**Search & Navigation**

**Software Assignment**

**Task Definition**

You work in an **e-commerce** platform as a **software engineer** and your mission is to build **batch** **and** **real-time** **data pipelines** together with high-performance **rest APIs** to create a real-time recommendation engine. Your rest API will be the source of two recommendation lists on the main page:

1. **Browsing History**
   1. You should return the **last** **ten products** **viewed** by a given user and sorted by view date.
      1. **GET** endpoint, inputs: user-id
      2. Sample response:

|  |
| --- |
| {  "user-id": "fdsfsdfs",  "products": ["a","b","c","d"."e"],  "type" : "personalized"  } |

* 1. You should also add a functionality for users which they can delete a **product from their history**.
     1. **DELETE** endpoint, inputs: user-id, product-id

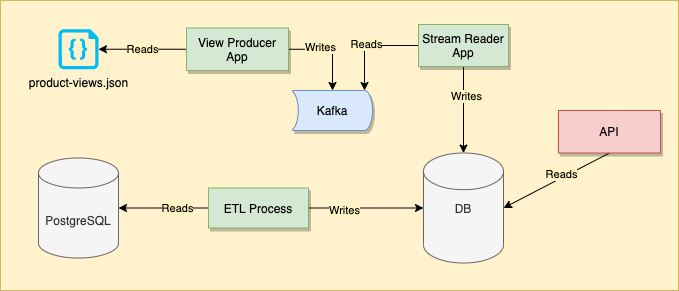
1. **Best Seller Products**
   1. You should understand the interest of a user using his/her browsing history items and recommend best seller products to him/her only from the categories of these items. Otherwise, you should return a general best seller product list without any filter.
      1. **GET** endpoint, inputs: user-id
      2. Sample response

|  |
| --- |
| {  "user-id": "fdsfsdfs",  "products": ["a","b","c","d"."e"],  "type" : "non-personalized"  } |

* + 1. **First Strategy Details (Personalized)**
       1. Use the **browsing history** of the user.
       2. Extract **at most three categories** from this history
       3. Offer **ten products** based on these **categories**.
       4. Bestsellers of a category means **top ten products of this category bought** (last month) by the most **distinct** users
    2. **Second Strategy Details (Non-Personalized)**
       1. If a user has **no browsing history** you should return a list as a **second strategy**
       2. It gives the top **ten products bought** (last month) by the most **distinct** users **without any filter.**

**Notes:**

* For any recommendation list, the **minimum** number of products to return is **five**. If there are fewer products than this number, API should return an **empty list.**
* To satisfy the requirements of the task you should have **product views** file data source and a database tables containing **product**, **order**(containing orders of the last month) and **order item** information. The definition of files and tables are in **technical details**.
* You should **create**
  + a clickstream product view producer application as described in technical details. (View Producer App)
  + a streaming app to read views from Kafka and integrate with API (Stream Reader App)
  + a batch category based and general best sellers flow and integrate with API (ETL Process)
  + a recommendation Rest API to provide given endpoints (API)



For write operations, you can use the PostgreSQL we provided to you or you can select a database of your preference that you think it will be useful for this case. You are free to edit **docker-compose.yml** file we provided to setup a custom database. In any case, don’t forget to send us custom database related operations(table/index/schema/mapping/etc. creation) and explanations.

**Technical Details**

**Environment**

**docker-compose.yml** file is provided for you. You do not have to use this environment but it simplifies the tool setup process of your environment. If you choose to use it, you need to install Docker and Docker Compose into your computer. Following commands will setup a **Kafka**, **PostgreSQL** and Zookeeper(which is required for Kafka to run. You won’t need to use this).

To setup databases, you should run following command:

docker-compose -f ./docker-compose.yml up -d

To tear down, you should run following command:

docker-compose -f ./docker-compose.yml down

**Important Note:** If you tear down docker containers after ingesting data, you will lose the data you ingested and you have to ingest data again.

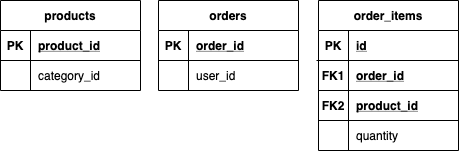
After setup is completed;

* You can access Kafka from port **9092** on your **localhost**
* You can access PostgreSQL from port **5432** on your **localhost**. Credentials are:
* **Username:** postgres
* **Password:** 123456

**Data**

There are two data sources:

1. The **PostgreSQL** we provided to you, which contains **data-db** databasewith following tables:



1. The **product-views.json** we provided to you**.** The **View Producer App** will read this file and publish **one event in a second** to Kafka. The current **timestamp** should be added to product view event data. An example of a product view event is given below. See the file for details.

|  |
| --- |
| {  "**event**": "ProductView",  "**messageid**": "c002a71a-9750-4604-8d70-d5ff3f1c4495",  "**userid**": "user-120",  "**properties**": {  "**productid**": "product-393"  },  "**context**": {  "**source**": "mobile-app"  }  } |

Fields descriptions:

* **event** is the name of the event
* **messageid** is a unique guid id for the event
* **userid** is the member who viewed product
* **properties.productid** is the viewed product
* **context.source** is the platform of the event source

**Expected Outcomes**

* Documentation of your solution which contains the schema/block diagram of your solution and any other details.
* Codes of any application (scripts, ETLs, streaming app, rest API, tests) developed for the given task.
* Especially for API application, unit testing must be satisfied

**Notes**

* You are free to choose any programming/scripting languages.
* You are free to choose and install any other tools/DBS into your environment. Please give details if used.
* Following criteria will be considered during the evaluation of the assignment:
  + Satisfaction of requirements
  + Architectural Decisions & Tool Choices
  + Code Quality & Readability
  + Testing, packaging and deployment. (Containerization)
* Package your codes and documents and send them to us.

You have 7 days to fulfill the assignment. Please, do not hesitate to contact us in case of any questions or suggestions.

Good luck :)