SAMN'S MERCADO: END TO END E-COMMERCE PLATFORM

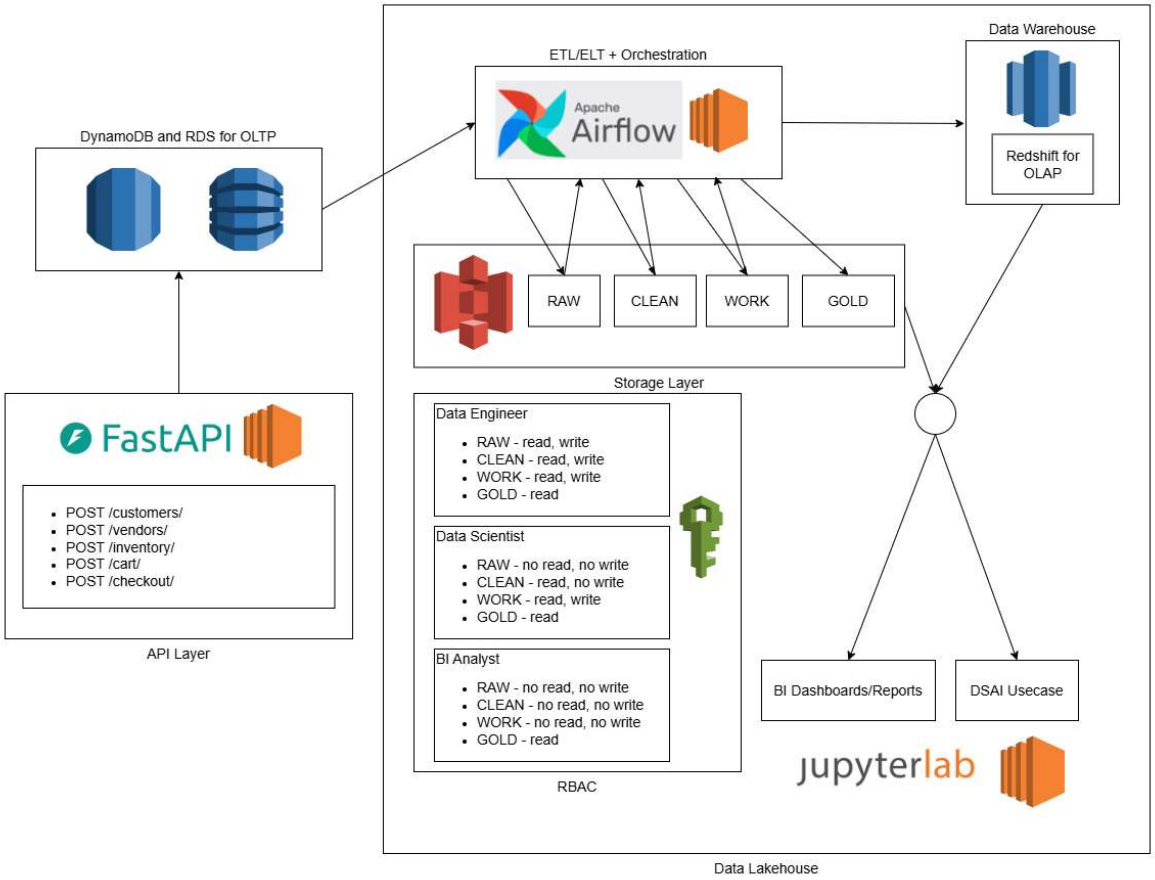
LEARNING TEAM 3 - ASIS, ITUCAL, MERCADO, RIZADA

Introduction

This project demonstrates the data engineering skills learned in the elective by developing an e-commerce platform that not only enables customers to browse and purchase a diverse selection of goods online but also empowers vendors with an analytics platform built on data lakes and data warehouses. By leveraging these technologies, vendors can track sales performance, gain actionable insights, and enable Data Science, Analytics, and AI (DSAI) capabilities to optimize their business strategies.

Architecture Overview

End to end system architecture:

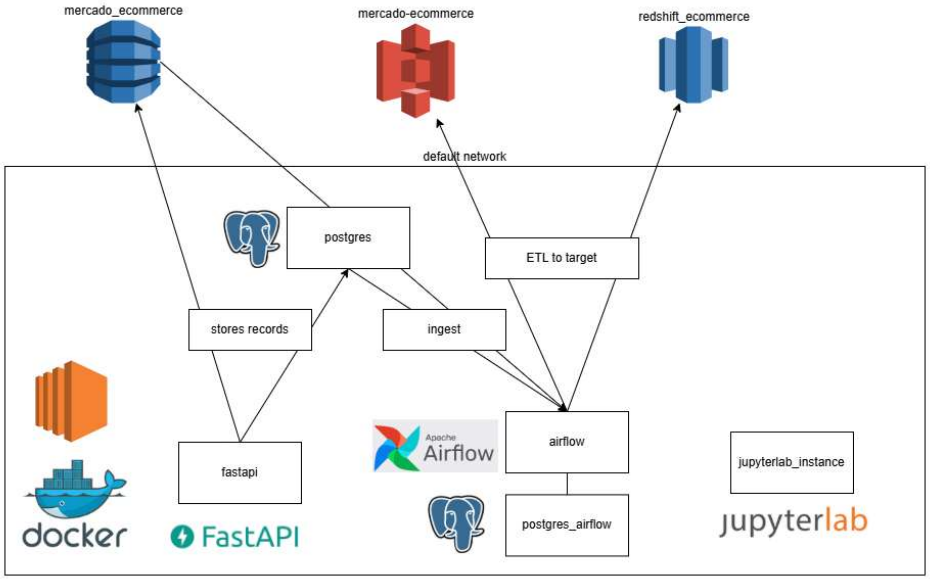


The end to end solution will include:

- API interfaces for data upload
- OLTP processing (DynamoDB+Postgres)
- Orchestration and ETL/ELT
- Data Lakehouse

- Storage Layers
- RBAC (Role Based Access Control)
- OLTP Data Warehouse
- Consumption

Docker Architecture:



Services are deployed via Docker in the EC2 instance for ease of deployment and easy to spin up and tear down.

- Postgres for OLTP - Instead of using RDS for postgres we will just use a postgres contained inside our docker network
- Airflow + Postgres - Orchestration and workflow
- Jupyterlab container - to run inserts, and provide DSAI use case in consumption.

Directory Structure

- `src` - source codes
  - `airflow` - airflow source codes
  - `api` - fastapi source codes
  - `dockerfiles` - Dockerfile definition for the services
  - `jupyterlab` - Notebooks and Images
  - `sql` - sql references for create tables
- `.github/workflows` - pre-commit hooks via github actions
- `sample_env` - sample env file to be filled up in an actual `.env file`, should specify details about connection details, and aws access keys.
- `.pre-commit-config.yaml` - pre-commit dependencies
- `docker-compose.yml` - docker compose definitions

API Layer

In this layer we use the API endpoints served via fastapi to:

- Submit customers
- Submit vendors
- Submit inventory of vendors
- Simulate customer adding to cart, checking out and leaving items in their carts

In [1]:
```
%load_ext sql
```

In [2]:
```
import boto3
import requests
import os
import pandas as pd
from sqlalchemy import create_engine
from faker import Faker
import redshift_connector
```

In [3]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [4]:
```
pd.set_option('display.max_rows', 12)
pd.set_option('display.min_rows', 6)
```

We will use faker to submit dummy API requests based on the definitions above.

In [5]:
```
fake = Faker()

base_url = "http://fastapi:8000"
headers = {"Content-Type": "application/json"}

print("customers")
# Create customers
for customer_id in range(1, 11):
    data = {
        "first_name": fake.first_name(),
        "last_name": fake.last_name(),
        "email": fake.email(),
        "joined_at": fake.iso8601()
    }
    response = requests.post(f"{base_url}/customers/", json=data, headers=headers)
    print(response.status_code, response.json())

print("vendors")
# Create vendors
for vendor_id in range(1, 6):
    data = {
        "vendor_name": fake.company(),
        "region": fake.country_code()
    }
    response = requests.post(f"{base_url}/vendors/", json=data, headers=headers)
    print(response.status_code, response.json())

print("inventory per vendor")
# Add inventory per vendor
for vendor_id in range(1, 6):
    for item_id in range(1, 11):
        params = {"vendor_id": vendor_id}
        data = {
            "item_name": fake.word(),
            "category": str(1),
            "price": fake.random_int(min=10, max=500),
        }
        response = requests.post(f"{base_url}/inventory/", json=data, headers=headers, params=params)
        print(response.status_code, response.json())

print("cart and checkout twice and add to cart again!")
for customer_id in range(1, 11):
    for _ in range(2):  # Checkout twice
        price1 = fake.random_int(min=10, max=500)
        price2 = fake.random_int(min=10, max=500)
        qty1 = fake.random_int(min=10, max=500)
        qty2 = fake.random_int(min=10, max=500)
        cart_data = {
            "user_id": customer_id,
            "cart": [
                {
                    "item_id": str(fake.random_int(min=1, max=10)),
                    "vendor_id": str(fake.random_int(min=1, max=5)),
                    "qty": qty1,
                    "unit_price": price1,
                    "total_price": qty1*price1,
                },
                {
                    "item_id": str(fake.random_int(min=1, max=10)),
                    "vendor_id": str(fake.random_int(min=1, max=5)),
                    "qty": qty2,
                    "unit_price": price2,
                    "total_price": qty2*price2,
                }
            ]
```

```
    }
    response = requests.post(f"{base_url}/cart/", json=cart_data, headers=headers)
    print(response.status_code, response.json())

    response = requests.post(f"{base_url}/checkout/", json={"user_id": customer_id}, headers=headers)
    print(response.status_code, response.text)

price1 = fake.random_int(min=10, max=500)
price2 = fake.random_int(min=10, max=500)
qty1 = fake.random_int(min=10, max=500)
qty2 = fake.random_int(min=10, max=500)
# Add to cart one more time
cart_data = {
    "user_id": customer_id,
    "cart": [
        {
            "item_id": str(fake.random_int(min=1, max=10)),
            "vendor_id": str(fake.random_int(min=1, max=5)),
            "qty": qty1,
            "unit_price": price1,
            "total_price": qty1*price1,
        },
        {
            "item_id": str(fake.random_int(min=1, max=10)),
            "vendor_id": str(fake.random_int(min=1, max=5)),
            "qty": qty1,
            "unit_price": price1,
            "total_price": qty1*price1,
        }
    ]
}
response = requests.post(f"{base_url}/cart/", json=cart_data, headers=headers)
print(response.status_code, response.json())
```

customers
201 {'id': 1, 'first_name': 'Leslie', 'last_name': 'Richards', 'email': 'kylerussell@example.com', 'joined_at': '2025-03-16T18:07:07.204516'}
201 {'id': 2, 'first_name': 'Gabriela', 'last_name': 'Crawford', 'email': 'millerjessica@example.org', 'joined_at': '2025-03-16T18:07:07.215455'}
201 {'id': 3, 'first_name': 'Melissa', 'last_name': 'Herrera', 'email': 'mendezdavid@example.org', 'joined_at': '2025-03-16T18:07:07.223133'}
201 {'id': 4, 'first_name': 'Jeffrey', 'last_name': 'Fleming', 'email': 'stacey45@example.org', 'joined_at': '2025-03-16T18:07:07.232465'}
201 {'id': 5, 'first_name': 'Jeffrey', 'last_name': 'Hoover', 'email': 'thomaskenneth@example.org', 'joined_at': '2025-03-16T18:07:07.242187'}
201 {'id': 6, 'first_name': 'Logan', 'last_name': 'Rodriguez', 'email': 'juliemorgan@example.org', 'joined_at': '2025-03-16T18:07:07.251264'}
201 {'id': 7, 'first_name': 'Kevin', 'last_name': 'Rangel', 'email': 'schmidtrobert@example.net', 'joined_at': '2025-03-16T18:07:07.260664'}
201 {'id': 8, 'first_name': 'Shawn', 'last_name': 'Stewart', 'email': 'qjordan@example.com', 'joined_at': '2025-03-16T18:07:07.269037'}
201 {'id': 9, 'first_name': 'Susan', 'last_name': 'Lester', 'email': 'nancy49@example.net', 'joined_at': '2025-03-16T18:07:07.277155'}
201 {'id': 10, 'first_name': 'Laura', 'last_name': 'Robinson', 'email': 'pthornton@example.org', 'joined_at': '2025-03-16T18:07:07.285038'}
vendors
201 {'id': 1, 'vendor_name': 'Hamilton, Horton and Ingram', 'region': 'TV', 'joined_at': '2025-03-16T18:07:07.295645'}
201 {'id': 2, 'vendor_name': 'Hester-Ramirez', 'region': 'GY', 'joined_at': '2025-03-16T18:07:07.304975'}
201 {'id': 3, 'vendor_name': 'Rogers, Santos and Flowers', 'region': 'TR', 'joined_at': '2025-03-16T18:07:07.313088'}
201 {'id': 4, 'vendor_name': 'Lam-Smith', 'region': 'CH', 'joined_at': '2025-03-16T18:07:07.320201'}
201 {'id': 5, 'vendor_name': 'Ali-Thompson', 'region': 'EC', 'joined_at': '2025-03-16T18:07:07.327773'}
inventory per vendor
200 {'id': 1, 'item_name': 'experience', 'category': '1', 'price': 216.0, 'vendor_id': 1, 'updated_at': '2025-03-16T18:07:07.336667'}
200 {'id': 2, 'item_name': 'statement', 'category': '1', 'price': 49.0, 'vendor_id': 1, 'updated_at': '2025-03-16T18:07:07.348005'}
200 {'id': 3, 'item_name': 'list', 'category': '1', 'price': 283.0, 'vendor_id': 1, 'updated_at': '2025-03-16T18:07:07.357084'}
200 {'id': 4, 'item_name': 'name', 'category': '1', 'price': 134.0, 'vendor_id': 1, 'updated_at': '2025-03-16T18:07:07.366226'}
200 {'id': 5, 'item_name': 'lot', 'category': '1', 'price': 303.0, 'vendor_id': 1, 'updated_at': '2025-03-16T18:07:07.374240'}
200 {'id': 6, 'item_name': 'stock', 'category': '1', 'price': 206.0, 'vendor_id': 1, 'updated_at': '2025-03-16T18:07:07.383193'}
200 {'id': 7, 'item_name': 'trial', 'category': '1', 'price': 278.0, 'vendor_id': 1, 'updated_at': '2025-03-16T18:07:07.391071'}
200 {'id': 8, 'item_name': 'build', 'category': '1', 'price': 284.0, 'vendor_id': 1, 'updated_at': '2025-03-16T18:07:07.400003'}
200 {'id': 9, 'item_name': 'rather', 'category': '1', 'price': 366.0, 'vendor_id': 1, 'updated_at': '2025-03-16T18:07:07.408077'}
200 {'id': 10, 'item_name': 'certain', 'category': '1', 'price': 418.0, 'vendor_id': 1, 'updated_at': '2025-03-16T18:07:07.417665'}
200 {'id': 11, 'item_name': 'including', 'category': '1', 'price': 199.0, 'vendor_id': 2, 'updated_at': '2025-03-16T18:07:07.427733'}
200 {'id': 12, 'item_name': 'a', 'category': '1', 'price': 373.0, 'vendor_id': 2, 'updated_at': '2025-03-16T18:07:07.437183'}
200 {'id': 13, 'item_name': 'ago', 'category': '1', 'price': 298.0, 'vendor_id': 2, 'updated_at': '2025-03-16T18:07:07.447092'}
200 {'id': 14, 'item_name': 'really', 'category': '1', 'price': 448.0, 'vendor_id': 2, 'updated_at': '2025-03-16T18:07:07.454642'}
200 {'id': 15, 'item_name': 'production', 'category': '1', 'price': 364.0, 'vendor_id': 2, 'updated_at': '2025-03-16T18:07:07.462280'}
200 {'id': 16, 'item_name': 'generation', 'category': '1', 'price': 355.0, 'vendor_id': 2, 'updated_at': '2025-03-16T18:07:07.470316'}
200 {'id': 17, 'item_name': 'result', 'category': '1', 'price': 161.0, 'vendor_id': 2, 'updated_at': '2025-03-16T18:07:07.477692'}
200 {'id': 18, 'item_name': 'idea', 'category': '1', 'price': 171.0, 'vendor_id': 2, 'updated_at': '2025-03-16T18:07:07.488007'}
200 {'id': 19, 'item_name': 'effort', 'category': '1', 'price': 446.0, 'vendor_id': 2, 'updated_at': '2025-03-16T18:07:07.498017'}
200 {'id': 20, 'item_name': 'reach', 'category': '1', 'price': 448.0, 'vendor_id': 2, 'updated_at': '2025-03-16T18:07:07.509458'}
200 {'id': 21, 'item_name': 'research', 'category': '1', 'price': 370.0, 'vendor_id': 3, 'updated_at': '2025-03-16T18:07:07.517477'}
200 {'id': 22, 'item_name': 'feeling', 'category': '1', 'price': 426.0, 'vendor_id': 3, 'updated_at': '2025-03-16T18:07:07.527701'}
200 {'id': 23, 'item_name': 'especially', 'category': '1', 'price': 148.0, 'vendor_id': 3, 'updated_at': '2025-03-16T18:07:07.538512'}
200 {'id': 24, 'item_name': 'little', 'category': '1', 'price': 282.0, 'vendor_id': 3, 'updated_at': '2025-03-16T18:07:07.547291'}
200 {'id': 25, 'item_name': 'girl', 'category': '1', 'price': 71.0, 'vendor_id': 3, 'updated_at': '2025-03-16T18:07:07.557983'}
200 {'id': 26, 'item_name': 'from', 'category': '1', 'price': 388.0, 'vendor_id': 3, 'updated_at': '2025-03-16T18:07:07.566198'}
200 {'id': 27, 'item_name': 'room', 'category': '1', 'price': 296.0, 'vendor_id': 3, 'updated_at': '2025-03-16T18:07:07.574760'}
200 {'id': 28, 'item_name': 'star', 'category': '1', 'price': 349.0, 'vendor_id': 3, 'updated_at': '2025-03-16T18:07:07.583960'}
200 {'id': 29, 'item_name': 'final', 'category': '1', 'price': 321.0, 'vendor_id': 3, 'updated_at': '2025-03-16T18:07:07.593661'}
200 {'id': 30, 'item_name': 'so', 'category': '1', 'price': 236.0, 'vendor_id': 3, 'updated_at': '2025-03-16T18:07:07.601333'}
200 {'id': 31, 'item_name': 'according', 'category': '1', 'price': 140.0, 'vendor_id': 4, 'updated_at': '2025-03-16T18:07:07.609452'}
200 {'id': 32, 'item_name': 'still', 'category': '1', 'price': 22.0, 'vendor_id': 4, 'updated_at': '2025-03-16T18:07:07.617462'}
200 {'id': 33, 'item_name': 'region', 'category': '1', 'price': 100.0, 'vendor_id': 4, 'updated_at': '2025-03-16T18:07:07.626903'}
200 {'id': 34, 'item_name': 'nature', 'category': '1', 'price': 29.0, 'vendor_id': 4, 'updated_at': '2025-03-16T18:07:07.634948'}
200 {'id': 35, 'item_name': 'son', 'category': '1', 'price': 325.0, 'vendor_id': 4, 'updated_at': '2025-03-16T18:07:07.646302'}
200 {'id': 36, 'item_name': 'third', 'category': '1', 'price': 323.0, 'vendor_id': 4, 'updated_at': '2025-03-16T18:07:07.656938'}
200 {'id': 37, 'item_name': 'enough', 'category': '1', 'price': 112.0, 'vendor_id': 4, 'updated_at': '2025-03-16T18:07:07.665983'}
200 {'id': 38, 'item_name': 'every', 'category': '1', 'price': 366.0, 'vendor_id': 4, 'updated_at': '2025-03-16T18:07:07.673695'}
200 {'id': 39, 'item_name': 'partner', 'category': '1', 'price': 344.0, 'vendor_id': 4, 'updated_at': '2025-03-16T18:07:07.680712'}
200 {'id': 40, 'item_name': 'international', 'category': '1', 'price': 27.0, 'vendor_id': 4, 'updated_at': '2025-03-16T18:07:07.690561'}
200 {'id': 41, 'item_name': 'writer', 'category': '1', 'price': 232.0, 'vendor_id': 5, 'updated_at': '2025-03-16T18:07:07.702012'}
200 {'id': 42, 'item_name': 'though', 'category': '1', 'price': 393.0, 'vendor_id': 5, 'updated_at': '2025-03-16T18:07:07.710193'}
200 {'id': 43, 'item_name': 'evening', 'category': '1', 'price': 249.0, 'vendor_id': 5, 'updated_at': '2025-03-16T18:07:07.718666'}
200 {'id': 44, 'item_name': 'others', 'category': '1', 'price': 165.0, 'vendor_id': 5, 'updated_at': '2025-03-16T18:07:07.727642'}
200 {'id': 45, 'item_name': 'social', 'category': '1', 'price': 440.0, 'vendor_id': 5, 'updated_at': '2025-03-16T18:07:07.736033'}
200 {'id': 46, 'item_name': 'side', 'category': '1', 'price': 41.0, 'vendor_id': 5, 'updated_at': '2025-03-16T18:07:07.742628'}
200 {'id': 47, 'item_name': 'language', 'category': '1', 'price': 302.0, 'vendor_id': 5, 'updated_at': '2025-03-16T18:07:07.751230'}
200 {'id': 48, 'item_name': 'growth', 'category': '1', 'price': 106.0, 'vendor_id': 5, 'updated_at': '2025-03-16T18:07:07.758669'}
200 {'id': 49, 'item_name': 'arm', 'category': '1', 'price': 155.0, 'vendor_id': 5, 'updated_at': '2025-03-16T18:07:07.767366'}
200 {'id': 50, 'item_name': 'range', 'category': '1', 'price': 17.0, 'vendor_id': 5, 'updated_at': '2025-03-16T18:07:07.775252'}
cart and checkout twice and add to cart again!
200 {'user_id': 1, 'cart': [{'item_id': '2', 'qty': 294, 'vendor_id': '5', 'unit_price': '189', 'total_price': '55566'}, {'item_id': '6', 'qty': 31, 'vendor_id': '3', 'unit_price': '23', 'total_price': '713'}]}
200 {"message":"Transaction completed","transaction_id":"dbb41f1d-cc3b-4a2b-9d66-b51cbd8b4d0d"}
200 {'user_id': 1, 'cart': [{'item_id': '5', 'qty': 102, 'vendor_id': '5', 'unit_price': '394', 'total_price': '40188'}, {'item_id': '8', 'qty': 45, 'vendor_id': '2', 'unit_price': '247', 'total_price': '11115'}]}
200 {"message":"Transaction completed","transaction_id":"b8ba1dfb-998c-4916-987d-699767096f7b"}
200 {'user_id': 1, 'cart': [{'item_id': '6', 'qty': 330, 'vendor_id': '1', 'unit_price': '32', 'total_price': '10560'}, {'item_id': '7', 'qty': 330, 'vendor_id': '2', 'unit_price': '32', 'total_price': '10560'}]}
200 {'user_id': 2, 'cart': [{'item_id': '4', 'qty': 20, 'vendor_id': '4', 'unit_price': '59', 'total_price': '1180'}, {'item_id': '1', 'qty': 359, 'vendor_id': '1', 'unit_price': '482', 'total_price': '173038'}]}
200 {"message":"Transaction completed","transaction_id":"1f660feb-c531-4eff-8d57-2de2f30e9d0d"}
200 {'user_id': 2, 'cart': [{'item_id': '3', 'qty': 21, 'vendor_id': '5', 'unit_price': '307', 'total_price': '6447'}, {'item_id': '1', 'qty': 471, 'vendor_id': '1', 'unit_price': '327', 'total_price': '154017'}]}
200 {"message":"Transaction completed","transaction_id":"2e015f82-2928-4de7-948b-0f19c6bd5df4"}
200 {'user_id': 2, 'cart': [{'item_id': '4', 'qty': 170, 'vendor_id': '5', 'unit_price': '318', 'total_price': '54060'}, {'item_id': '6', 'qty': 170, 'vendor_id': '3', 'unit_price': '318', 'total_price': '54060'}]}
200 {'user_id': 3, 'cart': [{'item_id': '1', 'qty': 183, 'vendor_id': '1', 'unit_price': '449', 'total_price': '82167'}, {'item_id': '1', 'qty': 232, 'vendor_id': '4', 'unit_price': '83', 'total_price': '19256'}]}
200 {"message":"Transaction completed","transaction_id":"05dfe9a6-08b4-4e04-81ac-bc92c0265c61"}
200 {'user_id': 3, 'cart': [{'item_id': '10', 'qty': 70, 'vendor_id': '5', 'unit_price': '169', 'total_price': '11830'}, {'item_id': '5', 'qty': 323, 'vendor_id': '2', 'unit_price': '412', 'total_price': '133076'}]}
200 {"message":"Transaction completed","transaction_id":"dba6ca4b-a0bf-4a57-a82a-902a26078354"}

```
200 {'user_id': 3, 'cart': [{'item_id': '7', 'qty': 447, 'vendor_id': '5', 'unit_price': '494', 'total_price': '220818'}, {'item_id': '5', 'qty': 447, 'vendor_id': '2', 'unit_price': '494', 'total_price': '220818'}]}
200 {'user_id': 4, 'cart': [{'item_id': '8', 'qty': 62, 'vendor_id': '4', 'unit_price': '19', 'total_price': '1178'}, {'item_id': '5', 'qty': 268, 'vendor_id': '4', 'unit_price': '255', 'total_price': '68340'}]}
200 {"message":"Transaction completed","transaction_id":"eba4945f-98b5-44db-aee8-656f9a457f34"}
200 {'user_id': 4, 'cart': [{'item_id': '7', 'qty': 483, 'vendor_id': '1', 'unit_price': '468', 'total_price': '226044'}, {'item_id': '4', 'qty': 479, 'vendor_id': '5', 'unit_price': '150', 'total_price': '71850'}]}
200 {"message":"Transaction completed","transaction_id":"d0def17d-d596-4623-a37a-07deb9de0355"}
200 {'user_id': 4, 'cart': [{'item_id': '3', 'qty': 345, 'vendor_id': '5', 'unit_price': '472', 'total_price': '162840'}, {'item_id': '9', 'qty': 345, 'vendor_id': '5', 'unit_price': '472', 'total_price': '162840'}]}
200 {'user_id': 5, 'cart': [{'item_id': '5', 'qty': 22, 'vendor_id': '4', 'unit_price': '457', 'total_price': '10054'}, {'item_id': '3', 'qty': 67, 'vendor_id': '1', 'unit_price': '477', 'total_price': '31959'}]}
200 {"message":"Transaction completed","transaction_id":"85a45ea5-e91c-4c56-ade0-03cc9413bc93"}
200 {'user_id': 5, 'cart': [{'item_id': '5', 'qty': 392, 'vendor_id': '3', 'unit_price': '421', 'total_price': '165032'}, {'item_id': '5', 'qty': 161, 'vendor_id': '3', 'unit_price': '18', 'total_price': '2898'}]}
200 {"message":"Transaction completed","transaction_id":"e46dee10-6792-4f0a-b26f-20e956135389"}
200 {'user_id': 5, 'cart': [{'item_id': '2', 'qty': 100, 'vendor_id': '4', 'unit_price': '315', 'total_price': '31500'}, {'item_id': '3', 'qty': 100, 'vendor_id': '5', 'unit_price': '315', 'total_price': '31500'}]}
200 {'user_id': 6, 'cart': [{'item_id': '4', 'qty': 310, 'vendor_id': '5', 'unit_price': '118', 'total_price': '36580'}, {'item_id': '6', 'qty': 368, 'vendor_id': '3', 'unit_price': '212', 'total_price': '78016'}]}
200 {"message":"Transaction completed","transaction_id":"9e632b46-5948-4569-9adf-def3b084185b"}
200 {'user_id': 6, 'cart': [{'item_id': '1', 'qty': 387, 'vendor_id': '5', 'unit_price': '233', 'total_price': '90171'}, {'item_id': '1', 'qty': 161, 'vendor_id': '4', 'unit_price': '373', 'total_price': '60053'}]}
200 {"message":"Transaction completed","transaction_id":"638283d8-18a4-4f26-ada8-9c2f29c68caa"}
200 {'user_id': 6, 'cart': [{'item_id': '3', 'qty': 253, 'vendor_id': '1', 'unit_price': '189', 'total_price': '47817'}, {'item_id': '9', 'qty': 253, 'vendor_id': '5', 'unit_price': '189', 'total_price': '47817'}]}
200 {'user_id': 7, 'cart': [{'item_id': '10', 'qty': 14, 'vendor_id': '3', 'unit_price': '33', 'total_price': '462'}, {'item_id': '5', 'qty': 219, 'vendor_id': '4', 'unit_price': '103', 'total_price': '22557'}]}
200 {"message":"Transaction completed","transaction_id":"9957d825-ecdf-4e3e-ac66-da1477506958"}
200 {'user_id': 7, 'cart': [{'item_id': '4', 'qty': 304, 'vendor_id': '1', 'unit_price': '322', 'total_price': '97888'}, {'item_id': '7', 'qty': 158, 'vendor_id': '5', 'unit_price': '23', 'total_price': '3634'}]}
200 {"message":"Transaction completed","transaction_id":"e33df40d-3dba-4228-be8c-519fe985425f"}
200 {'user_id': 7, 'cart': [{'item_id': '6', 'qty': 16, 'vendor_id': '3', 'unit_price': '191', 'total_price': '3056'}, {'item_id': '9', 'qty': 16, 'vendor_id': '2', 'unit_price': '191', 'total_price': '3056'}]}
200 {'user_id': 8, 'cart': [{'item_id': '5', 'qty': 89, 'vendor_id': '5', 'unit_price': '298', 'total_price': '26522'}, {'item_id': '5', 'qty': 101, 'vendor_id': '3', 'unit_price': '116', 'total_price': '11716'}]}
200 {"message":"Transaction completed","transaction_id":"772b70ec-08bb-4bfe-8646-6450b7776306"}
200 {'user_id': 8, 'cart': [{'item_id': '5', 'qty': 17, 'vendor_id': '4', 'unit_price': '336', 'total_price': '5712'}, {'item_id': '6', 'qty': 427, 'vendor_id': '5', 'unit_price': '92', 'total_price': '39284'}]}
200 {"message":"Transaction completed","transaction_id":"7051e008-ac0b-4b24-a107-dd6f5f887fbc"}
200 {'user_id': 8, 'cart': [{'item_id': '8', 'qty': 145, 'vendor_id': '2', 'unit_price': '153', 'total_price': '22185'}, {'item_id': '3', 'qty': 145, 'vendor_id': '5', 'unit_price': '153', 'total_price': '22185'}]}
200 {'user_id': 9, 'cart': [{'item_id': '8', 'qty': 162, 'vendor_id': '2', 'unit_price': '365', 'total_price': '59130'}, {'item_id': '3', 'qty': 421, 'vendor_id': '1', 'unit_price': '377', 'total_price': '158717'}]}
200 {"message":"Transaction completed","transaction_id":"0ea90e0f-b1aa-4a83-99ce-3d84466c7730"}
200 {'user_id': 9, 'cart': [{'item_id': '4', 'qty': 119, 'vendor_id': '2', 'unit_price': '340', 'total_price': '40460'}, {'item_id': '1', 'qty': 485, 'vendor_id': '4', 'unit_price': '147', 'total_price': '71295'}]}
200 {"message":"Transaction completed","transaction_id":"9e4be843-6388-4923-8b21-68e2cc6ed9f8"}
200 {'user_id': 9, 'cart': [{'item_id': '6', 'qty': 342, 'vendor_id': '2', 'unit_price': '269', 'total_price': '91998'}, {'item_id': '4', 'qty': 342, 'vendor_id': '5', 'unit_price': '269', 'total_price': '91998'}]}
200 {'user_id': 10, 'cart': [{'item_id': '10', 'qty': 351, 'vendor_id': '5', 'unit_price': '107', 'total_price': '37557'}, {'item_id': '7', 'qty': 272, 'vendor_id': '3', 'unit_price': '136', 'total_price': '36992'}]}
200 {"message":"Transaction completed","transaction_id":"e700194e-80bc-4553-a9c0-83a32ed2ab9d"}
200 {'user_id': 10, 'cart': [{'item_id': '3', 'qty': 415, 'vendor_id': '4', 'unit_price': '445', 'total_price': '184675'}, {'item_id': '8', 'qty': 168, 'vendor_id': '3', 'unit_price': '200', 'total_price': '33600'}]}
200 {"message":"Transaction completed","transaction_id":"d19bcd4a-1216-43ad-bb09-847c82ae49e1"}
200 {'user_id': 10, 'cart': [{'item_id': '5', 'qty': 160, 'vendor_id': '3', 'unit_price': '314', 'total_price': '50240'}, {'item_id': '10', 'qty': 160, 'vendor_id': '1', 'unit_price': '314', 'total_price': '50240'}]}
```

In [6]:
```
OLTP_USER = os.getenv("POSTGRES_USER")
OLTP_PASS = os.getenv("POSTGRES_PASSWORD")
OLTP_HOST = os.getenv("POSTGRES_HOST")
OLTP_DB_NAME = os.getenv("POSTGRES_DB")
OLAP_USER = os.getenv("REDSHIFT_USER")
OLAP_PASS = os.getenv("REDSHIFT_PASSWORD")
OLAP_HOST = os.getenv("REDSHIFT_HOST")
OLAP_DB_NAME = os.getenv("REDSHIFT_DB")
```

In [7]:
```
connection_string = f"postgresql://{OLTP_USER}:{OLTP_PASS}@{OLTP_HOST}:5432/{OLTP_DB_NAME}"
engine = create_engine(connection_string)
get_ipython().run_line_magic('sql', connection_string)
```

Connecting to 'postgresql://user:***@db:5432/ecommerce'

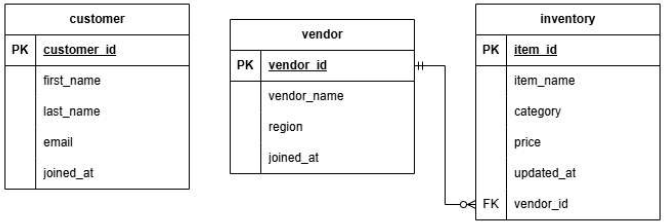
Viewing OLTP Layer - Postgres and NoSQL Contents

Implementing the OLTP layer both SQL and NoSQL storage was used to handle different use cases:

- Postgres in RDS
  - customer table
  - vendor table
  - inventory table
- NoSQL DynamoDB
  - cart details
  - customer transaction history

Postgres side

ERD:



The selected tables were stored in PostgreSQL to take advantage of its SQL-friendly search capabilities. On the customer side, this allows for efficient searching of products by item name or vendor name, enhancing the shopping experience. On the vendor side, PostgreSQL enables seamless lookup of specific clients using their name or email, making it easier to manage customer relationships and track transactions.

*Create table definition found in `src\sql\*` directory

In [8]:
```
%sql \dt
```

Running query in 'postgresql://user:***@db:5432/ecommerce'

Out[8]:

Schema	Name	Type	Owner
public	customer	table	user
public	inventory	table	user
public	vendor	table	user

In [9]:
```
table_list = [
    "customer", "inventory", "vendor"
]
```

In [10]:
```
for table in table_list:
    print(f"{table}")
    schema_query = f"columns --table {table}"
    display(get_ipython().run_line_magic('sqlcmd', schema_query))
```

customer

name	type	nullable	default	autoincrement	comment
id	INTEGER	False	nextval('customer_id_seq'::regclass)	True	None
first_name	TEXT	False	None	False	None
last_name	TEXT	False	None	False	None
email	TEXT	False	None	False	None
joined_at	TIMESTAMP	False	None	False	None

inventory

name	type	nullable	default	autoincrement	comment
id	INTEGER	False	nextval('inventory_id_seq'::regclass)	True	None
item_name	TEXT	False	None	False	None
category	TEXT	False	None	False	None
price	INTEGER	False	None	False	None
updated_at	TIMESTAMP	False	None	False	None
vendor_id	INTEGER	True	None	False	None

vendor

name	type	nullable	default	autoincrement	comment
id	INTEGER	False	nextval('vendor_id_seq'::regclass)	True	None
vendor_name	TEXT	False	None	False	None
region	TEXT	False	None	False	None
joined_at	TIMESTAMP	False	None	False	None

In [11]:
```
for table in table_list:
    print(f"{table}")
    display(pd.read_sql(f"select * from {table} limit 100", engine.raw_connection()))
```

customer

	id	first_name	last_name	email	joined_at
0	1	Leslie	Richards	kylerussell@example.com	2025-03-16 18:07:07.204516
1	2	Gabriela	Crawford	millerjessica@example.org	2025-03-16 18:07:07.215455
2	3	Melissa	Herrera	mendezdavid@example.org	2025-03-16 18:07:07.223133
3	4	Jeffrey	Fleming	stacey45@example.org	2025-03-16 18:07:07.232465
4	5	Jeffrey	Hoover	thomaskenneth@example.org	2025-03-16 18:07:07.242187
5	6	Logan	Rodriguez	juliemorgan@example.org	2025-03-16 18:07:07.251264
6	7	Kevin	Rangel	schmidtrobert@example.net	2025-03-16 18:07:07.260664
7	8	Shawn	Stewart	qjordan@example.com	2025-03-16 18:07:07.269037
8	9	Susan	Lester	nancy49@example.net	2025-03-16 18:07:07.277155
9	10	Laura	Robinson	pthornton@example.org	2025-03-16 18:07:07.285038

inventory

	id	item_name	category	price	updated_at	vendor_id
0	1	experience	1	216	2025-03-16 18:07:07.336667	1
1	2	statement	1	49	2025-03-16 18:07:07.348005	1
2	3	list	1	283	2025-03-16 18:07:07.357084	1
...	...	...	...	...	...	...
47	48	growth	1	106	2025-03-16 18:07:07.758669	5
48	49	arm	1	155	2025-03-16 18:07:07.767366	5
49	50	range	1	17	2025-03-16 18:07:07.775252	5

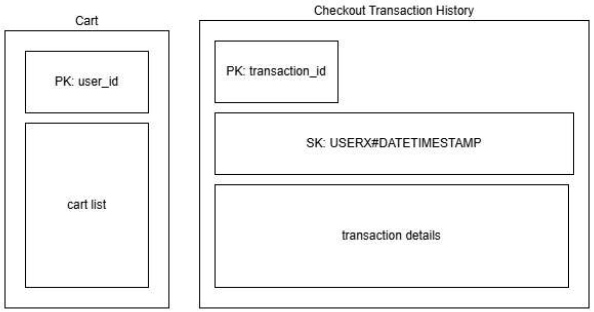
50 rows × 6 columns

vendor

	id	vendor_name	region	joined_at
0	1	Hamilton, Horton and Ingram	TV	2025-03-16 18:07:07.295645
1	2	Hester-Ramirez	GY	2025-03-16 18:07:07.304975
2	3	Rogers, Santos and Flowers	TR	2025-03-16 18:07:07.313088
3	4	Lam-Smith	CH	2025-03-16 18:07:07.320201
4	5	Ali-Thompson	EC	2025-03-16 18:07:07.327773

DynamoDB side

ERD/Entity Containment in NoSQL:



The Cart and Transaction entities are stored in DynamoDB to take advantage of its flexible schema and high-performance read/write capabilities. The Cart table uses `user_id` as the partition key (PK), allowing it to store a nested list of items efficiently, enabling quick updates and retrievals as users modify their carts. The Transaction table is structured with `transaction_id` as the partition key (PK) and `user_id + timestamp` as the sort key (SK), ensuring efficient lookups and chronological ordering of transactions per user. DynamoDB's scalability and fast access times make it an ideal choice for handling real-time cart updates and transaction history in a dynamic e-commerce environment.

In [12]:
```
dynamodb = boto3.resource("dynamodb", "us-east-1")
table = dynamodb.Table("mercado_ecommerce")
table.scan()['Items'][:3]
```

Out[12]:
```
[{'updated_at': '2025-03-16T18:07:08.335537',
  'sk': 'CART',
  'pk': 'USER#8',
  'cart': [{'unit_price': Decimal('153'),
    'total_price': Decimal('22185'),
    'item_id': '8',
    'qty': Decimal('145'),
    'vendor_id': '2'},
   {'unit_price': Decimal('153'),
    'total_price': Decimal('22185'),
    'item_id': '3',
    'qty': Decimal('145'),
    'vendor_id': '5'}]},
 {'sk': 'USER8#2025-03-16T18:07:08.294212',
  'created_at': '2025-03-16T18:07:08.294212',
  'pk': '772b70ec-08bb-4bfe-8646-6450b7776306',
  'cart': [{'unit_price': Decimal('298'),
    'total_price': Decimal('26522'),
    'item_id': '5',
    'qty': Decimal('89'),
    'vendor_id': '5'},
   {'unit_price': Decimal('116'),
    'total_price': Decimal('11716'),
    'item_id': '5',
    'qty': Decimal('101'),
    'vendor_id': '3'}]},
 {'sk': 'USER7#2025-03-16T18:07:08.259143',
  'created_at': '2025-03-16T18:07:08.259143',
  'pk': 'e33df40d-3dba-4228-be8c-519fe985425f',
  'cart': [{'unit_price': Decimal('322'),
    'total_price': Decimal('97888'),
    'item_id': '4',
    'qty': Decimal('304'),
    'vendor_id': '1'},
   {'unit_price': Decimal('23'),
    'total_price': Decimal('3634'),
    'item_id': '7',
    'qty': Decimal('158'),
    'vendor_id': '5'}]}]
```

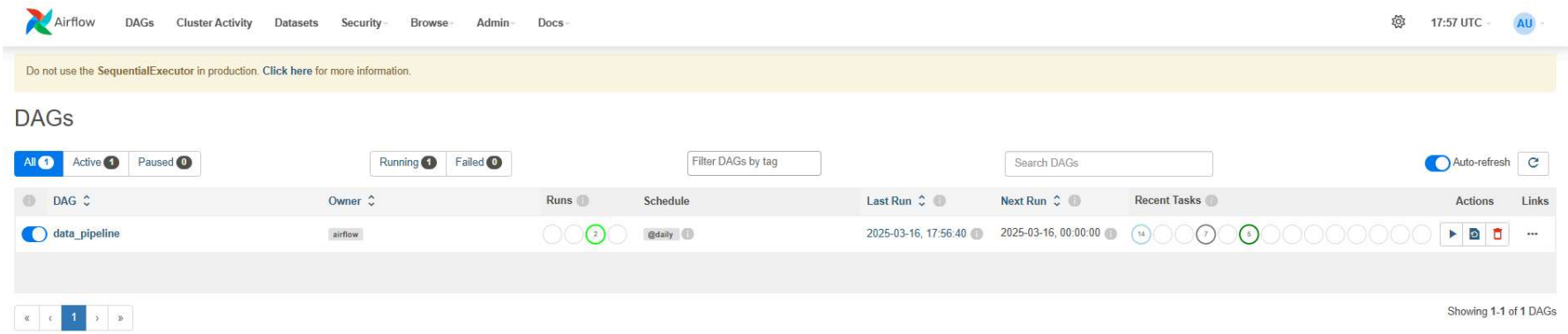
Worflow, Orchestration and ETL/ELT

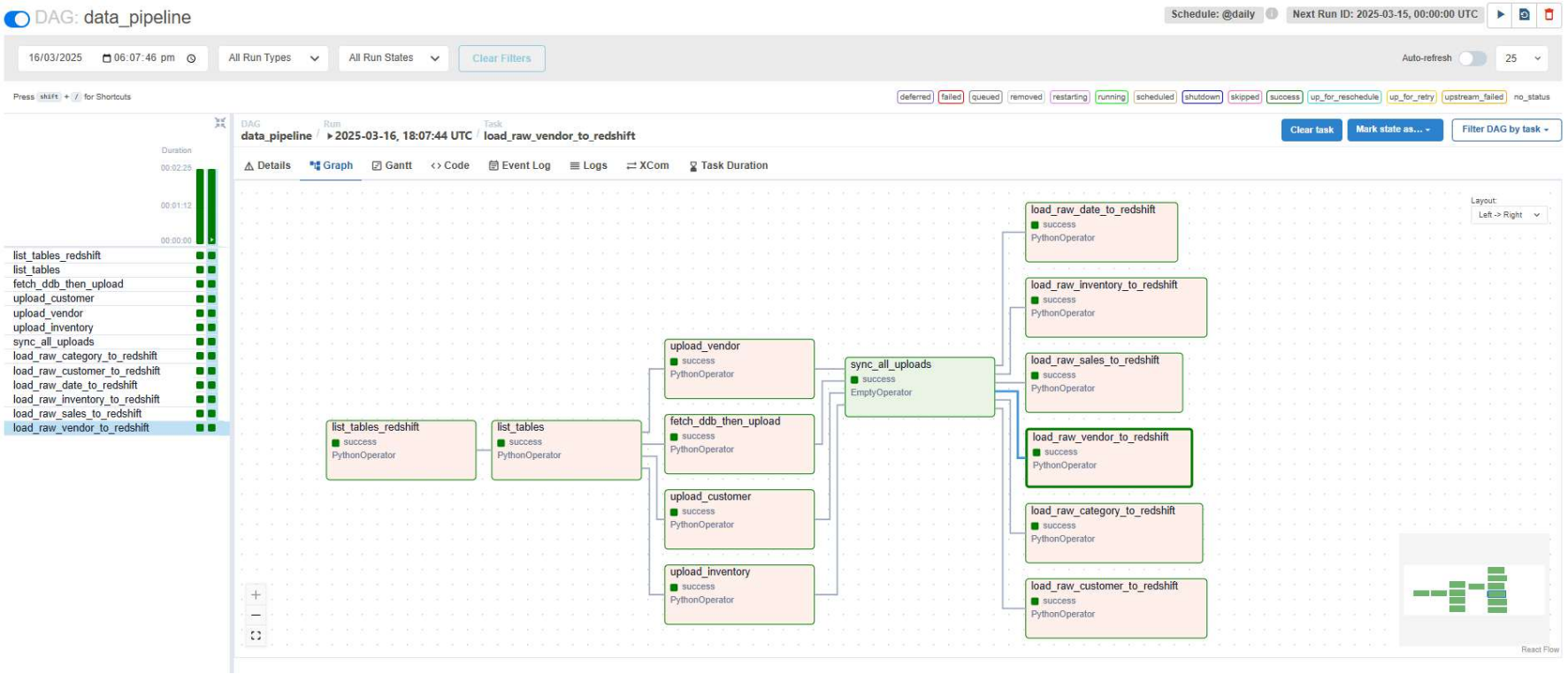
Here we will be discussing the workflow deployed in Airflow. The DAG Deployed does:

- Fetch from OLTP Data Source - Postgres and DynamoDB
- Dump to S3 zones - `raw`, `cleaned`, `work`, and `gold`
- Ingest Data from `gold` zone to Data Warehouse in Redshift

The DAG is a batch processing job that is scheduled to run daily.

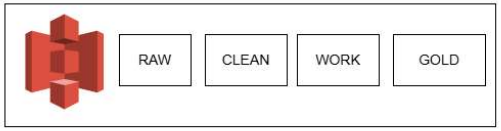
*DAG technical code definition found in `src\airflow\dags\*` directory





Data Lake

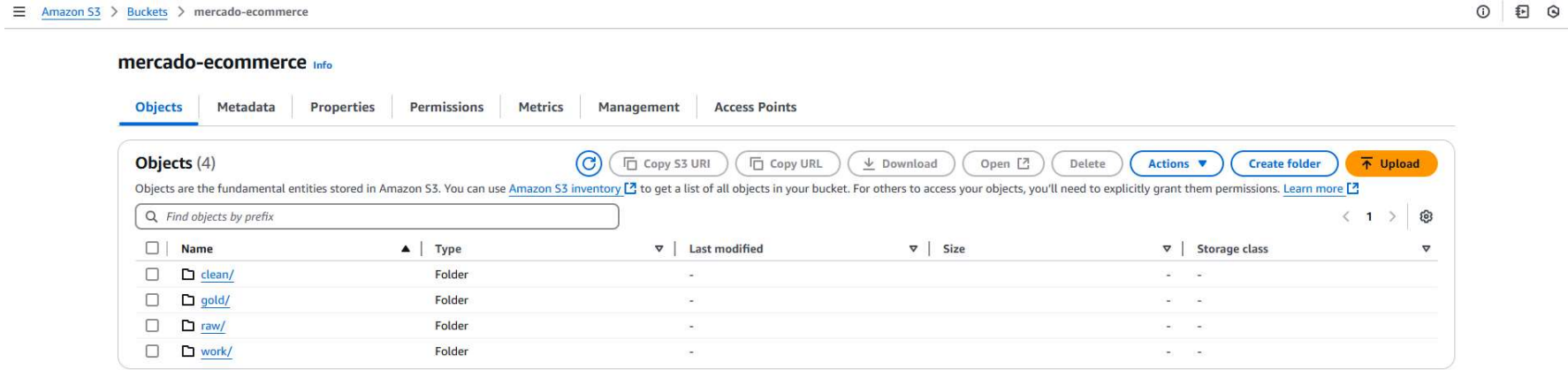
Storage Layer



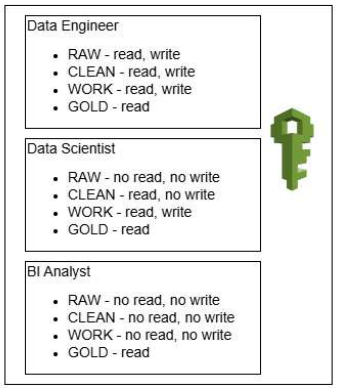
Storage Layer

The data lake storage layer is typically structured into four key zones: **raw, cleaned, work, and gold**, each serving a specific purpose in data processing. The **raw zone** acts as the landing area for unprocessed data, ingested from various sources in its original format, ensuring data fidelity and traceability. The **cleaned zone** contains data that has undergone basic transformations such as deduplication, format standardization, and schema validation, making it suitable for further processing. The **work zone** is a sandbox environment where analysts and data engineers perform transformations, enrichments, and exploratory analysis before finalizing datasets. Finally, the **gold zone** holds highly curated, business-ready datasets that are optimized for analytics, reporting, and machine learning applications, ensuring high data quality and governance.

S3 contents:



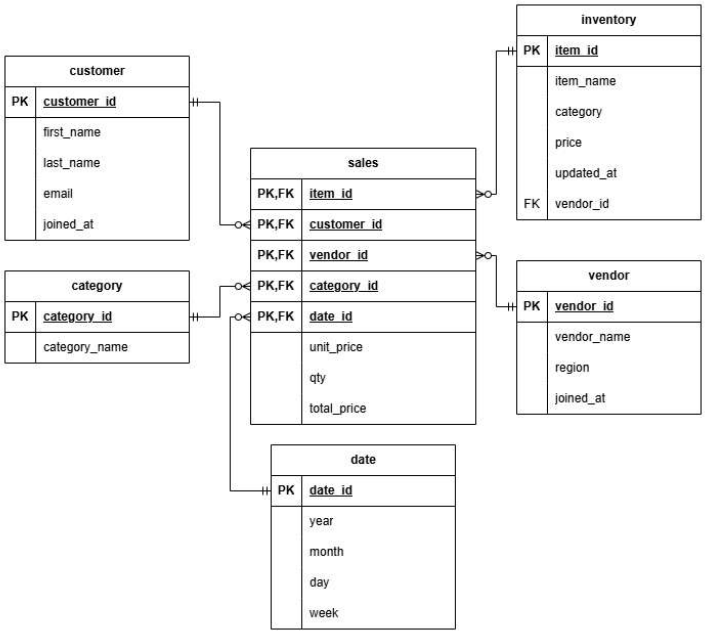
RBAC - Role-Based Access Management



RBAC

This **Role-Based Access Control (RBAC)** model is designed to enforce **data security, governance, and efficient collaboration** across different personas interacting with the data lake. **Data Engineers** require full access to ingest, clean, and prepare data, ensuring its quality and usability. **Data Scientists** need access to curated datasets while being able to experiment in the work zone without modifying raw or cleaned data. **BI Analysts** should only consume fully refined, business-ready data from the gold zone to maintain **data integrity and consistency** in reporting. This structure minimizes the risk of accidental data corruption while ensuring each role has the appropriate level of access for their tasks.

Data Warehouse Redshift



The Entity-Relationship Diagram (ERD) represents an e-commerce sales fact table with a transaction line item as the granularity. At the center of the schema is the **sales** fact table, which records transactional data, including `product_id`, `customer_id`, `date_id`, and `quantity_sold`. This fact table is connected to multiple dimension tables that provide contextual details for analysis. The **customer** dimension stores customer-related information, allowing insights into purchasing behavior. The **category** dimension classifies products, aiding in

sales performance analysis across different product groups. The **date** dimension enables time-based reporting and trend analysis. The **inventory** dimension tracks stock levels, ensuring supply chain efficiency. Lastly, the **vendor** dimension contains supplier details, supporting procurement and vendor performance evaluations. This star schema structure facilitates efficient querying and analytics for e-commerce sales data.

In [13]:
```
connection_string = f"postgresql://{OLAP_USER}:{OLAP_PASS}@{OLAP_HOST}:5439/{OLAP_DB_NAME}"
engine = redshift_connector.connect(
    host=OLAP_HOST,
    port=5439,
    database=OLAP_DB_NAME,
    user=OLAP_USER,
    password=OLAP_PASS
)
engine = create_engine(connection_string)
get_ipython().run_line_magic('sql', connection_string)
```

Connecting and switching to connection 'postgresql://vincent:***@samplecluster.cq68pg38qszs.us-east-1.redshift.amazonaws.com:5439/mercado_ecommerce'

*Create table definition found in `src\sql\*` directory

In [14]:
```
%sql \dt
```

Running query in 'postgresql://vincent:***@samplecluster.cq68pg38qszs.us-east-1.redshift.amazonaws.com:5439/mercado_ecommerce'

Out[14]:

schema	name	type	owner
public	category	table	vincent
public	customer	table	vincent
public	date	table	vincent
public	inventory	table	vincent
public	sales	table	vincent
public	vendor	table	vincent

In [15]:
```
table_list = [
    "category", "customer", "date", "inventory", "vendor", "sales"
]
```

In [16]:
```
for table in table_list:
    print(f"{table}")
    schema_query = f"columns --table {table}"
    display(get_ipython().run_line_magic('sqlcmd', schema_query))
```

category

name	type	nullable	default	autoincrement	comment
category_id	INTEGER	False	None	False	None
category_name	VARCHAR(100)	True	None	False	None

customer

name	type	nullable	default	autoincrement	comment
customer_id	INTEGER	False	None	False	None
first_name	VARCHAR(50)	True	None	False	None
last_name	VARCHAR(50)	True	None	False	None
email	VARCHAR(100)	True	None	False	None
joined_at	TIMESTAMP	True	None	False	None

date

name	type	nullable	default	autoincrement	comment
date_id	INTEGER	False	None	False	None
year	INTEGER	True	None	False	None
month	INTEGER	True	None	False	None
day	INTEGER	True	None	False	None

inventory

name	type	nullable	default	autoincrement	comment
item_id	INTEGER	False	None	False	None
item_name	VARCHAR(100)	True	None	False	None
category	INTEGER	True	None	False	None
price	NUMERIC(10, 2)	True	None	False	None
updated_at	TIMESTAMP	True	None	False	None
vendor_id	INTEGER	True	None	False	None

vendor

name	type	nullable	default	autoincrement	comment
vendor_id	INTEGER	False	None	False	None
vendor_name	VARCHAR(100)	True	None	False	None
region	VARCHAR(100)	True	None	False	None
joined_at	TIMESTAMP	True	None	False	None

sales

name	type	nullable	default	autoincrement	comment
item_id	INTEGER	True	None	False	None
customer_id	INTEGER	True	None	False	None
vendor_id	INTEGER	True	None	False	None
category_id	INTEGER	True	None	False	None
date_id	INTEGER	True	None	False	None
unit_price	NUMERIC(10, 2)	True	None	False	None
qty	INTEGER	True	None	False	None
total_price	NUMERIC(10, 2)	True	None	False	None

In [17]:
```
for table in table_list:
    print(f"{table}")
    display(pd.read_sql(f"select * from {table} limit 100", engine.raw_connection()))
```

category

	category_id	category_name
0	1	General
1	1	General

customer

	customer_id	first_name	last_name	email	joined_at
0	1	Leslie	Richards	kylerussell@example.com	2025-03-16 18:07:07.204516
1	2	Gabriela	Crawford	millerjessica@example.org	2025-03-16 18:07:07.215455
2	3	Melissa	Herrera	mendezdavid@example.org	2025-03-16 18:07:07.223133
...	...	...	...	...	...
17	8	Shawn	Stewart	qjordan@example.com	2025-03-16 18:07:07.269037
18	9	Susan	Lester	nancy49@example.net	2025-03-16 18:07:07.277155
19	10	Laura	Robinson	pthornton@example.org	2025-03-16 18:07:07.285038

20 rows × 5 columns

date

	date_id	year	month	day
0	20250316	2025	3	16
1	20250316	2025	3	16

inventory

	item_id	item_name	category	price	updated_at	vendor_id
0	1	experience	1	216.0	2025-03-16 18:07:07.336667	1
1	2	statement	1	49.0	2025-03-16 18:07:07.348005	1
2	3	list	1	283.0	2025-03-16 18:07:07.357084	1
...	...	...	...	...	...	...
97	48	growth	1	106.0	2025-03-16 18:07:07.758669	5
98	49	arm	1	155.0	2025-03-16 18:07:07.767366	5
99	50	range	1	17.0	2025-03-16 18:07:07.775252	5

100 rows × 6 columns

vendor

	vendor_id	vendor_name	region	joined_at
0	1	Hamilton, Horton and Ingram	TV	2025-03-16 18:07:07.295645
1	2	Hester-Ramirez	GY	2025-03-16 18:07:07.304975
2	3	Rogers, Santos and Flowers	TR	2025-03-16 18:07:07.313088
3	4	Lam-Smith	CH	2025-03-16 18:07:07.320201
4	5	Ali-Thompson	EC	2025-03-16 18:07:07.327773
5	1	Hamilton, Horton and Ingram	TV	2025-03-16 18:07:07.295645
6	2	Hester-Ramirez	GY	2025-03-16 18:07:07.304975
7	3	Rogers, Santos and Flowers	TR	2025-03-16 18:07:07.313088
8	4	Lam-Smith	CH	2025-03-16 18:07:07.320201
9	5	Ali-Thompson	EC	2025-03-16 18:07:07.327773

sales

	item_id	customer_id	vendor_id	category_id	date_id	unit_price	qty	total_price
0	5	8	5	1	20250316	298.0	89	26522.0
1	5	8	3	1	20250316	116.0	101	11716.0
2	4	7	1	1	20250316	322.0	304	97888.0
...	...	...	...	...	...	...	...	...
77	6	1	3	1	20250316	23.0	31	713.0
78	5	5	3	1	20250316	421.0	392	165032.0
79	5	5	3	1	20250316	18.0	161	2898.0

80 rows × 8 columns

Summary

In this end to end implementation of an e-commerce system we have run through important components

- API layer.
- OLTP layer with SQL and NoSQL.
- Workflow Management and Data Pipelines.
- Data Lake Zones and Role Based Access Management.
- Data Warehousing with Dimensional Modelling.

In [ ]: