```
import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
  raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

```
Found GPU at: /device:GPU:0
```

```
!pip install mlxtend pandas
```

```
Requirement already satisfied: mlxtend in
/usr/local/lib/python3.10/dist-packages (0.23.1)
Requirement already satisfied: pandas in
/usr/local/lib/python3.10/dist-packages (2.1.4)
Requirement already satisfied: scipy>=1.2.1 in
/usr/local/lib/python3.10/dist-packages (from mlxtend) (1.13.1)
Requirement already satisfied: numpy>=1.16.2 in
/usr/local/lib/python3.10/dist-packages (from mlxtend) (1.26.4)
Requirement already satisfied: scikit-learn>=1.0.2 in
/usr/local/lib/python3.10/dist-packages (from mlxtend) (1.3.2)
Requirement already satisfied: matplotlib>=3.0.0 in
/usr/local/lib/python3.10/dist-packages (from mlxtend) (3.7.1)
Requirement already satisfied: joblib>=0.13.2 in
/usr/local/lib/python3.10/dist-packages (from mlxtend) (1.4.2)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.1 in
/usr/local/lib/python3.10/dist-packages (from pandas) (2024.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0-
>mlxtend) (1.3.0)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0-
>mlxtend) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0-
>mlxtend) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0-
>mlxtend) (1.4.7)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0-
>mlxtend) (24.1)
Requirement already satisfied: pillow>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0-
>mlxtend) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0-
```

```
>mlxtend) (3.1.4)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2-
>pandas) (1.16.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.2-
>mlxtend) (3.5.0)
```

```
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules

data = pd.read_csv('~/Groceries_dataset.csv')

data.head()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call `transform_cell`
automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during
thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```


```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 38765,\n
\"fields\": [\n    {\n      \"column\": \"Member_number\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1153,\n        \"min\": 1000,\n        \"max\": 5000,\n
\"num_unique_values\": 3898,\n        \"samples\": [\n          3785,\
n          1384,\n          3330\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Date\",\n      \"properties\": {\n
\"dtype\": \"object\",\n        \"num_unique_values\": 728,\n
\"samples\": [\n          \"19-07-2015\",\n          \"28-03-2015\",\n
\"29-07-2015\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"itemDescription\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 167,\n
\"samples\": [\n          \"cookware\",\n          \"canned fruit\",\n
\"specialty cheese\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"data"}
```


```
basket = data.groupby(['Member_number', 'itemDescription'])
['itemDescription'].count().unstack().reset_index().fillna(0).set_inde
x('Member_number')

basket = basket.map(lambda x: 1 if x > 0 else 0)

basket.head()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call `transform_cell`
```

```
automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during
thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

```
{"type":"dataframe","variable_name":"basket"}
```

```
frequent_itemsets = apriori(basket, min_support=0.01,
use_colnames=True)

frequent_itemsets.head()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call `transform_cell`
automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during
thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
/usr/local/lib/python3.10/dist-packages/mlxtend/frequent_patterns/
fpcommon.py:109: DeprecationWarning: DataFrames with non-bool types
result in worse computationalperformance and their support might be
discontinued in the future.Please use a DataFrame with bool type
  warnings.warn(
```

```
{"summary":"{\n  \"name\": \"frequent_itemsets\",\n  \"rows\": 3016,\n
\"fields\": [\n    {\n      \"column\": \"support\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.02380151420221512,\n        \"min\": 0.010005130836326322,\n
\"max\": 0.4581836839404823,\n        \"num_unique_values\": 281,\n
\"samples\": [\n          0.01026167265264238,\n
0.03078501795792714,\n          0.06054386865059005\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"itemsets\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\":
3016,\n        \"samples\": [\n          \"frozenset({'misc.
beverages'})\",\n          \"frozenset({'tropical fruit', 'canned
beer', 'whole milk', 'rolls/buns'})\",\n
\"frozenset({'spices'})\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"frequent_itemsets"}
```

```
rules = association_rules(frequent_itemsets, metric="lift",
min_threshold=1)

rules.head(10)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call `transform_cell`
automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during
```

```
thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

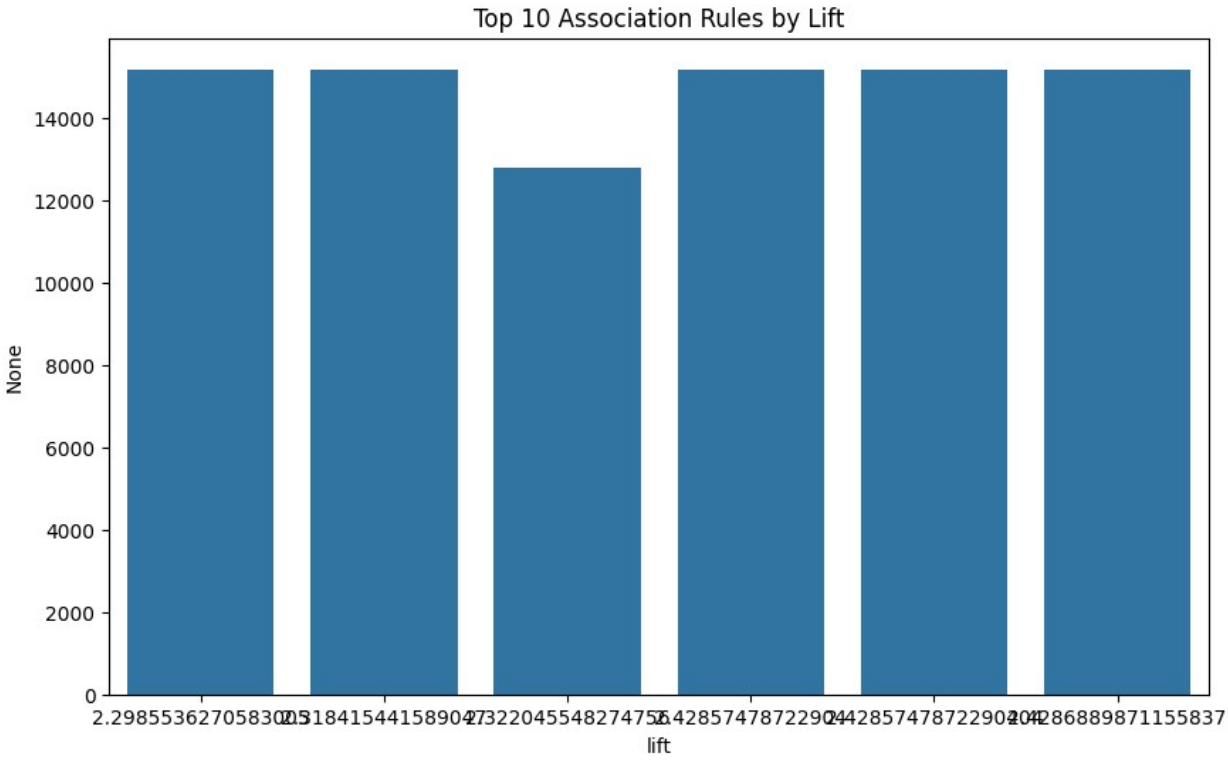

```
{"summary":"{\n  \"name\": \"rules\",\n  \"rows\": 15260,\n
\"fields\": [\n    {\n      \"column\": \"antecedents\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 1095,\n        \"samples\": [\n
\"frozenset({'fruit/vegetable juice', 'shopping bags'})\",\n
\"frozenset({'pastry', 'beef'})\",\n          \"frozenset({'bottled
water', 'UHT-milk'})\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"consequents\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 1095,\n
\"samples\": [\n          \"frozenset({'fruit/vegetable juice',
'shopping bags'})\",\n          \"frozenset({'pastry', 'other
vegetables'})\",\n          \"frozenset({'bottled water', 'UHT-
milk'})\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"antecedent support\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.12440029472281669,\n        \"min\":
0.013083632632119035,\n        \"max\": 0.4581836839404823,\n
\"num_unique_values\": 267,\n        \"samples\": [\n
0.09389430477167779,\n          0.05643919958953309,\n
0.05720882503848127\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"consequent support\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.12440029472281669,\n        \"min\":
0.013083632632119035,\n        \"max\": 0.4581836839404823,\n
\"num_unique_values\": 267,\n        \"samples\": [\n
0.038994356080041044,\n          0.05643919958953309,\n
0.05720882503848127\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"support\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 0.010440086040183424,\n        \"min\":
0.010005130836326322,\n        \"max\": 0.1913801949717804,\n
\"num_unique_values\": 237,\n        \"samples\": [\n
0.05720882503848127,\n          0.02103642893791688,\n
0.08132375577219086\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"confidence\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.15915359238021315,\n        \"min\":
0.021836506159014557,\n        \"max\": 0.7843137254901961,\n
\"num_unique_values\": 6684,\n        \"samples\": [\n
0.04143337066069429,\n          0.11727078891257996,\n
0.05858310626702997\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"lift\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.18484771302913766,\n        \"min\": 1.0001439344159704,\n
\"max\": 2.4286889871155837,\n        \"num_unique_values\": 8974,\n
\"samples\": [\n          1.3822883563605617,\n
```

1.2294698476392407,\n   1.1814269261077774\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"leverage\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\":
0.0019026224830934217,\n   \"min\": 3.0274303618599285e-06,\n
\"max\": 0.020939814421151365,\n   \"num_unique_values\": 6997,\n
\"samples\": [\n   0.003648448468265026,\n
0.0018031638490065161,\n   0.0020264039313420915\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"conviction\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0.10113551089525315,\n   \"min\": 1.0000226505250491,\n
\"max\": 2.512057465366855,\n   \"num_unique_values\": 13865,\n
\"samples\": [\n   1.0141668091898979,\n
1.1200885744376334,\n   1.0545820257045457\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"zhangs_metric\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0.12016841240804954,\n   \"min\": 0.00016656045420299355,\n
\"max\": 0.7674479166666666,\n   \"num_unique_values\": 14102,\n
\"samples\": [\n   0.003891877101649603,\n
0.18715716062725568,\n   0.320969484116707\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe","variable_name":"rules"}

```
import seaborn as sns
import matplotlib.pyplot as plt

top_rules = rules.sort_values('lift', ascending=False).head(10)
plt.figure(figsize=(10, 6))
sns.barplot(x='lift', y=top_rules.index, data=top_rules)
plt.title('Top 10 Association Rules by Lift')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call `transform_cell`
automatically in the future. Please pass the result to
`transformed_cell` argument and any exception that happen during
thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```


Top 10 Association Rules by Lift
None
lift
0
2000
4000
6000
8000
10000
12000
14000