**eCommerce behavior data from a multi-category store**

* **Description**: This is a massive, real-world dataset containing user behavior events from a large online retailer. Crucially, it includes not just purchases, but also product views and cart interactions. This allows you to model the entire user funnel.
* **Why It's Better**:
  + **Massive Scale**: The October 2019 file alone has **over 42 million records**. The November file has over 67 million. This scale justifies the use of Spark, Kafka, and a distributed data warehouse.
  + **Rich Behavioral Features**: It contains the critical event\_type feature (view, add\_to\_cart, purchase), which is missing from the previous dataset. It also includes product brands and nested categories.
* **Key Fields**:
  + event\_time: Timestamp of the event.
  + event\_type: Can be view, cart, or purchase.
  + product\_id: Unique identifier for the product.
  + category\_id: Unique identifier for the product category.
  + category\_code: Nested category string (e.g., electronics.smartphone).
  + brand: Brand name of the product.
  + price: Price of the product.
  + user\_id: Persistent ID for the customer.
  + user\_session: ID for a specific Browse session.
* **Link**: [Kaggle - eCommerce behavior data](https://www.google.com/search?q=https://www.kaggle.com/datasets/mkechin/ecommerce-behavior-data-from-multi-category-store&authuser=2)

**The Project: Real-Time Customer Journey & Personalization Engine**

This project uses the richer dataset to build a platform that not only recommends products but also understands user intent in real-time to personalize the entire web experience (e.g., dynamically sorting category pages or offering timely promotions).

**Phase 1: Ingestion of High-Volume Event Streams**

A Python script reads from one of the massive monthly CSV files and streams events into a **Kafka** topic named user\_events to simulate a high-throughput, real-time firehose from a production website. A **Spark Structured Streaming** job consumes this stream, performs initial validation, and writes the raw events into a partitioned "raw" zone in a data lake (AWS S3, Azure Data Lake Storage) using an efficient format like Parquet.

* **Skills Used**: **Kafka**, **Spark**, **AWS**.

**Phase 2: Sessionization & Customer Feature Engineering (ETL/ELT)**

This is the core engineering challenge. An **Airflow** DAG triggers a daily **Spark** batch job that processes all the raw data from the previous day.

1. **Sessionization**: The job groups all events by user\_session. Within each session, it reconstructs the customer's journey, calculating metrics like time\_spent\_on\_product, session\_duration, and identifying abandoned carts.
2. **Feature Engineering**: It performs complex transformations. A key task is parsing the nested category\_code (e.g., 'electronics.smartphone') into separate columns (category\_level\_1, category\_level\_2) for easier querying.
3. **Customer 360 Profile**: The job updates a central "Customer 360" table in the data warehouse. This table aggregates data to create a rich profile for each user\_id, including their brand affinities, most viewed categories, average purchase value, and typical buying journey.

The transformed and aggregated data is loaded into a cloud data warehouse like **Snowflake** or **Databricks** (using Delta Tables).

* **Infra Used**: **Airflow**, **Spark**, **Snowflake**.

**Phase 3: Real-Time Personalization Serving**

The goal is to serve insights with low latency.

1. **Machine Learning Models**: Offline models are trained on the **Snowflake** data to predict churn risk, next likely purchase category, and generate product recommendations.
2. **Low-Latency Serving**: The generated user profiles and product recommendations are exported to a low-latency **NoSQL** database (like DynamoDB or Cassandra). This database is designed to be queried in milliseconds by the e-commerce website to personalize the user experience in real-time.
3. **Operational Dashboards**: Key metrics are served to an operational database like **PostgreSQL** for business analysts to monitor customer behavior trends.

<!-- end list -->

* **Infra Used**: **NoSQL**, **PostgreSQL**, **Data warehousing**.

**Phase 4: Infrastructure & MLOps**

The entire platform is managed as a professional software project.

1. **Infrastructure as Code**: **Terraform** is used to define and provision the entire cloud stack: the Kubernetes cluster, the Kafka service, the Spark environment on **Databricks**, and the **Snowflake** warehouse.
2. **Containerization**: All applications (the streaming jobs, the Airflow workers, the API serving the recommendations) are packaged into **Docker** containers.
3. **Deployment & Orchestration**: **Kubernetes** is used to deploy and manage the containerized applications, ensuring they are scalable and resilient. A **CI/CD** pipeline managed in **Git** automates the building, testing, and deployment of any code changes.

<!-- end list -->

* **Infra Used**: **Terraform**, **Docker**, **Kubernetes**, **Git**, **Agile** (methodology for managing the project).