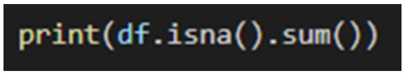
Testing and Validation

System Integration Testing and User Acceptance Testing

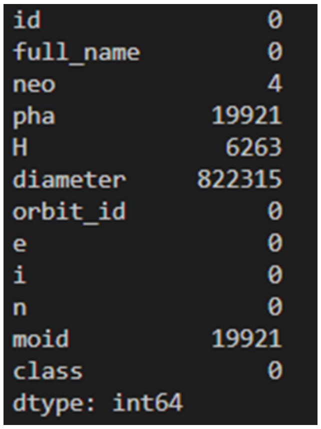
Test 1: Checking Number of Null Values

Explanation: Number of null values was accessed before and after data wrangling. Null values were not removed but were altered for use in data analysis.



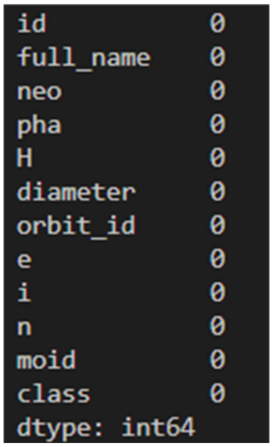
```
print(df.isna().sum())
```

Before handling null values



```
id                0
full_name         0
neo               4
pha           19921
H              6263
diameter     822315
orbit_id          0
e                 0
i                 0
n                 0
moid          19921
class             0
dtype: int64
```

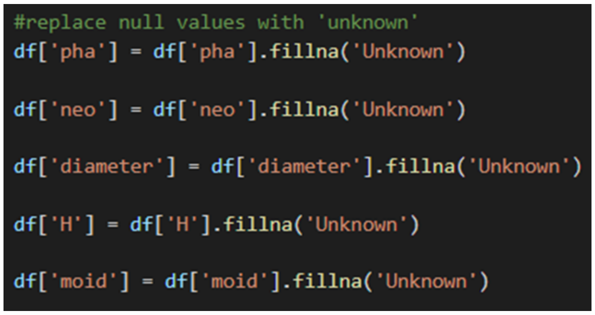
After handling null values



```
id           0
full_name    0
neo          0
pha          0
H            0
diameter     0
orbit_id     0
e            0
i            0
n            0
moid         0
class        0
dtype: int64
```

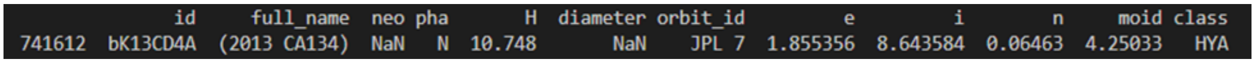
Test 2: Null Values Replaced

Explanation: These metrics were used in our data analysis and we wanted to account for unknown values instead of just removing them from the dataset.



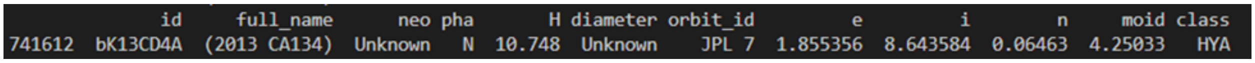
```
#replace null values with 'unknown'
df['pha'] = df['pha'].fillna('Unknown')

df['neo'] = df['neo'].fillna('Unknown')

df['diameter'] = df['diameter'].fillna('Unknown')

df['H'] = df['H'].fillna('Unknown')

df['moid'] = df['moid'].fillna('Unknown')
```

Before replacing columns containing null values (same id as reference)



```
              id     full_name  neo pha       H  diameter orbit_id         e         i        n     moid class
741612  bK13CD4A  (2013 CA134)  NaN   N  10.748       NaN    JPL 7  1.855356  8.643584  0.06463  4.25033   HYA
```

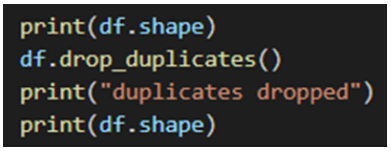
After replacing columns containing null values



```
             id     full_name      neo pha       H diameter orbit_id        e        i       n    moid class
741612  bK13CD4A  (2013 CA134)  Unknown   N  10.748  Unknown    JPL 7  1.855356  8.643584  0.06463  4.25033   HYA
```

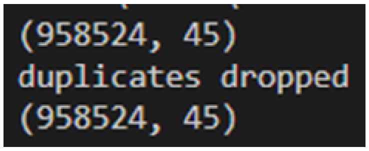
Test 3: Dropping Duplicates

Explanation: No Duplicates were found in the dataset. The number of rows and columns were unchanged after performing a duplicate drop function.



```
print(df.shape)
df.drop_duplicates()
print("duplicates dropped")
print(df.shape)
```

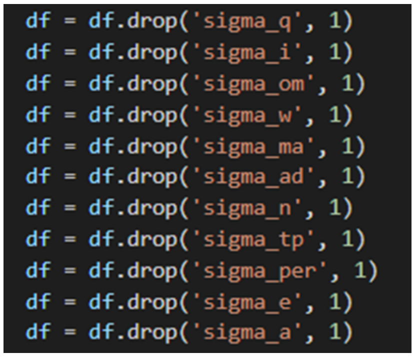
Before and after dropping duplicates



```
(958524, 45)
duplicates dropped
(958524, 45)
```

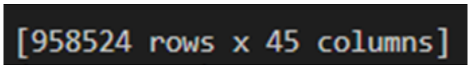
Test 4: Dropped Columns Not Being Used for Analysis

Explanation: Columns dropped were sigma values or scientific values that we could not fully understand scientifically for use in our data analysis.



```
df = df.drop('sigma_q', 1)
df = df.drop('sigma_i', 1)
df = df.drop('sigma_om', 1)
df = df.drop('sigma_w', 1)
df = df.drop('sigma_ma', 1)
df = df.drop('sigma_ad', 1)
df = df.drop('sigma_n', 1)
df = df.drop('sigma_tp', 1)
df = df.drop('sigma_per', 1)
df = df.drop('sigma_e', 1)
df = df.drop('sigma_a', 1)
```

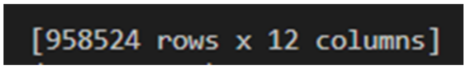
Before dropping columns



```
[958524 rows x 45 columns]
```

→

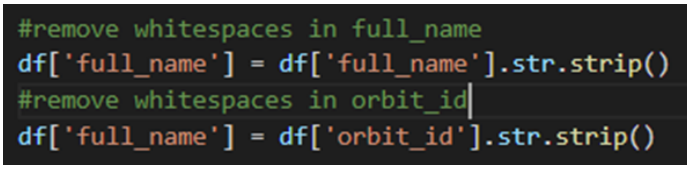
After dropping columns



```
[958524 rows x 12 columns]
```

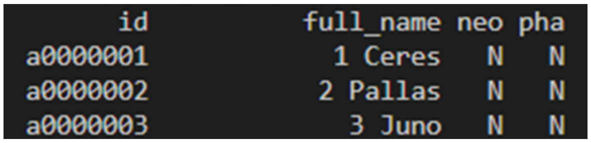
Test 5: Remove Whitespaces

Explanation: We removed whitespaces from our columns that had string values.



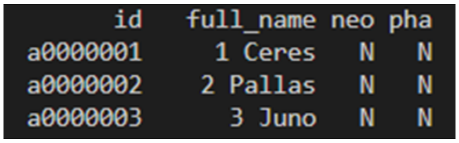
```
#remove whitespaces in full_name
df['full_name'] = df['full_name'].str.strip()
#remove whitespaces in orbit_id
df['full_name'] = df['orbit_id'].str.strip()
```

Before removing whitespaces from column "full_name"



```
      id            full_name neo pha
a0000001              1 Ceres   N   N
a0000002             2 Pallas   N   N
a0000003               3 Juno   N   N
```

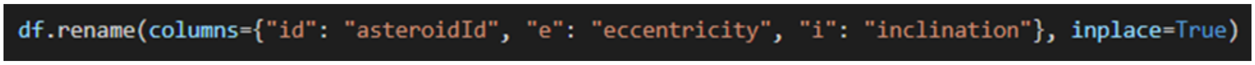
After removing whitespaces



```
      id    full_name neo pha
a0000001      1 Ceres   N   N
a0000002     2 Pallas   N   N
a0000003       3 Juno   N   N
```

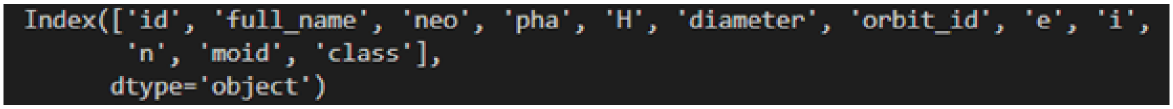
Test 6: Renamed columns

Explanation: Renamed specific columns for better understanding pertaining to business terms.



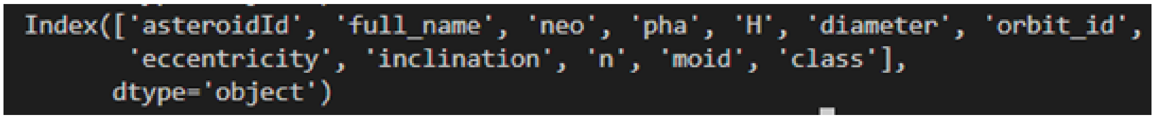
```
df.rename(columns={"id": "asteroidId", "e": "eccentricity", "i": "inclination"}, inplace=True)
```

Before renaming columns in dataset



```
Index(['id', 'full_name', 'neo', 'pha', 'H', 'diameter', 'orbit_id', 'e', 'i',
       'n', 'moid', 'class'],
      dtype='object')
```

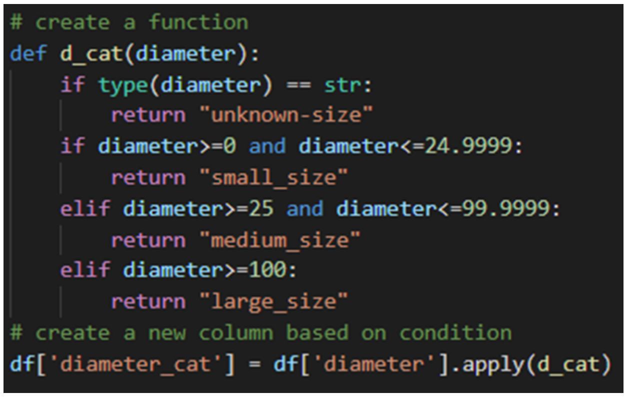
After renaming columns



```
Index(['asteroidId', 'full_name', 'neo', 'pha', 'H', 'diameter', 'orbit_id',
       'eccentricity', 'inclination', 'n', 'moid', 'class'],
      dtype='object')
```

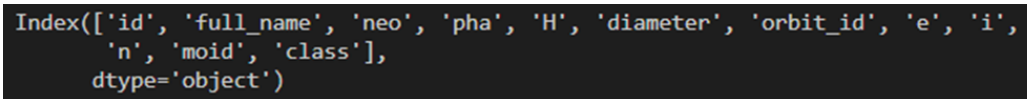
Test 7: Creating new columns based on value ranges

Explanation: We created new column categories for our metrics based on ranges to classify asteroids.



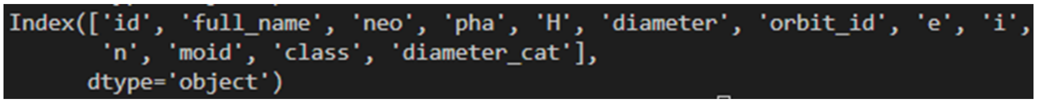
```
# create a function
def d_cat(diameter):
    if type(diameter) == str:
        return "unknown-size"
    if diameter>=0 and diameter<=24.9999:
        return "small_size"
    elif diameter>=25 and diameter<=99.9999:
        return "medium_size"
    elif diameter>=100:
        return "large_size"
# create a new column based on condition
df['diameter_cat'] = df['diameter'].apply(d_cat)
```

Before adding new diameter category column

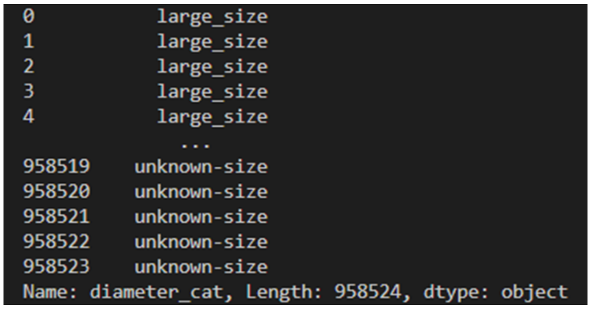


```
Index(['id', 'full_name', 'neo', 'pha', 'H', 'diameter', 'orbit_id', 'e', 'i',
       'n', 'moid', 'class'],
      dtype='object')
```

After adding the diameter category column to our dataset



```
Index(['id', 'full_name', 'neo', 'pha', 'H', 'diameter', 'orbit_id', 'e', 'i',
       'n', 'moid', 'class', 'diameter_cat'],
      dtype='object')
```



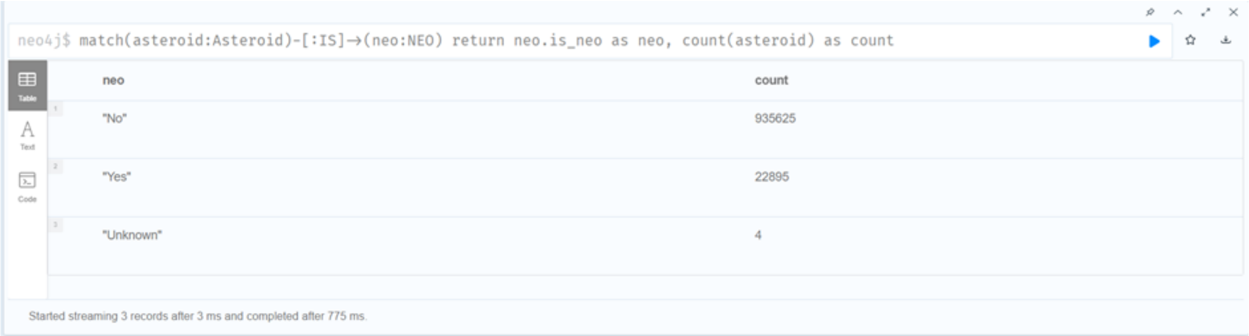
```
0           large_size
1           large_size
2           large_size
3           large_size
4           large_size
               ...
958519    unknown-size
958520    unknown-size
958521    unknown-size
958522    unknown-size
958523    unknown-size
Name: diameter_cat, Length: 958524, dtype: object
```

Neo4j Testing screenshots using Cypher Queries

Steps for testing and verification of data using Neo4j:

- Copy the cypher code(neo4j_testing_script.txt) containing the test queries
- Run the code to view graphs and tables verifying data load to Neo4j
- The scripts will run one by one and will take a few minutes displaying the below results

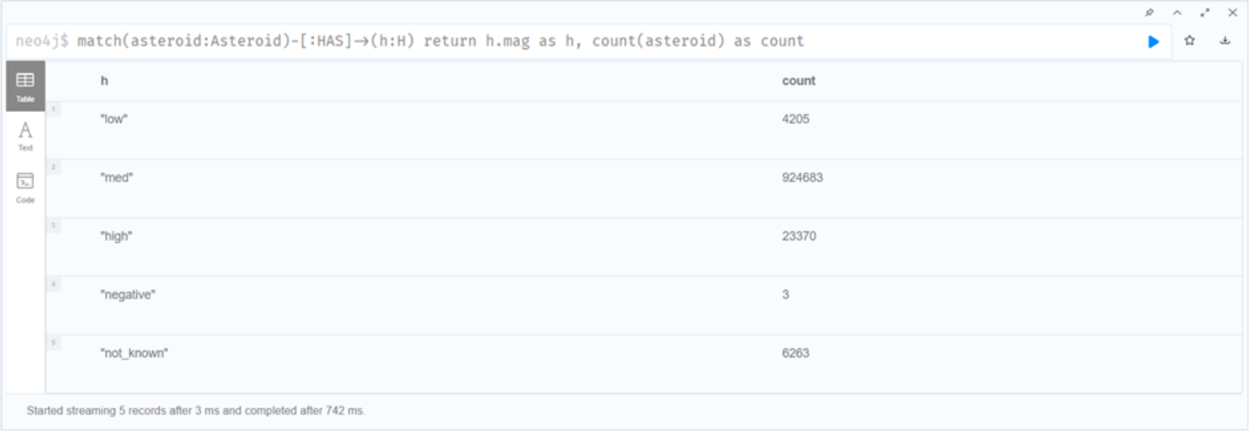
The queries provided below can be run to verify data stored in Neo4j matches the filtered python dataset



```
neo4j$ match(asteroid:Asteroid)-[:IS]→(neo:NEO) return neo.is_neo as neo, count(asteroid) as count
```

neo	count
"No"	935625
"Yes"	22895
"Unknown"	4

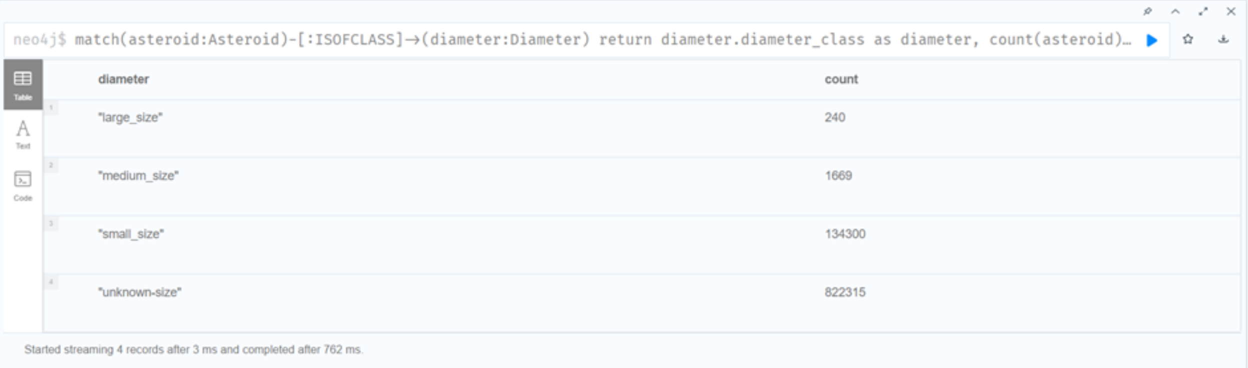
Started streaming 3 records after 3 ms and completed after 775 ms.



```
neo4j$ match(asteroid:Asteroid)-[:HAS]→(h:H) return h.mag as h, count(asteroid) as count
```

h	count
"low"	4205
"med"	924683
"high"	23370
"negative"	3
"not_known"	6263

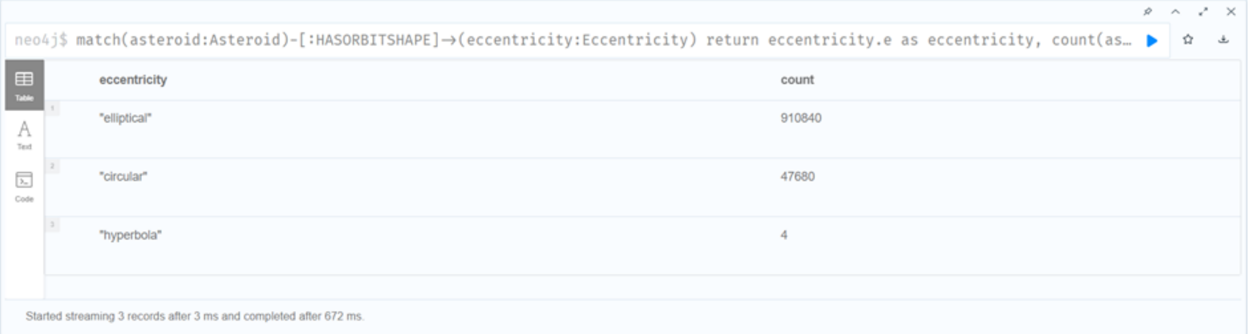
Started streaming 5 records after 3 ms and completed after 742 ms.



```
neo4j$ match(asteroid:Asteroid)-[:ISOFCLASS]→(diameter:Diameter) return diameter.diameter_class as diameter, count(asteroid)…
```

	diameter	count
1	"large_size"	240
2	"medium_size"	1669
3	"small_size"	134300
4	"unknown-size"	822315

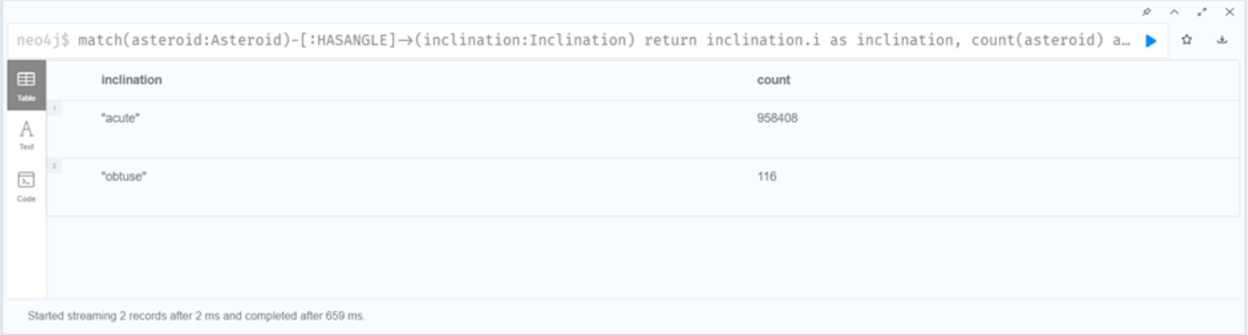
Started streaming 4 records after 3 ms and completed after 762 ms.



```
neo4j$ match(asteroid:Asteroid)-[:HASORBITSHAPE]→(eccentricity:Eccentricity) return eccentricity.e as eccentricity, count(as…
```

	eccentricity	count
1	"elliptical"	910840
2	"circular"	47680
3	"hyperbola"	4

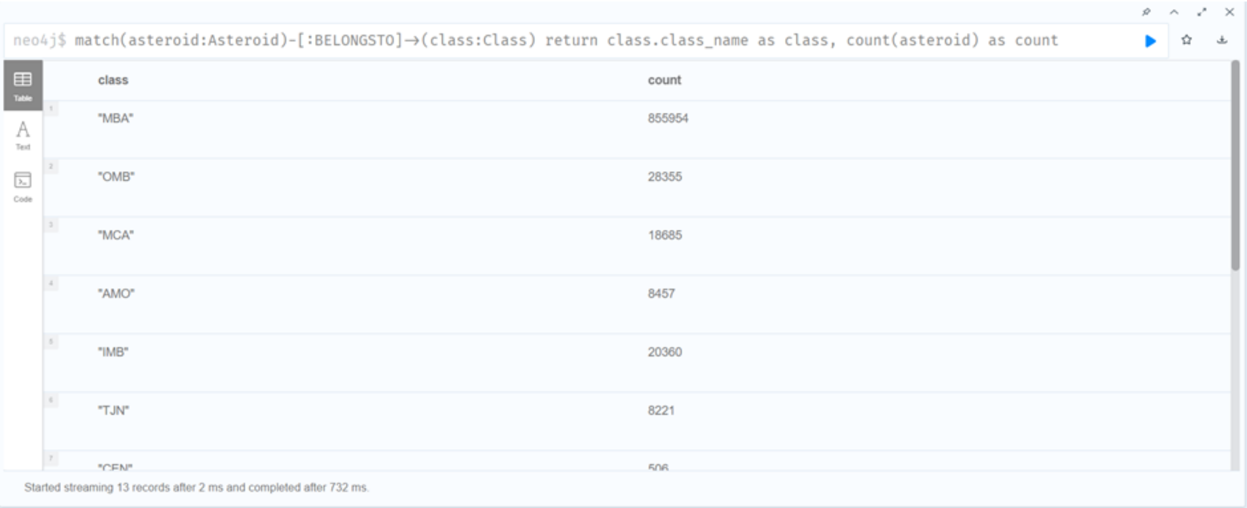
Started streaming 3 records after 3 ms and completed after 672 ms.



```
neo4j$ match(asteroid:Asteroid)-[:HASANGLE]→(inclination:Inclination) return inclination.i as inclination, count(asteroid) a…
```

	inclination	count
1	"acute"	958408
2	"obtuse"	116

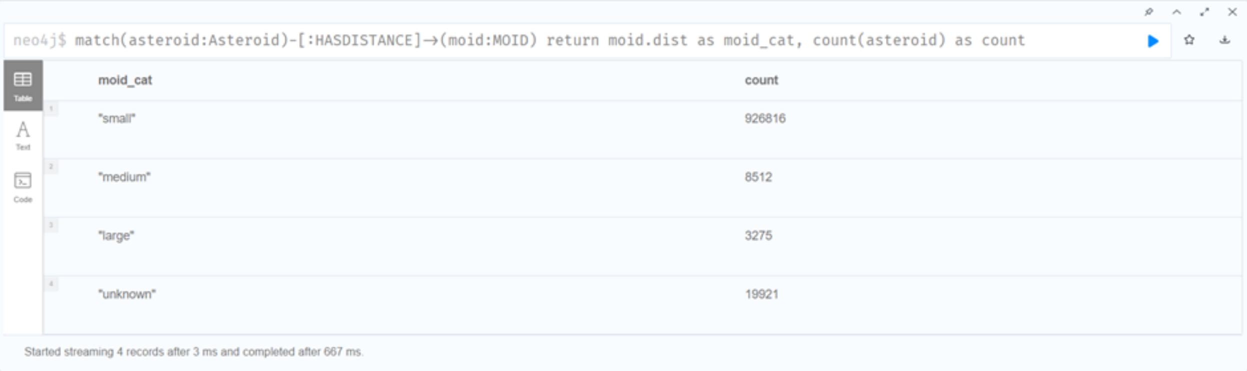
Started streaming 2 records after 2 ms and completed after 659 ms.



```
neo4j$ match(asteroid:Asteroid)-[:BELONGSTO]→(class:Class) return class.class_name as class, count(asteroid) as count
```

class	count
"MBA"	855954
"OMB"	28355
"MCA"	18685
"AMO"	8457
"IMB"	20360
"TJN"	8221
"CEN"	506

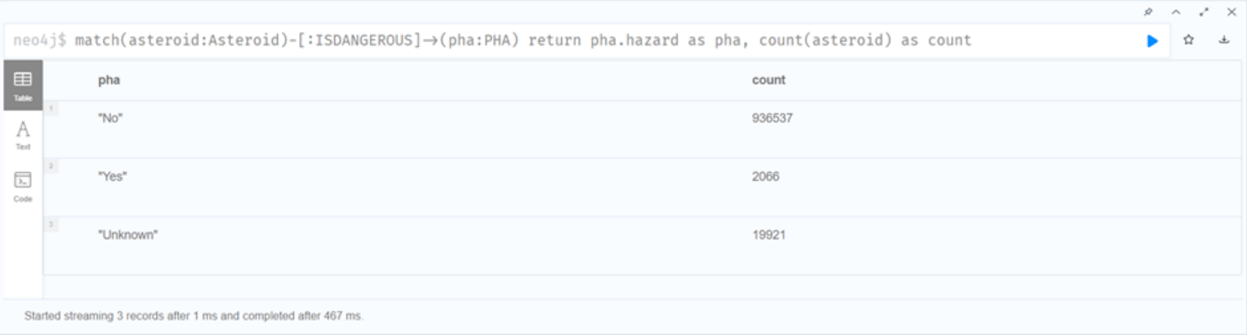
Started streaming 13 records after 2 ms and completed after 732 ms.



```
neo4j$ match(asteroid:Asteroid)-[:HASDISTANCE]→(moid:MOID) return moid.dist as moid_cat, count(asteroid) as count
```

moid_cat	count
"small"	926816
"medium"	8512
"large"	3275
"unknown"	19921

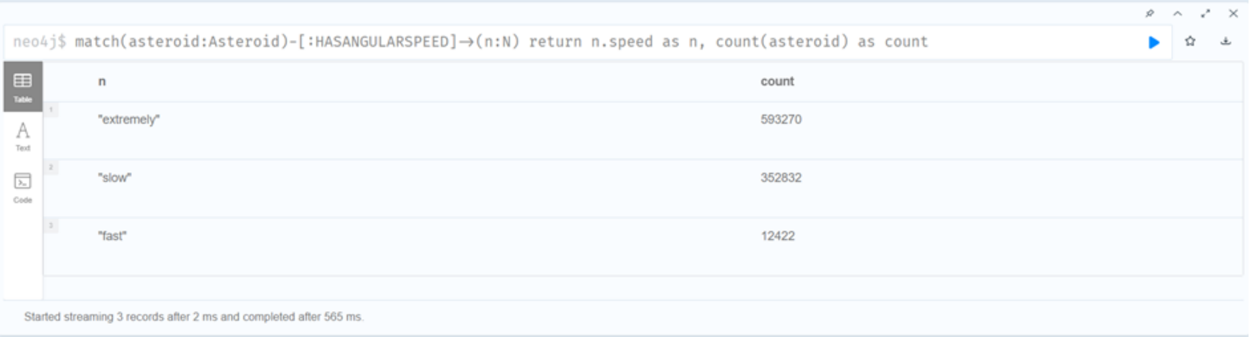
Started streaming 4 records after 3 ms and completed after 667 ms.



```
neo4j$ match(asteroid:Asteroid)-[:ISDANGEROUS]→(pha:PHA) return pha.hazard as pha, count(asteroid) as count
```

pha	count
"No"	936537
"Yes"	2066
"Unknown"	19921

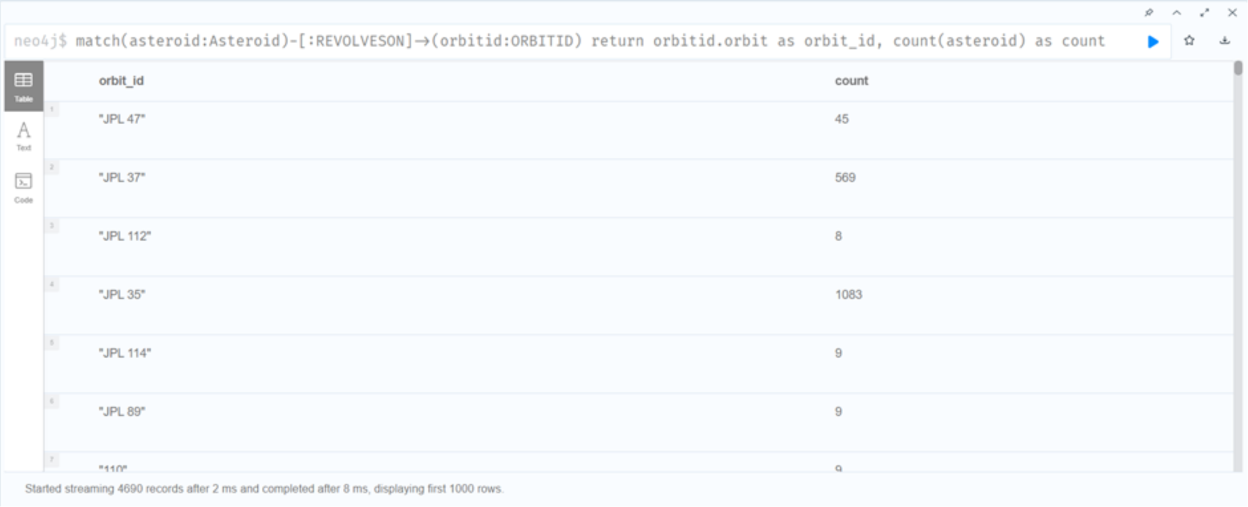
Started streaming 3 records after 1 ms and completed after 467 ms.



```
neo4j$ match(asteroid:Asteroid)-[:HASANGULARSPEED]→(n:N) return n.speed as n, count(asteroid) as count
```

n	count
"extremely"	593270
"slow"	352832
"fast"	12422

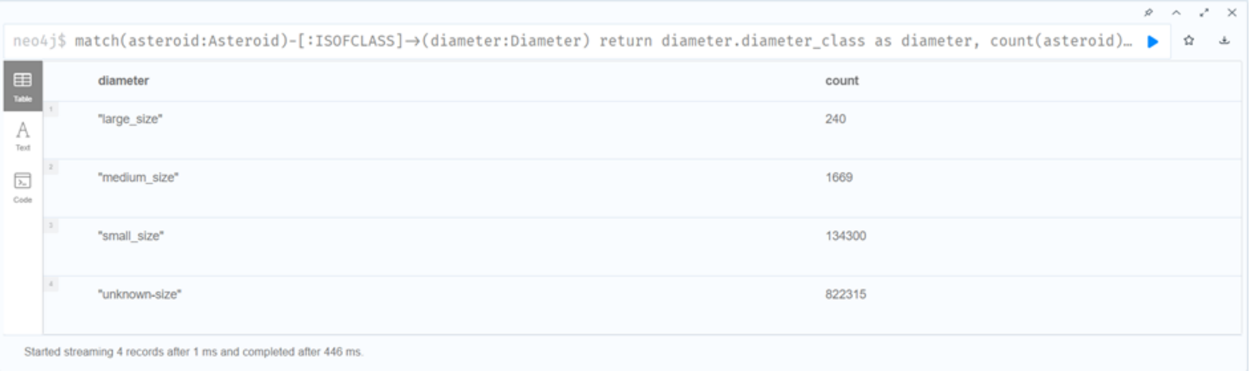
Started streaming 3 records after 2 ms and completed after 565 ms.



```
neo4j$ match(asteroid:Asteroid)-[:REVOLVESON]→(orbitid:ORBITID) return orbitid.orbit as orbit_id, count(asteroid) as count
```

orbit_id	count
"JPL 47"	45
"JPL 37"	569
"JPL 112"	8
"JPL 35"	1083
"JPL 114"	9
"JPL 89"	9
"110"	9

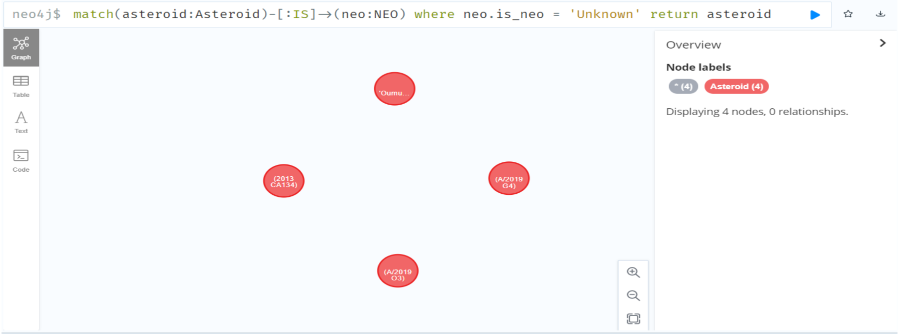
Started streaming 4690 records after 2 ms and completed after 8 ms, displaying first 1000 rows.



```
neo4j$ match(asteroid:Asteroid)-[:ISOFCLASS]→(diameter:Diameter) return diameter.diameter_class as diameter, count(asteroid)…
```

diameter	count
"large_size"	240
"medium_size"	1669
"small_size"	134300
"unknown-size"	822315

Started streaming 4 records after 1 ms and completed after 446 ms.



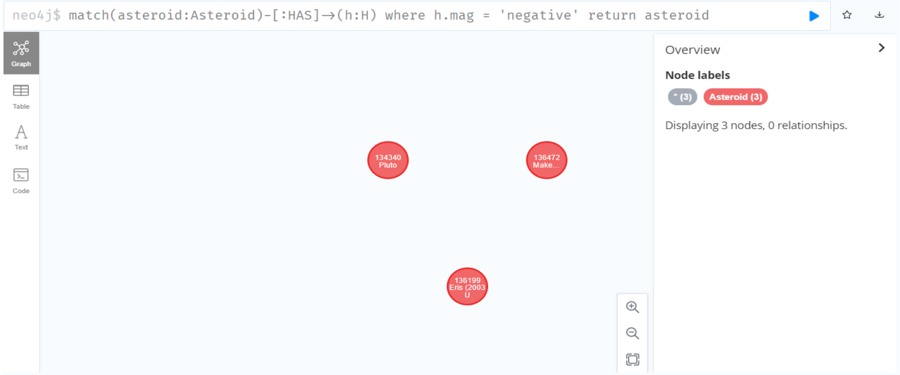
```
neo4j$ match(asteroid:Asteroid)-[:IS]→(neo:NEO) where neo.is_neo = 'Unknown' return asteroid
```

Overview

Node labels

* (4) Asteroid (4)

Displaying 4 nodes, 0 relationships.



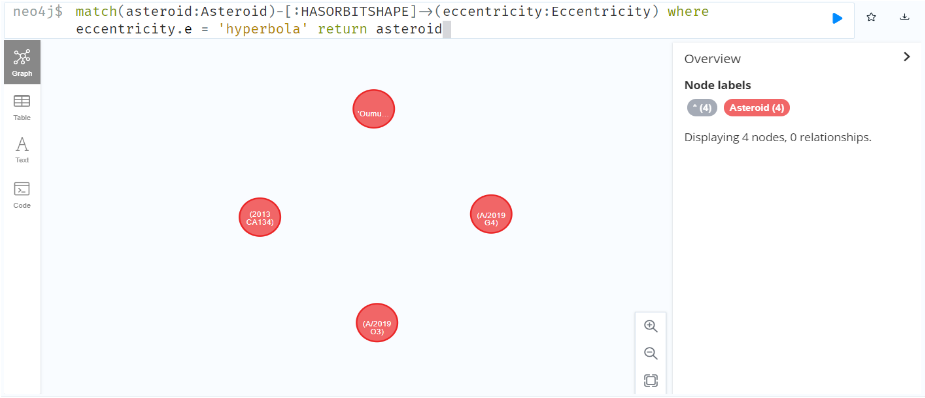
```
neo4j$ match(asteroid:Asteroid)-[:HAS]→(h:H) where h.mag = 'negative' return asteroid
```

Overview

Node labels

* (3) Asteroid (3)

Displaying 3 nodes, 0 relationships.



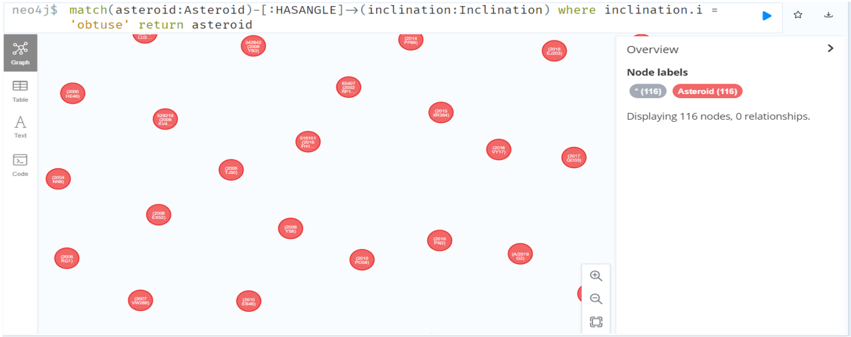
```
neo4j$ match(asteroid:Asteroid)-[:HASORBITSHAPE]→(eccentricity:Eccentricity) where
       eccentricity.e = 'hyperbola' return asteroid
```

Overview

Node labels

* (4) Asteroid (4)

Displaying 4 nodes, 0 relationships.



```
neo4j$ match(asteroid:Asteroid)-[:HASANGLE]→(inclination:Inclination) where inclination.i =
       'obtuse' return asteroid
```

Overview

Node labels

* (116) Asteroid (116)

Displaying 116 nodes, 0 relationships.