redis_exploratio

May 6, 2025

[1]:
```
!pip install redis
import redis

# Connect to Redis
r = redis.Redis(host='redis', port=6379, decode_responses=True)


r.ping()
```

```
Collecting redis
  Downloading redis-6.0.0-py3-none-any.whl.metadata (10 kB)
Downloading redis-6.0.0-py3-none-any.whl (268 kB)
                        268.9/268.9 kB
964.5 kB/s eta 0:00:00a 0:00:01
Installing collected packages: redis
Successfully installed redis-6.0.0
```

[1]:
```
True
```

[2]:
```
# Get all actor keys
actor_keys = r.keys("actor:*")
num_actors = len(actor_keys)
print(f"Number of actors: {num_actors}")

# Get all movie keys
movie_keys = r.keys("movie:*")
num_movies = len(movie_keys)
print(f"Number of movies: {num_movies}")
```

```
Number of actors: 1319
Number of movies: 922
```

[13]:
```
actors_born_before_1980 = []
count = 0

for actor_key in actor_keys:
    actor_data = r.hgetall(actor_key)
```

```
    if 'date_of_birth' in actor_data and int(actor_data['date_of_birth']) <␣
↪1980:
        print(f"{actor_data.get('first_name', '')} {actor_data.get('last_name',␣
↪'')} (Born: {actor_data['date_of_birth']})")
        count += 1
        if count >= 5:
            break
```

```
Vin Diesel (Born: 1967)
Tom Hanks (Born: 1956)
Pascale Hutton (Born: 1979)
Deanna Dunagan (Born: 1940)
Stephanie Faracy (Born: 1952)
```

[14]:
```
target_movie_title = "The Imitation Game"
movie_found = False

for movie_key in movie_keys:
    movie_data = r.hgetall(movie_key)
    if movie_data.get('title') == target_movie_title:
        genre = movie_data.get('genre', 'N/A')
        rating = movie_data.get('rating', 'N/A')
        print(f"Movie: {target_movie_title}")
        print(f"- Genre: {genre}")
        print(f"- Rating: {rating}")
        movie_found = True
        break

if not movie_found:
    print(f"Movie '{target_movie_title}' not found.")
```

```
Movie: The Imitation Game
- Genre: Biography
- Rating: 8.5
```

[15]:
```
all_movies_with_ratings = []


for movie_key in movie_keys:
    movie_data = r.hgetall(movie_key)
    title = movie_data.get('title')
    rating_str = movie_data.get('rating')
    if title and rating_str:
        rating = float(rating_str)
        all_movies_with_ratings.append({'title': title, 'rating': rating})
```

```
sorted_movies = sorted(all_movies_with_ratings, key=lambda x: x['rating'],␣
↪reverse=True)

print("Top 5 highest-rated movies:")
for i, movie in enumerate(sorted_movies[:5]):
    print(f"{i+1}. {movie['title']} (Rating: {movie['rating']})")
```

```
Top 5 highest-rated movies:
1. Boy 9 (Rating: 9.4)
2. Vegas (doc) (Rating: 9.4)
3. The Shawshank Redemption (Rating: 9.3)
4. Ween Live in Chicago (Rating: 9.2)
5. Over Canada: An Aerial Adventure (Rating: 9.1)
```

[6]:
```
count_movies_above_7_5 = 0

for movie_key in movie_keys:
    rating_str = r.hget(movie_key, 'rating')
    if rating_str:
        try:
            rating = float(rating_str)
            if rating > 7.5:
                count_movies_above_7_5 += 1
        except ValueError:
            print(f"Warning: Could not parse rating for movie key {movie_key}")


print(f"Number of movies with a rating above 7.5: {count_movies_above_7_5}")
```

```
Number of movies with a rating above 7.5: 183
```

[20]:
```
target_movie_title_update = "The Imitation Game"
new_rating = "8.5"

for movie_key in movie_keys:
    if r.hget(movie_key, 'title') == target_movie_title_update:
        r.hset(movie_key, 'rating', new_rating)
        print(f"Rating for '{target_movie_title_update}' (key: {movie_key})␣
↪updated to {new_rating}.")

    # Verify the update (optional, but good practice)
        updated_data = r.hgetall(movie_key)
        print(f"Verified new rating: {updated_data.get('rating')}")
        break
```

```
Rating for 'The Imitation Game' (key: movie:5) updated to 8.5.
```

```
Verified new rating: 8.5
```

[21]:
```python
max_id = 0

for key in actor_keys:
    try:
        current_id = int(key.split(':')[1])
        if current_id > max_id:
            max_id = current_id
    except (IndexError, ValueError):
        print(f"Warning: Could not parse ID from actor key: {key}")

new_actor_id = max_id + 1
new_actor_key = f"actor:{new_actor_id}"


new_actor_data = {
    "first_name": "Zendaya",
    "last_name": "",
    "date_of_birth": 1996
}

r.hset(new_actor_key, mapping=new_actor_data)

print(f"Added new actor: Zendaya, with key {new_actor_key}")

# Verify
retrieved_zendaya = r.hgetall(new_actor_key)
print(f"Verified data for {new_actor_key}: {retrieved_zendaya}")
```

```
Added new actor: Zendaya, with key actor:1320
Verified data for actor:1320: {'first_name': 'Zendaya', 'last_name': '',
'date_of_birth': '1996'}
```

[9]:
```python
target_movie_title_delete = "The Room"
movie_key_to_delete = None

for movie_key in movie_keys:
    if r.hget(movie_key, 'title') == target_movie_title_delete:
        movie_key_to_delete = movie_key
        break

if movie_key_to_delete:
    result = r.delete(movie_key_to_delete)
    if result == 1:
        print(f"Movie '{target_movie_title_delete}' (key:␣
 ↪{movie_key_to_delete}) has been deleted.")
```

```
        # Verify (optional)
        if r.exists(movie_key_to_delete):
            print(f"Verification FAILED: Key {movie_key_to_delete} still exists.
 ↪")
        else:
            print(f"Verification PASSED: Key {movie_key_to_delete} no longer␣
 ↪exists.")
    else:
        print(f"Movie '{target_movie_title_delete}' (key:␣
 ↪{movie_key_to_delete}) was targeted for deletion, but r.delete returned␣
 ↪{result}.")
else:
    print(f"Movie '{target_movie_title_delete}' not found. Could not delete.")
```

```
Movie 'The Room' not found. Could not delete.
```