02/06/25 - Redis + Python

Redis-py

Redis-py is the standard client for Python

- Maintained by the Redis Company itself
- GitHub Repo: https://github.com/redis/redis-py
- In your 4300 Conda Environment: `pip install redis`

Connecting to the Server

```
import redis
redis_client = redis.Redis(host='localhost',
          port=6379,
          db=2,
          decode_responses=True)
```

For your Docker deployment, `host` could be localhost or 127.0.0.1

`port` is the port mapping given when you created the container (probably the default 6379)

`db` is the database (0-15) you want to connect to

`decode_responses` → data comes back from server as bytes

- Setting this true converts them (decodes) to strings

Redis Command List

Full List: https://redis.io/docs/latest/commands/

- Use Filter to get to command for the particular data structure you're targeting (list, hash, set, etc.)

Redis.py Documentation: https://redis-py.readthedocs.io/en/stable/

String Commands

```
# r represents the Redis client object
r.set('clickCount:/abc', 0)
val = r.get('clickCount:/abc')
r.incr('clickCount:/abc')
ret_val = r.get('clickCount:/abc')
print(f'click count = {ret_val}')

redis_client.mset({'key1': 'val1',
                   'key2': 'val2',
                   'key3': 'val3'})
print(redis_client.mget('key1',
                        'key2',
                        'key3'))
# returns as list [‘val1’, ‘val2’, ‘val3’]



- set(), mset(), setex(), msetnx(), setnx()
- get(), mget(), getex(), getdel()
- incr(), decr(), incrby(), decrby()
- strlen(), append()
```

List Commands

```
# create list: key = 'names'
# values = ['mark', 'sam', 'nick']
redis_client.rpush('names',
                                     'mark', 'sam', 'nick')

# prints ['mark', 'sam', 'nick']
print(redis_client.lrange('names', 0, -1))

- lpush(), lpop(), lset(), lrem()
- rpush(), rpop()
- lrange(), llen(), lpos()
- Other commands include moving elements between lists,
    popping from multiple lists at the same time, etc.
```

Hash Commands

```
redis_client.hset('user-session:123',
    mapping={'first': 'Sam',
              'last': 'Uelle',
              'company': 'Redis',
              'age': 30
})

# prints:
# {'name': 'Sam', 'surname': 'Uelle', 'company': 'Redis',
# 'age': '30'}
print(redis_client.hgetall('user-session:123'))

- hset(), hget(), hgetall()
- hkeys()
- hdel(), hexists(), hlen(), hstrlen()
```

Redis Pipelines

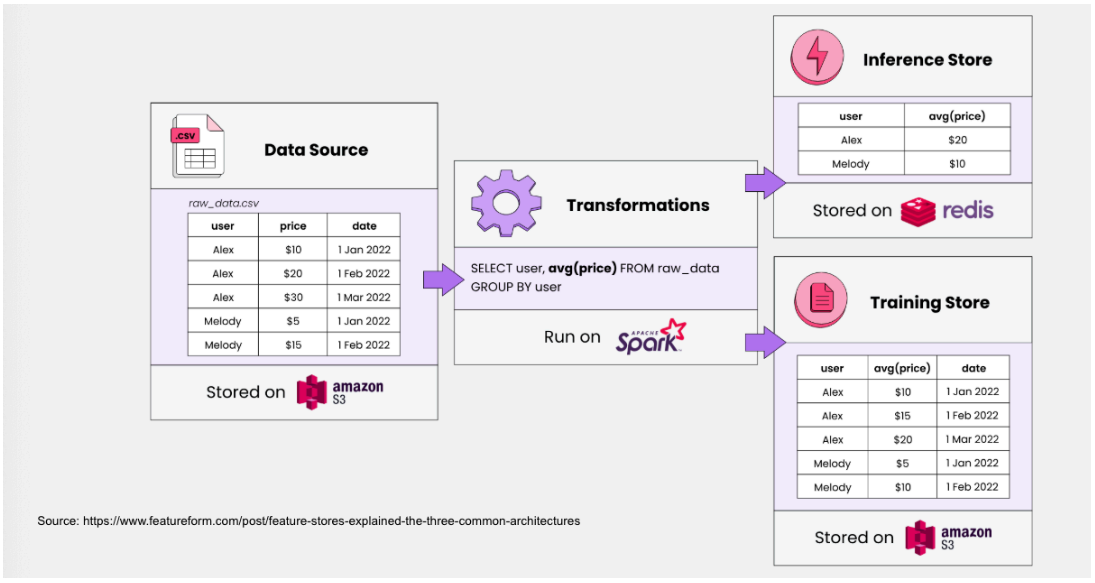
Helps avoid multiple related calls to the server → less network overhead

```
r = redis.Redis(decode_responses = True)
pipe = r.pipeline()

for i in range(5):
    pipe.set(f"seat:{i}", f"#{i}")

set_5_result = pipe.execute()
print(set_5_result)  # >>> [True, True, True, True, True]
```

```
pipe = r.pipeline()

# "Chain" pipeline commands together
get_3_result = pipe.get("seat:0").get("seat:3").get("seat:4").execute()
print(get_3_result)  # >>> ['#0', '#3', '#4']
```

Redis in Context

Redis in ML - Simplified Example



user	price	date
Alex	$10	1 Jan 2022
Alex	$20	1 Feb 2022
Alex	$30	1 Mar 2022
Melody	$5	1 Jan 2022
Melody	$15	1 Feb 2022

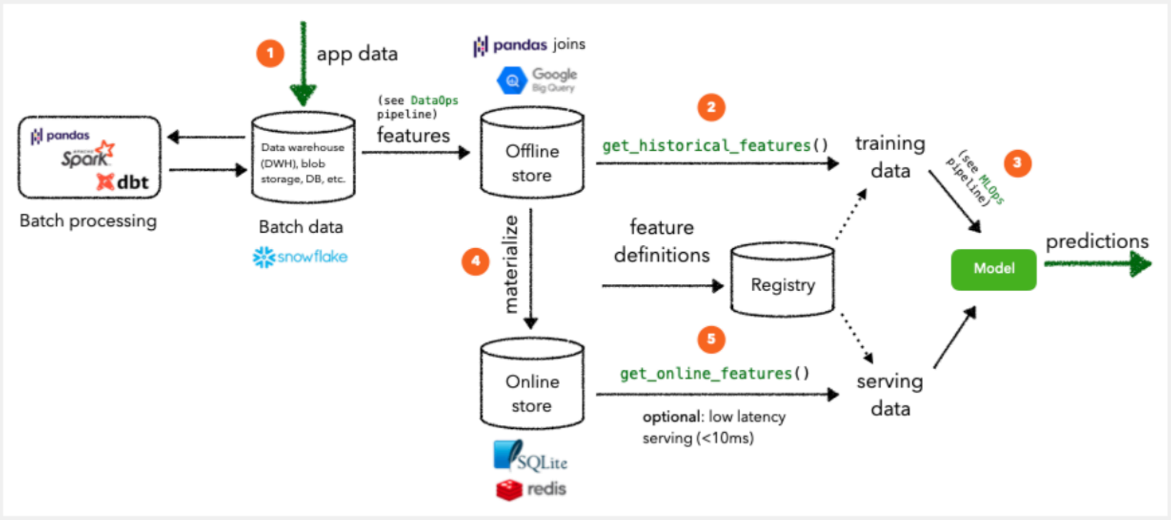
```
SELECT user, avg(price) FROM raw_data
GROUP BY user
```

user	avg(price)
Alex	$20
Melody	$10

user	avg(price)	date
Alex	$10	1 Jan 2022
Alex	$15	1 Feb 2022
Alex	$20	1 Mar 2022
Melody	$5	1 Jan 2022
Melody	$10	1 Feb 2022

Source: https://www.featureform.com/post/feature-stores-explained-the-three-common-architectures

Redis in DS/ML



Source: https://madewithml.com/courses/mlops/feature-store/