1. Real-time data Streaming Platform

One of the most significant differences between kafka and kinesis is the patform/ environment. Kafka is an open-source event streaming platform which requires full management in order to deploy on-premises. This requires the design of a cluster and setup of machines within to manage scaling/availability & all services desired. Amazon Kinesis on the other hand, is a fully managed and is designed to scale automatically (within the limitations set by the user) reducing the need for infrastructure engineers.

The next layer of comparison involves features/capabilities. Kinesis was built off the framework of kafka, then developed into an elastic, secure service easily integrable into the AWS architecture. Because Kafka introduces the ability to stream transactions in real time, this is carried through into kinesis. While kafka can be used in ETL pipelines, it only acts as the messaging service, with the ability only to classify events by producers which then determines the flow. For Kinesis to use this same functionality, Streams must be integrated with sharding which also causes latencies.

Kinesis Data Analytics adds functions to transform, aggregate, and filter streaming data. Although kafka is stateful, it provides no built-in analytic capabilities. Kinesis' integration also introduces streaming queries which easily use the state of transactions for analytics such as windows functions. Kafka has made recent improvements to the Kafka Streams DSL improving features/performance, but still requires much more customization than Kinesis. While this extra work is more difficult, it has always been more customizable, while kinesis only extended retention options a few months ago while this may be much more expensive.

2. Computing Infrastructure

- What processor does your personal development computer have?

Intel(R) Core(TM) i7-10875H CPU @ 2.30GHz   2.30 GHz

- Does your computer have a GPU? If yes, what type?

NVIDIA GeForce RTX 2060

- How much memory does your computer have?

2*16GB 3200MHz (dual channel)

- Have you used Google Colab (Links to an external site.) before? If yes, in what capacity?

Yes, various shareable data science projects. For a while it was my go-to for autosklearn; now I use docker

3. Tools, Programming Languages and Packages

- Python

I've ben using professionally for ~3.5yrs. not an expert because I go weeks without

- Java

no

- Kafka

no

- Google TensorFlow

Yes, v1.X tried for exposure to ML with CNNs & some others. Most recent was refactoring some code ~1yr ago from 1.12 to 1.14

- Facebook Prophet

no

- Surprise

no

- Statmodel

no

- Anaconda

Yes, this is my go-to python env/package handler. (cmd line) when I get a laptop (work or personal) my setup includes downloading miniconda & loading my .condarc default

- Neo4j and GDS

Used Neo4j for several projects; never enterprise/production size. GDS some but not for a professional deliverable

4. Mock Data Generator

Comment on the real-time transaction fields/attributes that you think must be part of the mock data generated.

I understand this task to be relative to the transactional purchase data from the store. There are many other layers of data to be collected (user browsing [within session on site & lead-in], history, cart data, network stats) but this question I believe focuses on the place to start – the actual purchases. These can be normalized out into a star/snowflake schema for storing purchase dimensions. See `Project_Phase_1_Stevens-mock_data.csv`.

'trans_id'	ID for each transaction, may repeat across lines due to multiple products in single purchase. No need for sub_trans_id for each item (that I can think of yet….cart history should be separate [in what order were items placed in cart & when])
'trans_datetime'	Datetime of purchase
‘product_id’	FK to product table
'user_id'	FK to user table which stores user metadata. Browsing, purchase, & location history will be joined for analysis
'store_id'	If this onmart has online & B&M, each location should have an ID for its metadata
'trans_long'	Longitudinal location of purchase. Important for analyzing where online purchases are made
'trans_lat'	Latitude; see long

5. Analytics

Much of these analytics are ripe for use of statistical/machine learning tools. These are often easiest to implement with python packages due to the large open-source user community. Recommender engines are the class of tools used to perform this task. Collaborative filtering is one of the most common. k-NN is one of the algorithms which can be used for product recommendations and cross-promotions. Simple regression is the easiest place to start for forecasting sales, product demands, and returns/zip. Network science is a likely path to pursue for centrality in social networks (for influential customers). Anomaly detectors can be used for fraud, with NLP providing some of the features on ratings (other features like generation rate/location/profile similarity should be factored into the model). Inventory management & contingency planning may be more easily analyzed using the R built-in tools. Trends, seasonality & other time-series features can be recognized by the forecast package autoregressive/arima models.