Real-Time Spark + Scala ETL Project Ideas (with Kaggle Datasets)

---

1. Retail Sales Analytics (ETL + Aggregation + Reporting)

Dataset: Online Retail Dataset

https://www.kaggle.com/datasets/vijayuv/onlineretail

ETL Pipeline Tasks:

- **Extract**: Load CSV/Excel sales data from Kaggle into Spark DataFrame.
- **Transform**:
  - Clean missing values (e.g., customer IDs, product descriptions).
  - Convert transaction dates to Spark `TimestampType`.
  - Enrich with calculated fields: revenue = `Quantity * UnitPrice`.
  - Use **window functions** to calculate customer's rolling purchase behavior.
- **Load**: Store cleaned/aggregated data in **Parquet** or **Postgres**.

Real-Time Flavor:

- Simulate streaming by ingesting transactions from a **Kafka topic** and processing with Spark Structured Streaming.

Business Outcome:

- Daily sales trends per country.
- Top 10 most loyal customers.
- Revenue contribution by category.

---

2. Movie Recommendation Pipeline (ETL + ML Integration)

Dataset: MovieLens 20M Dataset

https://www.kaggle.com/datasets/grouplens/movielens-20m-dataset

ETL Pipeline Tasks:

- **Extract**: Load `ratings.csv`, `movies.csv`.
- **Transform**:
  - Clean movie titles (remove year info).
  - Parse genres into array column.
  - Join `ratings` with `movies`.
  - Create user-movie rating matrix.
- **Load**: Save transformed dataset into **Hive table** or **Delta Lake**.

Real-Time Flavor:

- Stream in **new ratings** via Kafka → update recommendation scores using Spark MLlib ALS model.

Business Outcome:

- Recommend top-N movies for each user.
- Trending movies dashboard (based on recent ratings).

---

3. Stock Market ETL & Real-Time Analytics

Dataset: Stock Prices Dataset

https://www.kaggle.com/datasets/borismarjanovic/price-volume-data-for-all-us-stocks-etfs

ETL Pipeline Tasks:

- **Extract**: Load stock CSV files.
- **Transform**:
  - Handle missing values in stock prices.
  - Calculate moving averages (5-day, 20-day).
  - Compute volatility per stock using window functions.
- **Load**: Write results into **HDFS Parquet** and push aggregated insights to a **NoSQL DB** (e.g., Cassandra).

Real-Time Flavor:

- Stream stock tick data (simulate with Kafka producer).
- Spark Structured Streaming updates moving averages in real-time.

Business Outcome:

- Detect stock anomalies.
- Provide a dashboard with live moving averages.

---

4. Airline Delay Prediction ETL Pipeline

Dataset: US Flight Delay Dataset

https://www.kaggle.com/datasets/shubhendra/airline-dataset

ETL Pipeline Tasks:

- **Extract**: Load flight delays + weather data.
- **Transform**:
  - Join flights with weather conditions.
  - Feature engineering: day-of-week, seasonality, holiday flag.
  - Encode categorical variables (airline, origin airport).
- **Load**: Store cleaned dataset for ML training.

Real-Time Flavor:

- Streaming flight data → Predict **delay probability** in real-time using pre-trained MLlib model.

Business Outcome:

- Airport operations dashboard predicting delays.

---

5. COVID-19 Data ETL + Analytics

Dataset: COVID-19 Dataset (Johns Hopkins)

https://www.kaggle.com/datasets/sudalairajkumar/novel-corona-virus-2019-dataset

ETL Pipeline Tasks:

- **Extract**: Load daily COVID case reports.
- **Transform**:
    - Handle missing region/country values.
    - Aggregate by country, continent, rolling 7-day averages.
    - Use **Spark SQL** for time-series transformations.
- **Load**: Store into **Hive tables** for BI dashboards.

Real-Time Flavor:

- Simulate streaming daily reports → Spark updates global COVID dashboards in near real-time.

Business Outcome:

- Country-level COVID trends.
- Forecasting spread using Spark MLlib time series models.

Tech Stack for All Projects

- **Spark 3.x + Scala 2.12**
- **Kafka** (for streaming simulation)
- **HDFS / Hive / Delta Lake** (for storage)
- **Parquet/ORC** (optimized formats)
- **Postgres / Cassandra** (optional sink)
- **Spark MLlib** (if ML is needed)

```
spark-scala-etl-project/
├── data/              # raw Kaggle dataset
├── src/main/scala/    # Scala ETL code
│   ├── Extract.scala
│   ├── Transform.scala
│   ├── Load.scala
├── conf/              # log4j, Hadoop, Spark configs
├── output/            # cleaned parquet/csv files
```