Forecasting Trends and Demand

Capstone Project II

In this project we need to monitor various KPI that will be used by the RetailCart, Inc for making real-time discount related decisions which are offered for a limited period of time like whether discount should be continued, or modified and which discount to be removed during the season sales. Also, the same KPI will be used to monitor the store performance during the season sales.

Our main focus is on sales related data so we will create data mart from product sales only. A data mart around sales will generate actionable insights such as the following:

- Total sales in a day/week/quarter (both in terms of sales revenue and units sold)
- Top 5 most selling item categories in a day (both in terms of amount and quantity)
- Top 5 most profitable items in a day/week/quarter
- Top 5 most profitable stores in a day/week/month/quarter
- Total profit or loss in U.S. dollars every day in percentage

Real-time KPIs:

- Total sales in dollars
- Top 5 most selling items
- Top 5 most profitable items
- Top 5 stores with most sales
- Average discount on items in percentage
- Top 5 discounted Items

Steps

1) Loaded historical data to MySql tables stored at HDFS location which is then loaded to hive.
2) Loaded historical events data stored at HDFS location to hive.
3) Send real time events data from MySQL to Hbase using Kafka topics
4) Historical data from Hive and Real-time data from Hbase are used for regenerating all the KPIs

Tools used:

1) **Hive** - used for storing all the dimensions and facts
2) **MySQL** - for storing all the real time events as they are generated and then moved to HDFS.
3) **Kafka** - used as a message broker for sending real time events from MYSQL to Kafka.
4) **Hbase-** used to store the real-time events and then used for real-time KPIs

IMPLEMENTATION DETAILS

1. Perform batch ingestion from MySQL to Hive tables for static dimension data

Create MySQL tables and load data directly from HDFS Location to MySql using SQOOP.

```
-- Create project folder
mkdir edureka_788309_retailcart
cd edureka_788309_retailcart

-- Copy data from common hdfs location to user location
hadoop fs -mkdir edureka_788309_retailcart
hadoop fs -cp /bigdatapgp/common_folder/project_retailcart/batchdata/*
edureka_788309_retailcart/

-- Login to MySQL Database
mysql -u edu_labuser -h dbserver.edu.cloudlab.com -p -D labuser_database
```

a. retailcart_calendar_details.txt - Calendar table for retailCart

```
-- Create temporary/staging for storing calendar details
CREATE TABLE IF NOT EXISTS edureka_788309_retailcart_calendar_details_stage
(
      calendar_date varchar(100),
      date_desc varchar(100),
      week_day_nbr varchar(100),
      week_number varchar(100),
      week_name varchar(100),
      year_week_number varchar(100),
      month_number varchar(100),
      month_name varchar(100),
      quarter_number varchar(100),
      quarter_name varchar(100),
      half_year_number varchar(100),
      half_year_name varchar(100),
      geo_region_cd varchar(100)
);
```

```
-- Create actual table for calendar details
CREATE TABLE IF NOT EXISTS edureka_788309_retailcart_calendar_details
(
      calendar_date date,
      date_desc varchar(50),
      week_day_nbr smallint,
      week_number smallint,
      week_name varchar(50),
      year_week_number int,
      month_number smallint,
      month_name varchar(50),
      quarter_number smallint,
      quarter_name varchar(50),
      half_year_number smallint,
      half_year_name varchar(50),
      geo_region_cd char(2),
      PRIMARY KEY(calendar_date)
);

-- Export Calendar details data from HDFS file to MySQL stage table
sqoop export \
--connect jdbc:mysql://dbserver.edu.cloudlab.com:3306/labuser_database \
--username edu_labuser \
--password edureka \
--table edureka_788309_retailcart_calendar_details_stage \
--fields-terminated-by "\t" \
--optionally-enclosed-by "\"" \
--input-optionally-enclosed-by "\"" \
--export-dir
hdfs://nameservice1/user/edureka_788309/edureka_788309_retailcart/retailcart_c
alendar_details.txt

-- Insert from staging calendar table to actual MySQL table
INSERT INTO edureka_788309_retailcart_calendar_details
SELECT * FROM edureka_788309_retailcart_calendar_details_stage WHERE
calendar_date != 'calendar_date';
```

b. retailcart_currency_details.txt - Different types of currencies and their exchange rate for USD

```
-- Create staging table for currency details
CREATE TABLE IF NOT EXISTS edureka_788309_retailcart_currency_details_stage
(
      currency_id varchar(100),
      currency_code varchar(100),
      currency_name varchar(100),
      usd_exchange_rate varchar(100)
);
```

```
-- Create actual currency details table
CREATE TABLE IF NOT EXISTS edureka_788309_retailcart_currency_details
(
	currency_id int,
	currency_code varchar(10),
	currency_name varchar(50),
	usd_exchange_rate decimal(18,4),
	PRIMARY KEY(currency_id)
);

-- Export Currency details data from HDFS file to MySQL stage table
sqoop export \
--connect jdbc:mysql://dbserver.edu.cloudlab.com:3306/labuser_database \
--username edu_labuser \
--password edureka \
--table edureka_788309_retailcart_currency_details_stage \
--fields-terminated-by "\t" \
--optionally-enclosed-by "\"" \
--input-optionally-enclosed-by "\"" \
--export-dir
hdfs://nameservice1/user/edureka_788309/edureka_788309_retailcart/retailcart_c
urrency_details.txt

--Insert from staging currency table to actual MySQL table
INSERT INTO edureka_788309_retailcart_currency_details
SELECT REPLACE(currency_id,'"',''), currency_code, currency_name,
usd_exchange_rate FROM edureka_788309_retailcart_currency_details_stage
WHERE currency_id != 'currency_id';
```

c. retailcart_department_details.txt - Department and sub-department details of an item

```
-- Create staging table for department details
CREATE TABLE IF NOT EXISTS edureka_788309_retailcart_department_details_stage
(
	department_number varchar(100),
	department_category_number varchar(100),
	department_sub_catg_number varchar(100),
	department_description varchar(100),
	department_category_description varchar(100),
	department_sub_catg_desc varchar(100),
	geo_region_cd varchar(100)
);
```

```
-- Create actual department details table
CREATE TABLE IF NOT EXISTS edureka_788309_retailcart_department_details
(
      department_number smallint,
      department_category_number int,
      department_sub_catg_number int,
      department_description varchar(80),
      department_category_description varchar(80),
      department_sub_catg_desc varchar(80),
      geo_region_cd varchar(10)
);


-- Export Department details data from HDFS file to MySQL stage table
sqoop export \
--connect jdbc:mysql://dbserver.edu.cloudlab.com:3306/labuser_database \
--username edu_labuser \
--password edureka \
--table edureka_788309_retailcart_department_details_stage \
--fields-terminated-by "\t" \
--optionally-enclosed-by "\"" \
--input-optionally-enclosed-by "\"" \
--export-dir
hdfs://nameservice1/user/edureka_788309/edureka_788309_retailcart/retailcart_d
epartment_details.txt

--Insert from stage department details table to actual MySQL table
INSERT INTO edureka_788309_retailcart_department_details
SELECT
       department_number,
       REPLACE(department_category_number,"NULL",NULL),
       REPLACE(department_sub_catg_number,"NULL",NULL),
       REPLACE(department_description,"NULL",NULL),
       REPLACE(department_category_description,"NULL",NULL),
       REPLACE(department_sub_catg_desc,"NULL",NULL),
       REPLACE(geo_region_cd,"NULL",NULL)
from edureka_788309_retailcart_department_details_stage
where department_number != 'department_number';
```

d. retailcart_item_details.txt - Item details for retailCart

```
-- Create stage table for item details table
CREATE TABLE IF NOT EXISTS edureka_788309_retailcart_item_details_stage
(
      item_id varchar(100),
      geo_region_cd varchar(100),
      item_description varchar(100),
      unique_product_cd varchar(100),
      unique_product_cd_desc varchar(100),
      department_number varchar(100),
      department_category_number varchar(100),
      department_sub_catg_number varchar(100),
      vendor_name varchar(100),
      vendor_number varchar(100),
      item_status_cd varchar(100),
      item_status_desc varchar(100),
      unit_cost varchar(100)
);

-- Create actual table for item details
CREATE TABLE IF NOT EXISTS edureka_788309_retailcart_item_details
(
      item_id int,
      geo_region_cd char(2),
      item_description varchar(80),
      unique_product_cd decimal(18,0),
      unique_product_cd_desc varchar(80),
      department_number smallint,
      department_category_number int,
      department_sub_catg_number int,
      vendor_name varchar(80),
      vendor_number int,
      item_status_cd char(1),
      item_status_desc varchar(80),
      unit_cost decimal(15,2),
      PRIMARY KEY(item_id)
);
```

```
-- Export item details data from HDFS file to MySQL temp table
sqoop export \
--connect jdbc:mysql://dbserver.edu.cloudlab.com:3306/labuser_database \
--username edu_labuser \
--password edureka \
--table edureka_788309_retailcart_item_details_stage \
--fields-terminated-by "\t" \
--export-dir
hdfs://nameservice1/user/edureka_788309/edureka_788309_retailcart/retailcart_i
tem_details.txt

--Insert from temp item details table to actual MySQL table
INSERT INTO edureka_788309_retailcart_item_details
SELECT
      item_id,
      geo_region_cd,
      item_description,
      unique_product_cd,
      unique_product_cd_desc,
      REPLACE(REPLACE(department_number,"dept-num:",""),"NULL",NULL),
REPLACE(REPLACE(department_category_number,"dept-catg-num:",""),"NULL",NULL),
REPLACE(REPLACE(department_sub_catg_number,"dept-sub-catg-num:",""),"NULL",NUL
L),
      vendor_name,
      vendor_number,
      item_status_cd,
      item_status_desc,
      REPLACE(unit_cost,"NULL",NULL)
FROM edureka_788309_retailcart_item_details_stage
WHERE item_id != 'item_id';
```

e. retailcart_store_details.txt - Store details for retailCart

```
-- Create stage table for store details table
CREATE TABLE edureka_788309_retailcart_store_details_stage
(
      store_id varchar(100),
      geo_region_cd varchar(100),
      store_name varchar(100),
      sub_division_name varchar(100),
      sub_division_number varchar(100),
      region_number varchar(100),
      region_name varchar(100),
      market_number varchar(100),
      market_name varchar(100),
      city_name varchar(80),
      open_date varchar(100),
      open_status_desc varchar(100),
      postal_cd varchar(100),
	state_prov_cd varchar(100)
);

-- Create actual store details table
CREATE TABLE edureka_788309_retailcart_store_details
(
	store_id int,
	geo_region_cd char(2),
	store_name varchar(100),
	sub_division_name varchar(100),
	sub_division_number char(2),
	region_number smallint,
	region_name varchar(100),
	market_number smallint,
	market_name varchar(100),
	city_name varchar(80),
	open_date date,
	open_status_desc varchar(40),
	postal_cd char(10),
	state_prov_cd char(2),
	PRIMARY KEY(store_id)
);

-- Export store details data from HDFS file to MySQL stage table
sqoop export \
--connect jdbc:mysql://dbserver.edu.cloudlab.com:3306/labuser_database \
--username edu_labuser \
--password edureka \
--table edureka_788309_retailcart_store_details_stage \
--fields-terminated-by "\t" \
--export-dir
hdfs://nameservice1/user/edureka_788309/edureka_788309_retailcart/retailcart_s
tore_details.txt

--Insert from temp store details table to actual MySQL table
INSERT INTO edureka_788309_retailcart_store_details
SELECT *
FROM edureka_788309_retailcart_store_details_stage
WHERE store_id != 'store_id';
```

f. Drop all staging tables

```
DROP TABLE IF EXISTS edureka_788309_retailcart_calendar_details_stage;
DROP TABLE IF EXISTS edureka_788309_retailcart_currency_details_stage;
DROP TABLE IF EXISTS edureka_788309_retailcart_department_details_stage;
DROP TABLE IF EXISTS edureka_788309_retailcart_item_details_stage;
DROP TABLE IF EXISTS edureka_788309_retailcart_store_details_stage;
```

2. Perform batch ingestion from MySQL to Hive tables for static dimension data

```
-- Create and select Hive database
CREATE DATABASE IF NOT EXISTS edureka_788309;

USE edureka_788309;

-- Create and import calendar details table in hive.
sqoop import \
--connect jdbc:mysql://dbserver.edu.cloudlab.com:3306/labuser_database \
--username edu_labuser \
--password edureka \
--split-by calendar_date \
--query 'select a.*,now() AS row_insertion_dttm FROM
edureka_788309_retailcart_calendar_details a WHERE $CONDITIONS' \
--hcatalog-database edureka_788309 \
--hcatalog-table edureka_788309_retailcart_calendar_details \
--create-hcatalog-table \
--hcatalog-storage-stanza "stored as orcfile"

-- Create and import currency details table in hive
sqoop import \
--connect jdbc:mysql://dbserver.edu.cloudlab.com:3306/labuser_database \
--username edu_labuser \
--password edureka \
--split-by currency_id \
--query 'select a.*,now() AS row_insertion_dttm FROM
edureka_788309_retailcart_currency_details a WHERE $CONDITIONS' \
--hcatalog-database edureka_788309 \
--hcatalog-table edureka_788309_retailcart_currency_details \
--create-hcatalog-table \
--hcatalog-storage-stanza "stored as orcfile"

-- Create and import department details table in hive
sqoop import \
--connect jdbc:mysql://dbserver.edu.cloudlab.com:3306/labuser_database \
--username edu_labuser \
--password edureka \
--split-by department_number \
--query 'select a.*,now() AS row_insertion_dttm FROM
edureka_788309_retailcart_department_details a WHERE $CONDITIONS' \
--hcatalog-database edureka_788309 \
--hcatalog-table edureka_788309_retailcart_department_details \
--create-hcatalog-table \
--hcatalog-storage-stanza "stored as orcfile"
```

```
-- Create and insert item details in hive
sqoop import \
--connect jdbc:mysql://dbserver.edu.cloudlab.com:3306/labuser_database \
--username edu_labuser \
--password edureka \
--split-by item_id \
--query 'select a.*,now() AS row_insertion_dttm FROM
edureka_788309_retailcart_item_details a WHERE $CONDITIONS' \
--hcatalog-database edureka_788309 \
--hcatalog-table edureka_788309_retailcart_item_details \
--create-hcatalog-table \
--hcatalog-storage-stanza "stored as orcfile"

-- Create and insert store details in hive
sqoop import \
--connect jdbc:mysql://dbserver.edu.cloudlab.com:3306/labuser_database \
--username edu_labuser \
--password edureka \
--split-by store_id \
--query 'select a.*,now() AS row_insertion_dttm FROM
edureka_788309_retailcart_store_details a WHERE $CONDITIONS' \
--hcatalog-database edureka_788309 \
--hcatalog-table edureka_788309_retailcart_store_details \
--create-hcatalog-table \
--hcatalog-storage-stanza "stored as orcfile"
```

3. Perform batch load of historical data for sales and price change transactions from existing ORC files placed in the HDFS path mentioned above

Sales Transaction Events

```
-- Enable dynamic partition
set hive.exec.dynamic.partition.mode=nonstrict;

-- Copy Historical sales transaction event data
hadoop fs -cp
/bigdatapgp/common_folder/project_retailcart/history_data/store_item_sales_dat
a/ edureka_788309_retailcart/
```

```
-- Create staging hive table for historical sales item store data
CREATE TABLE IF NOT EXISTS
edureka_788309_retailcart_sales_transaction_events_stage
(
      sales_id INT,
      sales_date DATE,
      store_id INT,
      item_id INT,
      scan_type TINYINT,
      geo_region_cd CHAR(2),
      currency_code VARCHAR(20),
      scan_id INT,
      sold_unit_quantity decimal(9,2),
      scan_date DATE,
      scan_time varchar(20),
      scan_dept_nbr SMALLINT
)
STORED AS ORC
tblproperties("skip.header.line.count"="1");

-- Create hive table for historical sales item store data
CREATE TABLE IF NOT EXISTS edureka_788309_retailcart_sales_transaction_events
(
      sales_id INT,
      sales_date DATE,
      store_id INT,
      item_id INT,
      scan_type TINYINT,
      currency_code VARCHAR(20),
      scan_id INT,
      sold_unit_quantity decimal(9,2),
      scan_date DATE,
      scan_time varchar(20),
      scan_dept_nbr SMALLINT,
      row_insertion_dttm string
) PARTITIONED BY(geo_region_cd varchar(2))
STORED AS ORC;

-- Load historical store item sales store item data to hive table
LOAD DATA INPATH
'hdfs://nameservice1/user/edureka_788309/edureka_788309_retailcart/store_item_
sales_data/*'
OVERWRITE INTO TABLE
edureka_788309.edureka_788309_retailcart_sales_transaction_events_stage;

-- Insert sales transaction events from stage table
INSERT OVERWRITE TABLE edureka_788309_retailcart_sales_transaction_events
PARTITION(geo_region_cd)
```

```
SELECT sales_id,cast(sales_date as date),
store_id,item_id,scan_type,currency_code,scan_id,sold_unit_quantity,scan_date,
scan_time,scan_dept_nbr,current_timestamp() AS row_insertion_dttm,
geo_region_cd
FROM edureka_788309_retailcart_sales_transaction_events_stage;
```

Item Price Events

```
-- Copy Historical item price change event data
hadoop fs -cp
/bigdatapgp/common_folder/project_retailcart/history_data/store_item_price_cha
nge/ edureka_788309_retailcart/

-- Create staging hive table for historical store item price change data
CREATE TABLE IF NOT EXISTS edureka_788309_retailcart_price_change_events_stage
(
      item_id INT,
      store_id INT,
      price_chng_activation_ts TIMESTAMP,
      geo_region_cd char(2),
      price_change_reason varchar(100),
      business_date DATE,
      prev_price_amt decimal(15,2),
      curr_price_amt decimal(15,2)
)
STORED AS ORC
tblproperties("skip.header.line.count"="1");

-- Create hive table for historical store item price change events
CREATE TABLE IF NOT EXISTS edureka_788309_retailcart_price_change_events
(
      item_id INT,
      store_id INT,
      price_chng_activation_ts TIMESTAMP,
      price_change_reason varchar(100),
      business_date DATE,
      prev_price_amt decimal(15,2),
      curr_price_amt decimal(15,2),
      row_insertion_dttm string
) PARTITIONED BY(geo_region_cd varchar(2))
STORED AS ORC;

-- Load historical store item sales price change data to hive table
LOAD DATA INPATH
'hdfs://nameservice1/user/edureka_788309/edureka_788309_retailcart/store_item_
price_change/'
OVERWRITE INTO TABLE
edureka_788309.edureka_788309_retailcart_price_change_events_stage;
```

```
-- Insert item price events from stage table
INSERT OVERWRITE TABLE edureka_788309_retailcart_price_change_events
PARTITION(geo_region_cd)
SELECT
item_id,store_id,price_chng_activation_ts,price_change_reason,business_date,pr
ev_price_amt,curr_price_amt,current_timestamp() as row_insertion_dttm,
geo_region_cd
FROM edureka_788309_retailcart_price_change_events_stage;

-- Drop staging tables
DROP TABLE IF EXISTS edureka_788309_retailcart_price_change_events_stage;
DROP TABLE IF EXISTS edureka_788309_retailcart_sales_transaction_events_stage;
```

4. Capture new sales transactions and item price change events from respective MySQL tables using real-time simulator and produce them to a Kafka topic

```
-- Login to MySQL Database
mysql -u edu_labuser -h dbserver.edu.cloudlab.com -p -D labuser_database

-- Create Sales transaction event table
CREATE TABLE IF NOT EXISTS
labuser_database.edureka_788309_retailcart_sales_transaction_events(
      sales_id INT,
      Sales_date DATE,
      store_id INT,
      item_id INT,
      scan_type TINYINT,
      geo_region_cd CHAR(2),
      currency_code VARCHAR(20),
      scan_id INT,
      sold_unit_quantity decimal(9,2),
      scan_date DATE,
      scan_time varchar(20),
      scan_dept_nbr SMALLINT,
      row_insertion_dttm timestamp
);

-- Create price change event table
CREATE TABLE IF NOT EXISTS
labuser_database.edureka_788309_retailcart_price_change_events(
      item_id INT,
      store_id INT,
      price_chng_activation_ts TIMESTAMP,
      geo_region_cd char(2),
      price_change_reason varchar(100),
      business_date DATE,
      prev_price_amt decimal(15,2),
      curr_price_amt decimal(15,2),
      row_insertion_dttm TIMESTAMP
);
```

Activate the real time stimulator

```
hdfs dfs -get
/bigdatapgp/common_folder/project_retailcart/realtimedata
python2 realtimedata/real_time_simulator.py edureka_788309
```

Create Kafka topic so that real time data can be sent to it

```
-- Delete kafka topic before creating
kafka-topics --delete --bootstrap-server ip-20-0-31-221.ec2.internal:9092
--topic edureka788309salestransaction

-- Create new kafka topic for sales transactions
kafka-topics --create --bootstrap-server ip-20-0-31-221.ec2.internal:9092
--topic edureka788309salestransaction --partitions 1 --replication-factor 1

-- Delete kafka topic before creating
kafka-topics --delete --bootstrap-server ip-20-0-31-221.ec2.internal:9092
--topic edureka788309pricechange

-- create new kafka topic for item price change
kafka-topics --create --bootstrap-server  ip-20-0-31-221.ec2.internal:9092
--topic edureka788309pricechange --partitions 1 --replication-factor 1
```

Create HBase Table

-- Login on Hbase shell

```
hbase shell

-- Create sales transaction event table
create 'edureka_788309_retailcart_sales_transaction_events', 'cf1'

-- Create price change event table
create 'edureka_788309_retailcart_price_change_events', 'cf1'
```

Real time Kafka Producer/Consumer code for moving data from MySQL to HBase

```
/* Main Class */
public class RdbmsToKafkaProcessor {

   public static void main(String[] args) throws Exception{

       SalesTransactionProducer stProducer = new SalesTransactionProducer();
       ItemPriceChangeProducer ipcProducer = new ItemPriceChangeProducer();

       // Call To Producer Functions
       stProducer.copyMySqlRecordsToKafkaTopic();
       ipcProducer.copyMySqlRecordsToKafkaTopic();
   }

}
```

/*Sales Transaction Producer Code */

```
import java.util.Properties;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.DriverManager;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerRecord;

public class SalesTransactionProducer {
    private static KafkaProducer<String, String> producer;

    private static final String topic = "edureka788309salestransaction";

    SalesTransactionProducer() throws Exception {

        Properties producerProps = new Properties();
        producerProps.put("bootstrap.servers",
"ip-20-0-31-221.ec2.internal:9092");
        producerProps.put("acks", "all");
        producerProps.put("retries", 0);
        producerProps.put("batch.size", 16384);
        producerProps.put("linger.ms", 1);
        producerProps.put("buffer.memory", 33554432);
        producerProps.put("key.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
        producerProps.put("value.serializer",
"org.apache.kafka.common.serialization.StringSerializer");

        producer = new KafkaProducer<String,String>(producerProps);
    }

    public void copyMySqlRecordsToKafkaTopic() throws Exception {

        String msg = null;
        String sales_id =  null;
        String store_id =  null;
        String item_id =  null;
        String scan_type =  null;
        String geo_region_cd =  null;
        String currency_code = null;
        String scan_id =  null;
        String sold_unit_quantity =  null;
        String sales_timestamp =  null;
        String scan_dept_nbr = null;
        String row_insertion_dttm =  null;

        String myDriver = "com.mysql.jdbc.Driver";
        String myUrl = 
"jdbc:mysql://dbserver.edu.cloudlab.com/labuser_database";
        Class.forName(myDriver);
        Connection conn = DriverManager.getConnection(myUrl,"edu_labuser", 
"edureka");
        String query = "SELECT * FROM 
edureka_788309_retailcart_sales_transaction_events";
        Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(query);
        while (rs.next())
        {
            sales_id = rs.getString("sales_id");
            store_id = rs.getString("store_id");
            item_id = rs.getString("item_id");
            scan_type = rs.getString("scan_type");
            geo_region_cd = rs.getString("geo_region_cd");
            currency_code = rs.getString("currency_code");
            scan_id = rs.getString("scan_id");
            sold_unit_quantity = rs.getString("sold_unit_quantity");
            sales_timestamp = rs.getString("sales_timestamp");
            scan_dept_nbr = rs.getString("scan_dept_nbr");
            row_insertion_dttm = rs.getString("row_insertion_dttm");

            msg = 
sales_id+","+store_id+","+item_id+","+scan_type+","+geo_region_cd+","+currency
_code+","+scan_id+","+sold_unit_quantity+","+sales_timestamp+","+
            scan_dept_nbr+","+row_insertion_dttm ;
            ProducerRecord<String,String> producerRecord = new 
ProducerRecord<String,String>(topic,msg);
            producer.send(producerRecord);
        }
        conn.close();
    }
}
```

/* Producer Class for Item Price Change */

```
import java.util.Properties;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.DriverManager;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerRecord;

public class ItemPriceChangeProducer {
    private static KafkaProducer<String, String> producer;
    private static final String topic = "edureka788309pricechange";

    ItemPriceChangeProducer() {
        Properties producerProps = new Properties();

        // Set Producer Properties
        producerProps.put("bootstrap.servers",
"ip-20-0-31-221.ec2.internal:9092");
        producerProps.put("acks", "all");
        producerProps.put("retries", 0);
        producerProps.put("batch.size", 16384);
        producerProps.put("linger.ms", 1);
        producerProps.put("buffer.memory", 33554432);
        producerProps.put("key.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
        producerProps.put("value.serializer",
"org.apache.kafka.common.serialization.StringSerializer");

        producer = new KafkaProducer<String,String>(producerProps);
    }

    public void copyMySqlRecordsToKafkaTopic() throws Exception {

        String msg = null;
        String event_id = null;
        String item_id = null;
        String store_id = null;
        String price_chng_activation_ts = null;
        String geo_region_cd = null;
        String price_change_reason = null;
        String prev_price_amt = null;
        String curr_price_amt = null;
        String row_insertion_dttm = null;

        String myDriver = "com.mysql.jdbc.Driver";
        String myUrl =
"jdbc:mysql://dbserver.edu.cloudlab.com/labuser_database";
        Class.forName(myDriver);
        Connection conn = DriverManager.getConnection(myUrl,"edu_labuser",
"edureka");
        String query = "SELECT * FROM
edureka_788309_retailcart_price_change_events";
        Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(query);
        while (rs.next()) {

            event_id = rs.getString("event_id");
            item_id = rs.getString("item_id");
            store_id = rs.getString("store_id");
            price_chng_activation_ts =
rs.getString("price_chng_activation_ts");
            geo_region_cd = rs.getString("geo_region_cd");
            price_change_reason = rs.getString("price_change_reason");
            prev_price_amt = rs.getString("prev_price_amt");
            curr_price_amt = rs.getString("curr_price_amt");
            row_insertion_dttm = rs.getString("row_insertion_dttm");

            msg =
event_id+","+item_id+","+store_id+","+price_chng_activation_ts+","+geo_region_
cd+","+price_change_reason+","+prev_price_amt+","+curr_price_amt+","+row_inser
tion_dttm;
            ProducerRecord<String,String> producerRecord = new
ProducerRecord<String,String>(topic,msg);
            producer.send(producerRecord);
        }
        conn.close();
    }
}
```

6. Create a consumer application that will consume the incoming messages from the topic and ingest

```
/* Consumer code for sales transactions */
import java.nio.charset.Charset;
import java.util.Arrays;
import java.time.Duration;

import java.util.Properties;
import org.apache.kafka.clients.consumer.ConsumerRecord;
import org.apache.kafka.clients.consumer.ConsumerRecords;
import org.apache.kafka.clients.consumer.KafkaConsumer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.Connection;
import org.apache.hadoop.hbase.client.ConnectionFactory;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.client.Table;


public class SalesTransactionHbaseConsumer {
    public static void main(String[] args) throws Exception {
        Properties consumerProps = new Properties();
        consumerProps.put("bootstrap.servers",
"ip-20-0-31-221.ec2.internal:9092");
        consumerProps.put("group.id", "retail-group1");
        consumerProps.put("enable.auto.commit", "true");
        consumerProps.put("auto.commit.interval.ms", "1000");
        consumerProps.put("session.timeout.ms", "30000");
        consumerProps.put("key.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
        consumerProps.put("value.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");

        System.out.println("Initiaization Completed ...");

        KafkaConsumer<String, String> consumer = new
KafkaConsumer<>(consumerProps);

        final Configuration hbaseConf = HBaseConfiguration.create();
        hbaseConf.set("hbase.master", "ip-20-0-21-196.ec2.internal:60020");
        hbaseConf.set("hbase.zookeeper.quorum","ip-20-0-21-196");
        hbaseConf.set("hbase.zookeeper.property.clientPort", "2181");
        Connection hBase = ConnectionFactory.createConnection(hbaseConf);
        Table table =
hBase.getTable(TableName.valueOf("edureka_788309_retailcart_sales_transaction_
events"));

        consumer.subscribe(Arrays.asList("edureka788309salestransaction"));

        Long timeout = new Long(1000L);
        byte[] cf = "cf1".getBytes(Charset.forName("UTF-8"));

        String sales_id =  null;
        String store_id =  null;
        String item_id =  null;
        String scan_type =  null;
        String geo_region_cd =  null;
        String currency_code = null;
        String scan_id =  null;
        String sold_unit_quantity =  null;
        String sales_timestamp =  null;
        String scan_dept_nbr = null;
        String row_insertion_dttm =  null;

        while (true) {

            ConsumerRecords<String, String> records =
consumer.poll(Duration.ofMillis(timeout));

            System.out.println("Total records:" + records.count());
            for (ConsumerRecord<String, String> record : records) {

                String[] key = record.value().toString().split(",");

                System.out.println("Total Breaks:" + key.length);

                sales_id = key[0];
                store_id = key[1];
                geo_region_cd = key[4];
                item_id = key[2];
                scan_type = key[3];
                currency_code = key[5];
                scan_id = key[6];
                sold_unit_quantity = key[7];
                sales_timestamp = key[8];
                scan_dept_nbr = key[9];
                row_insertion_dttm = key[10];

                Put p = new
Put(sales_id.toString().getBytes(Charset.forName("UTF-8")));


                p.addColumn(cf,"sales_id".getBytes(Charset.forName("UTF-8")),
sales_id.getBytes(Charset.forName("UTF-8")));
                p.addColumn(cf,"store_id".getBytes(Charset.forName("UTF-8")),
store_id.getBytes(Charset.forName("UTF-8")));
                p.addColumn(cf,"item_id".getBytes(Charset.forName("UTF-8")),
item_id.getBytes(Charset.forName("UTF-8")));
                p.addColumn(cf,"scan_type".getBytes(Charset.forName("UTF-8")),
scan_type.getBytes(Charset.forName("UTF-8")));

p.addColumn(cf,"geo_region_cd".getBytes(Charset.forName("UTF-8")),
geo_region_cd.getBytes(Charset.forName("UTF-8")));

p.addColumn(cf,"currency_code".getBytes(Charset.forName("UTF-8")),
currency_code.getBytes(Charset.forName("UTF-8")));
                p.addColumn(cf,"scan_id".getBytes(Charset.forName("UTF-8")),
scan_id.getBytes(Charset.forName("UTF-8")));

p.addColumn(cf,"sold_unit_quantity".getBytes(Charset.forName("UTF-8")),
sold_unit_quantity.getBytes(Charset.forName("UTF-8")));

p.addColumn(cf,"sales_timestamp".getBytes(Charset.forName("UTF-8")),
sales_timestamp.getBytes(Charset.forName("UTF-8")));

p.addColumn(cf,"scan_dept_nbr".getBytes(Charset.forName("UTF-8")),
scan_dept_nbr.getBytes(Charset.forName("UTF-8")));
p.addColumn(cf,"row_insertion_dttm".getBytes(Charset.forName("UTF-8")),
row_insertion_dttm.getBytes(Charset.forName("UTF-8")));

               System.out.println("Record Inserted into hbase: " + sales_id);

               table.put(p);
            }
        }
    }
}
```

/* Consumer code for Item change data */

```
import java.nio.charset.Charset;
import java.util.Arrays;
import java.time.Duration;

import java.util.Properties;
import org.apache.kafka.clients.consumer.ConsumerRecord;
import org.apache.kafka.clients.consumer.ConsumerRecords;
import org.apache.kafka.clients.consumer.KafkaConsumer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.Connection;
import org.apache.hadoop.hbase.client.ConnectionFactory;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.client.Table;

public class ItemPriceChangeHbaseConsumer
{
   public static void main(String[] args) throws Exception{
        Properties consumerProps = new Properties();
        consumerProps.put("bootstrap.servers",
"ip-20-0-31-221.ec2.internal:9092");
        consumerProps.put("group.id", "retail-price-group1");
        consumerProps.put("enable.auto.commit", "true");
        consumerProps.put("auto.commit.interval.ms", "1000");
        consumerProps.put("session.timeout.ms", "300000");
        consumerProps.put("key.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
        consumerProps.put("value.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");

        KafkaConsumer<String, String> consumer = new
KafkaConsumer<>(consumerProps);

        final Configuration hbaseConf = HBaseConfiguration.create();
        hbaseConf.set("hbase.master", "ip-20-0-21-196.ec2.internal:60020");
        hbaseConf.set("hbase.zookeeper.quorum","ip-20-0-21-196");
        hbaseConf.set("hbase.zookeeper.property.clientPort", "2181");
        Connection hBase = ConnectionFactory.createConnection(hbaseConf);

        Table table =
hBase.getTable(TableName.valueOf("edureka_788309_retailcart_price_change_event
s"));

        consumer.subscribe(Arrays.asList("edureka788309pricechange"));

        Long timeout = new Long(1000L);
        byte[] cf = "cf1".getBytes(Charset.forName("UTF-8"));

        String event_id = null;
        String item_id = null;
        String store_id = null;
        String price_chng_activation_ts = null;
        String geo_region_cd = null;
        String price_change_reason = null;
        String prev_price_amt = null;
        String curr_price_amt = null;
        String row_insertion_dttm = null;

        while (true) {

            ConsumerRecords<String, String> records =
consumer.poll(Duration.ofMillis(timeout));

            System.out.println("Total records:" + records.count());

            for (ConsumerRecord<String, String> record : records) {
                String[] key = record.value().toString().split(",");

                System.out.println("Total Breaks:" + key.length);

                event_id = key[0];
                item_id = key[1];
                store_id = key[2];
                price_chng_activation_ts = key[3];
                geo_region_cd = key[4];
                price_change_reason = key[5];
                prev_price_amt = key[6];
                curr_price_amt = key[7];
                row_insertion_dttm = key[8];

                Put p = new
Put(event_id.toString().getBytes(Charset.forName("UTF-8")));
```

```
                p.addColumn(cf,"event_id".getBytes(Charset.forName("UTF-8")),
event_id.getBytes(Charset.forName("UTF-8")));
                p.addColumn(cf,"item_id".getBytes(Charset.forName("UTF-8")),
item_id.getBytes(Charset.forName("UTF-8")));
                p.addColumn(cf,"store_id".getBytes(Charset.forName("UTF-8")),
store_id.getBytes(Charset.forName("UTF-8")));

p.addColumn(cf,"price_chng_activation_ts".getBytes(Charset.forName("UTF-8")),
price_chng_activation_ts.getBytes(Charset.forName("UTF-8")));

p.addColumn(cf,"geo_region_cd".getBytes(Charset.forName("UTF-8")),
geo_region_cd.getBytes(Charset.forName("UTF-8")));

p.addColumn(cf,"price_change_reason".getBytes(Charset.forName("UTF-8")),
price_change_reason.getBytes(Charset.forName("UTF-8")));

p.addColumn(cf,"prev_price_amt".getBytes(Charset.forName("UTF-8")),
prev_price_amt.getBytes(Charset.forName("UTF-8")));

p.addColumn(cf,"curr_price_amt".getBytes(Charset.forName("UTF-8")),
curr_price_amt.getBytes(Charset.forName("UTF-8")));

p.addColumn(cf,"row_insertion_dttm".getBytes(Charset.forName("UTF-8")),
row_insertion_dttm.getBytes(Charset.forName("UTF-8")));
                table.put(p);

                System.out.println("Record Inserted into hbase: " + event_id);
            }
        }
    }

}
```

Execute the Producer and Consumer JARs

```
-- Producer Jar
hadoop jar
edureka_788309_retailcart/RdbmsToKafka-1.0-SNAPSHOT-jar-with-dependencies.jar
RdbmsToKafkaProcessor

-- Sales transaction event consumer Jar
hadoop jar
edureka_788309_retailcart/KafkaToHbaseSalesTransactionConsumer-1.0-SNAPSHOT-ja
r-with-dependencies.jar SalesTransactionHbaseConsumer

-- Item Price event consumer Jar
hadoop jar
edureka_788309_retailcart/KafkaToHbaseItemPriceChangeConsumer-1.0-SNAPSHOT-jar
-with-dependencies.jar ItemPriceChangeHbaseConsumer
```

Using Hive to access an existing HBase table

Load sales transaction table in hive from hbase

```
CREATE EXTERNAL TABLE edureka_788309_retailcart_sales_transaction_events_hbase
(
      sales_id INT,
      store_id INT,
      item_id INT,
      scan_type TINYINT,
      geo_region_cd CHAR(2),
      currency_code VARCHAR(20),
      scan_id INT,
      sold_unit_quantity decimal(9,2),
      sales_timestamp TIMESTAMP,
      scan_dept_nbr SMALLINT,
      row_insertion_dttm TIMESTAMP
)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping" =
"cf1:store_id,cf1:item_id,cf1:scan_type,cf1:geo_region_cd,cf1:currency_code,cf
1:scan_id,cf1:sold_unit_quantity,cf1:sales_timestamp,cf1:scan_dept_nbr,cf1:row
_insertion_dttm")
TBLPROPERTIES("hbase.table.name" =
"edureka_788309_retailcart_sales_transaction_events");
```

Load Item Change Price table in hive from hbase

```
CREATE EXTERNAL TABLE edureka_788309_retailcart_price_change_events_hbase
(
      item_id INT,
      store_id INT,
      price_chng_activation_ts TIMESTAMP,
      geo_region_cd char(2),
      price_change_reason varchar(100),
      business_date DATE,
      prev_price_amt decimal(15,2),
      curr_price_amt decimal(15,2),
      row_insertion_dttm TIMESTAMP
)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping" =
"cf1:store_id,cf1:price_chng_activation_ts,cf1:geo_region_cd,cf1:price_change_
reason,cf1:business_date,cf1:prev_price_amt,cf1:curr_price_amt,cf1:row_inserti
on_dttm")
```

```
TBLPROPERTIES("hbase.table.name" =
"edureka_788309_retailcart_price_change_events");
```

Creating view for historical and real time data as view for easy querying
-- Sales Transaction data

```
CREATE VIEW edureka_788309_retailcart_sales_transaction_events_view AS
SELECT
sales_id,store_id,item_id,scan_type,geo_region_cd,currency_code,scan_id,sold_u
nit_quantity,concat(scan_date,' ',scan_time) AS
sales_timestamp,scan_dept_nbr,row_insertion_dttm
FROM
edureka_788309_retailcart_sales_transaction_events WHERE scan_date IS NOT NULL
AND scan_time IS NOT NULL
UNION ALL
SELECT
sales_id,store_id,item_id,scan_type,geo_region_cd,currency_code,scan_id,sold_u
nit_quantity,sales_timestamp,scan_dept_nbr,row_insertion_dttm
FROM
edureka_788309_retailcart_sales_transaction_events_hbase;
```

-- Price Change data

```
CREATE VIEW edureka_788309_retailcart_price_change_events_view AS
SELECT
item_id,store_id,price_chng_activation_ts,geo_region_cd,price_change_reason,bu
siness_date,prev_price_amt,curr_price_amt,row_insertion_dttm
FROM
edureka_788309_retailcart_price_change_events
UNION ALL
SELECT
item_id,store_id,price_chng_activation_ts,geo_region_cd,price_change_reason,bu
siness_date,prev_price_amt,curr_price_amt,row_insertion_dttm
FROM
edureka_788309_retailcart_price_change_events_hbase;
```

7. Once we have captured both batch and real-time data in stage tables, perform a data modeling around business KPIs in Hive to create facts and dimensions

1. Total sales in dollars

```
SELECT SUM(sold_unit_quantity * applied_price * usd_exchange_rate) as
total_sales_dollars
FROM (
SELECT
   a.sold_unit_quantity, (CASE WHEN a.sales_timestamp <
b.price_chng_activation_ts THEN prev_price_amt ELSE curr_price_amt END) as
applied_price, c.usd_exchange_rate, row_number() OVER (PARTITION BY b.item_id
ORDER BY b.price_chng_activation_ts) as rnk
FROM
    edureka_788309_retailcart_sales_transaction_events_view a
    INNER JOIN edureka_788309_retailcart_price_change_events_view b
ON(a.store_id = b.store_id AND a.item_id = b.item_id AND a.geo_region_cd =
b.geo_region_cd )
    INNER JOIN edureka_788309_retailcart_currency_details c ON(
a.currency_code = c.currency_code)
WHERE
    a.sales_timestamp > b.price_chng_activation_ts
) as temp_tbl WHERE rnk = 1;
```

total_sales_dollars
8775984.86710673

2. Top 5 most selling items

```
SELECT
    a.item_id, d.item_description, sum(a.sold_unit_quantity) AS sold_items
FROM
    edureka_788309_retailcart_sales_transaction_events_view a
    INNER JOIN edureka_788309_retailcart_item_details d ON(a.item_id =
d.item_id AND a.geo_region_cd=d.geo_region_cd)
GROUP BY
    a.item_id,d.item_description
ORDER BY
    sold_items DESC
LIMIT 5;
```

a.item_id	d.item_description	sold_items
203811437	CD BD 5-10 SIGN CARD	5633.42
208858154	BLUEBERRY D.PT OR NF	5292.73
62594704	40 CAN ROLLER-RED	5212.57
173538177	BELL GREEN BD	4699.31
37512336	BELL GREEN JMB CHR	4620.6

3. Top 5 most profitable items

```
SELECT
    tempTbl.item_id,
    d.item_description,
    sum(tempTbl.sold_unit_quantity * applied_price * c.usd_exchange_rate) as
total_sales_dollars
FROM(
    SELECT
        a.item_id,
        a.sold_unit_quantity,
        (CASE WHEN a.sales_timestamp < b.price_chng_activation_ts THEN
prev_price_amt else curr_price_amt end) AS applied_price,
        a.currency_code,
        a.geo_region_cd,
        row_number() OVER (PARTITION BY b.item_id ORDER BY
b.price_chng_activation_ts) as rnk
    FROM
        edureka_788309_retailcart_sales_transaction_events_view a
        LEFT JOIN edureka_788309_retailcart_price_change_events b
ON(a.store_id = b.store_id AND a.item_id = b.item_id AND a.geo_region_cd =
b.geo_region_cd)
    WHERE
        a.sales_timestamp > b.price_chng_activation_ts
)tempTbl
INNER JOIN edureka_788309_retailcart_currency_details c
ON(tempTbl.currency_code = c.currency_code)
INNER JOIN edureka_788309_retailcart_item_details d ON(tempTbl.item_id =
d.item_id AND tempTbl.geo_region_cd = d.geo_region_cd)
WHERE
    rnk = 1
GROUP BY
    tempTbl.item_id,d.item_description
ORDER BY
    total_sales_dollars desc limit 5;
```

tempTbl.item_id	d.item_description	total_sales_dollars
61229355	0.50 AIR AQUA,8.6	18568.050592
57550380	SS CRSS W/PAINT RSE	11074.56
62594824	18G SPOILER	9736.09875
184473785	BG PEPLUM BODYSUIT	8809.68
61181042	PT CHOCOLATE CHIP 16	8649.99

4. Top 5 stores with most sales

```
SELECT
    tempTbl.store_id,
    d.store_name,
    SUM(sold_unit_quantity * applied_price * usd_exchange_rate) AS
total_sales_dollars
FROM
(
    SELECT
        a.store_id,
        a.currency_code,
        a.sold_unit_quantity,
        a.geo_region_cd,
        (CASE WHEN a.sales_timestamp < b.price_chng_activation_ts THEN
prev_price_amt ELSE curr_price_amt END) AS applied_price,
        row_number() OVER (PARTITION BY b.item_id ORDER BY
b.price_chng_activation_ts) as rnk
    FROM
        edureka_788309_retailcart_sales_transaction_events_view a
        INNER JOIN edureka_788309_retailcart_price_change_events_view b
ON(a.store_id=b.store_id AND a.item_id = b.item_id AND a.geo_region_cd =
b.geo_region_cd)
    WHERE
        a.sales_timestamp > b.price_chng_activation_ts
    )tempTbl
    INNER JOIN edureka_788309_retailcart_currency_details c
ON(tempTbl.currency_code = c.currency_code)
    INNER JOIN edureka_788309_retailcart_store_details d ON(tempTbl.store_id =
d.store_id AND tempTbl.geo_region_cd = d.geo_region_cd)
WHERE rnk = 1
GROUP BY
    tempTbl.store_id,d.store_name
ORDER BY
    total_sales_dollars DESC
LIMIT 5;
```

temptbl.store_id	d.store_name	total_sales_dollars
1877	PORTERVILLE, CA	26770.6137165
1782	VIENNA WV	25344.33913623
598	KEARNEY, NE	18990.81066834
131	MT PLEASANT TX	18823.34607546
3478	HONOLULU HI	18372.55933

5. Average discount on items in percentage

```
SELECT
    tempTbl1.item_id, avg(tempTbl1.discount) AS Average_Discount
FROM
(
    SELECT
        tempTbl.item_id,d.item_description,((tempTbl.prev_price_amt -
tempTbl.curr_price_amt)/tempTbl.prev_price_amt) * 100 as discount
    FROM (
        SELECT
        a.item_id,
            b.prev_price_amt,
            b.curr_price_amt,
            a.geo_region_cd,
            row_number() OVER (PARTITION BY b.item_id ORDER BY
b.price_chng_activation_ts) as rnk
        FROM
            edureka_788309_retailcart_sales_transaction_events_view a
            INNER JOIN edureka_788309_retailcart_price_change_events_view b
ON(a.store_id = b.store_id AND a.item_id = b.item_id AND a.geo_region_cd =
b.geo_region_cd)
        WHERE
            a.sales_timestamp > b.price_chng_activation_ts
    ) tempTbl
        INNER JOIN edureka_788309_retailcart_item_details d ON(tempTbl.item_id
= d.item_id and tempTbl.geo_region_cd = d.geo_region_cd)
    WHERE
        rnk = 1
)tempTbl1
GROUP BY
    tempTbl1.item_id
order by
    Average_Discount desc limit 5;
```

item_id	Average_Discount
52534366	66.4283644092358226
92635315	66.3554046406338427
136770038	66.3009728834609812
39291210	66.300251256281407
72492902	66.1341616744323645

6. Top 5 discounted Items

```
SELECT
    tempTbl1.item_id, max(tempTbl1.discount) AS Max_Discount
FROM
      (
          SELECT
               tempTbl.item_id,d.item_description,((tempTbl.prev_price_amt -
tempTbl.curr_price_amt)/tempTbl.prev_price_amt) * 100 as discount
          FROM (
            SELECT
                  a.item_id,
                  b.prev_price_amt,
                  b.curr_price_amt,
                  a.geo_region_cd,
                  row_number() OVER (PARTITION BY b.item_id ORDER BY
b.price_chng_activation_ts) as rnk
            FROM
                edureka_788309_retailcart_sales_transaction_events_view a
                INNER JOIN edureka_788309_retailcart_price_change_events_view
b ON(a.store_id = b.store_id AND a.item_id = b.item_id AND a.geo_region_cd =
b.geo_region_cd)
            WHERE
                a.sales_timestamp > b.price_chng_activation_ts
        ) tempTbl
            INNER JOIN edureka_788309_retailcart_item_details d
ON(tempTbl.item_id = d.item_id and tempTbl.geo_region_cd = d.geo_region_cd)
            WHERE rnk = 1
      )tempTbl1
GROUP BY tempTbl1.item_id
order by
    Max_Discount desc limit 5;
```

item_id	Average_Discount
52534366	66.4283644092358226
92635315	66.3554046406338427
136770038	66.3009728834609812
39291210	66.300251256281407
72492902	66.1341616744323645

8. Join facts and dimensions and load pivot table

```
CREATE TABLE IF NOT EXISTS edureka_788309_retail_pivot AS
    SELECT tempTbl.*,
            c.usd_exchange_rate,
            d.week_number,
            d.quarter_number,
            d.week_name,
            d.quarter_name,
            d.month_number,
            d.month_name,
            e.item_description
        FROM(
            SELECT
                DISTINCT TO_DATE(a.sales_timestamp) as sales_date,
                a.sold_unit_quantity,
                from_unixtime(UNIX_TIMESTAMP(sales_timestamp), 'HH:mm:ss') as
sales_time,
                a.store_id,
                b.item_id,
                a.currency_code,
                a.geo_region_cd,
                b.price_chng_activation_ts,
                b.prev_price_amt,
                b.curr_price_amt,
                row_number() OVER (PARTITION BY b.item_id ORDER BY
b.price_chng_activation_ts) as rnk
            FROM
                edureka_788309_retailcart_sales_transaction_events_view a
            INNER JOIN edureka_788309_retailcart_price_change_events_view b ON
(a.store_id=b.store_id AND a.item_id = b.item_id AND a.geo_region_cd =
b.geo_region_cd AND TO_DATE(a.sales_timestamp) = b.business_date)
        WHERE
            a.sales_timestamp > b.price_chng_activation_ts
    )tempTbl
    INNER JOIN edureka_788309_retailcart_currency_details c ON
(tempTbl.currency_code = c.currency_code)
    INNER JOIN edureka_788309_retailcart_calendar_details d
ON(tempTbl.sales_date=d.calendar_date)
    INNER JOIN edureka_788309_retailcart_item_details e ON(tempTbl.item_id =
e.item_id AND tempTbl.geo_region_cd = e.geo_region_cd)
WHERE
    rnk = 1;
```

9. On the real-time feeds, develop a real-time analysis framework that will create real-time KPIs and publish them to a dashboard. For this, you might have to join both real-time feeds and static MySQL Tables.

In this project I have mapped the Hbase table with the hive table so that as the events generated in the MySQL table they will directly push to the Hbase table via Kafka topic and thus to hive KPI's query. So that hive will give the real-time time KPI's.

Actionable insights

i) Total sales in a day/week/quarter (both in terms of sales revenue and units sold)

```
SELECT
  sales_date,
  week_name,
  quarter_name,
  sum(
    sold_unit_quantity * (
      case when cast(concat_ws(' ', sales_date, sales_time) as timestamp) <
price_chng_activation_ts then prev_price_amt else curr_price_amt end ) *
usd_exchange_rate
  ) as sales_revenue,
  sum(sold_unit_quantity) as units_sold
from
  edureka_788309_retail_pivot
group by
  sales_date,
  week_name,
  quarter_name
order by
  sales_date;
```

sales_date	week_name	quarter_name	sales_revenue	units_sold
2020-02-01	Week 01	Q1	3585244.7554977	109914.83
2020-02-02	Week 01	Q1	1232637.72259051	36883.29
2020-02-03	Week 01	Q1	763919.22205488	24598.24
2020-02-04	Week 01	Q1	701577.33983418	21406.22
2020-02-05	Week 01	Q1	627901.16140196	18855.76
2020-02-06	Week 01	Q1	687744.92436865	20738.02

2020-02-07	Week 01	Q1	836457.912676 68	25775.34

ii) Top 5 most selling item categories in a day (both in terms of amount and quantity)

```
SELECT
  sales_date,item_description, sales_revenue, row_num
FROM(
    SELECT
      sales_date,item_description,
      sum (sold_unit_quantity * (case when cast(concat_ws(' ', sales_date,
sales_time) as timestamp) < price_chng_activation_ts then prev_price_amt else
curr_price_amt end
        ) * usd_exchange_rate
      ) as sales_revenue,
      rank() over (partition by sales_date order by sum (sold_unit_quantity *
( case when cast(concat_ws(' ', sales_date, sales_time) as timestamp) <
price_chng_activation_ts then prev_price_amt else curr_price_amt end) *
usd_exchange_rate
          ) desc
      ) as row_num
    FROM
      edureka_788309_retail_pivot
    GROUP BY sales_date, item_description
  ) T
WHERE row_num <= 5;
```

sales_date	item_description	sales_revenue	row_num
2020-02-01	GE LS CREW TEE	5922.82076998	1
2020-02-01	W AW MESH TRAINER	4411.04423286	2
2020-02-01	WN SOLID CREW TEE	4327.27352451	3
2020-02-01	COLOR CLUB NL POLISH	3453.24792297	4
2020-02-01	TS SCOOP NECK TUNIC	3019.87158305	5
2020-02-02	KEMPSELECT 1% MILK	3346.21812	1
2020-02-02	GE LS CREW TEE	2271.0387738	2
2020-02-02	GE LS PLAID FLANNEL	2092.544286	3
….	….	...	...

```
SELECT
  sales_date, item_description,units_sold,row_num
FROM
  (
    SELECT
      sales_date,
      item_description,
      sum(sold_unit_quantity) as units_sold,
      rank() over (
        partition by sales_date
        order by
          sum(sold_unit_quantity) desc
      ) as row_num
    FROM
      edureka_788309_retail_pivot
    GROUP BY
      sales_date,
      item_description
  ) T
WHERE
  row_num <= 5;
```

sales_date	item_description	units_sold	row_num
2020-02-01	GE LS CREW TEE	179.92	1
2020-02-01	W AW MESH TRAINER	124.74	2
2020-02-01	WN SOLID CREW TEE	105.98	3
2020-02-01	OPP CORD RIBBON ASTM	100	4
2020-02-01	COLOR CLUB NL POLISH	99.15	5
2020-02-02	HB 1.1 MWO BLK	63	1
2020-02-02	KEMPSELECT 1% MILK	60	2
2020-02-02	W DS ASPIRE WW	57.08	3
2020-02-02	GE LS CREW TEE	56.19	4
...	...	...	...

iii) Top 5 most profitable items in a day/week/quarter

```
SELECT
  sales_date,
  week_name,
  quarter_name,
  item_description,
  diff,
  row_num
FROM
  (
    SELECT
      sales_date,
      week_name,
      quarter_name,
      item_description,
      sum(
        (curr_price_amt - prev_price_amt) * sold_unit_quantity *
usd_exchange_rate
      ) as diff,
      rank() over (
        partition by sales_date,
        week_name,
        quarter_name,
        item_description
        order by
          sum(
            (curr_price_amt - prev_price_amt) * sold_unit_quantity *
usd_exchange_rate
          ) desc
      ) as row_num
    FROM
      edureka_788309_retail_pivot
    GROUP BY
      sales_date,
      week_name,
      quarter_name,
      item_description
  ) T
WHERE
  row_num <= 5
  and diff > 0;
```

sales_date	week_name	quarter_name	item_description	diff	row_num
2020-02-01	Week 01	Q1	"COFFEE" 2PK TOWELS	7.84224	1
2020-02-01	Week 01	Q1	"E" CRY DISC CHM	10.03296	1
2020-02-01	Week 01	Q1	"FAMILY FUN" TOWELS	0.3563	1
2020-02-01	Week 01	Q1	"J" CRY DISC CHM	2.38512928	1
2020-02-01	Week 01	Q1	"K" CRY DISC CHM	6.125165	1
2020-02-01	Week 01	Q1	#10 AVENGERS	8.094708	1
2020-02-01	Week 01	Q1	#10 DISNEY FAIRIES	0.045227	1
2020-02-01	Week 01	Q1	#10 HANDY MANNY	15.72272	1
...	...	...			...

iv) Top 5 most profitable stores in a day/week/month/quarter

```
SELECT
  store_id,
  sales_date,
  week_name,
  quarter_name,
  sales_revenue,
  row_num
FROM
  (
    SELECT
      store_id,
      sales_date,
      week_name,
      quarter_name,
      sum (
        sold_unit_quantity * (
          case when cast(concat_ws(' ', sales_date, sales_time) as timestamp)
< price_chng_activation_ts then prev_price_amt else curr_price_amt end
        ) * usd_exchange_rate
      ) as sales_revenue,
      rank() over (
        partition by sales_date
        order by
          sum (
            sold_unit_quantity * (
              case when cast(concat_ws(' ', sales_date, sales_time) as
timestamp) < price_chng_activation_ts then prev_price_amt else curr_price_amt
end
            ) * usd_exchange_rate
          ) desc
      ) as row_num
    FROM
      edureka_788309_retail_pivot
    GROUP BY
      store_id,
      sales_date,
      week_name,
      quarter_name
  ) T
WHERE
  row_num <= 5;
```

store_id	sales_date	week_name	quarter_name	sales_revenue	row_num
1877	2020-02-01	Week 01	Q1	26803.7381895	1
3395	2020-02-01	Week 01	Q1	17622.36876708	2
3761	2020-02-01	Week 01	Q1	13923.50262736	3
1782	2020-02-01	Week 01	Q1	13768.54457774	4
1880	2020-02-01	Week 01	Q1	12391.91213095	5
3478	2020-02-02	Week 01	Q1	6277.543778	1
198	2020-02-02	Week 01	Q1	5861.60562522	2
221	2020-02-02	Week 01	Q1	5185.74791425	3
...	...	...	...	...	...

v) Total profit or loss in U.S. dollars every day in percentage

```
SELECT
  sales_date,
  sum(
    sold_unit_quantity * usd_exchange_rate * prev_price_amt) / sum
(sold_unit_quantity
      * ( case when cast(concat_ws(' ', sales_date, sales_time) as timestamp)
< price_chng_activation_ts then prev_price_amt else curr_price_amt end) *
usd_exchange_rate
  ) * 100 as proft_Loss
from
  edureka_788309_retail_pivot
group by
  sales_date;
```

sales_date	proft_Loss
2020-02-01	99.7233704128070549
2020-02-02	99.3370249234685464
2020-02-03	100.5321960714568759
2020-02-04	100.4334929581931506
2020-02-05	99.3988593827948576
2020-02-06	99.4568663606613929
2020-02-07	100.0315097465485842
2020-02-01	99.7233704128070549

Project Resources

Project Folder Path

```
/mnt/bigdatapgp/edureka_788309/edureka_788309_retailcart
```

Kafka Jars

```
/mnt/bigdatapgp/edureka_788309/edureka_788309_retailcart/KafkaToHbaseItemPrice
ChangeConsumer-1.0-SNAPSHOT-jar-with-dependencies.jar

/mnt/bigdatapgp/edureka_788309/edureka_788309_retailcart/RdbmsToKafka-1.0-SNAP
SHOT-jar-with-dependencies.jar

/mnt/bigdatapgp/edureka_788309/edureka_788309_retailcart/KafkaToHbaseSalesTran
sactionConsumer-1.0-SNAPSHOT-jar-with-dependencies.jar
```

MySQL, HiveQL and SQOOP Scripts
All MySQL, HiveQL and SQOOP script code are written in this file only.

GitHub
https://github.com/vikramraj-sahu/retail-cart.git

--------XXXXX------