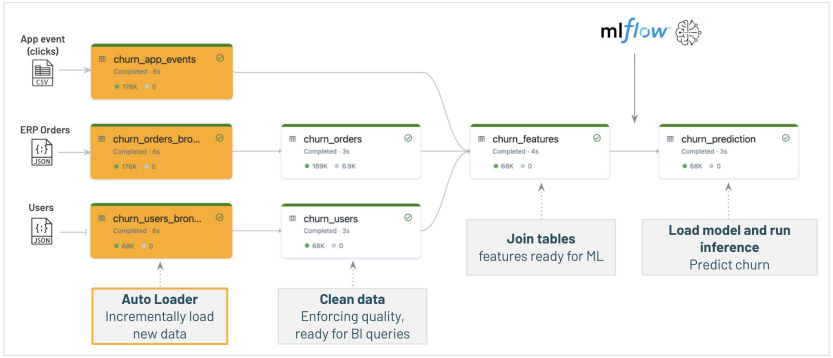
databricks 01.2 - Delta Live Tables - SQL

(https://databricks.com)

1/ Loading our data using Databricks Autoloader (cloud_files)

Autoloader allow us to efficiently ingest millions of files from a cloud storage, and support efficient schema inference and evolution at scale.

Let's use it to our pipeline and ingest the raw JSON & CSV data being delivered in our blob cloud storage.



Ingest raw app events stream in incremental mode

```
CREATE STREAMING LIVE TABLE churn_app_events (
  CONSTRAINT correct_schema EXPECT (_rescued_data IS NULL)
)
COMMENT "Application events and sessions"
AS SELECT * FROM cloud_files("/cloud_lakehouse_labs/retail/raw/events", "csv", map("cloudFiles.inferColumnTypes", "true"))
```

Ingest raw orders from ERP

```
CREATE STREAMING LIVE TABLE churn_orders_bronze (
  CONSTRAINT orders_correct_schema EXPECT (_rescued_data IS NULL)
)
COMMENT "Spending score from raw data"
AS SELECT * FROM cloud_files("/cloud_lakehouse_labs/retail/raw/orders", "json", map("cloudFiles.inferColumnTypes", "true"))
```

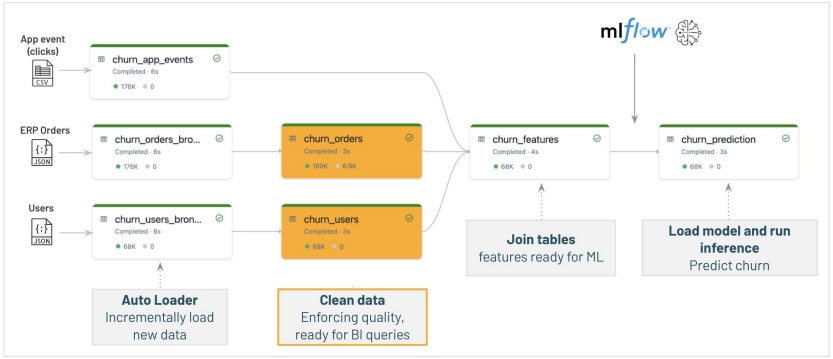
Ingest raw user data

```
CREATE STREAMING LIVE TABLE churn_users_bronze (
  CONSTRAINT correct_schema EXPECT (_rescued_data IS NULL)
)
COMMENT "raw user data coming from json files ingested in incremental with Auto Loader to support schema inference and evolution"
AS SELECT * FROM cloud_files("/cloud_lakehouse_labs/retail/raw/users", "json", map("cloudFiles.inferColumnTypes", "true"))
```

2/ Enforce quality and materialize our tables for Data Analysts

The next layer often call silver is consuming **incremental** data from the bronze one, and cleaning up some information.

We're also adding an expectation (https://docs.databricks.com/workflows/delta-live-tables/delta-live-tables-expectations.html) on different field to enforce and track our Data Quality. This will ensure that our dashboard are relevant and easily spot potential errors due to data anomaly.



Clean and anonymise User data

```
CREATE STREAMING LIVE TABLE churn_users (
  CONSTRAINT user_valid_id EXPECT (user_id IS NOT NULL) ON VIOLATION DROP ROW
)
TBLPROPERTIES (pipelines.autoOptimize.zOrderCols = "id")
COMMENT "User data cleaned and anonymized for analysis."
AS SELECT
  id as user_id,
  sha1(email) as email,
  to_timestamp(creation_date, "MM-dd-yyyy HH:mm:ss") as creation_date,
  to_timestamp(last_activity_date, "MM-dd-yyyy HH:mm:ss") as last_activity_date,
  initcap(firstname) as firstname,
  initcap(lastname) as lastname,
  address,
  channel,
  country,
  cast(gender as int),
  cast(age_group as int),
  cast(churn as int) as churn
from STREAM(live.churn_users_bronze)
```

Clean orders

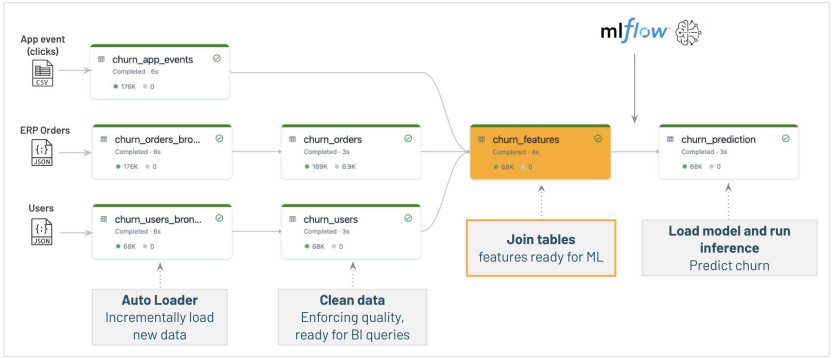
```
CREATE STREAMING LIVE TABLE churn_orders (
  CONSTRAINT order_valid_id EXPECT (order_id IS NOT NULL) ON VIOLATION DROP ROW,
  CONSTRAINT order_valid_user_id EXPECT (user_id IS NOT NULL) ON VIOLATION DROP ROW
)
COMMENT "Order data cleaned and anonymized for analysis."
AS SELECT
  cast(amount as int),
  id as order_id,
  user_id,
  cast(item_count as int),
  to_timestamp(transaction_date, "MM-dd-yyyy HH:mm:ss") as creation_date

from STREAM(live.churn_orders_bronze)
```

3/ Aggregate and join data to create our ML features

We're now ready to create the features required for our Churn prediction.

We need to enrich our user dataset with extra information which our model will use to help predicting churn, sucj as:

- last command date
- number of item bought
- number of actions in our website
- device used (ios/iphone)
- ...



Create the feature table