- Copy all the input and output files from the “TF_IDF (Python)” folder to your Neo4j project import folder.
- You can find the files needed for Neo4j in the “4. Recommendation (Neo4j)/import” folder.
- You can find the findal results of different output in the xlsx files in the “4. Recommendation (Neo4j)/results” folder.

### Loading Data

1. Load user data in “Users” nodes.

```
LOAD CSV WITH HEADERS FROM 'file:/user.csv' AS row
FIELDTERMINATOR '\t'
CREATE (:Users {user_id: row.user_id, cnt: row.cnt, attr: row.attr})
```

2. Load book data “Books” nodes.

```
LOAD CSV WITH HEADERS FROM 'file:/book.csv' AS row
FIELDTERMINATOR '\t'
CREATE (:Books {book_id:row.book_id, authors:row.authors, year:row.year, title:row.title,
language:row.language})
```

3. Load rating data in “Rated” relationship. [Users-Rated->Books]

```
LOAD CSV WITH HEADERS FROM 'file:/rating.csv' AS row
FIELDTERMINATOR '\t'
MATCH (u:Users {user_id: row.user_id})
MATCH (b:Books {book_id: row.book_id})
MERGE (u)-[r:RATED]->(b)
ON CREATE SET r.rating = toInteger(row.rating)
```

4. Load IDF values for all book features in “IDF_title, IDF_authors, IDF_language, IDF_year” nodes.

```
LOAD CSV WITH HEADERS FROM 'file:/IDF_title.csv' AS row
FIELDTERMINATOR '\t'
CREATE (:IDF_title {word: row.word, IDF: row.IDF_val})

LOAD CSV WITH HEADERS FROM 'file:/IDF_authors.csv' AS row
FIELDTERMINATOR '\t'
CREATE (:IDF_authors {author: row.author, IDF: row.IDF_val})
```

```
LOAD CSV WITH HEADERS FROM 'file:/IDF_language.csv' AS row
FIELDTERMINATOR '\t'
CREATE (:IDF_language {language: row.language, IDF: row.IDF_val})

LOAD CSV WITH HEADERS FROM 'file:/IDF_year.csv' AS row
FIELDTERMINATOR '\t'
CREATE (:IDF_year {year: row.year, IDF: row.IDF_val})
```

5. Load TF values for all book features in “TF_title, TF_authors, TF_language, TF_year” relationship [Books-TF…->IDF…].

```
LOAD CSV WITH HEADERS FROM 'file:/TF_title.csv' AS row
FIELDTERMINATOR '\t'
MATCH (b:Books {book_id: row.book_id})
MATCH (i:IDF_title {word: row.word})
MERGE (b)-[v:TF_title]->(i)
ON CREATE SET v.val = toFloat(row.TF_title_val)

LOAD CSV WITH HEADERS FROM 'file:/TF_authors.csv' AS row
FIELDTERMINATOR '\t'
MATCH (b:Books {book_id: row.book_id})
MATCH (i:IDF_authors {author: row.author})
MERGE (b)-[v:TF_authors]->(i)
ON CREATE SET v.val = toFloat(row.TF_author_val)

LOAD CSV WITH HEADERS FROM 'file:/TF_language.csv' AS row
FIELDTERMINATOR '\t'
MATCH (b:Books {book_id: row.book_id})
MATCH (i:IDF_language {language: row.language})
MERGE (b)-[v:TF_language]->(i)
ON CREATE SET v.val = toFloat(row.TF_language_val)

LOAD CSV WITH HEADERS FROM 'file:/TF_year.csv' AS row
FIELDTERMINATOR '\t'
MATCH (b:Books {book_id: row.book_id})
MATCH (i:IDF_year {year: row.year})
MERGE (b)-[v:TF_year]->(i)
ON CREATE SET v.val = toFloat(row.TF_year_val)
```

6. Load User profiles for all book features in “UA_title, UA_authors, UA_language, UA_year” relationship [Users->UA… ->IDF…] .

```
LOAD CSV WITH HEADERS FROM 'file:/UA_title.csv' AS row
FIELDTERMINATOR '\t'
MATCH (u:Users {user_id: row.user_id})
MATCH (i:IDF_title {word: row.word})
MERGE (u)-[v:UA_title]->(i)
ON CREATE SET v.val = toFloat(row.UA_title_val)

LOAD CSV WITH HEADERS FROM 'file:/UA_authors.csv' AS row
FIELDTERMINATOR '\t'
MATCH (u:Users {user_id: row.user_id})
MATCH (i:IDF_authors {author: row.author})
MERGE (u)-[v:UA_authors]->(i)
ON CREATE SET v.val = toFloat(row.UA_author_val)

LOAD CSV WITH HEADERS FROM 'file:/UA_language.csv' AS row
FIELDTERMINATOR '\t'
MATCH (u:Users {user_id: row.user_id})
MATCH (i:IDF_language {language: row.language})
MERGE (u)-[v:UA_language]->(i)
ON CREATE SET v.val = toFloat(row.UA_language_val)

LOAD CSV WITH HEADERS FROM 'file:/UA_year.csv' AS row
FIELDTERMINATOR '\t'
MATCH (u:Users {user_id: row.user_id})
MATCH (i:IDF_year {year: row.year})
MERGE (u)-[v:UA_year]->(i)
ON CREATE SET v.val = toFloat(row.UA_year_val)
```

### Recommend Books for Registry to the couple ###

1. Store average rating by each user of type ‘Couple’ and ‘Married’ [Needed for Pearson Similarity] in “avg_rating” property of Users

```
MATCH (u:Users)-[r:RATED]->(b:Books)
WHERE u.attr = 'M' OR u.attr = 'C'
WITH u, toFloat(SUM(r.rating))/COUNT(r) as avg_rating
SET u.avg_rating = avg_rating
```

2. Calculate Cosine, Pearson, & MeanSquare Similarity of each user of type ‘Couple’ to ‘Married’ in “Similarity” relationship [Users(c) – Similarity – Users(r)].
   And put the similarity values in “consine”, “pearson”, “meansquare” properties in the “Similarity” relationship.

```
MATCH (u1:Users{attr:'C'})-[x:RATED]->(b:Books)<-[y:RATED]-(u2:Users{attr:'M'})
WITH u1, u2,
COUNT(b)/SUM((x.rating-y.rating)^2) as MeanSquare,
SUM(x.rating * y.rating) AS xy,
SQRT(SUM(x.rating^2)) as sqrtx2,
SQRT(SUM(y.rating^2)) as sqrty2,
SUM((x.rating-u1.avg_rating) * (y.rating-u2.avg_rating)) AS xx_yy_,
SQRT(SUM((x.rating-u1.avg_rating)^2)) as sqrtxx_2,
SQRT(SUM((y.rating-u2.avg_rating)^2)) as sqrtyy_2
MERGE (u1)-[s:SIMILARITY]-(u2)
ON CREATE SET s.cosine = xy / (sqrtx2*sqrty2),
s.pearson = CASE WHEN (sqrtxx_2*sqrtyy_2)=0 THEN 0 ELSE xx_yy_ / (sqrtxx_2*sqrtyy_2) END,
s.meansquare = MeanSquare
```

3. Calculate SImilarity of Users of type ‘Married’ with the both Users of type ‘Couple’ (as a sum) in “cosine”, “pearson”, “meansquare” property of Users of type ‘Married’

```
MATCH (c1:Users{user_id:'12874'})-[s1:SIMILARITY]-(u1:Users{attr:'M'})
MATCH (c2:Users{user_id:'30944'})-[s2:SIMILARITY]-(u2:Users{attr:'M'})
WHERE u1.user_id=u2.user_id
WITH u1, (s1.cosine+s2.cosine) AS cosine,
(s1.pearson+s2.pearson) AS pearson,
(s1.meansquare+s2.meansquare) AS meansquare
SET u1.cosine=cosine,
u1.pearson=pearson,
u1.meansquare=meansquare
```

4. Calculate 100 highest similar Books for the Users of type ‘Couple’ calculated by the User Similarity of Users of type ‘Married’ and
    a. put in “cosine”, “pearson”, “meansquare” property of those books and
    b. set property equal ‘true’ in “registry_cosine”, “registry_pearson”, “registry_meansquare” property of those books.

```
MATCH (u2:Users{attr:'M'})-[r:RATED]->(b:Books)
MATCH (u1:Users {attr:'C'})
WHERE NOT((u1)-[:RATED]->(b))
WITH b, SUM(u2.cosine*r.rating) AS Weighted_Sim
ORDER BY Weighted_Sim DESC
SET b.registry_cosine = true, b.cosine = Weighted_Sim
RETURN b.book_id, Weighted_Sim
LIMIT 100

MATCH (u2:Users{attr:'M'})-[r:RATED]->(b:Books)
MATCH (u1:Users {attr:'C'})
WHERE NOT((u1)-[:RATED]->(b))
WITH b, SUM(u2.pearson*r.rating) AS Weighted_Sim
ORDER BY Weighted_Sim DESC
SET b.registry_pearson = true, b.pearson = Weighted_Sim
RETURN b.book_id, Weighted_Sim
LIMIT 100

MATCH (u2:Users{attr:'M'})-[r:RATED]->(b:Books)
MATCH (u1:Users {attr:'C'})
WHERE NOT((u1)-[:RATED]->(b))
WITH b, SUM(u2.meansquare*r.rating) AS Weighted_Sim
ORDER BY Weighted_Sim DESC
SET b.registry_meansquare = true, b.meansquare = Weighted_Sim
RETURN b.book_id, Weighted_Sim
LIMIT 100
```

5. See the common books suggested by all the similarity method

```
MATCH (b1:Books{registry_cosine:true})
MATCH (b2:Books{registry_pearson:true})
MATCH (b3:Books{registry_meansquare:true})
WHERE b1.book_id = b2.book_id AND b2.book_id = b3.book_id
WITH b1.book_id as book_id,
b1.cosine as cosine,
b2.pearson as pearson,
b3.meansquare as meansquare
RETURN book_id, cosine, pearson, meansquare
```

### Recommend Books for Gift to the invited Friends ###

1. Create similarity of Users to Books of “registry_cosine” on all four features in “UB_title, UB_authors, UB_language, UB_year” relationship [Users-UB…->Books (registry_cosine)]. Set the similarity value in “cosine” property of this “UB…” relationship.

```
MATCH (u:Users {attr:'F'})-[uat:UA_title]->(idft1:IDF_title)
MATCH (b:Books {registry_cosine:true})-[tft:TF_title]->(idft2:IDF_title)
WHERE idft1.word=idft2.word
WITH u, b,
SUM(tft.val*toFloat(idft1.IDF)*uat.val) as sim_title
MERGE (u)-[t:UB_title]->(b)
ON CREATE SET t.cosine = sim_title

MATCH (u:Users {attr:'F'})-[uaa:UA_authors]->(idfa1:IDF_authors)
MATCH (b:Books {registry_cosine:true})-[tfa:TF_authors]->(idfa2:IDF_authors)
WHERE idfa1.author=idfa2.author
WITH u, b,
SUM(tfa.val*toFloat(idfa1.IDF)*uaa.val) as sim_authors
MERGE (u)-[a:UB_authors]->(b)
ON CREATE SET a.cosine = sim_authors

MATCH (u:Users {attr:'F'})-[ual:UA_language]->(idfl1:IDF_language)
MATCH (b:Books {registry_cosine:true})-[tfl:TF_language]->(idfl2:IDF_language)
WHERE idfl1.language=idfl2.language
WITH u, b,
SUM(tfl.val*toFloat(idfl1.IDF)*ual.val) as sim_language
MERGE (u)-[l:UB_language]->(b)
ON CREATE SET l.cosine = sim_language

MATCH (u:Users {attr:'F'})-[uay:UA_year]->(idfy1:IDF_year)
MATCH (b:Books {registry_cosine:true})-[tfy:TF_year]->(idfy2:IDF_year)
WHERE  idfy1.year=idfy2.year
WITH u, b,
SUM(tfy.val*toFloat(idfy1.IDF)*uay.val) as sim_year
MERGE (u)-[y:UB_year]->(b)
ON CREATE SET y.cosine = sim_year
```

2. Create similarity of Users to Books of “registry_pearson” on all four features in “UB_title, UB_authors, UB_language, UB_year” relationship [Users-UB…->Books (registry_pearson)]. Set the similarity value in “pearson” property of this “UB…” relationship.

```
MATCH (u:Users {attr:'F'})-[uat:UA_title]->(idft1:IDF_title)
MATCH (b:Books {registry_pearson :true})-[tft:TF_title]->(idft2:IDF_title)
WHERE idft1.word=idft2.word
WITH u, b,
SUM(tft.val*toFloat(idft1.IDF)*uat.val) as sim_title
MERGE (u)-[t:UB_title]->(b)
SET t.pearson = sim_title

MATCH (u:Users {attr:'F'})-[uaa:UA_authors]->(idfa1:IDF_authors)
MATCH (b:Books {registry_pearson:true})-[tfa:TF_authors]->(idfa2:IDF_authors)
WHERE idfa1.author=idfa2.author
WITH u, b,
SUM(tfa.val*toFloat(idfa1.IDF)*uaa.val) as sim_authors
MERGE (u)-[a:UB_authors]->(b)
SET a.pearson = sim_authors

MATCH (u:Users {attr:'F'})-[ual:UA_language]->(idfl1:IDF_language)
MATCH (b:Books {registry_pearson:true})-[tfl:TF_language]->(idfl2:IDF_language)
WHERE idfl1.language=idfl2.language
WITH u, b,
SUM(tfl.val*toFloat(idfl1.IDF)*ual.val) as sim_language
MERGE (u)-[l:UB_language]->(b)
SET l.pearson = sim_language

MATCH (u:Users {attr:'F'})-[uay:UA_year]->(idfy1:IDF_year)
MATCH (b:Books {registry_pearson:true})-[tfy:TF_year]->(idfy2:IDF_year)
WHERE  idfy1.year=idfy2.year
WITH u, b,
SUM(tfy.val*toFloat(idfy1.IDF)*uay.val) as sim_year
MERGE (u)-[y:UB_year]->(b)
SET y.pearson = sim_year
```

3. Create similarity of Users to Books in “registry_meansquare” on all four features in “UB_title, UB_authors, UB_language, UB_year” relationship [Users-UB…->Books (registry_meansquare)]. Set the similarity value in “meansquare” property of this “UB…” relationship.

```
MATCH (u:Users {attr:'F'})-[uat:UA_title]->(idft1:IDF_title)
MATCH (b:Books {registry_meansquare:true})-[tft:TF_title]->(idft2:IDF_title)
WHERE idft1.word=idft2.word
WITH u, b,
SUM(tft.val*toFloat(idft1.IDF)*uat.val) as sim_title
MERGE (u)-[t:UB_title]->(b)
SET t.meansquare = sim_title

MATCH (u:Users {attr:'F'})-[uaa:UA_authors]->(idfa1:IDF_authors)
MATCH (b:Books {registry_meansquare:true})-[tfa:TF_authors]->(idfa2:IDF_authors)
WHERE idfa1.author=idfa2.author
WITH u, b,
SUM(tfa.val*toFloat(idfa1.IDF)*uaa.val) as sim_authors
MERGE (u)-[a:UB_authors]->(b)
SET a.meansquare = sim_authors

MATCH (u:Users {attr:'F'})-[ual:UA_language]->(idfl1:IDF_language)
MATCH (b:Books {registry_meansquare:true})-[tfl:TF_language]->(idfl2:IDF_language)
WHERE idfl1.language=idfl2.language
WITH u, b,
SUM(tfl.val*toFloat(idfl1.IDF)*ual.val) as sim_language
MERGE (u)-[l:UB_language]->(b)
SET l.meansquare = sim_language

MATCH (u:Users {attr:'F'})-[uay:UA_year]->(idfy1:IDF_year)
MATCH (b:Books {registry_meansquare:true})-[tfy:TF_year]->(idfy2:IDF_year)
WHERE  idfy1.year=idfy2.year
WITH u, b,
SUM(tfy.val*toFloat(idfy1.IDF)*uay.val) as sim_year
MERGE (u)-[y:UB_year]->(b)
SET y.meansquare = sim_year
```

4. Show the top 10 Gift suggestion of Users of type ‘Friend’ to the Users of type ‘Couple’ for 100 Registry books found using Cosine, Pearson, & MeanSquare user similarity. In the following 3 queries, you can
    a. Change the user_id and put any user_id of Users of type ‘Friend’.
    b. Change the weight of title/authors/language/year to calculate recommendation.

- For registry_cosine

```
WITH '21228' as id,
6 as weight_title,
3 as weight_authors,
1 as weight_language,
3 as weight_year
MATCH (u:Users {user_id:id})-[ubt:UB_title]->(b:Books {registry_cosine:true})
MATCH (u:Users {user_id:id})-[uba:UB_authors]->(b:Books {registry_cosine:true})
MATCH (u:Users {user_id:id})-[ubl:UB_language]->(b:Books {registry_cosine:true})
MATCH (u:Users {user_id:id})-[uby:UB_year]->(b:Books {registry_cosine:true})
WITH b, ubt.cosine as sim_title, uba.cosine as sim_authors, ubl.cosine as sim_language,
uby.cosine as sim_year, (weight_title*ubt.cosine + weight_authors*uba.cosine +
weight_language*ubl.cosine + weight_year*uby.cosine) as sim_total
ORDER BY (weight_title*sim_title + weight_authors*sim_authors +
weight_language*sim_language + weight_year*sim_year) DESC
RETURN b.title, b.authors, b.language, b.year,
round(sim_title*10000)/10000 as sim_title,
round(sim_authors*10000)/10000 as sim_authors,
round(sim_language*10000)/10000 as sim_language,
round(sim_year*10000)/10000 as sim_year,
round(sim_total*10000)/10000 as sim_total
LIMIT 10
```

- For registry_pearson

```
WITH '21228' as id,
6 as weight_title,
3 as weight_authors,
1 as weight_language,
3 as weight_year
MATCH (u:Users {user_id:id})-[ubt:UB_title]->(b:Books {registry_pearson:true})
MATCH (u:Users {user_id:id})-[uba:UB_authors]->(b:Books {registry_pearson:true})
MATCH (u:Users {user_id:id})-[ubl:UB_language]->(b:Books {registry_pearson:true})
MATCH (u:Users {user_id:id})-[uby:UB_year]->(b:Books {registry_pearson:true})
WITH b, ubt.pearson as sim_title, uba.pearson as sim_authors, ubl.pearson as sim_language,
uby.pearson as sim_year, (weight_title*ubt.pearson + weight_authors*uba.pearson +
weight_language*ubl.pearson + weight_year*uby.pearson) as sim_total
```

```
ORDER BY (weight_title*sim_title + weight_authors*sim_authors +
weight_language*sim_language + weight_year*sim_year) DESC
RETURN b.title, b.authors, b.language, b.year,
round(sim_title*10000)/10000 as sim_title,
round(sim_authors*10000)/10000 as sim_authors,
round(sim_language*10000)/10000 as sim_language,
round(sim_year*10000)/10000 as sim_year,
round(sim_total*10000)/10000 as sim_total
LIMIT 10
```

- For registry_meansquare

```
WITH '21228' as id,
6 as weight_title,
3 as weight_authors,
1 as weight_language,
3 as weight_year
MATCH (u:Users {user_id:id})-[ubt:UB_title]->(b:Books {registry_meansquare:true})
MATCH (u:Users {user_id:id})-[uba:UB_authors]->(b:Books {registry_meansquare:true})
MATCH (u:Users {user_id:id})-[ubl:UB_language]->(b:Books {registry_meansquare:true})
MATCH (u:Users {user_id:id})-[uby:UB_year]->(b:Books {registry_meansquare:true})
WITH b, ubt.meansquare as sim_title, uba.meansquare as sim_authors, ubl.meansquare as
sim_language, uby.meansquare as sim_year, (weight_title*ubt.meansquare +
weight_authors*uba.meansquare + weight_language*ubl.meansquare +
weight_year*uby.meansquare) as sim_total
ORDER BY (weight_title*sim_title + weight_authors*sim_authors +
weight_language*sim_language + weight_year*sim_year) DESC
RETURN b.title, b.authors, b.language, b.year,
round(sim_title*10000)/10000 as sim_title,
round(sim_authors*10000)/10000 as sim_authors,
round(sim_language*10000)/10000 as sim_language,
round(sim_year*10000)/10000 as sim_year,
round(sim_total*10000)/10000 as sim_total
LIMIT 10
```