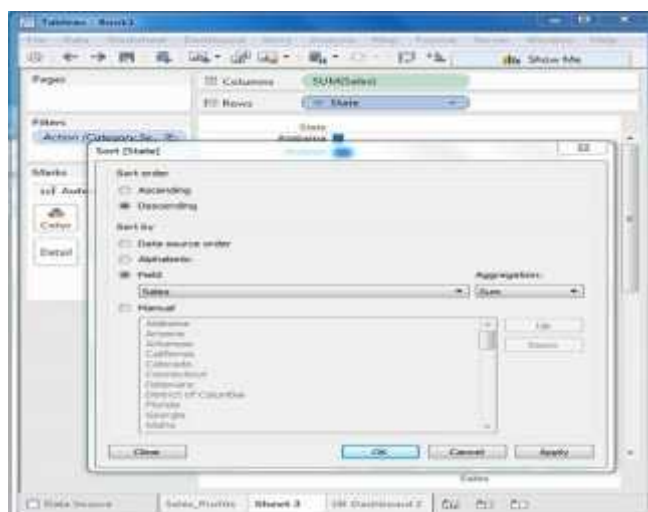
Creating a Dashboard:

Using the Sample-superstore, plan to create a dashboard showing the sales and profits for different segments and Sub-Category of products across all the states. To achieve this objective, following are the steps.

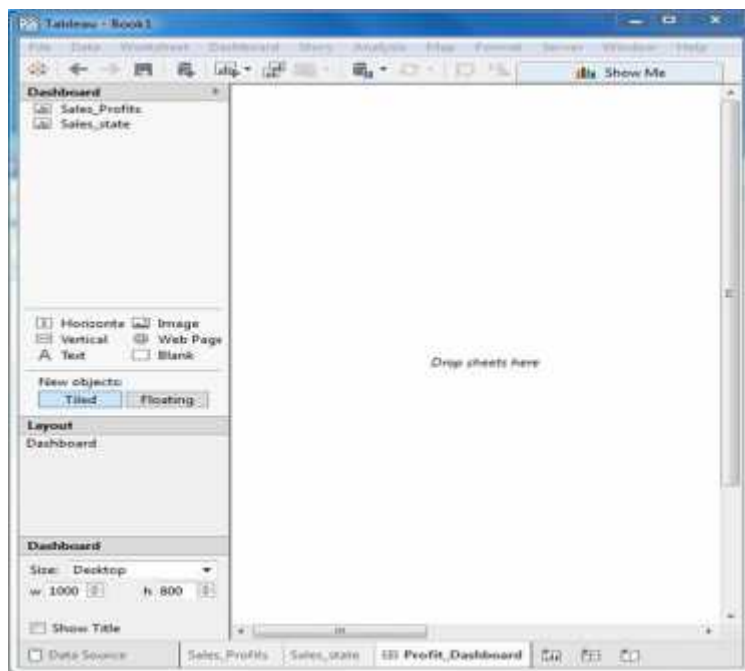
Step 1 – Create a blank worksheet by using the add worksheet icon located at the bottom of the workbook. Drag the dimension Segment to the columns shelf and the dimension Sub-Category to the Rows Shelf. Drag and drop the measure Sales to the Color shelf and the measure Profit to the Size shelf. This worksheet is referred as the Master worksheet. Right-click and rename this worksheet as **Sales_Profits**. The following chart appears.



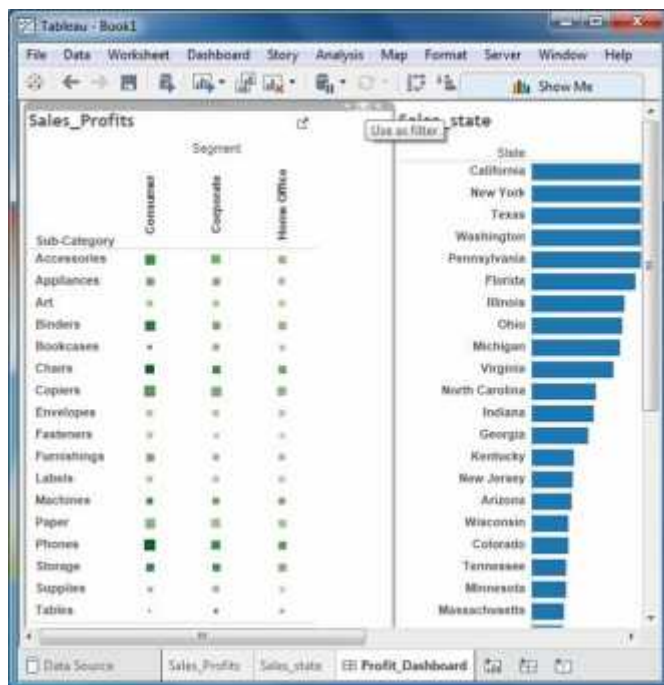
Step 2 – Create another sheet to hold the details of the Sales across the States. For this, drag the dimension State to the Rows shelf and the measure Sales to the Columns shelf as shown in the following screenshot. Next, apply a filter to the State field to arrange the Sales in a descending order. Right-click and rename this worksheet as **Sales_state**.



Step 3 – Next, create a blank dashboard by clicking the Create New Dashboard link at the bottom of the workbook. Right-click and rename the dashboard as Profit_Dashboard.

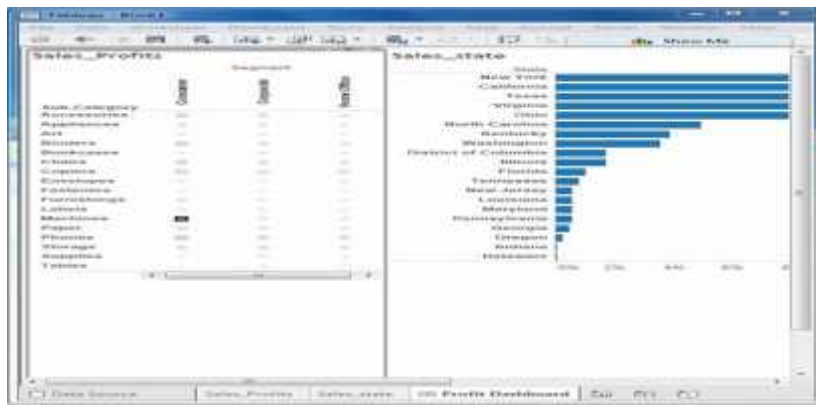


Step 4 – Drag the two worksheets to the dashboard. Near the top border line of Sales Profit worksheet, you can see three small icons. Click the middle one, which shows the prompt Use as Filter on hovering the mouse over it.



Step 5 – Now in the dashboard, click the box representing Sub-Category named Machines and segment named Consumer.

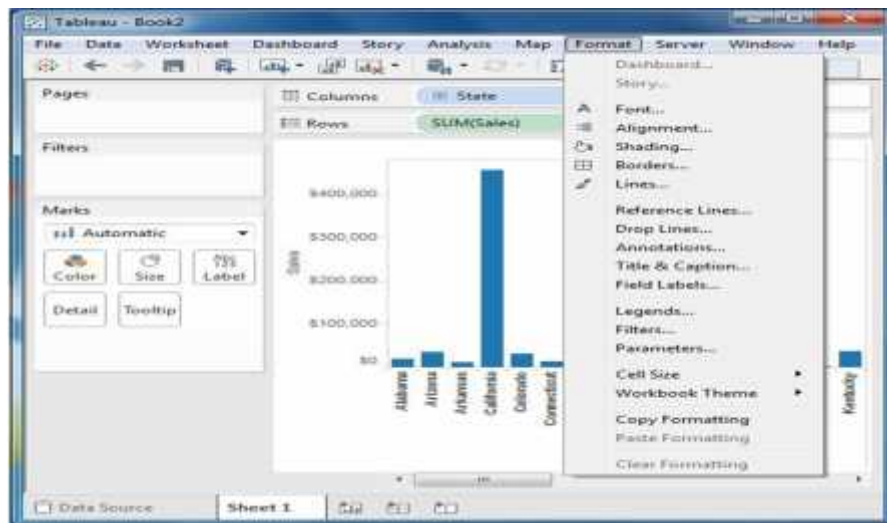
You can notice that only the states where the sales happened for this amount of profit are filtered out in the right pane named **Sales_state**. This illustrates how the sheets are linked in a dashboard.



DASHBOARD FORMATTING:

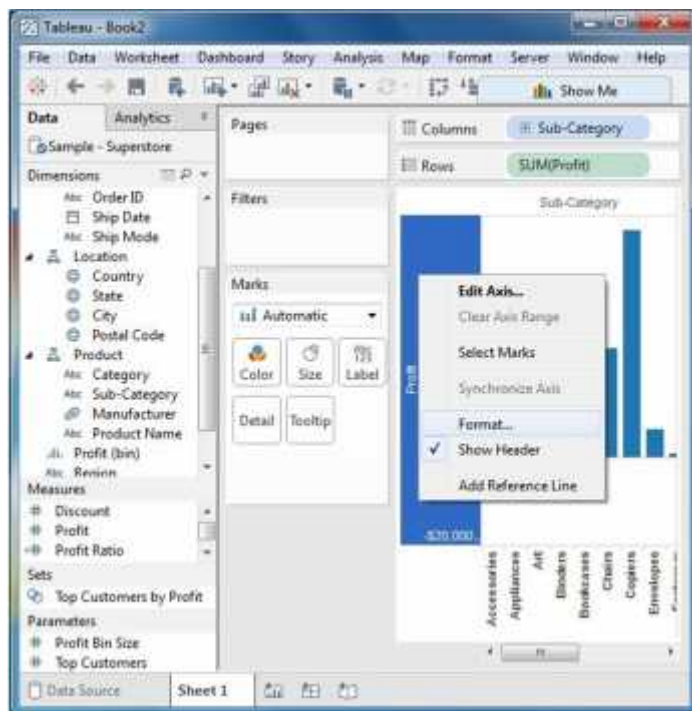
Tableau has a very wide variety of formatting options to change the appearance of the visualizations created. You can modify nearly every aspect such as font, color, size, layout, etc. You can format both the content and containers like tables, labels of axes, and workbook theme, etc.

The following diagram shows the Format Menu which lists the options. In this chapter, you will touch upon some of the frequently used formatting options.



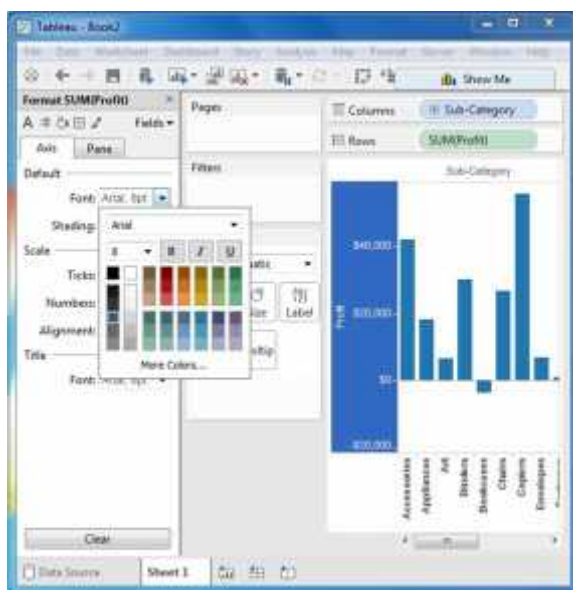
Formatting the Axes

You can create a simple bar chart by dragging and dropping the dimension Sub-Category into the Columns Shelf and the measure Profit into the Rows shelf. Click the vertical axis and highlight it. Then right-click and choose format.



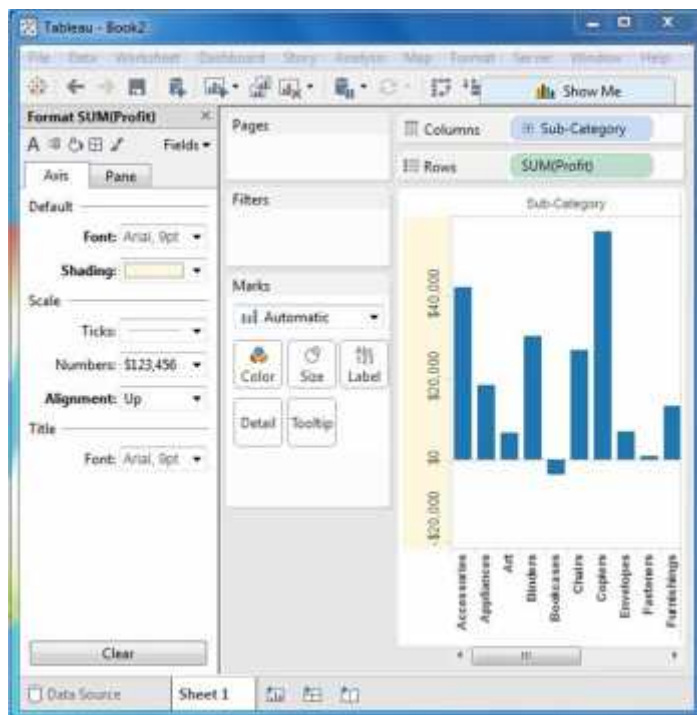
Change the Font

Click the font drop-down in the Format bar, which appears on the left. Choose the font type as Arial and size as 8pt. as shown in the following screenshot.



Change the Shade and Alignment

You can also change the orientation of the values in the axes as well as the shading color as shown in the following screenshot.



Format Borders

Consider a crosstab chart with Sub-Category in the Columns shelf and State in the Rows shelf. Now, you can change the borders of the crosstab table created by using the formatting options. Right-click on crosstab chart and choose Format.

The Format Borders appear in the left pane. Choose the options as shown in the following screenshot.

