GDS - Final Project Queries

Graph Projections:

```
CALL gds.graph.project(
  'graph1',
  'City',
  'FLIGHT_TO'
)
```

Graph Statistics:

- Node Count:

```
MATCH (a)
RETURN count(a)
```

- Operator Count:

```
MATCH (o:Operator)
RETURN count(o)
```

- City Count

```
MATCH (c:City)
RETURN count(c)
```

- Crash City Count

```
MATCH (cl:CrashLocation)
RETURN count(cl)
```

- Air Craft Count

```
MATCH (ac:AircraftType)
RETURN count(ac)
```

- Air Craft to Operator Relationship Count

```
MATCH (a)-[w:WITHIN]-(b)
RETURN count(w)
```

- Operator to Start City in Route Relationship Count

```
MATCH (a)-[oa:OPERATED_AT]-(b)
RETURN count(oa)
```

- City to City in Route Relationship Count

```
MATCH (a)-[ft:FLIGHT_TO]-(b)
RETURN count(ft)
```

Contextual Questions:

- Most Frequent Crash Starting Cities & Count

```
MATCH (a:City)-[ft:FLIGHT_TO]->(b)
RETURN a.Name, count(ft)
ORDER BY count(ft) DESC
LIMIT 5
```

- Most Frequent Crash Site Cities & Count

```
MATCH (:City)-[ca:CRASHED_At]-(a:CrashLocation)
RETURN a.Name, count(ca)
ORDER BY count(ca) DESC
LIMIT 5
```

- Most Frequent Crashing Operator Type & Count

```
MATCH (a:Operator)-[oa:OPERATED_AT]->(:City)
RETURN a.Name, count(oa)
ORDER BY count(oa) DESC
```

- Crashing Operators Count

```
MATCH (a:Operator)-[oa:OPERATED_AT]->(:City)
```

```
RETURN oa.OperatorName, count(oa)
ORDER BY count(oa) DESC
LIMIT 5
```

- Most Frequent Crashing Aircraft Types & Count

```
MATCH (ac:AircraftType)-[w:WITHIN]->(:Operator)
RETURN ac.Make, count(w)
ORDER BY count(w) DESC
LIMIT 5
```

- Least Frequent Crashed Cities & Count

```
MATCH (a:City)-[ft:FLIGHT_TO]->(b)
RETURN a.Name, count(ft)
ORDER BY count(a) ASC
LIMIT 5
```

- Least Frequent Crash Route Starting Cities & Count

```
MATCH (:Operator)-[oa:OPERATED_AT]->(a:City)
RETURN a.Name , count(oa)
ORDER BY count(oa) ASC
LIMIT 5
```

- Least Frequent Crash Site Cities & Count

```
MATCH (:City)-[ca:CRASHED_At]-(a:CrashLocation)
RETURN a.Name, count(ca)
ORDER BY count(ca) ASC
LIMIT 5
```

- Least Frequent Crashing Aircraft Types & Count

```
MATCH (ac:AircraftType)-[w:WITHIN]->(:Operator)
RETURN ac.Make, count(w)
ORDER BY count(w) ASC
LIMIT 5
```

- Dates with Max Accidents

```
MATCH (o:Operator)-[oa:OPERATED_AT]->(:City)
RETURN o.Name, oa.Date, count(oa)
ORDER BY count(oa) DESC
LIMIT 5
```

- Worst Accidents

```
MATCH (o:Operator)-[oa:OPERATED_AT]->(:City)
RETURN o.Name, oa.Fatalities as Killed
ORDER BY Killed DESC
LIMIT 5
```

Relational Questions:

- Most Common City and Crash City Affiliation

```
MATCH (c:City)-[ca:CRASHED_At]->(cl:CrashLocation)
RETURN c.Name as City1, cl.Name as CrashSite, count(ca) as count
ORDER BY count DESC
LIMIT 5
```

- Most Common City and Crash City (Without Landing Crash) Affiliation

```
MATCH (c:City)-[ca:CRASHED_At]->(cl:CrashLocation)
WHERE c.Name <> cl.Name
RETURN c.Name as City1, cl.Name as CrashSite, count(ca) as count
ORDER BY count DESC
LIMIT 5
```

- Most Common Operator and Aircraft Type Affiliation

```
MATCH (o:Operator)<-[w:WITHIN]-(at:AircraftType)
RETURN o.Name as Operator, at.Make as Make, count(w) as count
ORDER BY count DESC
LIMIT 5
```

- Most Common City to City Route Affiliation

```
MATCH (c1:City)<-[ft:FLIGHT_TO]-(c2:City)
RETURN c1.Name as City1, c2.Name as City2, count(ft) as count
ORDER BY count DESC
```

```
LIMIT 5
```

Centrality Analysis:

- Eigenvector Centrality

```
CALL gds.eigenvector.stream('graph1')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).Name AS name, score
ORDER BY score DESC
LIMIT 5
```

- Closeness Centrality

```
CALL gds.closeness.stream('graph1')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).Name AS name, score
ORDER BY score DESC
LIMIT 5
```

- Degree Centrality

```
CALL gds.degree.stream('graph1')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).Name AS name, score
ORDER BY score DESC
LIMIT 5
```

- Betweenness Centrality

```
CALL gds.betweenness.stream('graph1')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).Name AS name, score
ORDER BY score DESC
LIMIT 5
```

- PageRank Centrality

```
CALL gds.pageRank.stream('graph1')
```

```
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).Name AS name, score
ORDER BY score DESC
LIMIT 5
```

Community Detection Analysis:

- Strongly Connected Components

```
CALL gds.scc.stream('graph1')
YIELD nodeId, componentId
WITH componentId, count(nodeId) AS componentSize, collect(gds.util.asNode(nodeId).Name) as CityNames
RETURN componentId,componentSize, CityNames
ORDER BY size(CityNames) ASC;
```

- Weakly Connected Components

```
CALL gds.wcc.stream('graph1')
YIELD nodeId, componentId
WITH componentId, count(nodeId) AS componentSize, collect(gds.util.asNode(nodeId).Name) as CityNames
RETURN componentId,componentSize, CityNames
ORDER BY size(CityNames) ASC;
```

- Label Propagation

```
CALL gds.labelPropagation.stream('graph1')
YIELD nodeId, communityId
RETURN gds.util.asNode(nodeId).Name AS name, communityId
ORDER BY communityId DESC
```

- Louvain Modularity

```
CALL gds.louvain.stream('graph1') YIELD nodeId, communityId RETURN gds.util.asNode(nodeId).Name AS name, communityId ORDER BY communityId DESC
```