# 7. Feature Store

## 7.1 Feature Store Approach

A **custom feature store** is implemented to manage, document, and retrieve machine learning features in a consistent and versioned manner.

The feature store is designed to:

- Centralize access to engineered features
- Ensure consistency between training and inference
- Support feature versioning and metadata management

PostgreSQL is used as the **offline feature storage layer**, while a lightweight Python-based registry manages feature definitions and versions.

---

## 7.2 Feature Store Architecture

The feature store is built on top of the feature engineering pipeline and consists of the following components:

```
Prepared Dataset
      ↓
Feature Engineering (Step 6)
      ↓
PostgreSQL Feature Tables
      ↓
Feature Registry (Metadata + Versions)
      ↓
Feature Retrieval APIs
```

This architecture mirrors the design of production-grade feature stores while remaining simple and easy to maintain.

---

## 7.3 Feature Storage Layer

Engineered features are stored in PostgreSQL tables created during Step 6:

- `user_features`
- `item_features`
- `user_item_features`

These tables act as the **offline feature store**, enabling efficient querying and reuse across different stages of the ML pipeline.

---

## 7.4 Feature Metadata Registry

A centralized **feature registry** is used to document feature metadata and manage feature versions.

### 📄 Registry Location

`src/feature_store/registry.py`

The registry captures:

- Feature group (user, item, user–item)
- Feature names
- Source tables
- Feature descriptions
- Feature version identifiers

This ensures feature discoverability and traceability.

---

## 7.5 Feature Versioning Strategy

Feature versions are managed using explicit version identifiers (e.g., `v1`).

Each version:

- Represents a stable set of feature definitions
- Can be independently retrieved for training or inference
- Enables backward compatibility when features evolve

Versioning guarantees reproducibility of model training and evaluation.

---

## 7.6 Feature Retrieval Mechanism

Feature retrieval is implemented using reusable Python functions that query PostgreSQL based on the requested feature version.

Retrieval functions support:

- User-level feature lookup
- Item-level feature lookup
- User–item interaction feature lookup

The same retrieval logic is used during **model training and inference**, ensuring feature consistency across the ML lifecycle.

---

## 7.7 Sample Feature Retrieval Demonstration

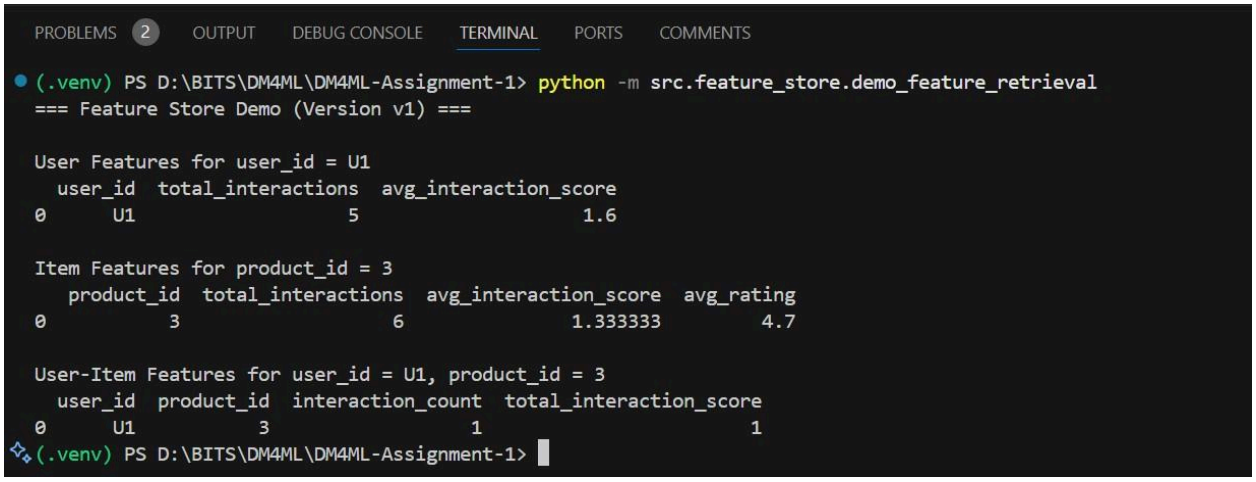
A standalone script demonstrates feature retrieval from the feature store.

### 📄 Script Location

`src/feature_store/demo_feature_retrieval.py`

The script:

- Retrieves user features for a given user ID
- Retrieves item features for a given product ID
- Retrieves user–item interaction features
- Uses versioned feature access (`v1`)




This demonstrates how downstream ML components can consume features reliably.

---

## 7.8 Training and Inference Consistency

- Both training and inference pipelines access features through the same APIs
- Feature definitions and transformations remain synchronized
- Versioned retrieval prevents feature drift between environments

This ensures reliable and repeatable model behavior.

---

## 7.9 Summary

The feature store provides a structured, versioned, and documented mechanism for managing machine learning features. It bridges the gap between feature engineering and model consumption, enabling scalable and reproducible recommendation workflows.