# Data Processing and Transformation in Hive using Azure VM

**Business Overview**

Big Data is a collection of massive quantities of semi-structured and unstructured data created by a heterogeneous group of high-performance devices spanning from social networks to scientific computing applications. Companies have the ability to collect massive amounts of data, and they must ensure that the data is in highly usable condition by the time it reaches data scientists and analysts. The profession of data engineering involves designing and constructing systems for acquiring, storing, and analyzing vast volumes of data. It is a broad field with applications in nearly every industry.

Apache Hadoop is a Big Data solution that allows for the distributed processing of enormous data volumes across computer clusters by employing basic programming techniques. It is meant to scale from a single server to thousands of computers, each of which will provide local computation and storage.

Apache Hive is a fault-tolerant distributed data warehouse system that allows large-scale analytics. Hive allows users to access, write, and manage petabytes of data using SQL. It is built on Apache Hadoop, and as a result, it is tightly integrated with Hadoop and is designed to manage petabytes of data quickly. Hive is distinguished by its ability to query enormous datasets utilizing a SQL-like interface and an Apache Tez, MapReduce, or Spark engine.

**Data Pipeline**

A data pipeline is a technique for transferring data from one system to another. The data may or may not be updated, and it may be handled in real-time (or streaming) rather than in batches. The data pipeline encompasses everything from harvesting or acquiring data using various methods to storing raw data, cleaning, validating, and transforming data into a query-worthy format, displaying KPIs, and managing the above process.

**Dataset Description**

In this project, we will use the Airlines dataset to demonstrate the issues related to massive amounts of data and how various Hive components can be used to tackle them. Following are the files used in this project, along with a few of their fields :

- airlines.csv - IATA_code, airport_name, city, state, country
- carrier.csv - code, description
- plane-data.csv - tail_number, type, manufacturer, model, engine_type
- Flights data (yearly) - flight_num, departure, arrival, origin, destination, distance

**Tech Stack**

➔ Language: HQL
➔ Services: Azure VM, Hive, Hadoop

**Key Takeaways**

- Introduction to Hadoop and Hive
- Understanding the Dataset
- Creating Azure VM
- Installation and configuration of Hadoop and Hive
- Setting up Hive metastore
- Accessing Hive server using Beeline
- Creating and understanding tables in Hive
- Implementing Hive table operations
- Partitioning in Hive
- Creating Hive Buckets
- Sampling using Hive
- Understanding Joins and Views in Hive
- Understanding different file formats in Hive and their usage
- Performance analysis using Explain and Analyze commands

## Note:

**For Error - " WARN jdbc.HiveConnection: Failed to connect to localhost:10000 Could not open connection to the HS2 server. Please check the server URI and if the URI is correct, then ask the administrator to check the server status.
Error: Could not open client transport with JDBC Uri: jdbc:hive2://localhost:10000: java.net.ConnectException: Connection refused (Connection refused) (state=08S01,code=0)
Beeline version 3.1.2 by Apache Hive"**

**Solution - start hiveserver2 from the right directory i.e cd $HIVE_HOME/bin;**

**Error 2 - if issues connecting to Hiveserver2 through Beeline -
Add these inbound and outbound port rules in Azure VM**


**1) reinstalled Hadoop and hive**

**2) installed net-tools**
**sudo apt install net-tools**

**3) check for all the ports that are active**
**sudo netstat -tulpn | grep LISTEN**

**4) then execute the below for more logging. simple hiveserver2 will also work.**
**cd $HIVE_HOME/bin;**

**hive --service hiveserver2 --hiveconf hive.server2.thrift.port=10000 --hiveconf hive.root.logger=INFO,console**

**5) Wait for the port 1000 to be appreared in the following command output**
**sudo netstat -tulpn | grep LISTEN**

**6) Open a new PuTTy Session**

**7) once 10000 port is active then the user can execute the beeline command**
**cd $HIVE_HOME/bin;**
**beeline -u jdbc:hive2://localhost:10000 -n ubuntuhive**