JI22-2 (Jaysie Lestari)

Big Data – Case 1

1. Command to Load data from CSV to Hive

hadoop fs -copyFromLocal product\_category.csv /user/cloudera

hadoop fs -copyFromLocal location.csv /user/cloudera

CREATE DATABASE BluejackStore

USE BluejackStore

**Product:**

CREATE EXTERNAL TABLE MsProductCategory (

ProductCategoryID INT,

ProductCategoryName VARCHAR(150)

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE

**Location:**

CREATE EXTERNAL TABLE MsLocation(

LocationID INT,

CountryName VARCHAR(150),

ProvinceName VARCHAR(150),

CityName VARCHAR(150)

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE

LOAD DATA INPATH '/user/cloudera/product\_category.csv'

INTO TABLE MsProductCategory

LOAD DATA INPATH '/user/cloudera/location.csv'

INTO TABLE MsLocation

SELECT \* FROM msproductcategory

Text

Description automatically generatedSELECT \* FROM mslocation

1. Command to Load data from MySQL Hive

mysql -u root -p

Text

Description automatically generatedEnter password: cloudera

show databases;

CREATE DATABASE BluejackStore

USE BluejackStore

source insert.sql

A picture containing text

Description automatically generatedshow tables;

exit;

// Scoop

sudo sqoop import-all-tables --connect jdbc:mysql://quickstart/BluejackStore/ --username=root -P --hive-import --hive-database=BluejackStore

Enter password: cloudera

1. Hive query for analysis

a.

SELECT MAX(TotalPrice.AfterDiscountedPrice) AS MostProfitablePrice,

TotalPrice.productcategoryname AS CategoryName

FROM (

SELECT ((PS.product\_price \* (quantity - COALESCE(is\_cancelled, 0))) - (PS.product\_price \* (quantity - COALESCE(is\_cancelled, 0)) \* PS.discount)) AS AfterDiscountedPrice,

PC.productcategoryname

FROM sales\_detail SD

JOIN sales SL

ON SD.sales\_id = SL.sales\_id

JOIN products PS

ON PS.product\_id = SD.product\_id

JOIN msproductcategory PC

ON PS.product\_category\_id = PC.productcategoryid

WHERE YEAR(sales\_date) = 2019

) AS TotalPrice

GROUP BY TotalPrice.productcategoryname, TotalPrice.AfterDiscountedPrice

ORDER BY MostProfitablePrice DESC

LIMIT 1

b.

SELECT MAX(salestotal.SalesCount) AS TotalSalesCount,

salestotal.provincename,

salestotal.cityname

FROM (

SELECT COUNT(SS.sales\_id) AS SalesCount,

YEAR(SS.sales\_date) AS SalesYear,

ML.provincename,

ML.cityname

FROM sales SS

JOIN mslocation ML

ON ML.locationid = SS.location\_id

GROUP BY YEAR(SS.sales\_date), ML.provincename, ML.cityname

) AS salestotal

GROUP BY salestotal.provincename, salestotal.cityname

ORDER BY TotalSalesCount DESC

LIMIT 1

c.

SELECT MAX(Prices.AfterDiscountedPrice) AS ProductPrice,

Prices.product\_name AS ProductName

FROM (

SELECT ((PS.product\_price \* (SD.quantity - COALESCE(SD.is\_cancelled, 0))) - (PS.product\_price \* (SD.quantity - COALESCE(SD.is\_cancelled, 0)) \* PS.discount)) AS AfterDiscountedPrice,

PS.product\_name

FROM sales\_detail SD

JOIN products PS

ON PS.product\_id = SD.product\_id

) AS Prices

GROUP BY Prices.product\_name

ORDER BY ProductPrice DESC

LIMIT 1

d.

SELECT COUNT(SS.sales\_id) AS TotalTransaction,

CT.customer\_id AS CustomerID,

CT.customer\_name AS CustomerName,

CT.customer\_dob AS CustomerDOB,

CT.customer\_email AS CustomerEmail,

CT.customer\_phone AS CustomerPhone

FROM sales SS

JOIN customer CT

ON SS.customer\_id = CT.customer\_id,

(

SELECT AVG(TransactionGrouping.TransactionCount) AS Average

FROM (

SELECT COUNT(SS.sales\_id) AS TransactionCount,

CT.customer\_id AS CustomerID

FROM sales SS

JOIN customer CT

ON SS.customer\_id = CT.customer\_id

GROUP BY CT.customer\_id

) AS TransactionGrouping

GROUP BY TransactionGrouping.CustomerID

) AS AverageGrouping

GROUP BY CT.customer\_id, CT.customer\_name, CT.customer\_dob, CT.customer\_email, CT.customer\_phone, AverageGrouping.Average

HAVING TotalTransaction > AverageGrouping.Average

e.

SELECT SUM((PS.product\_price \* (SD.quantity - COALESCE(SD.is\_cancelled, 0))) - (PS.product\_price \* (SD.quantity - COALESCE(SD.is\_cancelled, 0)) \* PS.discount)) AS AfterDiscountedPrice,

CT.customer\_id AS CustomerID,

CT.customer\_name AS CustomerName,

CT.customer\_dob AS CustomerDOB,

CT.customer\_email AS CustomerEmail,

CT.customer\_phone AS CustomerPhone,

CASE

WHEN ((PS.product\_price \* (SD.quantity - COALESCE(SD.is\_cancelled, 0))) - (PS.product\_price \* (SD.quantity - COALESCE(SD.is\_cancelled, 0)) \* PS.discount)) >= 10000000 AND ((PS.product\_price \* (SD.quantity - COALESCE(SD.is\_cancelled, 0))) - (PS.product\_price \* (SD.quantity - COALESCE(SD.is\_cancelled, 0)) \* PS.discount)) <= 24999999 THEN '1000000'

WHEN ((PS.product\_price \* (SD.quantity - COALESCE(SD.is\_cancelled, 0))) - (PS.product\_price \* (SD.quantity - COALESCE(SD.is\_cancelled, 0)) \* PS.discount)) >= 25000000 AND ((PS.product\_price \* (SD.quantity - COALESCE(SD.is\_cancelled, 0))) - (PS.product\_price \* (SD.quantity - COALESCE(SD.is\_cancelled, 0)) \* PS.discount)) <= 49999999 THEN '5000000'

WHEN ((PS.product\_price \* (SD.quantity - COALESCE(SD.is\_cancelled, 0))) - (PS.product\_price \* (SD.quantity - COALESCE(SD.is\_cancelled, 0)) \* PS.discount)) >= 50000000 THEN '10000000'

END AS Voucher

FROM sales\_detail SD

JOIN products PS

ON PS.product\_id = SD.product\_id

JOIN sales SS

ON SD.sales\_id = SS.sales\_id

JOIN customer CT

ON CT.customer\_id = SS.customer\_id

WHERE YEAR(SS.sales\_date) = 2019 AND MONTH(SS.sales\_date) = 12 AND (PS.product\_price \* (SD.quantity - COALESCE(SD.is\_cancelled, 0))) - (PS.product\_price \* (SD.quantity - COALESCE(SD.is\_cancelled, 0)) \* PS.discount) >= 1000000

GROUP BY CT.customer\_id, CT.customer\_name, CT.customer\_dob, CT.customer\_email, CT.customer\_phone, PS.product\_price, SD.quantity, SD.is\_cancelled, PS.discount

NB. In case the queries above are hard to read, there’s a query.sql file for all the queries.