Data Lake ETL /Data warehouse  
using opensource (Hive)

## Introduction to Hive

### Creating Hive Tables

สร้างตาราง aha ประกอบด้วย 2 คอลัมน์เป็นจำนวนเต็ม (Integer) และสายอักขระ (String)

CREATE TABLE aha (num INT, name STRING)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ',';

### Loading data from files into Hive:

สร้างไฟล์ที่ชื่อว่า aha.txt ขั้นคอลัมน์ด้วยคอมมา (,) แล้ว upload ไปไว้ในโฟลเดอร์ /user/admin แล้วเข้าไปโหลดใน Hive UI

LOAD DATA INPATH '/user/admin/aha.txt'

INTO TABLE aha;

\*\* Challenge สร้าง ไฟล์ aha2.txt เพื่อเพิ่มข้อมูล แล้วโหลดเข้าไปเพิ่มในตาราง aha\*\*

\*\* Challenge สร้าง ไฟล์ aha3.txt เพื่อเพิ่มข้อมูล โดยการ upload เข้าไปที่โฟล์เดอร์ /apps/have/warehouse/aha แล้วทดของ query ข้อมูล\*\*

## Hive and structured text file (CSV)

### เปิด Zeppelin เพื่อช่วย Download ข้อมูลตัวอย่าง ([cityofchicago.org](https://data.cityofchicago.org/Transportation/Chicago-Traffic-Tracker-Historical-Congestion-Esti/emtn-qqdi)) (แนะนำให้พิมพ์ใหม่เองหรือให้ใช้ ไฟล์ zeppelin notebook ที่ให้ไว้ ไม่แนะนำให้ copy คำสั่งดาวโหลดส่วนนี้จาก word)

~~%sh~~

~~hdfs dfs -mkdir -p /user/zeppelin/chicago~~

~~wget https://data.cityofchicago.org/api/views/emtn-qqdi/rows.csv -O rows.csv~~

~~hdfs dfs -put rows.csv /user/zeppelin/chicago~~

### สร้าง table จาก csv file

create external table chicago\_bus (

timelog String,

region int,

buscount int,

logreads int,

speed float)

row format delimited fields terminated by ','

lines terminated by "\n"

location /user/zeppelin/chicago"

tblproperties('skip.header.line.count'='1');

### Processing and querying Hive data using [UDFs](https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF)

ต้องการแปลงวันที่และเวลาให้เป็นเวลามาตรฐาน

1. สร้าง Hive managed table

create table chicago (timelog timestamp, region int, buscount int, logreads int, speed float) ;

2. แปลงเวลาแล้วโหลดข้อมูลเข้าไปในตารางใหม่

INSERT OVERWRITE TABLE chicago

SELECT cast(to\_utc\_timestamp(unix\_timestamp(timelog, 'MM/dd/yyyy hh:mm:ss a'),'America/Chicago') AS TIMESTAMP) as timelog, region , buscount ,logreads ,speed FROM chicago\_bus

3. วิเคราะห์ข้อมูลตามวันที่และเขตพื้นที่

select to\_date(timelog) as date,region,avg(speed) as average\_speed   
from chicago group by to\_date(timelog),region;

4. สร้างตารางใหม่เพื่อเก็บความเร็วเฉลี่ยของแต่ละพื้นที่ตามเขตต่าง ๆ

CREATE TABLE chicago\_region\_speed (logdate timestamp, region int, speed float);

INSERT OVERWRITE TABLE chicago\_region\_speed SELECT to\_date(timelog) as timelog,region,avg(speed) as speed from chicago group by to\_date(timelog),region;

## Hive and relational database

### เปิด Zeppelin เพื่อช่วย Download ข้อมูลตัวอย่าง (แนะนำให้พิมพ์ใหม่เองหรือให้ใช้ ไฟล์ zeppelin notebook ที่ให้ไว้ ไม่แนะนำให้ copy คำสั่งดาวโหลดที่ให้ไว้นี้)

~~wget https://raw.githubusercontent.com/praisan/hello-world/master/retail\_hive\_export.zip~~

~~unzip retail\_hive\_export.zip~~

~~hdfs dfs -mkdir -p /user/zeppelin/retail~~

~~hdfs dfs -put retail\_export /user/zeppelin/~~

### สร้าง table จาก csv file

CREATE DATABASE retail;

USE retail;

IMPORT FROM '/user/zeppelin/retail\_export/categories';

IMPORT FROM '/user/zeppelin/retail\_export/customers';

IMPORT FROM '/user/zeppelin/retail\_export/departments';

IMPORT FROM '/user/zeppelin/retail\_export/order\_items';

IMPORT FROM '/user/zeppelin/retail\_export/orders';

IMPORT FROM '/user/zeppelin/retail\_export/products';

ทดสอบ

select count(\*) from retail.order\_items;

## Hive and log file

### เปิด Zeppelin เพื่อช่วย Download ข้อมูลตัวอย่าง (แนะนำให้พิมพ์ใหม่เองหรือให้ใช้ ไฟล์ zeppelin notebook ที่ให้ไว้ ไม่แนะนำให้ copy คำสั่งดาวโหลดส่วนนี้)

~~%sh~~

~~hdfs dfs -mkdir -p /user/zeppelin/accesslog~~

~~wget wget https://github.com/praisan/hello-world/raw/master/access.log.2.zip~~

~~unzip access.log.2.zip~~

~~hdfs dfs -put access.log.2 /user/zeppelin/accesslog~~

### สร้างตารางใน Hive (External table)

CREATE EXTERNAL TABLE access\_logs (

ip STRING,

thedate STRING,

method STRING,

url STRING,

http\_version STRING,

code1 STRING,

code2 STRING,

dash STRING,

user\_agent STRING)

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'

WITH SERDEPROPERTIES (

'input.regex' = '([^ ]\*) - - \\[([^\\]]\*)\\] "([^\ ]\*) ([^\ ]\*) ([^\ ]\*)" (\\d\*) (\\d\*) "([^"]\*)" "([^"]\*)"',

'output.format.string' = "%1$$s %2$$s %3$$s %4$$s %5$$s %6$$s %7$$s %8$$s %9$$s")

LOCATION '/user/zeppelin/accesslog';

ทดสอบ query

SELECT \* FROM access\_logs limit 10;

SELECT ip,count(\*) FROM access\_logs GROUP BY ip;

## Partitioning and Bucketing

### เปิด Zeppelin เพื่อช่วย Download ข้อมูลตัวอย่าง (แนะนำให้พิมพ์ใหม่เองหรือให้ใช้ ไฟล์ zeppelin notebook ที่ให้ไว้ ไม่แนะนำให้ copy คำสั่งดาวโหลดส่วนนี้)

~~%sh~~

~~hdfs dfs -mkdir -p /user/zeppelin/OnlineRetail~~

~~wget wget https://github.com/praisan/hello-world/raw/master/OnlineRetail.zip~~

~~unzip OnlineRetail.zip~~

~~hdfs dfs -put OnlineRetail.csv /user/zeppelin/OnlineRetail~~

### Partitioning

Partitioning gives effective results when,

* There are limited number of partitions
* Comparatively equal sized partitions

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| InvoiceNo | StockCode | Description | Quantity | InvoiceDate | UnitPrice | CustomerID | Country |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

#### Create temporary table

CREATE EXTERNAL TABLE temp\_sales(invoice\_no BIGINT, stock\_code STRING, description STRING, quantity INT, invoice\_date STRING, unit\_price FLOAT, customer\_id BIGINT, country STRING ) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION '/user/zeppelin/OnlineRetail/' TBLPROPERTIES('skip.header.line.count'='1')

#### \*\*\*Challenge: Write the query to answer the following question.

1. แสดงจำนวนลูกค้าทั้งหมดในแต่ละประเทศ
2. แสดงข้อมูลประเทศและยอดซื้อสินค้าทั้งหมดในแต่ละประเทศ

#### Create partitioned table

CREATE TABLE sales(invoice\_no BIGINT, stock\_code STRING, description STRING, quantity INT, invoice\_date STRING, unit\_price STRING, customer\_id BIGINT ) PARTITIONED BY (country STRING);

#### Enable dynamic partitioning

SET hive.exec.dynamic.partition = true;

SET hive.exec.dynamic.partition.mode = nonstrict;

#### Load data into the partitioned table

INSERT OVERWRITE TABLE sales

PARTITION (country)

SELECT \*

FROM temp\_sales

### Hive Bucketing

CREATE TABLE sales\_with\_buckets(invoice\_no BIGINT, stock\_code STRING, description STRING, quantity INT, invoice\_date STRING, unit\_price STRING, customer\_id BIGINT, country STRING) CLUSTERED BY (customer\_id) INTO 10 BUCKETS;

#### Load data in to bucketed table

Enable dynamic bucketing while loading data into hive table

set hive.enforce.bucketing = true;

INSERT OVERWRITE TABLE sales\_with\_buckets

SELECT \*

FROM temp\_sales;

### Hive Partitioning and Bucketing

CREATE TABLE sales\_with\_partitions\_buckets(invoice\_no BIGINT, stock\_code STRING, description STRING, quantity INT, invoice\_date STRING, unit\_price STRING, customer\_id BIGINT) PARTITIONED BY (country STRING) CLUSTERED BY (customer\_id) INTO 10 BUCKETS;

Enable dynamic partitioning and bucketing while loading data into hive table

SET hive.exec.dynamic.partition = true;

SET hive.exec.dynamic.partition.mode = nonstrict;

set hive.enforce.bucketing = true;

Load data

INSERT OVERWRITE TABLE sales\_with\_partitions\_buckets

PARTITION (country)

SELECT \*

FROM temp\_sales;