The Snowflake Schema Data Model comes to Feature Stores

Javier de la Rúa Martínez
Research Engineer
Hopsworks
javier@hopsworks.ai

FEATURE STORE SUMMIT 2024
DATA FOR AI:
REAL-TIME, BATCH, AND LLMS

Organized by HOPSWORKS

Recap! 💡 Feature Stores extend the Lakehouse

The **Offline store** is an open Lakehouse

- Historical data and contextual information
- Time-series data
- Support for complex analytical queries. e.g., temporal queries

Lakehouse

AI Lakehouse

The **Online store** enables real-time and LLM-RAG inference pipelines

- Low-latency feature vector retrieval
- Real-time transformation functions

The **Vector index** can improve LLM-RAG inference pipelines

- Similarity search on vector embeddings

Recap! 💡 Feature Stores extend the Lakehouse

The **Offline store** is an open Lakehouse

- Historical data and contextual information *Columnar Store*
- Time-series data
- Support for complex analytical queries. e.g., temporal queries

Lakehouse

AI Lakehouse

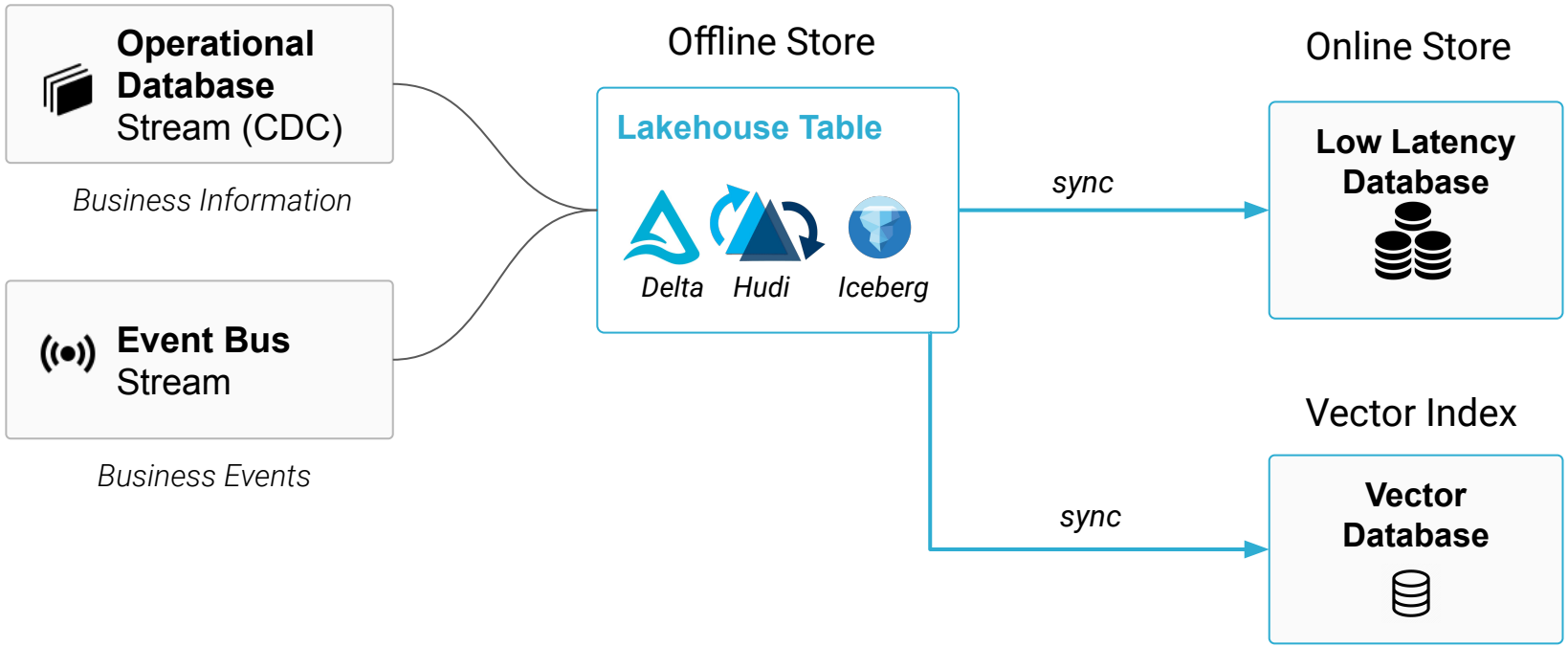
The **Online store** enables real-time and LLM-RAG inference pipelines

- Low-latency feature vector retrieval
- Real-time transformation functions *Row-oriented Store*

The **Vector index** can improve LLM-RAG inference pipelines

- Similarity search on vector embeddings *Vector database*

Common Non Real-Time AI Lakehouse



Common Non Real-Time AI Lakehouse

3NF

Operational Database Stream (CDC)

Business Information

Dimensions

Event Bus Stream

Business Events

Facts

Offline Store

Lakehouse Table

Delta *Hudi* *Iceberg*

Dimensional Data Model

sync

Online Store

Low Latency Database

Star Schema

- Duplicated columns are allowed among dimension tables
- Redundancy is allowed for query optimization
- No need for Nested Left Joins (query engines not optimized for joins)

Common Non Real-Time AI Lakehouse

3NF

Operational Database Stream (CDC)

Business Information

Dimensions

Event Bus Stream

Business Events

Facts

Offline Store

Lakehouse Table

Delta *Hudi* *Iceberg*

Dimensional Data Model

Feature Groups
Shared Schema

sync

Online Store

Low Latency Database

Star Schema

- Duplicated columns are allowed among dimension tables
- Redundancy is allowed for query optimization
- No need for Nested Left Joins (query engines not optimized for joins)

Common Non Real-Time AI Lakehouse

3NF

Operational Database
Stream (CDC)

Business Information
Dimensions

Event Bus
Stream

Business Events
Facts

Offline Store

Lakehouse Table

Delta *Hudi* *Iceberg*

Dimensional Data Model

Feature Groups
Shared Schema
sync

Online Store

Low Latency Database

Key-Value Store
- Performance for PK lookup
- It can accommodate the Star schema
- Scalability

Star Schema
- Duplicated columns are allowed among dimension tables
- Redundancy is allowed for query optimization
- No need for Nested Left Joins (query engines not optimized for joins)

Common Non Real-Time AI Lakehouse

3NF ?

Key-Value Store
- Limited (if any) support for joins
- No Relational Data Model

3NF

Operational Database
Stream (CDC)

Business Information
Dimensions

Event Bus
Stream

Business Events
Facts

Offline Store

Lakehouse Table

Delta *Hudi* *Iceberg*

Dimensional Data Model

Feature Groups
Shared Schema
sync

Online Store

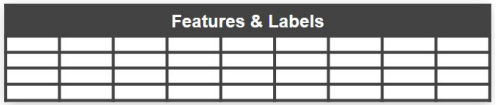
Low Latency Database

Star Schema
- Duplicated columns are allowed among dimension tables
- Redundancy is allowed for query optimization
- No need for Nested Left Joins (query engines not optimized for joins)

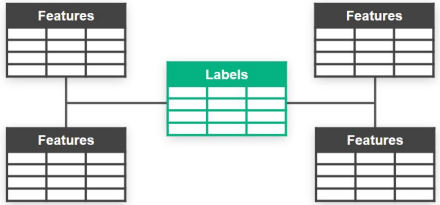
What’s the best Data Model for a Feature Store?

Offline Tables

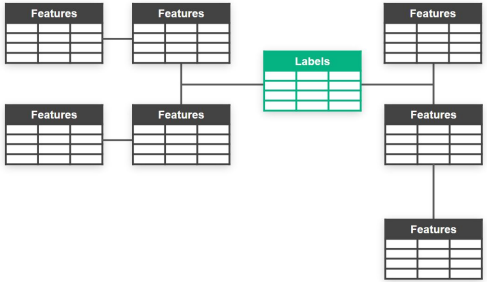
OBT schema



Star schema

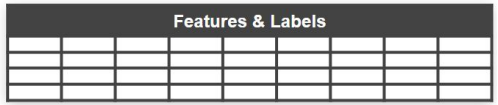


Snowflake schema

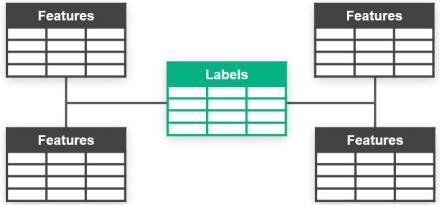


Online Tables

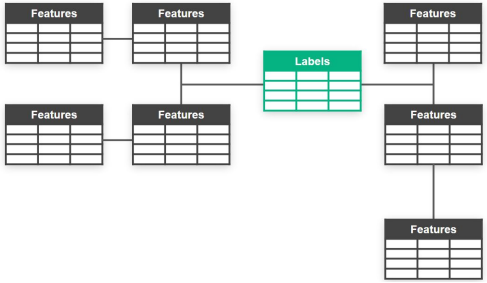
OBT schema



Star schema



Snowflake schema



Example 💡 Data Modeling for Credit Card Fraud Detection

Business information:

- Credit card transactions
- Fraudulent transactions
- Aggregates computed on recent transaction activity
- Bank accounts
- Bank details
- Merchant details

Example 💡 Data Modeling for Credit Card Fraud Detection

Business information:

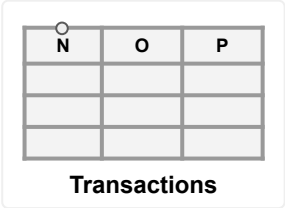
- Credit card transactions
- Fraudulent transactions 👈
- Aggregates computed on recent transactions
- Bank accounts
- Bank details
- Merchant details

1. **Prediction Problem**: Whether a credit card transaction is fraud or not → Label: fraud

Example 💡 Data Modeling for Credit Card Fraud Detection

Business information:

- Credit card transactions
- Fraudulent transactions 👈
- Aggregates computed on recent transactions
- Bank accounts
- Bank details
- Merchant details

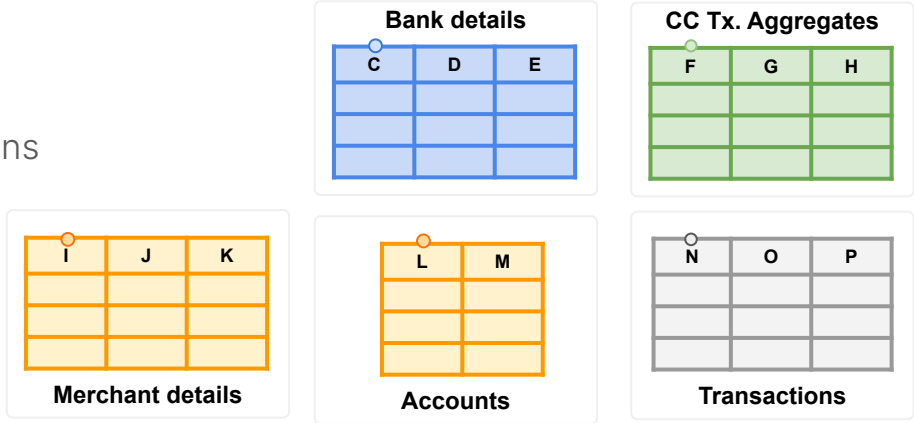


1. **Prediction Problem**: Whether a credit card transaction is fraud or not → Label: fraud
2. **Label Feature Group**: Fact table → Credit card transactions

Example 💡 Data Modeling for Credit Card Fraud Detection

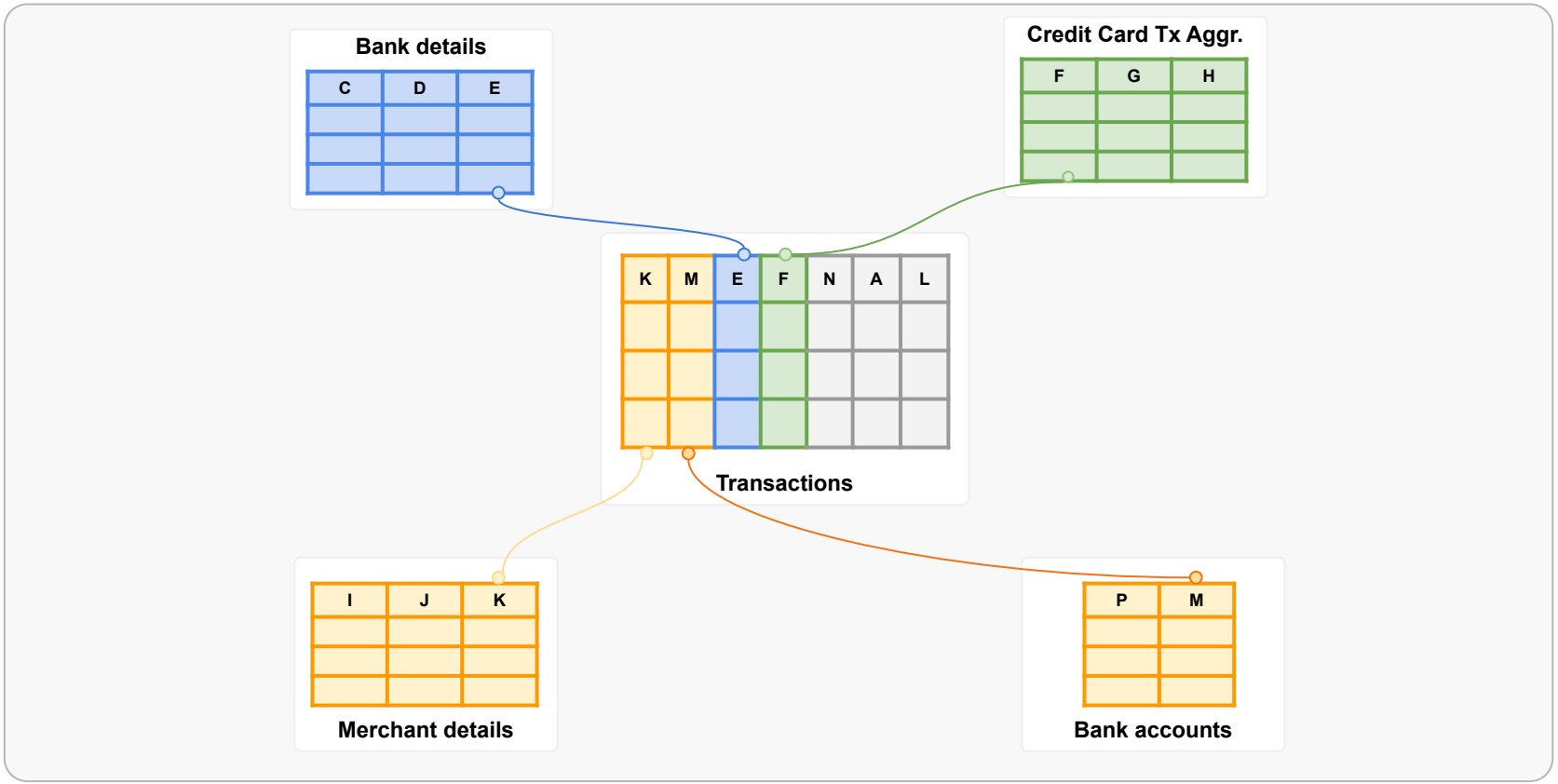
Business information:

- Credit card transactions
- Fraudulent transactions
- Aggregates computed on recent transactions
- Bank accounts
- Bank details
- Merchant details

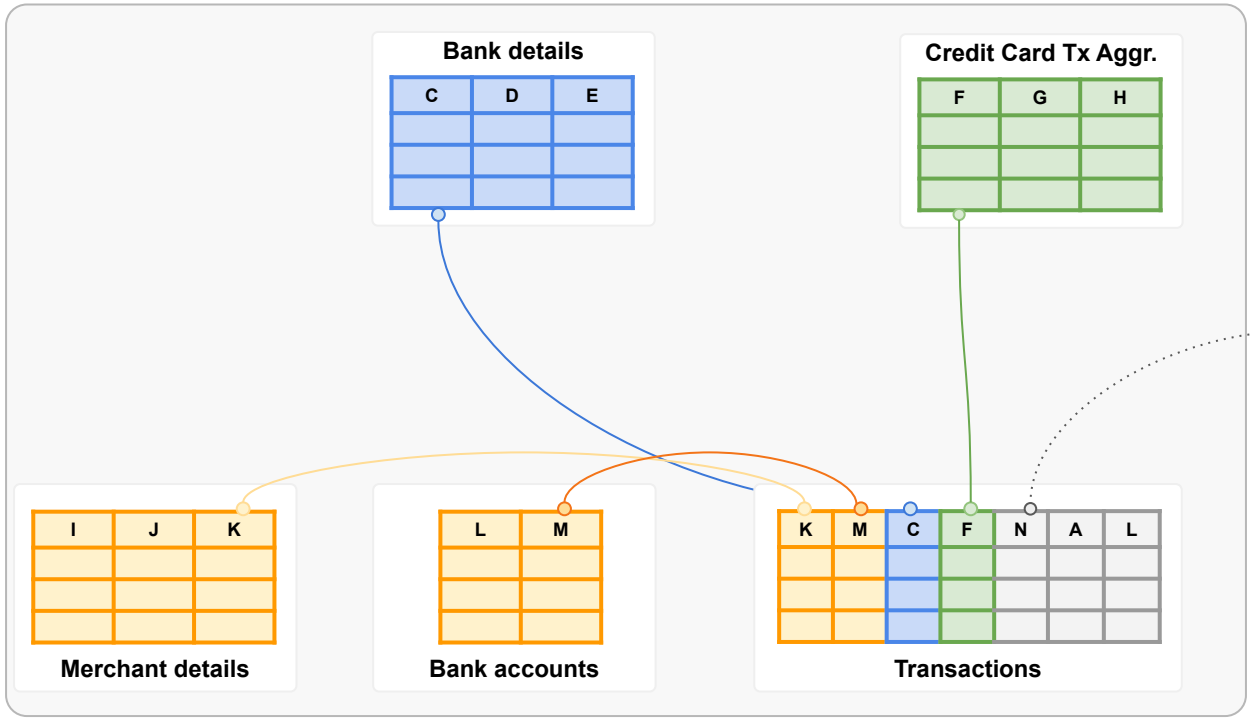


1. **Prediction Problem**: Whether a credit card transaction is fraud or not → Label: fraud
2. **Label Feature Group**: Fact table → Credit card transactions
3. **Normalized Feature Groups**: Dimensions tables → Bank details, Merchant details...

Star Schema Data Model

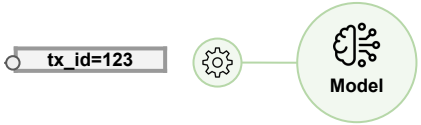


Star Schema Data Model in Online Inference Pipelines



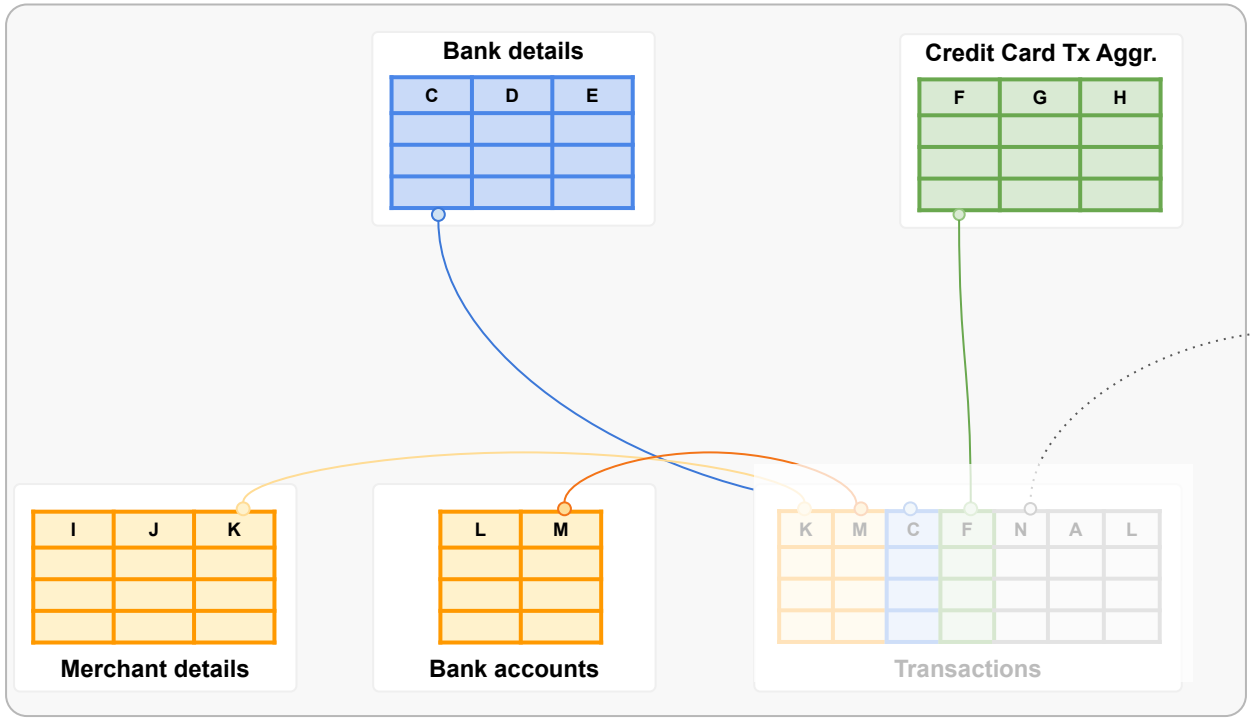
A new transaction is being processed, is it fraudulent?

AI-Enabled Application



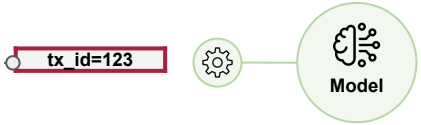
The model needs the bank details, merchant details, recent activity of the credit card, ...

Star Schema Data Model in Online Inference Pipelines



A new transaction is being processed, is it fraudulent?

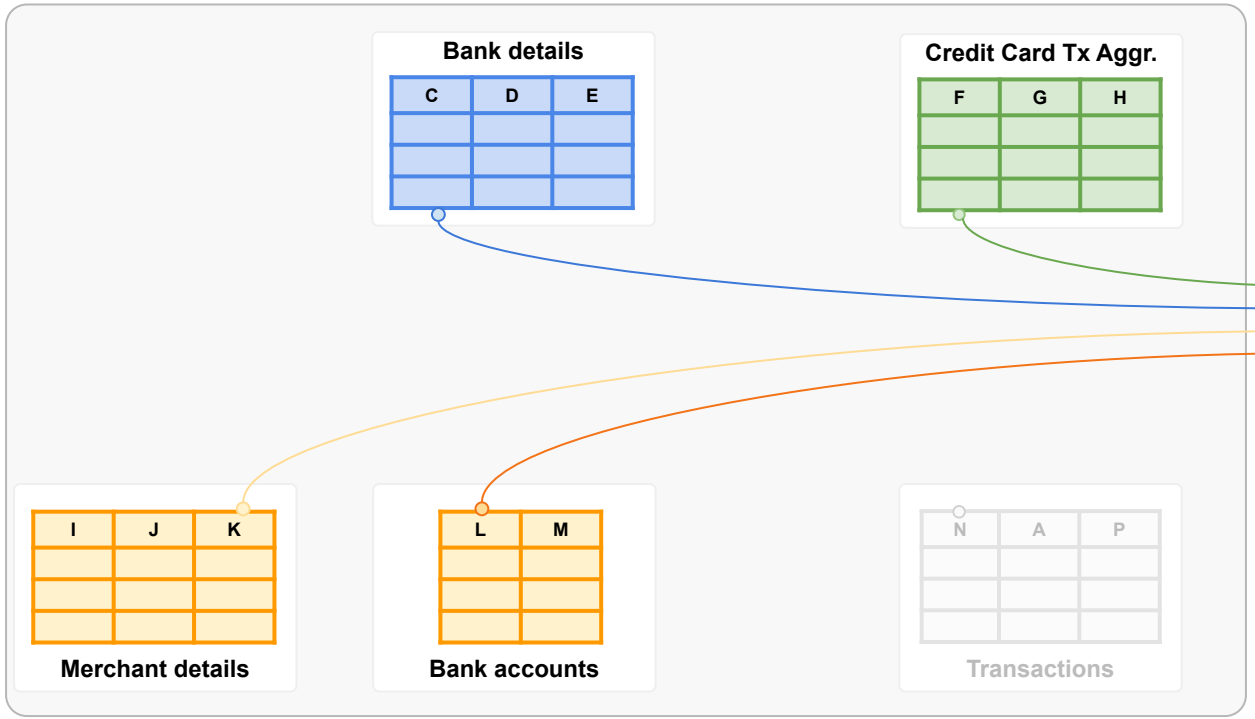
AI-Enabled Application



The model needs the bank details, merchant details, recent activity of the credit card, ...

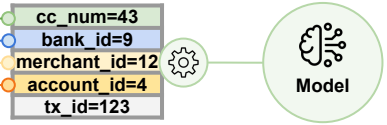
The transaction has not happened yet! it's not available at prediction time

Star Schema Data Model in Online Inference Pipelines



A new transaction is being processed, is it fraudulent?

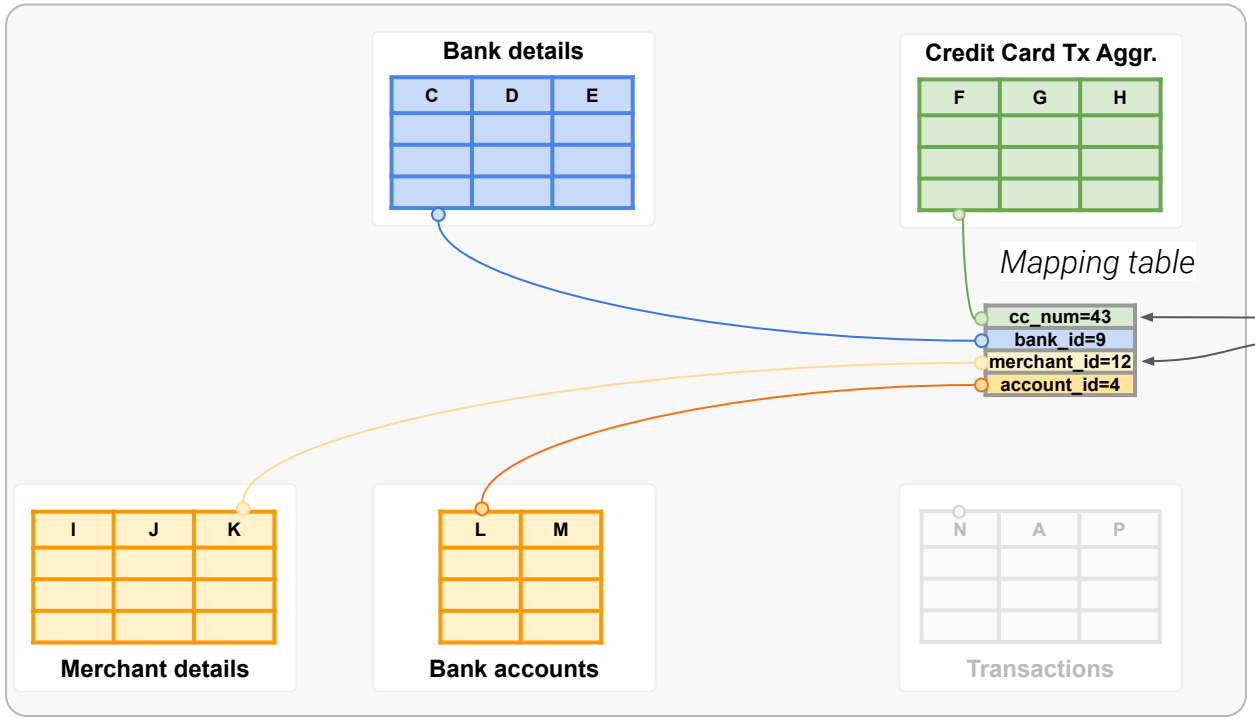
AI-Enabled Application



Problem nº1

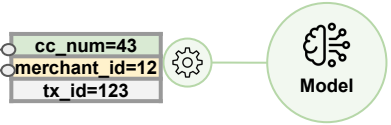
1. *Application needs to enter all IDs*

Star Schema Data Model in Online Inference Pipelines



What if we add a mapping table?

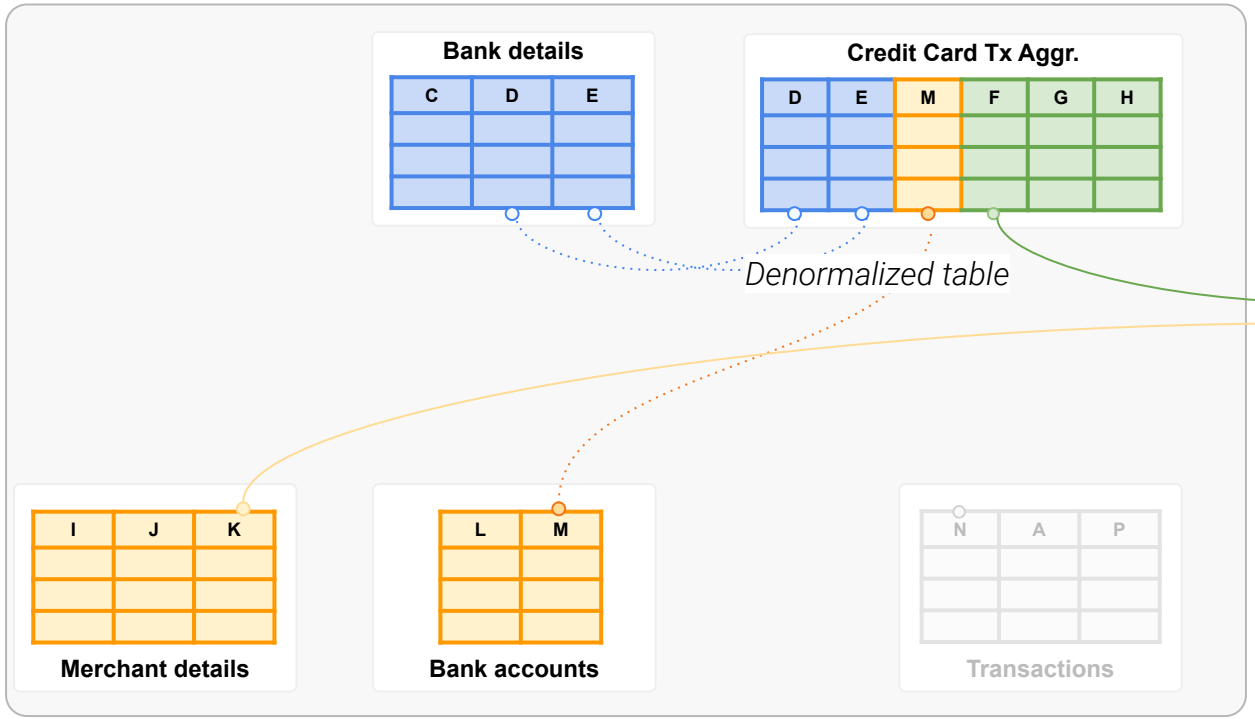
AI-Enabled Application



Problem nº2

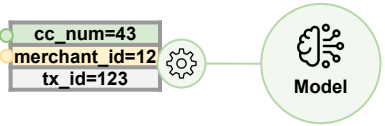
1. ~~*Application needs to enter all IDs*~~
2. *Extra ETL job(s) needed to maintain mapping tables to resolve IDs*

Star Schema Data Model in Online Inference Pipelines



What if we denormalize the Credit Card Tx. Aggr. table ?

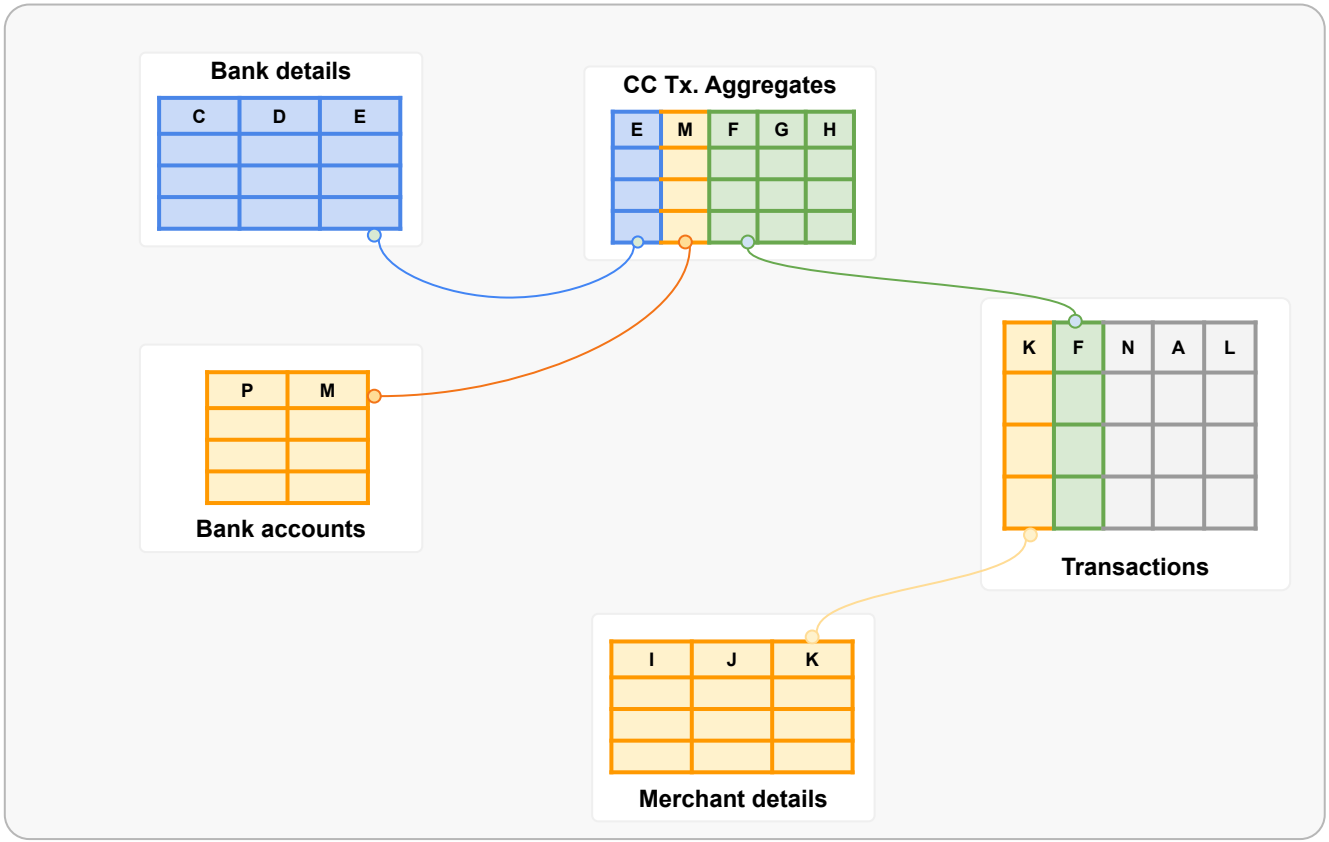
AI-Enabled Application



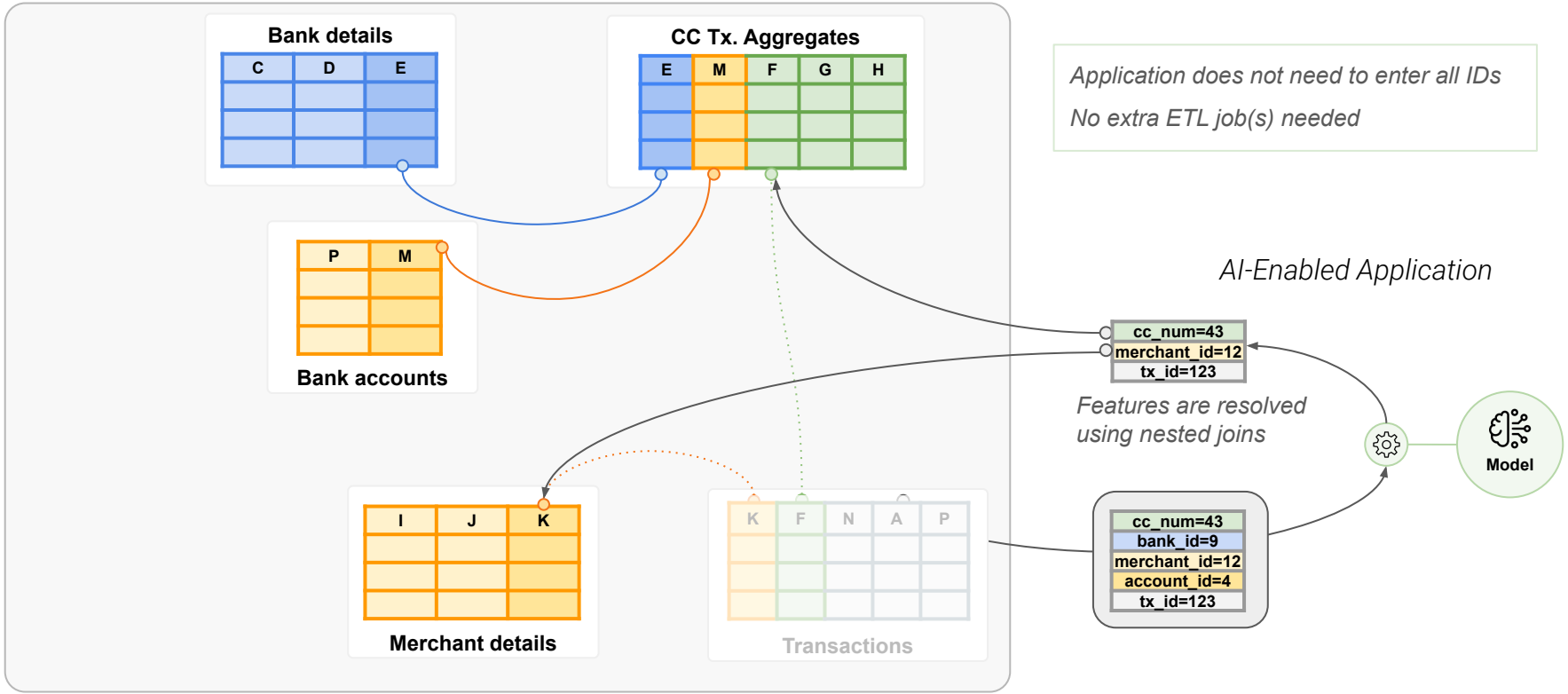
Problem nº3

1. ~~*Application needs to enter all IDs*~~
2. ~~*Extra ETL job(s) needed to maintain mapping tables to resolve IDs*~~
3. *Extra ETL job(s) to maintain denormalized table(s) with duplicated features*

What about the Snowflake Schema Data Model?



Snowflake Schema Data Model in Online Inference Pipelines



Snowflake Schema Data Model in Hopsworks

Offline + Online + RonDB = Snowflake Schema

→ Feature Groups shared schema

→ Hopsworks Feature Query Service

Nested temporal joins of Offline tables to create Point-In-Time correct training data

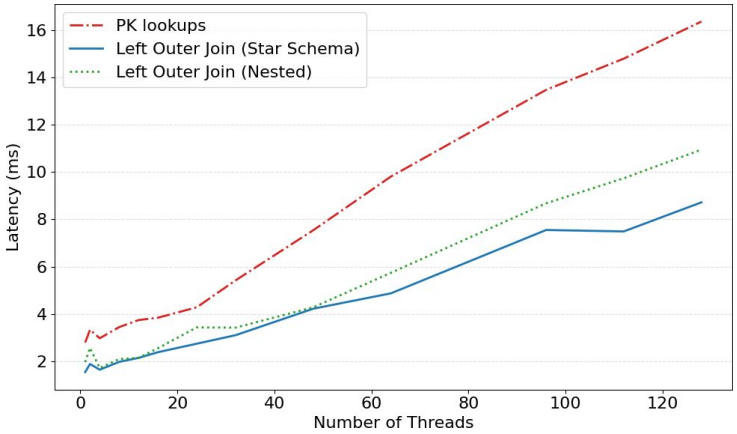
→ Support Relational Data Models (SQL)

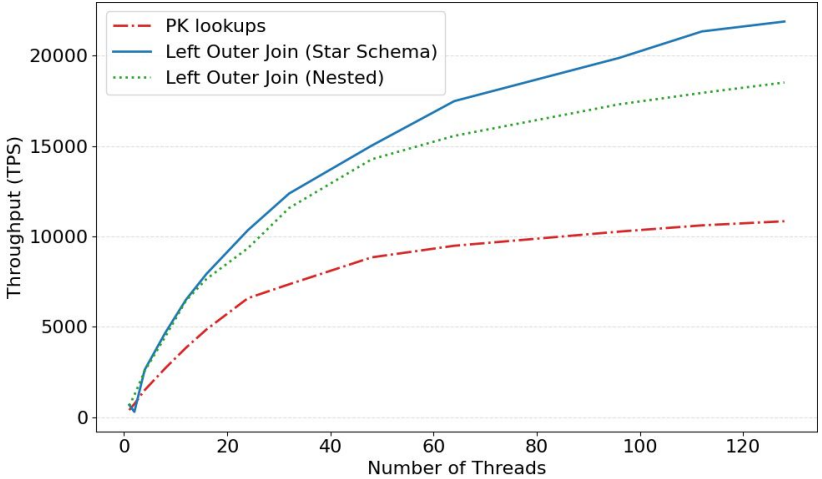
→ Pushdown Left (Nested) Joins

→ Projection pushdown

SIGMOD 2024 Research Paper - The Hopsworks Feature Store for Machine Learning - https://dl.acm.org/doi/pdf/10.1145/3626246.3653389

Pushdown Left Outer Joins in RonDB

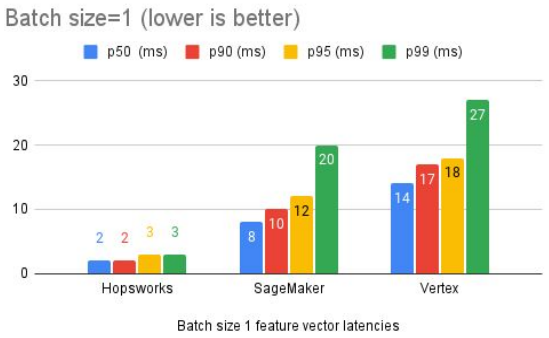


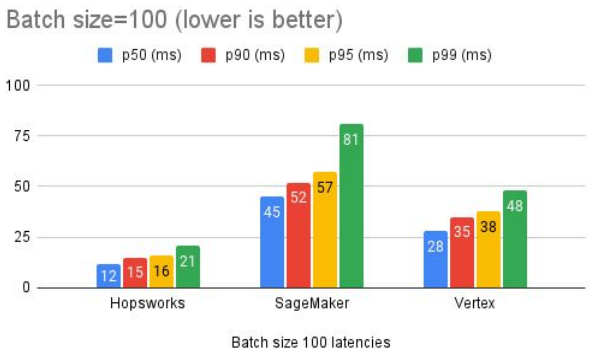


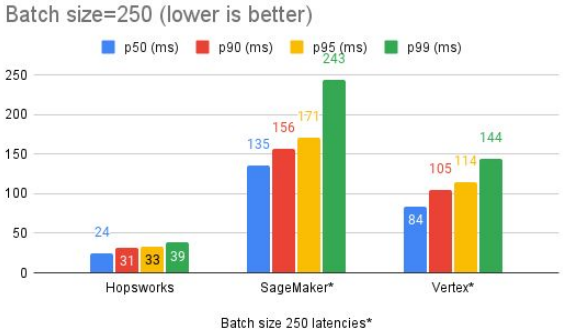
Nested Left Outer Joins (Snowflake Schema) **are comparable to** Left Outer Joins (Star schema)

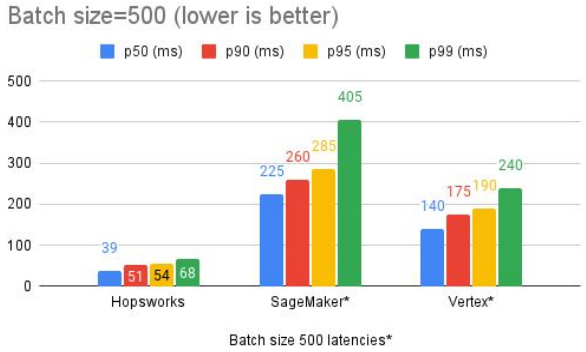
SIGMOD 2024 Research Paper - The Hopsworks Feature Store for Machine Learning - https://dl.acm.org/doi/pdf/10.1145/3626246.3653389

Online Feature Store Benchmarks → Feature Vector Retrieval









SIGMOD 2024 Research Paper - The Hopsworks Feature Store for Machine Learning - https://dl.acm.org/doi/pdf/10.1145/3626246.3653389
Feature Store Benchmarks - https://www.featurestore.org/benchmarks

Data Modeling with the Hopsworks Python API

```
labels = fs.get_feature_group("credit_card_transactions", version=1)
agg = fs.get_feature_group("aggregated_transactions", version=1)
merchant = fs.get_feature_group("merchant", version=1)
bank = fs.get_feature_group("bank_details", version=1)
account = fs.get_feature_group("account_details", version=1)
```

```
selection = labels.select_all()
      .join(merchant.select_all())
      .join(agg.select_all() # nested join
                  .join(bank.select_all())
                  .join(account.select_all()))
```

```
selection = labels.select_all() # flat join
      .join(merchant.select_all())
      .join(agg.select_all())
      .join(bank.select_all())
      .join(account.select_all())
```

```
feature_view = fs.create_feature_view(name='cc_fraud', query=selection, labels=["is_fraud"])
```

```
df = feature_view.get_feature_vectors(
    entry=[{ # less serving keys needed
      "cc_num": 1234567811112222,
      "merchant_id": 212
    }])
```

```
df = feature_view.get_feature_vectors(
    entry=[{ # more serving keys needed
      "cc_num": 123…, "bank_id": 23,
      “account_id”: 45, "merchant_id": 212
    }])
```

HOPSWORKS

- Join our slack community
- Explore our latest tutorials
- Ask us any questions

public-hopsworks.slack.com