In [1]:
```
import plotly.express as px
```

In [2]:
```
import pandas as pd
```

In [3]:
```
from neo4j import GraphDatabase
class Neo4jConnection:

    def __init__(self, uri, user, pwd):
        self.__uri = uri
        self.__user = user
        self.__pwd = pwd
        self.__driver = None
        try:
            self.__driver = GraphDatabase.driver(self.__uri, auth=(self.__user, self.__pwd))
        except Exception as e:
            print("Failed to create the driver:", e)

    def close(self):
        if self.__driver is not None:
            self.__driver.close()

    def query(self, query, db=None):
        assert self.__driver is not None, "Driver not initialized!"
        session = None
        response = None
        try:
            session = self.__driver.session(database=db) if db is not None else self.__driver.session()
            response = list(session.run(query))
        except Exception as e:
            print("Query failed:", e)
        finally:
            if session is not None:
                session.close()
        return response
```

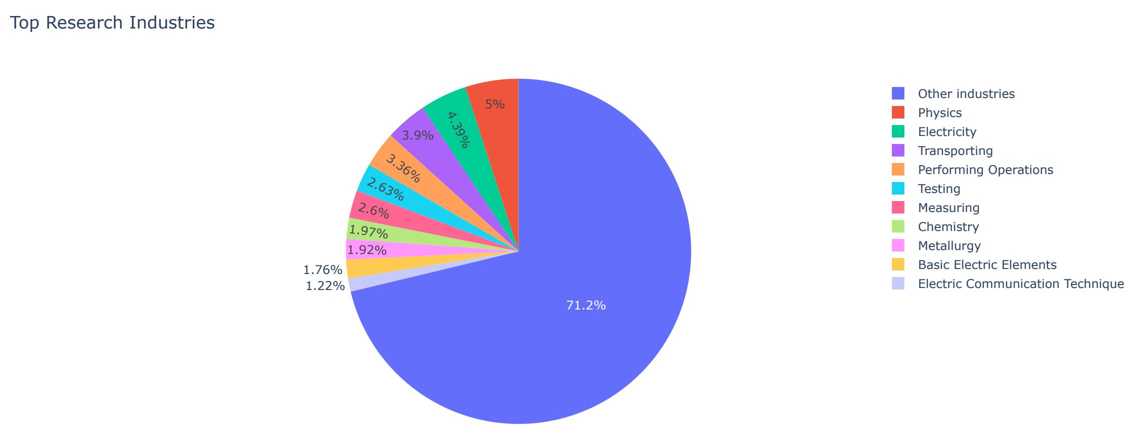
In [6]:
```
conn = Neo4jConnection(uri="bolt://localhost:11003", user="neo4j", pwd="dsc202")
```

Neo4j Cypher Queries :

Popular Research Industries (Degree Centrality)

In [7]:
```
pop_research_ind_query = '''MATCH (i:Industry)<-[r:BELONGS_TO]-()
WITH i, size((i)<--()) AS incomingCount
ORDER BY incomingCount DESC
RETURN DISTINCT i.industryId, incomingCount
LIMIT 100000;'''

pop_research_ind = conn.query(pop_research_ind_query, db='dsc202')
```

In [8]:
```
pop_research_ind_df = pd.DataFrame(pop_research_ind,columns=['Industry', 'PatentCount'])

# Represent only large countries
pop_research_ind_df.loc[pop_research_ind_df['PatentCount'] < 4200, 'Industry'] = 'Other industries'
fig = px.pie(pop_research_ind_df, values='PatentCount', names='Industry', title='Top Research Industries')
fig.show()
```



Most popular industries for research (top 5)

In [9]:
```
top_industries_query = '''
MATCH (i:Industry)<-[r:BELONGS_TO]-()
WITH i, size((i)<--()) AS incomingCount
ORDER BY incomingCount DESC
RETURN DISTINCT i.industryId, incomingCount
LIMIT 5;
'''
top_industries = conn.query(top_industries_query, db='dsc202')
```

In [10]:
```
top_industries_df = pd.DataFrame(top_industries,columns=['Industry','Patent'])
top_industries_df
```

Out[10]:

	Industry	Patent
0	Physics	17395
1	Electricity	15269
2	Transporting	13580
3	Performing Operations	11696
4	Testing	9166

Top 5 Assignees (companies) into research

In [11]:
```
top_assignee_query = '''MATCH (p:Patent)-[:ASSIGNED_TO]->(a:Assignee)
WITH p,a,size(()-->(a)) AS outgoingCount
ORDER BY outgoingCount DESC
RETURN DISTINCT a.assigneeName, outgoingCount
LIMIT 5;'''

top_assignee = conn.query(top_assignee_query, db='dsc202')
```

```
In [12]: top_assigness_df = pd.DataFrame(top_assignee,columns=['Assignee','Patents'])
         top_assigness_df
```

Out[12]:

	Assignee	Patents
0	Us Secretary Of Navy	13616
1	General Electric Co	8154
2	Boeing Co	3802
3	Safran Aircraft Engines Sas	2637
4	Rolls Royce Plc	2258

Assignee-Industry Count

```
In [13]: assignee_industry_count_query = '''MATCH
         (a:Assignee)<-[r:ASSIGNED_TO]-(p:Patent)-[b:BELONGS_TO]->(i:Industry) RETURN DISTINCT a.assigneeName,i.industryId,count(b) as count_b order by a.assigneeName,count_b desc'''

         assignee_industry_count = conn.query(assignee_industry_count_query,db='dsc202')
```

```
In [14]: assignee_industry_count_df = pd.DataFrame(assignee_industry_count,columns = ['Assignee','Industry','Patents'])
```

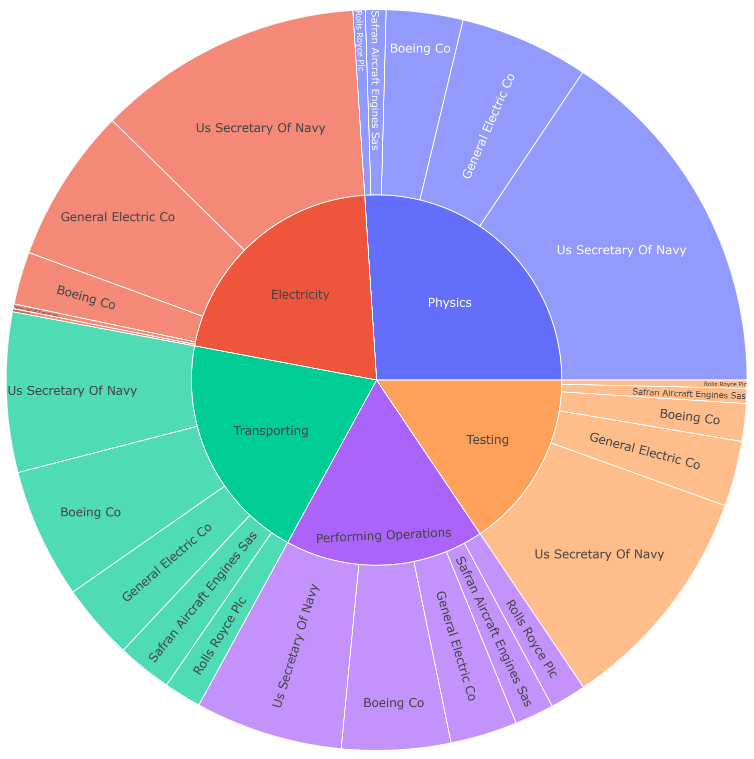
Top 5 Contributors (Assignees) for Top 5 Industries

```
In [15]: cond1 = assignee_industry_count_df['Assignee'].isin(top_assigness_df['Assignee'])
         cond2 = assignee_industry_count_df['Industry'].isin(top_industries_df['Industry'])
```

```
In [16]: top_assignee_top_industry = assignee_industry_count_df[(cond1 & cond2)].sort_values(['Assignee','Industry','Patents'])
```

```
In [17]: fig = px.sunburst(
             top_assignee_top_industry,
             path = ['Industry','Assignee'],
             values='Patents',width=900, height=900,
         )

         fig.show()
```

```
/opt/anaconda3/lib/python3.9/site-packages/plotly/express/_core.py:1637: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.

/opt/anaconda3/lib/python3.9/site-packages/plotly/express/_core.py:1637: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
```



Country Focus in Last Decade

```
In [18]: country_focus_query = '''MATCH
         (c:Country)<-[:ISSUED_BY]-(p:Patent)-[b:BELONGS_TO]->(i:Industry) RETURN DISTINCT c.countryName,i.industryId,p.fileDate.year,count(b) as count_b order by c.countryName,count_b desc
         '''

         country_focus = conn.query(country_focus_query,db='dsc202')
```

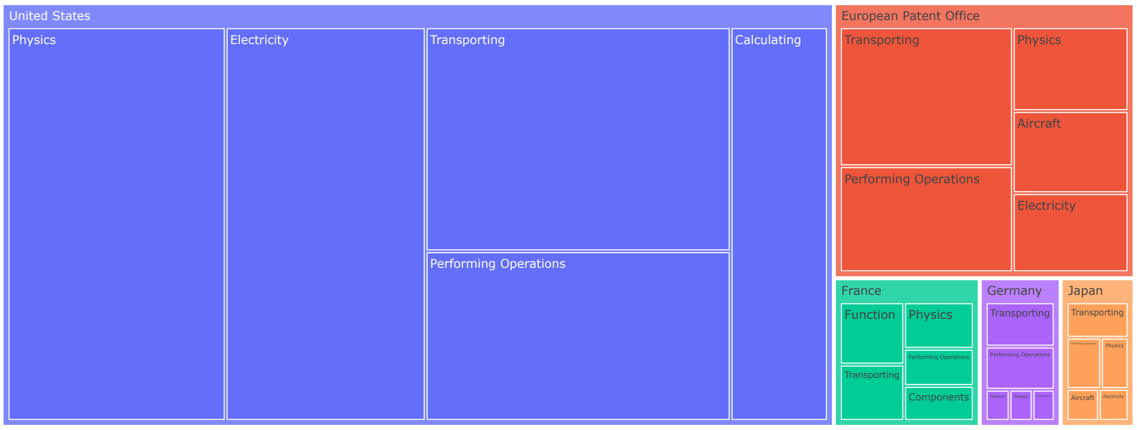
```
In [19]: country_focus_df = pd.DataFrame(country_focus,columns=['Country','Industry','Year','Patents'])

         country_focus_df_10yrs = country_focus_df[country_focus_df['Year'].isin(range(2010,2020))].groupby(['Country','Industry'])['Patents'].sum().reset_index()
```

```
In [20]: country_list = country_focus_df_10yrs.groupby('Country')['Patents'].sum().sort_values(ascending=False)[:5].index.tolist()
```

```
In [21]: country_focus_df_10yrs_top5 = country_focus_df_10yrs.sort_values(['Country','Patents'],ascending=False).groupby(['Country']).nth[:5].reset_index()
```

```
In [22]: fig = px.treemap(country_focus_df_10yrs_top5[country_focus_df_10yrs_top5['Country'].isin(country_list)], path=['Country','Industry'], values='Patents')
         fig.update_layout(margin = dict(t=50, l=25, r=25, b=25))
         fig.show()
```

```
/opt/anaconda3/lib/python3.9/site-packages/plotly/express/_core.py:1637: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.

/opt/anaconda3/lib/python3.9/site-packages/plotly/express/_core.py:1637: FutureWarning:

The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
```



Core Industry vs Research of Companies (assignees)

In [23]:
```
core_industry_vs_research_query = '''MATCH (ai:Industry)<-[c:BELONGS_TO]-(a:Assignee)<-[r:ASSIGNED_TO]-(p:Patent)-[b:BELONGS_TO]->(i:Industry)
RETURN DISTINCT a.assigneeName,collect(DISTINCT ai.industryId),i.industryId, COUNT(DISTINCT r) AS count_p ORDER BY count_p DESC'''
core_indusrty_vs_research = conn.query(core_industry_vs_research_query,db='dsc202')
```

In [24]:
```
core_indusrty_vs_research_df = pd.DataFrame(core_indusrty_vs_research,columns=['Organization','Core Industry','Research Industry','Patents'])
core_indusrty_vs_research_df
```

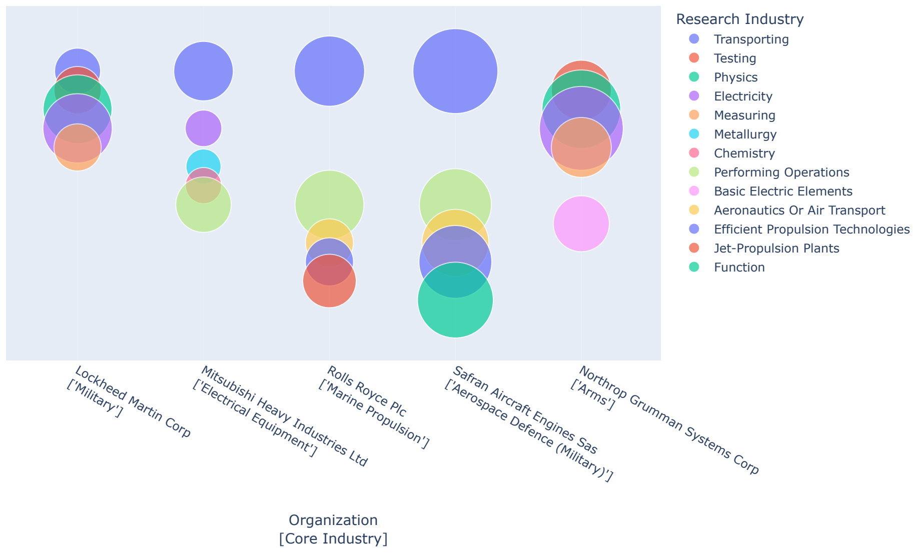
Out[24]:

	Organization	Core Industry	Research Industry	Patents
0	Safran Aircraft Engines Sas	[Aerospace Defence (Military)]	Transporting	817
1	Northrop Grumman Systems Corp	[Arms]	Electricity	796
2	Northrop Grumman Systems Corp	[Arms]	Physics	701
3	Safran Aircraft Engines Sas	[Aerospace Defence (Military)]	Function	651
4	Safran Aircraft Engines Sas	[Aerospace Defence (Military)]	Efficient Propulsion Technologies	601
...	...	...	...	...
17658	Demandware Inc	[Mobile Commerce]	Administration	1
17659	Demandware Inc	[Mobile Commerce]	Management	1
17660	Demandware Inc	[Mobile Commerce]	Details Thereof	1
17661	Demandware Inc	[Mobile Commerce]	Payment Architectures	1
17662	Demandware Inc	[Mobile Commerce]	Payment Protocols	1

17663 rows × 4 columns

In [25]:
```
core_indusrty_vs_research_df_top = core_indusrty_vs_research_df[core_indusrty_vs_research_df['Organization'].isin(core_indusrty_vs_research_df.groupby('Organization')['Patents'].sum().so
core_indusrty_vs_research_df_top = core_indusrty_vs_research_df_top.sort_values(by=['Organization','Patents'],ascending=False).groupby(['Organization']).nth[:5].reset_index()
core_indusrty_vs_research_df_top['Organization<br>[Core Industry]'] = core_indusrty_vs_research_df_top['Organization'] + "<br>" + core_indusrty_vs_research_df_top['Core Industry'].astype
```

In [26]:
```
fig = px.scatter(core_indusrty_vs_research_df_top, x="Organization<br>[Core Industry]",y='Research Industry',
                 size="Patents",color = 'Research Industry',hover_name='Core Industry', size_max=60,width=1000,height=600)

fig.update_yaxes(visible=False)
fig.update_yaxes(tickmode='array')

fig.show()
```



Top 5 Industry Research Over Time

In [27]:
```
assignee_pattern_query = '''MATCH
(a:Assignee)<-[r:ASSIGNED_TO]-(p:Patent)-[b:BELONGS_TO]->(i:Industry) RETURN DISTINCT a.assigneeName,p.fileDate.year as year_,i.industryId,count(b) as count_b order by a.assigneeName,yea

assignee_pattern = conn.query(assignee_pattern_query, db='dsc202')
```

In [28]:
```
assignee_pattern_df = pd.DataFrame(assignee_pattern,columns=['Assignee','Year','Industry','Patents'])
```

In [29]:
```
cond1 = assignee_pattern_df['Assignee'].isin(top_assigness_df['Assignee'])
cond2 = assignee_pattern_df['Industry'].isin(top_industries_df['Industry'])
```

In [30]:
```
top_assignee_top_industry = assignee_pattern_df[(cond1 & cond2)].sort_values(['Assignee','Industry','Year','Patents'])
```

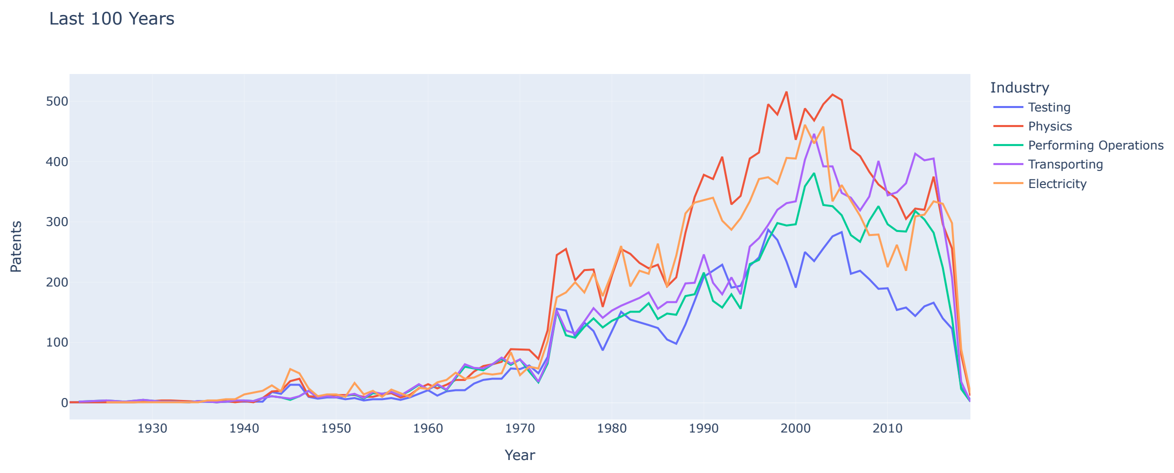
In [31]:
```
industries_year_query = '''MATCH (a:Assignee)<-[ap:ASSIGNED_TO]-(p:Patent)-[b:BELONGS_TO]->(i:Industry) RETURN DISTINCT p.fileDate.year as year_,i.industryId,count(distinct ap),count(dis
```

In [32]:
```
industries_year = conn.query(industries_year_query, db='dsc202')
```

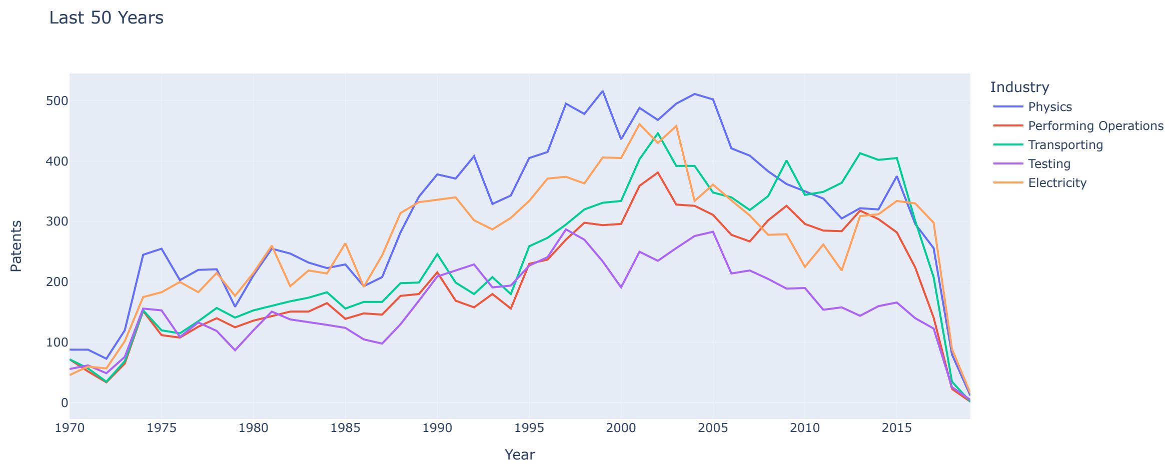
In [33]:
```
industries_year_df = pd.DataFrame(industries_year,columns = ['Year','Industry','ContributedBy','Patents'])
```

In [34]:
```
top_industries_year_df = industries_year_df[industries_year_df['Industry'].isin(top_industries_df['Industry'])]
```

In [35]:
```
fig = px.line(top_industries_year_df[top_industries_year_df['Year'].isin(range(1920,2020))],
              x="Year",y='Patents',
              color = 'Industry',hover_name='Industry',title='Last 100 Years')
fig.show()
```



In [36]:
```
fig = px.line(top_industries_year_df[top_industries_year_df['Year'].isin(range(1970,2020))],
              x="Year",y='Patents',color = 'Industry',
              hover_name='Industry',title='Last 50 Years')
fig.show()
```



PostgreSQL Query : Awards and Award Amount in various Industries

In [37]:
```
import psycopg2
```

In [38]:
```
conn = psycopg2.connect(
    host="awesome-hw.sdsc.edu",
    port="5432",
    database="postgres",
    user="ag_class",
    password="WUcgdfQ1")
conn
```

Out[38]:
```
<connection object at 0x7fb29c59ec10; dsn: 'user=ag_class password=xxx dbname=postgres host=awesome-hw.sdsc.edu port=5432', closed: 0>
```

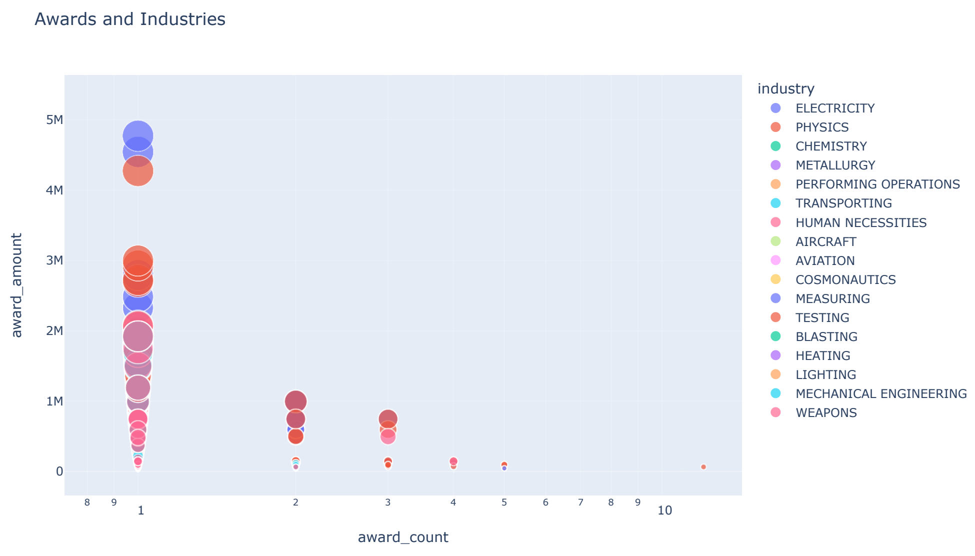
In [39]:
```
query2 = '''
select "Agency",Organization, Industry,regexp_replace("Award Amount", ',', '', 'g') Award_Amount ,sum(awards) award_count,sum(patents) Patents
from
     (select organization,industry,patents
      from
          (select unnest(assignee_name_current) Organization,
                  trim(unnest(string_to_array(unnest(classname),';'))) Industry,
                  array_length(regexp_split_to_array(trim(unnest(string_to_array(unnest(classname),';'))),E'\\W+'),1) len,
                  count(distinct patentid) as patents
           from patentdb
           group by 1,2
           order by 4 desc) as patent_info
      where len <= 3 and organization is not null) a
         join
         (select "Company",
                 "Agency",
                 "Award Amount",
                 count(*) as awards
          from sbir_award_data
          group by 1,2,3) b
        on a.organization=b."Company"
group by 1,2,3,4;
```

```
'''
data_df = pd.read_sql(query2, conn)
```

```
/opt/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:761: UserWarning:

pandas only support SQLAlchemy connectable(engine/connection) ordatabase string URI or sqlite3 DBAPI2 connectionother DBAPI2 objects are not tested, please consider using SQLAlchemy
```

In [41]:
```
data_df2 = data_df.astype({'award_amount': int})
```

In [42]:
```
fig = px.scatter(data_df2,
                 x="award_count", y="award_amount",
                 hover_name="organization",
                 color="industry", log_x=True,
                 size="award_amount", size_max=50,
                 title="Awards and Industries",width=1000, height=600)


fig.show()
```



In [ ]: