Big Data Analytics Assignment

Name : Singam Adikesava Reddy
VTU NO : VTU25083
course code : 10211IT201
Subject : Big Data Analytics
Task No : (3)

27/11/25

5/5

Dimensions

dim time

- time_id (int)
- date (Date)

day-of-week, day-of-month, week-of-year

is_holiday BOOLEN

- fiscal_month, fiscal-quator, fiscal-year

dim_store

- Store_id (int)
- Store_code, name
- region, city, state, country
- store_type (flagship, outlet)
- Store_open_date, store_close_date.
- Square_footage, manager_id

dim_Product

- Product_id (int)
- SKU, UPC
- name, brand, category sub category
- size, color
- sale_sart_date, sale_end_date
- current_price, cost_price
- product_status (active | discountinued)

dim_customer

- customer_id (int)

customer_type (guest)

- first_name, last_name
- gender, birth_year
- loyality_tier, Join_date

```
from pyspark import spark context
from pyspark.streaming import streming context
from pyspark.streaming.katfa import kafkautik
import Json.
Sc= spark contex (app Name = "video")
ssc = streaming contex (sc, batch Duration=5)
ssc.check point ("/path/check point")
kvs = kafkautils.Create Direct stream (ssc, topics
["video-events"], kafkaparams = {"metal broker}
events = kvs.map (lambla msg: Json.lcds (msg[1])
Plays = events.filter (lambda e:e['event-type'] in
('play-start', 'play', 'play-end'))
Content-play-seconds = plays.map (lambda
e:(e('content_id'), e.get ('play-Seconds', 0))
batch-play-seconds = content_play-seconds.reduce
By key (lambda a,b: a+b)
def update_running_sum (new-vals, running_totd):
    return sum(new-vals) + (running_total or 0)
running_total.foreachRDD (Push_to_store).
    ssc.start()
    ssc.awaittermination().
```

2) OLAP data Model for XYZ Retail (Sales Analysis)

Support fast multidimensional analysis of sales across store, time, products, customers, channel and promotions (kpis: sales amount)

Recommended model

A star schema with one central fact table and several conformed dimension tables

Fact - Sales

surrogate key: Sales_id (int)

fk- time_id (INT)

fk -store_id (int)

fk- store_product_id (int)

fk - customer_id (int)

fk- channel_id (int)

fk- promo- id (int)

quantity_sold (int)

Sales- amount (DECIMAL(12,2))

Unit- price (DECIMAL(10,2))

discount_amount (DECIMAL(12,2))

returned - flag (Boolean)

transaction- type (string)

created_at (time stamp)

load- batch_id (Varchar)

Partition this fact time (Year | month)

Implementing Apache ignite for AlphaBank

Goals

(3) • Speed up transaction and analytical operations via in-memory data grid

• Provide - low latency caching for customer profiles account balances reference data and accelerate OLTP

• Use ignite for disturbed in memory SQL, key-value, compute grid learning

High-leve architecture

• Data tier : persistent store (RDBMs / postgre SQL)+ ignite cluster as in-memor layer with optional

• ignite cluster : multiple node across data centers (replication, partitioning)

• client : banking application server

• integration: banking application use ignite cache store for read-through / write-through RDBMS.

Implementation steps

1. select deployment mode

• option A : ignite as distuibte cache in front of RDBMs (recommended for minimal change)

• Optional B : ignite with native persistance to replace some DB workloads.

2. Define data model
accounts, customer, transactions - recent, Product - rates

3. configure Cache store
• implement cache store interface to sync with RDMS, enabling persistance backing.
• configure write behind for better through put (tunable).

4. set up clustering
• Data Partitioned cache Replicated

```
Slect c.customer_id, account_id, a.balance
from accounts a.
Join customers (ON a.customer_id = C.
customer id
where C.region = 'south';
```

Implement Apache Hive for Retail Groove

Objectives

(4) • Centralize customer & transition data for large-scale analytics marketing

• Provide SQL-like interface (Hive QL) over data lakes

customer table

```
customer_id BIGINT,
first_name string
last_name string
email-hash string
sign-up date Date,
loyalty-tier string
gender string
brith-year int,
stored AS Parquet
location '/data/retail/customer/';
```

Transactions table

```
Create External Table retail-transactions
transaction_id, string,
customer_id Bigint,
store_id int,
Product_SKU string
quantity int
unit_price Decimal ((10,2))
```

1. Monthly revenue trend for a Product category

```
Select t.year , t.month , sum ( f. sales- amount )
As revenue from fact - sales f.
Join dim- product P ON f fk- product -id =
P. Product-id
Join dim-time t ON t fk- time-id = t.time id
where P.category = 'Electronics' AND t.date >=
DATE ADD (month, -12, current- date)
Group By t.year, t.month
ORDER By t.year, t. month;
```

2. Top 10 skus by margin last quater

```
Select P.SKU, P.name, s.region, sum ( f.margin-
amount ) As total- margin
from fact- sales f.
Join dim- product P ON f.fk - product -id = P. Product -id
Join dim- store S ON ffk - store -id = s. store - id
Join dim-time t ON ffk - time -id = t.time -id.
where t.fi sical - quater = 3 AND t.fiscal - year
=2025
  Group By P.Suk, P. name, S.region
ORDER By total- margin DESC.
    Limit 10;
```

Product table

```
Create external table retail_products (
    sku string,
    name string, brand string, category string,
price Decimal (10,2), cost Decimal (10,2),
cost Decimal (10,2), active Boolean)
stored As Parrequet
location '/data/retail/products/";
```

Predict the future of Hadoop

Key directions

1. Move from HDFS - centric to object - store - first

(5)

- Hadoop ecosystems will continue to integrate with cloud object stores (S3, GCS) HDFS will remain for on-prem but the trend favors objects store connector.

2. Smaller foot print, more modular

- Monolithic Hadoop distributions will give way to modular components (Yarn alternative)

3. Convergence towards SQL

- Data formats (Parquet / ORC) table formats (Apache Iceberg, Delta Lake) stack.

4. Stream + batch Unification

- Real time processing engine (Flink, Kafka streams, Spark structed streaming retired.

5. cloud - native, containerized

- Kubernete orchestration for data workloads, auto-scaling, improved resource operator patterns for components like Hive, HDFS alternatives.

7. integration with ML/AI workflows
- Native support for model traning integration

8. performance & cost effeciency
- More compute separation from storage with caching layer (Alluxio, ignite), tiered storage strategies (hot in memory), warm SSD, cold object store).

9. serverless analytics
- More serverless SQL-on-lake offerings (Athena, BigQuery like) reduce operator overhead, but hadoop tools will adapt to expose serverless capabilites.