**Question1: what are the most popular genres.**

Query file: /game\_scores/query2.py

Query used:

Mongodb:

json\_games=collection.find({},{"\_id":0,"sid":1,"gfq\_rating":1,"meta\_score":1,"meta\_uscore":1,"igdb\_score":1,"igdb\_uscore":1,"igdb\_popularity":1})

postgres:

with t as (select s.appid,s.name,s.release\_date,s.genres,j.gfq\_rating,j.meta\_score,

j.meta\_uscore,j.igdb\_score,j.igdb\_uscore,j.igdb\_popularity

from steam as s inner join json\_scores as j on j.sid=s.appid)

SELECT

unnested\_genres,

round(AVG(gfq\_rating),2) AS avg\_gfq\_rating,

round(AVG(meta\_score),2) AS avg\_meta\_score,

round(AVG(meta\_uscore),2) AS avg\_meta\_uscore,

round(AVG(igdb\_score),2) AS avg\_igdb\_score,

round(AVG(igdb\_uscore),2) AS avg\_igdb\_uscore,

round(AVG(igdb\_popularity),2) AS avg\_igdb\_popularity

FROM

(SELECT

unnest(string\_to\_array(genres, ';')) AS unnested\_genres,

gfq\_rating,

meta\_score,

meta\_uscore,

igdb\_score,

igdb\_uscore,

igdb\_popularity

FROM

t)

GROUP BY unnested\_genres

Python: calculate average and order result.

**Question2: who are the developers with highest game score.**

Query file: /game\_scores/query5.py

Query used:

Mongodb:

json\_games=collection.find({},{"\_id":0,"sid":1,"gfq\_rating":1,"meta\_score":1,"meta\_uscore":1,"igdb\_score":1,"igdb\_uscore":1,"igdb\_popularity":1})

neo4j:

MATCH (g:Game)-[:DEVELOPED\_BY]->(d:Developer)

RETURN g.name as Name,toInteger(g.id) as appid,d.name as Developer

postgres:

with t as (select g.appid,g."Developer",j.gfq\_rating,j.meta\_score,

j.meta\_uscore,j.igdb\_score,j.igdb\_uscore,j.igdb\_popularity

from game\_developer as g inner join json\_scores as j on j.sid=g.appid)

select

unnested\_developer,

count(\*) as num\_games,

round(AVG(gfq\_rating),2) AS avg\_gfq\_rating,

round(AVG(meta\_score),2) AS avg\_meta\_score,

round(AVG(meta\_uscore),2) AS avg\_meta\_uscore,

round(AVG(igdb\_score),2) AS avg\_igdb\_score,

round(AVG(igdb\_uscore),2) AS avg\_igdb\_uscore,

round(AVG(igdb\_popularity),2) AS avg\_igdb\_popularity

FROM

(SELECT

unnest(string\_to\_array(t."Developer", ';')) AS unnested\_developer, gfq\_rating, meta\_score,

meta\_uscore, igdb\_score, igdb\_uscore, igdb\_popularity

FROM t)

GROUP BY

unnested\_developer

Python: calculate average and order result.

**Question3: What are the trends of popularity for each genre over time.**

Query file: /game\_scores/query4.py

Query used:

Mongodb:

json\_games=collection.find({},{"\_id":0,"sid":1,"gfq\_rating":1,"meta\_score":1,"meta\_uscore":1,"igdb\_score":1,"igdb\_uscore":1,"igdb\_popularity":1})

postgres:

with t as (select s.appid,s.name,s.release\_date,s.genres,j.gfq\_rating,j.meta\_score,

j.meta\_uscore,j.igdb\_score,j.igdb\_uscore,j.igdb\_popularity

from steam as s inner join json\_scores as j on j.sid=s.appid)

SELECT

unnested\_genres,

release\_year,

count(\*) as num\_games,

round(AVG(gfq\_rating),2) AS avg\_gfq\_rating,

round(AVG(meta\_score),2) AS avg\_meta\_score,

round(AVG(meta\_uscore),2) AS avg\_meta\_uscore,

round(AVG(igdb\_score),2) AS avg\_igdb\_score,

round(AVG(igdb\_uscore),2) AS avg\_igdb\_uscore,

round(AVG(igdb\_popularity),2) AS avg\_igdb\_popularity

FROM

(SELECT

unnest(string\_to\_array(genres, ';')) AS unnested\_genres,

extract(year from to\_date(release\_date,'YYYY-MM-DD')) as release\_year,

gfq\_rating,

meta\_score,

meta\_uscore,

igdb\_score,

igdb\_uscore,

igdb\_popularity

FROM

t)

GROUP BY

release\_year,unnested\_genres

order by unnested\_genres,release\_year

Python: plot for each genre

**Question4: What are the other games with same genres given a game.**

Query file: /game\_scores/query1.py

Query used:

postgres:

cur.execute("select name,appid from steam where name ilike %s",(name,))

neo4j:

match p=(g:Game)-[:IS\_GENRE]->() where tolower(g.name) contains $name RETURN p

match p=(g:Game)-[:IS\_GENRE]->(genre:Genre) where g.id=$id RETURN genre

MATCH (g:Game)

WHERE ALL(genre IN $genres WHERE (g)-[:IS\_GENRE]->(:Genre {genre: genre}))

RETURN g.name as Name,g.id as Appid

MATCH (g:Game)-[:DEVELOPED\_BY]->(d:Developer)

RETURN g.name as Name,toInteger(g.id) as appid,d.name as Developer

Python: handle the return results and give output.

**Question5:**