Exercici 1. Importeu les dades en la BD de Neo4j del projecte.

Tractament preeliminar de les dades:

Creació de restriccions (constraints)

```
// Clau primaria individu
CREATE CONSTRAINT individu_id_unique IF NOT EXISTS
FOR (i:Individu)
REQUIRE i.id IS UNIQUE;

// Clau primaria habitatges
CREATE CONSTRAINT habitatge_composite_key IF NOT EXISTS
FOR (h:Habitatge)
REQUIRE (h.id_llar, h.municipi, h.any_padro) IS NODE KEY;
```

Creació dels Index:

```
// Index per any habitatge
CREATE INDEX habitatge_any_padro IF NOT EXISTS
FOR (h:Habitatge)
ON (h.any_padro);

// Index per municipi
CREATE INDEX habitatge_municipi IF NOT EXISTS
FOR (h:Habitatge)
ON (h.municipi);

// Index per direcció (carrer i número)
CREATE INDEX habitatge_adreca IF NOT EXISTS
FOR (h:Habitatge)
ON (h.carrer, h.numero);

// Index per cognoms
CREATE INDEX individu_cognoms IF NOT EXISTS
FOR (i:Individu)
ON (i.cognom1, i.cognom2);

// Index de búsqueda full-text sobre nom y cognoms
CREATE FULLTEXT INDEX individu_fulltext_index IF NOT EXISTS
FOR (i:Individu)
ON EACH [i.nom, i.cognom1, i.cognom2];

SET node.atribut = CASE WHEN row.atribut IS NOT NULL AND row.atribut <> ' ' AND
toLower(row.atribut) <> 'null' THEN row.atribut ELSE NULL END
```

Conversions de tipus

Carregar les dades:

```
// ====================
// 1. Crear nodes Individu
// ====================
LOAD CSV WITH HEADERS FROM 'file:///INDIVIDUAL.csv' AS row
WITH row
WHERE row.Id IS NOT NULL AND row.Id <> '' AND toLower(row.Id) <> 'null'
MERGE (i:Individu {id: toInteger(row.Id)})
SET i.nom = CASE WHEN row.name IS NOT NULL AND row.name <> '' AND
toLower(row.name) <> 'null' THEN row.name ELSE NULL END,
   i.cognom1 = CASE WHEN row.surname IS NOT NULL AND row.surname <> '' AND
toLower(row.surname) <> 'null' THEN row.surname ELSE NULL END,
   i.cognom2 = CASE WHEN row.second_surname IS NOT NULL AND
row.second_surname <> '' AND toLower(row.second_surname) <> 'null' THEN
row.second_surname ELSE NULL END,
   i.any_padro = CASE
      WHEN row.Year IS NOT NULL AND row.Year <> '' AND toLower(row.Year) <> 'null'
      THEN toInteger(row.Year)
      ELSE NULL
   END;

// ====================
// 2. Crear nodes Habitatge
// ====================
LOAD CSV WITH HEADERS FROM 'file:///HABITATGES.csv' AS row
WITH row
WHERE row.Id_Llar IS NOT NULL AND row.Id_Llar <> '' AND toLower(row.Id_Llar) <> 'null'
  AND row.Any_Padro IS NOT NULL AND row.Any_Padro <> '' AND
toLower(row.Any_Padro) <> 'null'
MERGE (h:Habitatge {
   id_llar: toInteger(row.Id_Llar),
   any_padro: toInteger(row.Any_Padro),
   municipi: row.Municipi
})
SET h.carrer = CASE WHEN row.Carrer IS NOT NULL AND row.Carrer <> '' AND
toLower(row.Carrer) <> 'null' THEN row.Carrer ELSE NULL END,
   h.numero = CASE
      WHEN row.Numero IS NOT NULL AND row.Numero <> '' AND toLower(row.Numero)
<> 'null'
      THEN toInteger(row.Numero)
      ELSE NULL
   END;
```

```
// ====================
// 3. Relació VIU (Individu)-[:VIU]->(Habitatge)
// ====================
LOAD CSV WITH HEADERS FROM 'file:///VIU.csv' AS row
WITH row
WHERE row.IND IS NOT NULL AND row.IND <> '' AND toLower(row.IND) <> 'null'
  AND row.HOUSE_ID IS NOT NULL AND row.HOUSE_ID <> '' AND
toLower(row.HOUSE_ID) <> 'null'
  AND row.Year IS NOT NULL AND row.Year <> '' AND toLower(row.Year) <> 'null'
MATCH (i:Individu {id: toInteger(row.IND)})
MATCH (h:Habitatge {id_llar: toInteger(row.HOUSE_ID)})
MERGE (i)-[:VIU {any_padro: toInteger(row.Year)}]->(h);
```

```
Set 68841 properties, created 68841 relationships, completed after 1305 ms.
```

```
// ====================
// 4. Relació SAME_AS (Individu)-[:SAME_AS]->(Individu)
// ====================
LOAD CSV WITH HEADERS FROM 'file:///SAME_AS.csv' AS row
WITH row
WHERE row.Id_A IS NOT NULL AND row.Id_A <> '' AND toLower(row.Id_A) <> 'null'
  AND row.Id_B IS NOT NULL AND row.Id_B <> '' AND toLower(row.Id_B) <> 'null'
MATCH (a:Individu {id: toInteger(row.Id_A)})
MATCH (b:Individu {id: toInteger(row.Id_B)})
MERGE (a)-[:SAME_AS]->(b);

// ====================
// 5. Relacions Familiars
// ====================
LOAD CSV WITH HEADERS FROM 'file:///FAMILIA.csv' AS row
WITH row
WHERE row.ID_1 IS NOT NULL AND row.ID_1 <> '' AND toLower(row.ID_1) <> 'null'
  AND row.ID_2 IS NOT NULL AND row.ID_2 <> '' AND toLower(row.ID_2) <> 'null'
  AND row.Relacio_Harmonitzada IS NOT NULL AND row.Relacio_Harmonitzada <> '' AND
toLower(row.Relacio_Harmonitzada) <> 'null'
MATCH (a:Individu {id: toInteger(row.ID_1)})
MATCH (b:Individu {id: toInteger(row.ID_2)})
MERGE (a)-[r:RELACIO]->(b)
SET r.tipus = row.Relacio_Harmonitzada;
```

Exercici 2. Resoleu les següents consultes Cypher:

Consulta a: Cognoms i habitants a Castellví de Rosanes per any de padró

```
MATCH (h:Habitatge {municipi: "CR"})<-[:VIU]-(i:Individu)
WITH h.any_padro AS year,
    collect(i.cognom1) + collect(i.cognom2) AS cognoms_combinats,
    count(DISTINCT i) AS habitants
UNWIND cognoms_combinats AS cognom
WITH year, habitants,
    collect(DISTINCT cognom) AS cognoms_totals
WITH year, habitants,
    [c IN cognoms_totals
     WHERE c <> "nan"] AS cognoms
RETURN year, habitants, cognoms
ORDER BY year;
```

Consulta b: Trobada de veïns a partir d'un individu concret

```
MATCH(r:Individu{nom:"rafel",cognom1:"marti"})-[v:VIU]->(h:Habitatge
{any_padro: 1838, municipi: "SFLL"})<-[VIU]-(vei:Individu)
RETURN r, v, h, vei;
```

Consulta c: Variants d'identitat d'un individu (relacions SAME_AS)

```
MATCH(:Individu{nom:"miguel",cognom1:"estape",cognom2:"bofill"})-[:S
AME_AS]-(p)
RETURN collect(distinct p.nom) as `Variants Nom`, collect(distinct
p.cognom1) as `Variants Primer Cognom`, collect(distinct p.cognom2)
as `Variants Segon Cognom`
```

Consulta d: Mitjana de fills per habitatge a Sant Feliu de Llobregat (1881)

```
CALL {
  MATCH (h:Habitatge {municipi: 'SFLL', any_padro: 1881})
  RETURN count(DISTINCT h) AS num_habitatges
}
WITH num_habitatges
MATCH (h:Habitatge {municipi: 'SFLL', any_padro: 1881})
MATCH(h)<-[:VIU{any_padro:1881}]-(:Individu)-[r:RELACIO]->(f:Individ
u)-[:VIU {any_padro: 1881}]->(h)
```

```
WHERE toLower(r.tipus) CONTAINS "fill"
WITH num_habitatges, count(DISTINCT f) AS total_fills
RETURN num_habitatges,total_fills,toFloat(total_fills)/num_habitatge
s AS mitjana_fills_per_habitatge
```

Consulta e: Famílies amb més de tres fills a Castellví de Rosanes

```
MATCH(i:Individu)-[:VIU]->(:Habitatge{municipi:"CR"}),(i)-[:RELACIO]
->(i)
where not exists{(i)<-[:SAME_AS]-(:Individu)}
match(i)-[r:RELACIO]->(f:Individu)
WHERE toLower(r.tipus) CONTAINS "fill"
with i, count(*) as fills
return "Familia "+i.cognom1+" " + i.cognom2 as Familia, fills
order by fills DESC
LIMIT 20
```

Consulta f: Carrers amb menys habitants per any a Sant Feliu de Llobregat

```
MATCH (i:Individu)-[:VIU]->(h:Habitatge {municipi: "SFLL"})
WHERE h.carrer IS NOT NULL AND h.any_padro IS NOT NULL
WITH h.any_padro AS year, h.carrer AS carrer, count(DISTINCT i) AS
habitants
CALL {
  WITH year
  MATCH (i2:Individu)-[:VIU]->(h2:Habitatge {municipi: "SFLL"})
  WHERE h2.carrer IS NOT NULL AND h2.any_padro = year
  WITH h2.carrer AS carrer2, count(DISTINCT i2) AS habitants2
  RETURN min(habitants2) AS minHabitants
}
WITH year, carrer, habitants, minHabitants
WHERE habitants = minHabitants
WITH year, carrer, habitants
ORDER BY year, carrer
WITH year, head(collect({carrer: carrer, habitants: habitants})) AS
calleMin
RETURN year, calleMin.carrer AS carrer, calleMin.habitants AS
minHabitants
ORDER BY year
```

Exercici 3. Analítica de Grafs:

a) Anàlisi de components connexes

Execució de l'algorisme en mode *stream*:

```
CALL gds.wcc.stream($generatedName, $config) YIELD nodeId, componentId AS community
WITH gds.util.asNode(nodeId) AS node, community
WITH collect(node) AS allNodes, community
RETURN community, allNodes[0..$communityNodeLimit] AS nodes, size(allNodes) AS size
ORDER BY size DESC
LIMIT 1;
```

a.1) Identificació de les components connexes més grans

```
CALL gds.wcc.stream($generatedName, $config)
YIELD nodeId, componentId AS community
WITH gds.util.asNode(nodeId) AS node, community
WITH community, collect(node) AS allNodes
RETURN community, size(allNodes) AS size
ORDER BY size DESC
LIMIT 10;
```

a.2) Quantitat de components sense cap node habitatge

```
CALL gds.wcc.stream($generatedName, $config)
YIELD nodeId, componentId AS community
WITH community, collect(gds.util.asNode(nodeId)) AS nodes
WITH community, [node IN nodes WHERE 'Habitatge' IN labels(node)] AS habitatges
WHERE size(habitatges) = 0
RETURN count(community) AS components_sense_habitatge;
```

Exercici 4. Comparativa sobre diferents esquemes de base de dades:

SCRIPT D'IMPORTACIÓ

```
//Creamos restriccion de año y de individuo
CREATE CONSTRAINT padro_year_unique IF NOT EXISTS
FOR (p:Padro)
REQUIRE p.year IS UNIQUE;
CREATE CONSTRAINT individu_id_unique IF NOT EXISTS
FOR (i:Individu)
REQUIRE i.id IS UNIQUE;

//Creamos individuo y año
LOAD CSV WITH HEADERS FROM 'file:///INDIVIDUAL.csv' AS row
WITH row
WHERE row.Id IS NOT NULL AND row.Year IS NOT NULL
WITH row, ToInteger(row.Year) AS year
MERGE (p:Padro {year: year})
```

```
MERGE (i:Individu {id: row.Id})
SET i.nom = row.name,
    i.cognom = row.surname,
    i.segon_cognom = row.second_surname,
    i.year = year

//Convertimos año en una lista encadenada
MATCH (p:Padro)
WITH p ORDER BY p.year
WITH collect(p) AS years
UNWIND range(0, size(years)-2) AS i
WITH years[i] AS from, years[i+1] AS to
MERGE (from)-[:NEXT]->(to);

//Carreguem SAME_AS
LOAD CSV WITH HEADERS FROM 'file:///SAME_AS.csv' AS row
WITH row
WHERE row.Id_A IS NOT NULL AND row.Id_B IS NOT NULL

MATCH (a:Individu {id: row.Id_A})
MATCH (b:Individu {id: row.Id_B})
MERGE (a)-[:SAME_AS]->(b)



//Creem cliques relació
MATCH (n:Individu)
WHERE n.componentId IS NULL
CALL apoc.path.subgraphNodes(n, {relationshipFilter: 'SAME_AS'}) YIELD node
WITH n, collect(node) AS component
WITH component, id(n) AS cid
UNWIND component AS x
SET x.componentId = cid


//Creem nodes de referencia
MATCH (a:Individu)
WITH a.componentId AS group, a,
     size([x IN [a.Id, a.Year, a.name, a.surname, a.second_surname] WHERE x IS NOT
NULL]) AS filled
ORDER BY filled DESC
WITH group, collect(a)[0] AS rep

SET rep:Representant

//Creem la aresta referencia
MATCH (n:Individu)
MATCH (rep:Representant)
WHERE n.componentId = rep.componentId
MERGE (rep)-[:REPRESENTA]->(n)

//Borramos same as, componentId y la autoaresta
MATCH ()-[r:SAME_AS]-()
DELETE r;

MATCH (n)
REMOVE n.componentId

//importamos familia filtrando malos
```

```
LOAD CSV WITH HEADERS FROM 'file:///FAMILIA.csv' AS row
WITH row
WHERE row.ID_1 IS NOT NULL AND row.ID_2 IS NOT NULL
  AND row.Relacio_Harmonitzada IS NOT NULL
  AND NOT row.Relacio_Harmonitzada IN ['null', 'ala', 'al']

MATCH (a:Individu {id: row.ID_1})<-[:REPRESENTA]-(repA:Representant)
MATCH (b:Individu {id: row.ID_2})<-[:REPRESENTA]-(repB:Representant)

MERGE (repA)-[r:FAMILIA]->(repB)
SET r.relacio = row.Relacio_Harmonitzada

//importamos ubicación y habitatges y conexion entre ellos y con los años
LOAD CSV WITH HEADERS FROM 'file:///HABITATGES.csv' AS row
MERGE (u:Ubicacio {carrer: row.Carrer, numero: row.Numero, municipi: row.Municipi})
MERGE (h:Habitatge {id: row.Id_Llar, year: toInteger(row.Any_Padro), municipi:
row.Municipi})
MERGE (h)-[:ESTA_A]->(u)
WITH h, row

MATCH (a:Padro {year: toInteger(row.Any_Padro)})
MERGE (h)-[:DEL_ANY]->(a)

//Conectamos habitatge con los individuos
LOAD CSV WITH HEADERS FROM 'file:///VIU.csv' AS row
WITH row
MATCH (h:Habitatge{id:row.HOUSE_ID, year:toInteger(row.Year), municipi:
row.Location})
MATCH (:Individu {id: row.IND})<-[:REPRESENTA]-(repA:Representant)
MERGE (repA)-[:VIU_EN]->(h)

//Creamos arestas para individuos no conectados
MATCH (i:Individu)<-[:REPRESENTA]-(rep:Representant)
WHERE NOT EXISTS ((rep)-[:VIU_EN]->(:Habitatge))
WITH rep, i.year AS y
MERGE (p:Padro {year: y})
MERGE (rep)-[:NO_UBICAT]->(p)

//Creamos la segunda etiqueta de ubicacio
MATCH (u:Ubicacio)
CALL apoc.create.addLabels(u, [u.municipi]) YIELD node
REMOVE node.municipi

//Borramos los atributos redundantes
MATCH (h:Habitatge)
REMOVE h.municipi, h.year
```

CONSULTES CYPHER

Model nou:

MATCH (rep:Representant)-[:VIU_EN]->(:Habitatge)-[:ESTA_A]->(u:Ubicacio)

```
WITH rep, size(collect(u)) AS n_ubicacions,  collect(DISTINCT u) AS ubicacions
WHERE size(ubicacions) = 1 AND n_ubicacions > 1
RETURN size(collect(rep)) AS SEDENTARIS
```

Model original:

```
MATCH (p:Person)

OPTIONAL MATCH (p)-[:SAME_AS]-(q:Person)
WITH p, collect(q) + p AS group_nodes
WITH [x IN group_nodes WHERE x IS NOT NULL | x.id] AS ids, group_nodes
WITH apoc.coll.sort(ids)[0] AS group_id, collect(DISTINCT p) AS persons
UNWIND persons AS person
MATCH (person)-[:VIU]->(h:Habitatge)
WITH group_id, collect(DISTINCT h.municipi + '|' + h.carrer + '|' + h.numero) AS adreces
WHERE size(adreces) = 1
RETURN count(*) AS SEDENTARIS
```

Persones que apareixen a dos padrons consecutius:

Model nou:

S'identifiquen els individus que viuen en habitatges associats a anys consecutius, utilitzant la relació `NEXT` per establir la continuïtat temporal.

```
MATCH (p1:Padro {year: $any_padro})-[:NEXT]->(p2:Padro)
MATCH (rep:Representant)-[:VIU_EN]->(:Habitatge)-[:DEL_ANY]->(p1)
WITH p1, p2, collect(DISTINCT rep) AS reps1
MATCH (rep:Representant)-[:VIU_EN]->(:Habitatge)-[:DEL_ANY]->(p2)
WITH reps1, collect(DISTINCT rep) AS reps2
WITH reps1, reps2,
    [r IN reps1 WHERE r IN reps2] AS enAmbdos
RETURN
  size(enAmbdos) AS persistents,
  size(reps1) AS primer_any,
  toFloat(size(enAmbdos)) / size(reps1) * 100 AS percentatge
```

Model original:

```
MATCH (:Habitatge)
WHERE any_padro > year_actual
WITH year_actual, min(any_padro) AS year_seguent
MATCH (p:Person)-[:VIU]->(h1:Habitatge)
WHERE h1.any_padro = year_actual
WITH DISTINCT p, year_actual, year_seguent
OPTIONAL MATCH (p)-[:SAME_AS]-(q:Person)
WITH p, year_actual, year_seguent, collect(q.id) + p.id AS ids
WITH reduce(min_id = p.id, x IN ids | CASE WHEN x < min_id THEN x ELSE min_id END)
AS group_id, ids, year_actual, year_seguent
WITH group_id, ids, year_actual, year_seguent
```

```
WITH collect(DISTINCT group_id) AS grups_originals, year_actual, year_seguent
UNWIND grups_originals AS gid
MATCH (p:Person)-[:SAME_AS*0..]-(p_equiv:Person)
WHERE p.id = gid
WITH DISTINCT gid, collect(DISTINCT p_equiv.id) AS all_ids
UNWIND all_ids AS pid
MATCH (p2:Person {id: pid})-[:VIU]->(h2:Habitatge)
WHERE h2.any_padro = year_seguent
WITH DISTINCT gid AS grups_en_ambdós, count(DISTINCT gid) AS n_ambdós
WITH n_ambdós, count(*) + 0.0 AS total_grups
RETURN
  total_grups AS total_cens_inicial, n_ambdós AS persones_amb_cens_repetit,
  round(n_ambdós * 100.0 / total_grups, 2) AS percentatge;
```