**Qualif Big Data Processing VK23-2 Case 1**

1. **Load data from CSV to HIVE**

hadoop fs – copyFromLocal SoalQualif/Case 01/location.csv

hadoop fs – copyFromLocal SoalQualif/Case 01/product\_category.csv

1. **Create Database and Table**

CREATE DATABASE BluejackStore;

--location table

CREATE EXTERNAL TABLE location(

location\_id INT,

location\_country VARCHAR(50),

location\_region VARCHAR(50),

location\_city VARCHAR(50)

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

--category table

CREATE EXTERNAL TABLE product\_category(

category\_id INT,

category\_name VARCHAR(50)

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

1. **Load data CSV to table**

LOAD DATA INPATH "/user/cloudera/location.csv" INTO TABLE location

LOAD DATA INPATH "/user/cloudera/ product \_category.csv" INTO TABLE product\_category

1. **Open mysql**

mysql -u root -p

password :cloudera

CREATE DATABASE bluejackstore

USE bluejackstore;

source insert.sql

sudo sqoop import-all-tables --connect jdbc:mysql://quickstart:3306/bluejackstore --username=root -P --hive-import --hive-database=bluejackstore

1. **Query Analysis**
   1. Show the most profitable product category in 2019.

SELECT pc.category\_name, SUM(p.product\_price – (p.discount \* p.product\_price)) \* quantity) AS profit

FROM sales s

JOIN sales\_detail sd ON s.sales\_id = sd.sales\_id

JOIN products p ON p.product\_id = sd.product\_id

JOIN product\_category pc ON pc.category\_id = p.product\_category\_id

WHERE YEAR(s.sales\_date) = 2019 AND is\_cancelled IS NULL

GROUP BY pc.category\_name

ORDER BY profit DESC

LIMIT 1

* 1. Show the location where the most transactions occurred in the most recent year that transactions occurred.

SELECT location\_country, location\_region, location\_city, COUNT(sales\_id) AS total\_transaction

FROM location l

JOIN sales s ON s.location\_id = l.location\_id

WHERE YEAR (sales\_data) = 2020

GROUP BY location\_country, location\_region, location\_city

ORDER BY total\_transaction DESC

LIMIT 1

* 1. Show the most profitable product by calculating its revenue. When there is a discount cut the product price by discount. Show data with profit more than 15.000.000.

SELECT product\_name, SUM((product\_price – (products.discount \* product\_price)) \* quantity ) AS Profit

FROM sales s

JOIN sales\_detail sd ON s.sales\_id = sd.sales\_id

JOIN products p ON p.product\_id = sd.product\_id

JOIN product\_category pc ON pc.id = p.product\_category\_id

WHERE p.discount > 0 AND is\_cancelled IS NULL

GROUP BY product\_name

HAVING Profit > 15000000

ORDER BY Profit DESC

LIMIT 1

* 1. Show customer who handled transactions more than the average of the total transactions by each customer.

SELECT c.customer\_name, COUNT(s.sales\_id) AS total\_transaction

FROM customer c

JOIN sales s ON c.customer\_id = s.customer\_id

JOIN (SELECT AVG(subquery.customerTransaction) AS avgCustomerTransaction

FROM

(SELECT COUNT (s.sales\_id) AS customerTransaction

FROM customer c

JOIN sales s ON c.customer\_id = s.customer\_id

GROUP BY c.customer\_id) AS subquery

) avg

GROUP BY

c.customer\_name, c.customer\_id, avg.avgCustomerTransaction

HAVING

total\_transaction > avg.avgCustomerTransaction

* 1. Show customers spending for last Christmas (December 2019). All customer who spends more than 10.000.000 will get a voucher for the next transaction with terms and condition as below:

A screenshot of a number of numbers

Description automatically generated

SELECT c.customer\_name, total,

CASE

WHEN x.total BETWEEN 10000000 AND 24999999 THEN "1.000.000"

WHEN x.total BETWEEN 25000000 AND 49999999 THEN "5.000.000"

WHEN x.total >= 50000000 THEN "10.000.000"

ELSE "0"

END AS Voucher

FROM customer c

JOIN (

SELECT s.customer\_id, SUM((p.product\_price \* (1 - p.discount)) \* IF(sd.is\_cancelled IS NULL, sd.quantity, 0)) AS total

FROM products p

JOIN

sales\_detail sd ON p.product\_id = sd.product\_id

JOIN

sales s ON s.sales\_id = sd.sales\_id

WHERE

YEAR(s.sales\_date) = 2019

AND MONTH(s.sales\_date) = 12

GROUP BY

s.customer\_id

) AS x ON c.customer\_id = x.customer\_id