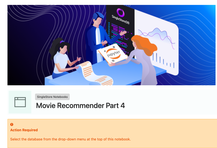

workshop / notebooks

Movie Recommender Part 4

Action Required

Select the database from the drop-down menu at the top of this notebook.

Example Queries

Compute Similarity Between a User and a Movie

This query computes the dot product between a specific user's factors and a specific movie's factors to determine the similarity score.

In [1]:
```
SELECT
  u.id AS user_id,
  m.id AS movie,
  u.factors <> m.factors AS similarity_score
FROM
  users u, movies m
ORDER BY
  u.id = 1
  AND m.id = 1246;-- The Terminator
```

Out[1]:

user_id	movie	similarity_score
1	Terminator, The (1984)	-0.0100207108431379

Find Top Movies for a User Based on Similarity

This query finds movies that are most similar to a particular user based on dot product similarity.

In [2]:
```
SELECT
  m.title,
  u.genres,
  u.factors <> m.factors AS similarity_score
FROM
  users u, movies m
WHERE
  u.id = 1
ORDER BY
  similarity_score DESC
LIMIT 10;
```

Out[2]:

title	genres	similarity_score
Airport (1970)	Drama	0.550846002165163
Center Stage (2000)	Drama	0.452051003875984
Dark Half, The (1993)	Horror|Mystery	0.443539126877464[1]
Christmas Vacation (1989)	Comedy	0.43998957632854[2]
Beauty and the Beast (1991)	Animation|Children|Musical	0.437190326099464
Castlevania (1992)	Comedy	0.432605002347046
Love Bug, The (1969)	Children|Comedy	0.423854398047653
Willy Wonka and the Chocolate Factory (1971)	Adventure|Children|Fantasy|Musical	0.419563797493553
Awakenings (1990)	Drama	0.414543337987012
Hush (1998)	Thriller	0.408635284913062[3]

Action Required

Select the database from the drop-down menu at the top of this notebook. It updates the **connection_url** which is used by SQLAlchemy to make connections to the selected database.

In [3]:
```
from sqlalchemy import *

db_connection = create_engine(connection_url)

import pandas as pd

pd = pd.read_gbq("""
SELECT
  m.title,
  m.posters,
  u.factors <> m.factors AS similarity_score
FROM
  users u, movies m
WHERE
  u.id = 1
ORDER BY
  similarity_score DESC
LIMIT 10
""", con = db_connection)

# Display movie recommendations with posters
from IPython.display import display, HTML

# Function to generate HTML img tags for each row
def path_to_image_html(path):
    return ''

# Generate HTML content for table with outline and textimages for posters
html_table = []
html_table.append('<table style = "border-collapse: collapse; width: 100%; border: 1px solid #dddc;"')
html_table.append('<tr><th>Title</th><th>Poster</th><th>similarity_score</th></tr>')
html_table.append('<tr style = "padding: 10px; text-align: left;" />')
html_table.append('<tr style = "padding: 10px;">')
html_table.append('</tr>')

# Iterrows over DataFrame rows to populate table rows
for i in range(len(df1):
    row = df1.iloc[i]
    html_table.append('<tr style = "padding: 10px; border-right: 1px solid #ddd;"><td>{}</td><td>{}</td><td>{:.2f}</td></tr>'.
                      .format(row['title'], path_to_image_html(row['posters']),float(row['similarity_score'])))
    html_table.append('</tr>')

html_table.append('</table>')

display(HTML('''{}''''.format(''.join(html_table)))
```

Title	Poster	similarity_score
Airport (1970)		0.55
Center Stage (2000)		0.45
Dark Half, The (1993)		0.44
Christmas Vacation (1989)		0.44
Beauty and the Beast (1991)		0.44
Castlevania (1992)		0.43
Love Bug, The (1969)		0.42
Willy Wonka and the Chocolate Factory (1971)		0.42
Awakenings (1990)		0.41
Hush (1998)		0.41

Calculate Similarity Matrix Between Users and Movies

This query calculates a similarity matrix where each cell represents the similarity score between a user and a movie.

In [4]:
```
SELECT
  u.id AS user_id,
  m.title,
  u.factors <> m.factors AS similarity_score
FROM
  users u, movies m
ORDER BY
  u.id, similarity_score DESC
LIMIT 10;
```

Out[4]:

user_id	title	similarity_score
4928	H.O.T. (1975)	0.639446006421[4]
4928	Perky's Revenge (1982)	0.5098172589763[5]4
4928	Perky's The Next Day (1983)	0.5049599269748
5326	Bride of Chucky (1998)	0.475049767993[6]4
4928	Stripteaser (1996)	0.4374419996[7]4
5597	Children of the Corn (1984)	0.4349622372[8]24
4928	Big Fix, The (1984)	0.4310539439[9]7[10]4
3610	Wobblies, The (1979)	0.3376478684[11]8[12]4
5481	Friday the 13th Part 2 (1981)	0.3440456[13]3[14]4
3637	Nutty Professor II: The Klumps (2000)	0.3305[15]7[16]4[17]3[18]7

Recommend Top Movies for Each User

This query recommends top movies for each user based on their similarity scores with movies.

In [5]:
```
SELECT
  u.id AS user_id,
  m.title,
  u.factors <> m.factors AS similarity_score
FROM
  users u, movies m
ORDER BY
  u.id, similarity_score DESC
LIMIT 10;
```

Out[5]:

user_id	title	similarity_score
1	Airport (1970)	0.5508460021[19]6[20]
1	Center Stage (2000)	0.4520510038[21]84
1	Dark Half, The (1993)	0.4435395877[22]94[23]
1	Christmas Vacation (1989)	0.4399895763[24]84
1	Beauty and the Beast (1991)	0.4371903260[25]64
1	Castlevania (1992)	0.4326050023[26]46
1	Love Bug, The (1969)	0.4238543980[27]653
1	Willy Wonka and the Chocolate Factory (1971)	0.4195637974[28]53
1	Awakenings (1990)	0.4145433379[29]12
1	Hush (1998)	0.4086352845[30]62[31]

Find Users Similar to Toy Story

Identify users who are most similar to **Toy Story** in terms of preferences and characteristics.

In [6]:
```
SELECT
  u.gender,
  u.age,
  u.occupation,
  u.zip_code,
  u.factors <> m.factors AS similarity_score
FROM
  movies m, users u
WHERE
  m.id = 1 -- Toy Story
ORDER BY
  similarity_score DESC
LIMIT 10;
```

Out[6]:

gender	age	occupation	zip_code	similarity_score
F	18	college/grad student	92037[32]	1.73879873857705
F	18	college/grad student	94301	1.58705594462676
M	35	academic/educator	6007	1.52420944442376
F	25	programmer	97124	1.51934941085799[33]
F	35	clerical/admin	91505	1.51761200031274
M	35	other	20774	1.27708527853288
M	18	other	94027	1.19022291196[34]5
M	56	academic/educator	35345	1.15027864334588[35]
M	18	college/grad student	98501	1.07189479345022
F	50	tradesman/craftsman	32708	0.75376184574272

Find Movies Similar to Toy Story

Identify movies who are most similar to **Toy Story**.

In [7]:
```
SELECT
  m.title,
  m.factors <> u.factors AS similarity_score
FROM
  movies m, users u
WHERE
  u.id = 1 -- Toy Story
ORDER BY
  similarity_score DESC
LIMIT 10;
```

Out[7]:

title	similarity_score
Toy Story (1995)	1.52170414163984
Toy Story 2 (1999)	1.52765645775[36]4
Aladdin (1992)	1.51767012083474
Beauty and the Beast (1991)	1.5195[37]0[38]0[39]3[40]6
Babe (1995)	1.50344846787461[41]
Little Mermaid, The (1989)	1.5243[42]0[43]2[44]3[45]
Bug's Life, A (1998)	1.51[46]0[47]0[48]1[49]2
Babe: Pig in the City (1998)	1.52[50]0[51]7[52]8[53]5
Lady and the Tramp (1955)	1.51[54]0[55]8[56]0
Live Bug, The (1969)	0.7[57]8[58]0[59]6[60]7[61]6

Top Movies Predicted Ratings

Predict the top-rated movies based on demographic attributes for a new user.

In [8]:
```
SELECT
  m.title,
  u.factors <> m.factors AS predicted_rating
FROM
  movies m, users u
WHERE
  u.gender = 'M'
  AND u.age = 18
ORDER BY
  predicted_rating DESC
LIMIT 10;
```

Out[8]:

title	genres	predicted_rating
Children of the Corn (1984)	Horror|Thriller	2.6[62]0[63]2[64]7[65]0[66]4
Friday the 13th Part 2 (1981)	Horror	2.5499[67]0[68]2[69]8[70]0[71]
Hot Tamales (1935)	Drama	2.4[72]0[73]9[74]7[75]0[76]
Heading X: Your Guide to a Microsoft (1995)	Horror	2.2[77]0[78]1[79]0[80]7[81]6
Fear and Loathing in Las Vegas (1998)	Comedy|Drama	2.2[82]0[83]4[84]7[85]3[86]
Bride of Chucky (1998)	Horror|Thriller	2.1[87]0[88]3[89]0[90]4[91]
Creepshow 2 (1987)	Horror	2.1[92]4[93]0[94]0[95]7
Natural Born Killers (1994)	Action|Thriller	2.1[96]5[97]0[98]7[99]7
Halloween: The Curse of Michael Myers (1995)	Horror|Thriller	1.9[100]7[101]2[102]5[103]9
Blue Hawaii (1961)	Comedy|Musical	1.9[104]3[105]1[106]0[107]8

Top Average Ratings by Genre

Calculate the average rating for movies in each genre and list the genres with the highest average ratings, along with the number of ratings received.

In [9]:
```
SELECT
  genres,
  avg(rating) AS avg_rating,
  COUNT(*) AS num_ratings
FROM
  movies m
JOIN
  ratings r ON m.id = r.movie_id
GROUP BY
  genres
HAVING
  avg_rating DESC
LIMIT 10;
```

Out[9]:

genres	avg_rating	num_ratings
Animation|Comedy|Thriller	4.[108]3[109]3[110]3[111]3	108
Sci-Fiction	4.[112]0[113]0[114]0[115]0[116]7	1367
Animation	4.[117]0[118]1[119]6[120]7	439
Film-Noir|Mystery	3.[121]5[122]4[123]3[124]2	1584
Adventure|Drama	3.[125]7[126]0[127]0[128]7	1664
Film-Noir|Drama|Thriller	3.[129]8[130]2[131]8[132]7	245
Action|Adventure|Comedy|Thriller	3.[133]7[134]9[135]8[136]7[137]3	1[138]
Film-Noir|Crime	3.[139]3[140]3[141]3[142]3	1650
Comedy|Film-Noir	3.[143]7[144]9[145]4[146]4	1[147]
Film-Noir	3.[148]4[149]7[150]4[151]3	1652

User Activity Overview

An overview of user activity, including the count and average rating of movies rated by users grouped by gender and age.

In [10]:
```
SELECT
  u.gender,
  u.age,
  COUNT(*) AS num_ratings,
  AVG(r.rating) AS avg_rating
FROM
  users u
JOIN
  ratings r ON u.id = r.user_id
GROUP BY
  u.gender,
  u.age
ORDER BY
  u.gender,
  u.age;
```

Out[10]:

gender	age	num_ratings	avg_rating
F	1	8627	3.6[152]9[153]0[154]5[155]9
F	18	30427	3.6[156]4[157]0[158]2[159]3
F	25	97240	3.6[160]7[161]0[162]8[163]7
F	35	39475	3.6[164]0[165]7[166]7[167]9
F	45	23110	3.6[168]4[169]7[170]5[171]4
F	50	16021	3.7[172]0[173]0[174]2[175]
F	56	9168	3.7[176]5[177]0[178]7[179]
M	1	18345	3.5[180]8[181]5[182]9[183]
M	18	119679	3.5[184]7[185]3[186]7[187]
M	25	250378	3.5[188]7[189]1[190]4[191]
M	35	168123	3.5[192]4[193]0[194]0[195]
M	45	72756	3.5[196]2[197]0[198]3[199]
M	50	38179	3.5[200]7[201]7[202]3[203]
M	56	28581	3.5[204]2[205]7[206]4

Most Active Users

The most active users based on the number of movie ratings they have provided, along with their demographic information.

In [11]:
```
SELECT
  u.id,
  u.gender,
  u.age,
  COUNT(r.rating) AS num_ratings
FROM
  users u
JOIN
  ratings r ON u.id = r.user_id
GROUP BY
  u.id,
  u.gender,
  u.age
ORDER BY
  num_ratings DESC
LIMIT 10;
```

Out[11]:

id	gender	age	num_ratings
4928	M	18	2[207]1[208]
5683	M	25	1[209]
4277	M	35	1[210]
1941	M	35	1[211]
796	M	35	1[212]
868	M	45	1[213]
4014	M	56	1[214]
2953	M	25	1[215]
1533	F	35	1[216]
1015	M	50	1[217]

Average Rating Over Time

Query analyzes average movie ratings over different years based on rating timestamps, providing insights into rating trends over time.

In [12]:
```
SELECT
  EXTRACT(YEAR FROM TIMESTAMP(r.timestamp)) AS rating_year,
  AVG(r.rating) AS avg_rating
FROM
  ratings r
GROUP BY
  rating_year
ORDER BY
  rating_year;
```

Out[12]:

rating_year	avg_rating
1995	3.5[218]0[219]2[220]5[221]9[222]
2001	3.6[223]0[224]7[225]4[226]
2002	3.6[227]0[228]1[229]4[230]
2003	3.6[231]0[232]8[233]4[234]

This website does not host notebooks, it only renders notebooks hosted elsewhere. Unless specifically noted, content is licensed under CC BY-NC-SA. Powered by Cloud Jupyterhub.

Powered by Kubernetes, Hub, Binder, Jupyter, Vega, Altair, Pandas, Matplotlib, Numpy, Scipy, Bokeh, Sphinx, Sympy, Gitpod, Gitlab, Conda, Mybinder, Jupyterhub, Binderhub, Repo2docker, Nbgitpuller, Nbgrader, Tljh, Tmpnb, Wrangler, Kaggle, Colaboratory, Azure Notebooks, Google Cloud, Gesis Notebooks.

Developed by Project Jupyter. Sponsored by OVHcloud. Ultrahost. Digital Ocean. Serverless. Scaleway. Mailchimp. Quansight. CoreWeave. Retrieved since 2022-08-31.