1

```
%sql
SELECT * FROM movie_recommendation.default.movies_csv LIMIT 5;
```

_sqldf: pyspark.sql.dataframe.DataFrame = [movieId: integer, title: string ... 1 more field]

Table

	movieId	title	genres
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	2	Jumanji (1995)	Adventure|Children|Fantasy
3	3	Grumpier Old Men (1995)	Comedy|Romance
4	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	5	Father of the Bride Part II (1995)	Comedy

5 rows

This result is stored as `_sqldf` and can be used in other Python and SQL cells.

2

```
%sql
SELECT * FROM movie_recommendation.default.ratings_csv LIMIT 5;
```

_sqldf: pyspark.sql.dataframe.DataFrame = [userId: integer, movieId: integer ... 2 more fields]

Table

	userId	movieId	rating	timestamp
1	1	296	5	1147880044
2	1	306	3.5	1147868817
3	1	307	5	1147868828
4	1	665	5	1147878820
5	1	899	3.5	1147868510

5 rows

This result is stored as `_sqldf` and can be used in other Python and SQL cells.

3

```
df_movies = spark.read.option("header", True).option("inferSchema", True).csv("/FileStore/tables/movies.csv")

df_movies.show()
```

df_movies: pyspark.sql.dataframe.DataFrame = [movieId: integer, title: string ... 1 more field]

```
|      3|Grumpier Old Men ...|      Comedy|Romance|
|      4|Waiting to Exhale...|Comedy|Drama|Romance|
|      5|Father of the Bri...|              Comedy|
|      6|         Heat (1995)|Action|Crime|Thri...|
|      7|      Sabrina (1995)|      Comedy|Romance|
|      8| Tom and Huck (1995)|  Adventure|Children|
|      9| Sudden Death (1995)|              Action|
|     10|    GoldenEye (1995)|Action|Adventure|...|
|     11|American Presiden...|Comedy|Drama|Romance|
|     12|Dracula: Dead and...|       Comedy|Horror|
|     13|        Balto (1995)|Adventure|Animati...|
|     14|        Nixon (1995)|               Drama|
|     15|Cutthroat Island ...|Action|Adventure|...|
|     16|       Casino (1995)|         Crime|Drama|
|     17|Sense and Sensibi...|       Drama|Romance|
|     18|   Four Rooms (1995)|              Comedy|
|     19|Ace Ventura: When...|              Comedy|
|     20|  Money Train (1995)|Action|Comedy|Cri...|
+-------+--------------------+--------------------+
only showing top 20 rows
```

;

```
df_ratings = spark.read.option("header", True).option("inferSchema", True).csv("/FileStore/tables/ratings.csv")

df_ratings.show()
```

df_ratings: pyspark.sql.dataframe.DataFrame = [userId: integer, movieId: integer ... 2 more fields]

```
|     1|    307|   5.0|1147868828|
|     1|    665|   5.0|1147878820|
|     1|    899|   3.5|1147868510|
|     1|   1088|   4.0|1147868495|
|     1|   1175|   3.5|1147868826|
|     1|   1217|   3.5|1147878326|
|     1|   1237|   5.0|1147868839|
|     1|   1250|   4.0|1147868414|
|     1|   1260|   3.5|1147877857|
|     1|   1653|   4.0|1147868097|
|     1|   2011|   2.5|1147868079|
|     1|   2012|   2.5|1147868068|
|     1|   2068|   2.5|1147869044|
|     1|   2161|   3.5|1147868609|
|     1|   2351|   4.5|1147877957|
|     1|   2573|   4.0|1147878923|
|     1|   2632|   5.0|1147878248|
|     1|   2692|   5.0|1147869100|
+------+-------+------+----------+
only showing top 20 rows
```

```
df_ratings.select("userId", "movieId",
"rating").write.format("delta").mode("overwrite").save("/dbfs/delta/ratings_delta")
```

```
from pyspark.ml.recommendation import ALS

ratings_df = spark.read.format("delta").load("/dbfs/delta/ratings_delta")

als = ALS(
    userCol="userId",
    itemCol="movieId",
    ratingCol="rating",
    nonnegative=True,
    coldStartStrategy="drop",
    implicitPrefs=False,
    rank=10,
    maxIter=10,
    regParam=0.1
)

model = als.fit(ratings_df)
```

ratings_df: pyspark.sql.dataframe.DataFrame = [userId: integer, movieId: integer ... 1 more field]

```
model.save("/dbfs/models/movie_recommender")
```

```
user_recs = model.recommendForAllUsers(10)
user_recs.show(truncate=False)
```

user_recs: pyspark.sql.dataframe.DataFrame = [userId: integer, recommendations: array]

```
|52    |[{203086, 6.222506}, {151989, 6.0586605}, {203882, 5.830687}, {194434, 5.7407126}, {183947, 5.501353}, {194334, 5.4694
304}, {205277, 5.4652166}, {194883, 5.416352}, {166812, 5.265595}, {184299, 5.213943}]      |
|53    |[{194334, 6.9836917}, {192089, 6.4453025}, {194883, 6.196027}, {203882, 6.1777706}, {194332, 6.164963}, {155923, 6.063
3526}, {159467, 6.0409594}, {203086, 6.0151234}, {183947, 5.93982}, {194434, 5.936064}]     |
|65    |[{205277, 7.259401}, {151989, 6.27239}, {144202, 6.1628313}, {135099, 5.992943}, {169606, 5.9657702}, {194268, 5.94386
5}, {158747, 5.866911}, {192261, 5.8125367}, {194434, 5.791329}, {203086, 5.7466793}]       |
|76    |[{151989, 6.374255}, {194434, 6.1283813}, {203882, 5.891994}, {203086, 5.887548}, {183947, 5.794066}, {196787, 5.52803
4}, {157791, 5.5205073}, {157789, 5.5205073}, {192089, 5.5080566}, {205277, 5.4857564}]     |
|78    |[{151989, 6.6395993}, {194434, 6.616261}, {203086, 6.44422}, {203882, 6.409883}, {192089, 6.4095297}, {200930, 6.28702
93}, {145871, 6.277875}, {194334, 6.2171636}, {157791, 6.1864796}, {157789, 6.1864796}]     |
|81    |[{151989, 4.6866813}, {207640, 4.4423537}, {193257, 4.4281144}, {203086, 4.390354}, {203882, 4.3540764}, {194434, 4.24
60127}, {161048, 4.2433467}, {183947, 4.235991}, {200930, 4.221349}, {196787, 4.2117767}]   |
|85    |[{203086, 5.725191}, {194334, 5.6194167}, {193257, 5.610821}, {151989, 5.5646186}, {205453, 5.512639}, {184299, 5.4478
28}, {151615, 5.431345}, {200930, 5.397193}, {176729, 5.364778}, {166812, 5.3621187}]       |
+------+------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------+
only showing top 20 rows
```

```python
from pyspark.sql.functions import explode

recs_exploded = user_recs.select(
    "userId",
    explode("recommendations").alias("rec")
)
```

recs_exploded: pyspark.sql.dataframe.DataFrame = [userId: integer, rec: struct]

```python
recs_flat = recs_exploded.select(
    "userId",
    recs_exploded.rec.movieId.alias("movieId"),
    recs_exploded.rec.rating.alias("predicted_rating")
)
```

recs_flat: pyspark.sql.dataframe.DataFrame = [userId: integer, movieId: integer ... 1 more field]

```python
recs_with_name = recs_flat.join(df_movies, on="movieId", how="left")
```

recs_with_name: pyspark.sql.dataframe.DataFrame = [movieId: integer, userId: integer ... 3 more fields]

```python
recs_with_name.select("userId", "movieId", "title", "predicted_rating").show(truncate=False)
```

```
|1     |194334 |Les Luthiers: El Grosso Concerto (2001)           |5.341385        |
|1     |183947 |NOFX Backstage Passport 2                         |5.3345475       |
|1     |203086 |Truth and Justice (2019)                          |5.3135343       |
|1     |203882 |Dead in the Water (2006)                          |5.2937703       |
|1     |192089 |National Theatre Live: One Man, Two Guvnors (2011)|5.199268        |
|1     |200930 |C'est quoi la vie? (1999)                         |5.1535807       |
|1     |155923 |Sing (1989)                                       |5.1285763       |
|1     |157791 |.hack Liminality In the Case of Kyoko Tohno       |5.107959        |
|6     |151989 |The Thorn (1971)                                  |6.462091        |
|6     |194434 |Adrenaline (1990)                                 |6.415434        |
|6     |202231 |Foster (2018)                                     |5.8129597       |
|6     |183947 |NOFX Backstage Passport 2                         |5.8091736       |
|6     |203882 |Dead in the Water (2006)                          |5.6967783       |
|6     |203086 |Truth and Justice (2019)                          |5.643959        |
|6     |201821 |Civilisation (1969)                               |5.6336117       |
|6     |157791 |.hack Liminality In the Case of Kyoko Tohno       |5.5766034       |
|6     |157789 |.hack Liminality In the Case of Yuki Aihara       |5.5766034       |
|6     |188569 |Elizabeth at 90: A Family Tribute (2016)          |5.5162764       |
+------+-------+--------------------------------------------------+----------------+
only showing top 20 rows
```

```
recs_with_name.filter("userId == 1").show(truncate=False)
```

```
+-------+------+----------------+--------------------------------------------------+------------------+
|movieId|userId|predicted_rating|title                                             |genres            |
+-------+------+----------------+--------------------------------------------------+------------------+
|151989 |1     |5.5198274       |The Thorn (1971)                                  |Comedy            |
|194434 |1     |5.3604155       |Adrenaline (1990)                                 |(no genres listed)|
|194334 |1     |5.341385        |Les Luthiers: El Grosso Concerto (2001)           |(no genres listed)|
|183947 |1     |5.3345475       |NOFX Backstage Passport 2                         |(no genres listed)|
|203086 |1     |5.3135343       |Truth and Justice (2019)                          |Drama             |
|203882 |1     |5.2937703       |Dead in the Water (2006)                          |Horror            |
|192089 |1     |5.199268        |National Theatre Live: One Man, Two Guvnors (2011)|Comedy            |
|200930 |1     |5.1535807       |C'est quoi la vie? (1999)                         |Drama             |
|155923 |1     |5.1285763       |Sing (1989)                                       |(no genres listed)|
|157791 |1     |5.107959        |.hack Liminality In the Case of Kyoko Tohno       |(no genres listed)|
+-------+------+----------------+--------------------------------------------------+------------------+
```

```
watched_list_u1 = df_ratings.select("userId","movieId","rating").filter("userId == 1")
```

▸ watched_list_u1: pyspark.sql.dataframe.DataFrame = [userId: integer, movieId: integer ... 1 more field]

```
new_recommendations = recs_with_name.filter("userId == 1").join(watched_list_u1.select("movieId"), on="movieId",
how="left_anti")

new_recommendations.show(truncate=False)
```

▸ new_recommendations: pyspark.sql.dataframe.DataFrame = [movieId: integer, userId: integer ... 3 more fields]

```
+-------+------+----------------+--------------------------------------------------+------------------+
|movieId|userId|predicted_rating|title                                             |genres            |
+-------+------+----------------+--------------------------------------------------+------------------+
|151989 |1     |5.5198274       |The Thorn (1971)                                  |Comedy            |
|194434 |1     |5.3604155       |Adrenaline (1990)                                 |(no genres listed)|
|194334 |1     |5.341385        |Les Luthiers: El Grosso Concerto (2001)           |(no genres listed)|
|183947 |1     |5.3345475       |NOFX Backstage Passport 2                         |(no genres listed)|
|203086 |1     |5.3135343       |Truth and Justice (2019)                          |Drama             |
|203882 |1     |5.2937703       |Dead in the Water (2006)                          |Horror            |
|192089 |1     |5.199268        |National Theatre Live: One Man, Two Guvnors (2011)|Comedy            |
|200930 |1     |5.1535807       |C'est quoi la vie? (1999)                         |Drama             |
|155923 |1     |5.1285763       |Sing (1989)                                       |(no genres listed)|
|157791 |1     |5.107959        |.hack Liminality In the Case of Kyoko Tohno       |(no genres listed)|
+-------+------+----------------+--------------------------------------------------+------------------+
```

```
recs_with_name.write.format("delta").mode("overwrite").save("/delta/user_recommendations")
```

▸ df: pyspark.sql.dataframe.DataFrame = [movieId: integer, userId: integer ... 3 more fields]

```
[Row(movieId=151989, userId=1, predicted_rating=5.519827365875244, title='The Thorn (1971)', genres='Comedy'),
 Row(movieId=194434, userId=1, predicted_rating=5.360415458679199, title='Adrenaline (1990)', genres='(no genres listed)'),
 Row(movieId=194334, userId=1, predicted_rating=5.3413848876953125, title='Les Luthiers: El Grosso Concerto (2001)', genres
='(no genres listed)'),
 Row(movieId=183947, userId=1, predicted_rating=5.334547519683838, title='NOFX Backstage Passport 2', genres='(no genres liste
d)'),
 Row(movieId=203086, userId=1, predicted_rating=5.313534259796143, title='Truth and Justice (2019)', genres='Drama'),
 Row(movieId=203882, userId=1, predicted_rating=5.2937703132629395, title='Dead in the Water (2006)', genres='Horror'),
 Row(movieId=192089, userId=1, predicted_rating=5.199267864227295, title='National Theatre Live: One Man, Two Guvnors (2011)',
genres='Comedy'),
 Row(movieId=200930, userId=1, predicted_rating=5.153580665588379, title="C'est quoi la vie? (1999)", genres='Drama'),
 Row(movieId=155923, userId=1, predicted_rating=5.128576278686523, title='Sing (1989)', genres='(no genres listed)'),
 Row(movieId=157791, userId=1, predicted_rating=5.107958793640137, title='.hack Liminality In the Case of Kyoko Tohno', genres
='(no genres listed)')]
```