Big Data Technologies

Agenda

- Hadoop Fundamentals
  - Hadoop Ecosystems
  - Hadoop Distributions
- Hive

Apache Hive

- Hive is data warehouse (for OLAP) built on Hadoop framework.
  - Data warehouse = (Huge) RDBMS for Analytical Processing
- Hive manage "structured data".
- Hive is client software that convert Hive QL (SQL-like queries) queries to MR.
- Hive QL is similar to SQL with many extended features.

Hive Documentation

- https://cwiki.apache.org/confluence/display/Hive/Home
  - https://cwiki.apache.org/confluence/display/Hive/LanguageManual

Hive History

- Facebook data ingestion into Hadoop
  - 10s GB/day – 2006
  - 1 TB/day – 2007
  - MySQL/Oracle database limitations
- Processing Hadoop data using MR is complex
- Developed Hive to convert SQL queries into MR
- Open sourced under Apache license (2010)

Hive advantages

- Data warehouse - data analysis
- Long running queries.
- Fault tolerant environment.

Hive Limitations

- Slower response time.
- Data manipulation is not supported (fully).

Hive Applications

- Batch processing (SQL based)
- ETL jobs
- Business Intelligence (Reports)
- Predictive Modeling
- Data mining
- Log processing

Hive Installation

- Install Hadoop.
- Install Hive.
    - hive-site.xml
    - set PATH in ~/.bashrc
- Start metastore service.
- Start hive CLI.
- Start hiveserver2 service.
- Start hive beeline.

Hive QL

- Hive QL is extended SQL.
- Supports DQL, DML, DDL and DCL.
- DQL supports filtering, ordering, grouping, joins, etc.
- Data will be read from HDFS (default location: /user/hive/warehouse).
- Store query result into HDFS (or in another table).
    - INSERT INTO atable SELECT * FROM another_table;
- Supports views and indexes (till Hive 2.x).
- Managed tables and External tables
- Partitioning & Bucketing.
- Various hive data types
    - Primitive Types:
        - BOOLEAN(1)
        - Integers: TINYINT(1), SMALLINT(2), INT(4), BIGINT(8)
        - Floating Point: FLOAT (single precision), DOUBLE (double precision), DECIMAL(m,n)
        - Characters: CHAR(n), VARCHAR(n), STRING
        - Date & Time: TIMESTAMP, DATE, DATETIME
    - Collection Types:
        - ARRAY: collection of same type of data
            - e.g. emails (separated by |)
        - STRUCT: collection of different type of data
            - e.g. addr (area STRING| dist STRING| pin INT)
        - MAP: collection of key-value pairs
            - e.g. phones (type:phone)
        - UNION
- Follows Schema-on-Read for better ingestion performance
    - While loading the data (LOAD DATA) schema is not verified.
    - While processing individual records schema is verified (SELECT, INSERT, UPDATE).
    - If data is not compatible with the type, value is considered null.

Hive QL - Schema on Read

- When data is ingested in Hive tables using "LOAD DATA" command, the data file is uploaded in Hive directly. Internally, it does HDFS put operation.

- While uploading file, no checks on data are performed against schema of the table.
- However, when data is read/processed record by record, it is verified against schema. If data is not compatible with its data type, the data is considered as null. If length/size of data is not matching, it will be truncated.
- This feature is called "Schema-on-Read". This enables high speed data ingestion.

Hive INSERT

- Inserts new records into hive table.
- Internally creates new files under HDFS (table directory).
- Produce MR job to insert data.
- While INSERT hive follows schema on write.

Assignments

1. Assignment 1 -

```
-- Customers table
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    Name VARCHAR(100),
    Age INT,
    LocationID INT
);

-- Products table
CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(100),
    Category VARCHAR(50),
    Price DECIMAL(10, 2)
);

-- Sales table
CREATE TABLE Sales (
```

```
    SaleID INT PRIMARY KEY,
    CustomerID INT,
    ProductID INT,
    SaleDate DATE,
    Quantity INT,
    TotalAmount DECIMAL(10, 2)
);

-- Locations table
CREATE TABLE Locations (
    LocationID INT PRIMARY KEY,
    City VARCHAR(50),
    State VARCHAR(50)
);

-- Customers
INSERT INTO Customers VALUES (1, 'John Doe', 30, 1);
INSERT INTO Customers VALUES (2, 'Jane Smith', 25, 2);
INSERT INTO Customers VALUES (3, 'Bob Johnson', 35, 1);
INSERT INTO Customers VALUES (4, 'Alice Brown', 28, 3);
INSERT INTO Customers VALUES (5, 'Charlie Davis', 32, 2);

-- Products
INSERT INTO Products VALUES (1, 'Laptop', 'Electronics', 800.00);
INSERT INTO Products VALUES (2, 'Smartphone', 'Electronics', 400.00);
INSERT INTO Products VALUES (3, 'T-shirt', 'Clothing', 20.00);
INSERT INTO Products VALUES (4, 'Shoes', 'Footwear', 50.00);
INSERT INTO Products VALUES (5, 'Bookshelf', 'Furniture', 150.00);

-- Sales
INSERT INTO Sales VALUES (1, 1, 1, '2023-01-01', 2, 1600.00);
INSERT INTO Sales VALUES (2, 2, 3, '2023-01-02', 3, 60.00);
INSERT INTO Sales VALUES (3, 3, 2, '2023-01-03', 1, 400.00);
INSERT INTO Sales VALUES (4, 4, 4, '2023-02-01', 2, 100.00);
INSERT INTO Sales VALUES (5, 5, 5, '2023-02-02', 1, 150.00);
```

```
-- Locations
INSERT INTO Locations VALUES (1, 'Pune', 'Maharashtra');
INSERT INTO Locations VALUES (2, 'Mumbai', 'Maharashtra');
INSERT INTO Locations VALUES (3, 'Bangalore', 'Karnataka');
INSERT INTO Locations VALUES (4, 'Delhi', 'Delhi');
INSERT INTO Locations VALUES (5, 'Chennai', 'Tamil Nadu');
```

- A. Retrieve the names of all customers who made a purchase.
- B. List the products and their total sales amounts for a given date range.
- C. Find the total sales amount for each product category.
- D. Identify the customers who made purchases in a specific city.
- E. Calculate the average age of customers who bought products in the 'Electronics' category.
- F. List the top 3 products based on total sales amount.
- G. Find the total sales amount for each month.
- H. Identify the products with no sales.
- I. Calculate the total sales amount for each state.
- J. Retrieve the customer names and their highest purchase amount.

2. Try MySQL classwork of Window functions and CTEs.