How to build a data pipeline without data

Synthetic data generation and testing with Python

Ruan Pretorius | October 2023

About me

Hi there, I'm Ruan Pretorius 👋

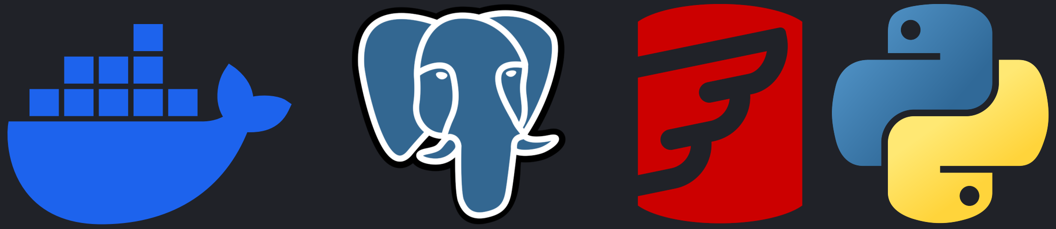
- 💻 I am a data scientist at *melio.ai*
- ☕ I turn coffee into data pipelines and AI
- 🔗 You can find me on GitHub *@ruankie*
- ✉️ Or contact me via email: *ruan@melio.ai*

🚏 Outline

- What are data pipelines and why do we need them?
- Challenges of building and testing data pipelines
- How to use synthetic data to test data pipelines
- Tools and methods to use when generating reliable synthetic data in Python
- Benefits and challenges of using synthetic data for testing data pipelines

Tools

Used in this demo



🛠️ What is a data pipeline?

- A data pipeline is a series of operations used to extract, load, transform, validate, or write data
- From various sources into a target file system, database, or data warehouse

🔍 Data pipelines without real data

- Sometimes, we may not have access to the real data that we want to process in our data pipeline.
- It could be:
  - Sensitive or confidential and can't be shared
  - Not yet collected or available
  - Too large or complex to handle for initial testing

🔍 Data pipelines without real data

- Without real data, it is challenging to:
  - Design the data model and schema
  - Develop the data extract, transform, and load (ETL) logic
  - Test the functionality and performance of the data pipeline
  - Debug and troubleshoot errors and issues

🧪 Synthetic data to test data pipelines

- Synthetic data is artificially generated data that mimics the characteristics and behavior of real data
- Synthetic data can help us to test our data pipelines by:
    - Providing realistic sample data
    - Allowing control of the size, shape, and distribution of the data
    - Enabling simulations of different scenarios and edge cases
    - Reducing the risk of exposing sensitive or confidential information

💃 Demo

- In this demo, I'll show you how you can create synthetic data
- Using a Python package called Faker
- And how to use Flyway to load the synthetic data into a Postgres database for repeatable deployments

💵 Our scenario

- Let's pretend we just started a new e-commerce website
- We have an idea of what kind of data we'll have for
  - Customers
  - Products
  - Transactions

📚 The data problem

- Now we want to start building different data pipelines and visualisations to see how well our business is doing
- We want our systems to work as soon as we get customers
- But we don't have data yet (so let's make some)

🪚 Import tools

- `SQLAlchemy` to create database objects

```python
from sqlalchemy import Column, Integer, String, DateTime
from sqlalchemy.orm import declarative_base
Base = declarative_base()
```

- `Faker` to generate synthetic data

```python
from faker import Faker
fake = Faker()
```

👥 Customer object

Class to store customer information

```
class Customer(Base):
  __tablename__ = "customers"
  id = Column(Integer, primary_key=True)
  name = Column(String(100))
  email = Column(String(100))
  phone = Column(String(25))
  address = Column(String(250))
  city = Column(String(100))
  country = Column(String(100))
```

👥 Customer data

Customer generator using `Faker` for synthetic data

```python
def generate_customer(id: int):
  customer = Customer(
      id=id,
      name=fake.name(),
      email=fake.email(),
      phone=fake.phone_number(),
      address=fake.street_address(),
      city=fake.city(),
      country=fake.country()
  )
  return customer
```

🍕 Product object

Class to store product information

```python
class Product(Base):
  __tablename__ = "products"
  id = Column(Integer, primary_key=True)
  name = Column(String(50))
  description = Column(String(200))
  category = Column(String(50))
  price = Column(Numeric(10, 2))
```

🍕 Product data

Product generator using `Faker` for synthetic data

```python
def generate_product(id: int):
  product = Product(
      id=id,
      name=fake.word(),
      description=fake.sentence(),
      category=fake.random_element(
        elements=("Electronics", "Fashion", "Books", "Games", "Sports", "Food")
      ),
      price=fake.pydecimal(left_digits=3, right_digits=2, positive=True)
  )
  return product
```

💻 Device object

Class to store device information

```
class Device(Base):
  __tablename__ = "devices"
  id = Column(Integer, primary_key=True)
  platform = Column(String(250))
  ipv4 = Column(String(50))
  macaddress = Column(String(50))
```

💻 Device data

Device generator using `Faker` for synthetic data

```python
def generate_device(id: int):
  device = Device(
      id=id,
      platform=fake.user_agent(),
      ipv4=fake.ipv4(),
      macaddress=fake.mac_address()
  )
  return device
```

💰 Transaction object

Class to store transaction information

```python
class Transaction(Base):
  __tablename__ = "transactions"
  id = Column(Integer, primary_key=True)
  date_time = Column(DateTime)
  customer_id = Column(Integer)
  product_id = Column(Integer)
  quantity = Column(Integer)
  device_id = Column(Integer)
  payment_method = Column(String(50))
```

💰 Transaction data

Transaction generator using `Faker` for synthetic data

```python
def generate_transaction(
  id: int,
  customers: list[Customer],
  products: list[Product],
  devices: list[Device]
):
  tr = Transaction(
      id=id,
      date_time=fake.date_between(start_date=START_DATE, end_date=END_DATE),
      customer_id=random.choice(customers).id,
      product_id=random.choice(products).id,
      quantity=fake.random_int(min=1, max=20),
      device_id=random.choice(devices).id,
      payment_method=fake.random_element(
        elements=("Credit Card", "EFT", "Bitcoin", "Reward Points")
      )
  )
  return tr
```

📊 Generate data

Use functions to generate synthetic data

```
customers = [generate_customer(i) for i in range(1000)]
products = [generate_product(i) for i in range(60)]
devices = [generate_device(i) for i in range(1000)]

transactions = [
  generate_transaction(i, customers, products, devices) for i in range(5000)
]
```

What are the benefits and challenges of using synthetic data for testing data pipelines?

- Some of the benefits of using synthetic data are:
  - It can speed up the development and testing process
  - It can increase the coverage and quality of testing
  - It can improve the scalability and reliability of the data pipeline
- Some of the challenges of using synthetic data are:
  - It may not capture all the nuances and variations of real data
  - It may introduce biases or errors in the synthetic data generation process
  - It may require additional effort and resources to create and maintain synthetic data

What are some best practices and tips for creating and using synthetic data effectively?

- Some of the best practices and tips are:
  - Define the scope and purpose of your synthetic data
  - Use existing tools and libraries to generate synthetic data
  - Validate and verify your synthetic data against your real data schema and business rules
  - Document your synthetic data generation process and code
  - Keep your synthetic data up-to-date with your real data changes

Summary

- In this talk, we learned how to build a data pipeline without real data using Python
- We discussed the challenges of building and testing data pipelines without real data
- We showed how we used synthetic data to test our data pipelines
- We demonstrated how we used Python packages such as Faker to generate realistic synthetic data
- We also showed how we used Flyway to load the synthetic data into a Postgres database

Thank you!

- I hope you enjoyed this talk and learned something new
- If you have any questions or feedback, please feel free to contact me
- You can find the code and slides for this talk on GitHub