Potentially Useful Queries

(Good for learning CYPHER. Works for 20240522 Version of FSLR Test Database)  
(Teresa Le)

# Simple Queries

////Return all Heat nodes where the heat temperature is less than or equal to 150 C and greater than 100C

MATCH (n:Heat)

WHERE 150 >= n.`Temp (C)` > 100

RETURN n

//// Solubility Testing: Return the list of ingredients in an Operation:Mix:MakePrecursor node with data pertaining to whether or not the ingredients dissolved. Columns: Ingredient 1, Molarity of Ing 1, Ingredient 2, Dissolve Result. Arrange in alphabetical order of first ingredient. Omit duplicate data.

MATCH (op:Mix:MakePrecursor) WHERE op.`Dissolved` IS NOT NULL

WITH op ORDER BY op.`Ing 1 Description` ASC

Return DISTINCT op.`Ing 1 Description` as Ing1, op.`Ing 1 Target Molarity` as `Ing1 Molarity`, op.`Ing 2 Description` as Ing2, op.`Dissolved` as Dissolved

//// Similar to above, but returns the results of solubility testing for heating operation. Also includes volume of acid because it affects how long you need to heat.

MATCH(m:Mix:MakePrecursor) -[:CREATED]-> (p:Precursor) -[:WENT\_TO]-> (h:Heat)

WHERE h.`Dissolved` IS NOT NULL

WITH m,h ORDER BY m.`Ing 1 Description` ASC

RETURN DISTINCT m.`Ing 1 Description` as Ing1, m.`Ing 1 Target Molarity` as `Molarity (M)`, m.`Ing 2 Description` as Ing2, m.`Ing 2 Actual Amt` as `Vol (mL)`,h.`Temp (C)` as `Temp (C)`, h.`Time (min)` as `Time (min)`, h.`Dissolved` as Dissolved

//// Similar to above, but only returns the results for the ones where Ing1 contains ‘Antimony’

MATCH(m:Mix:MakePrecursor) -[:CREATED]-> (p:Precursor) -[:WENT\_TO]-> (h:Heat)

WHERE h.`Dissolved` IS NOT NULL AND m.`Ing 1 Description` CONTAINS 'Antimony'

WITH m,h ORDER BY m.`Ing 1 Description` ASC

RETURN DISTINCT m.`Ing 1 Description` as Ing1, m.`Ing 1 Target Molarity` as `Molarity (M)`, m.`Ing 2 Description` as Ing2, m.`Ing 2 Actual Amt` as `Vol (mL)`, h.`Dissolved` as Dissolved

// Returns solubility results for the Rest operation

MATCH(m:Mix:MakePrecursor) -[:CREATED]-> (p:Precursor) -[:WENT\_TO]-> (h:Heat) -[:THEN]-> (r:Rest)

WHERE r.`Dissolved` IS NOT NULL

WITH m,r ORDER BY m.`Ing 1 Description` ASC

RETURN DISTINCT m.`Ing 1 Description` as Ing1, m.`Ing 1 Target Molarity` as `Molarity (M)`, m.`Ing 2 Description` as Ing2, m.`Ing 2 Actual Amt` as `Vol (mL)`, r.`Time (min)` as `Time (min)`, r.`Dissolved` as Dissolved

# More Complicated Queries

// WARNING: Can be a very expensive query depending on how you set it up.  
// Returns full list of OTC ingredients ever used to create target material CsBiBr4. 5 relationships in between, max, and direction must go from ingredient to sample.

MATCH (OTC: `OTC Ingredient`)-[\*0..5]->(s:Sample)

WHERE s.`Target Material` = "CsBiBr4"

Return DISTINCT OTC.`Description` as `OTC Ingredient`

// Returns the same thing as above.

MATCH (OTC: `OTC Ingredient`)--{1,5}(s:Sample)

WHERE s.`Target Material` = "CsBiBr4"

Return DISTINCT OTC.`Description` as `OTC Ingredient`

//// Return group of all non-'OTC Ingredient' and non-‘Operation’ nodes that went to an operation. Also return the operation alongside the nodes. Nodes with more than one operation listed more than once in the table.

// Find all nodes that are not `OTC Ingredient` and do not contain Operation as a label

MATCH (n) WHERE NOT n:Operation AND NOT n:`OTC Ingredient`

// Within the subgroup, find ones that went\_to an operation

MATCH (n)-[r:WENT\_TO]->(op:Operation)

// Order by chronological order of operations. ORDER BY requires WITH (or other statement) to proceed

WITH n,op ORDER BY op.`Timestamp`

// Return list

RETURN n AS Object, op AS Operation

//// Return group of all non-'OTC Ingredient' and non-‘Operation’ nodes that went to operations. Collect all operations belonging to same node.

// Find all nodes that are not `OTC Ingredient` and do not contain Operation as a label

MATCH (n) WHERE NOT n:Operation AND NOT n:`OTC Ingredient`

// Within the subgroup, find ones that went\_to an operation

MATCH (n)-[r:WENT\_TO]->(op:Operation)

// Order by chronological order of operations. ORDER BY requires WITH (or other statement) to proceed

WITH n,op ORDER BY op.`Timestamp`

// Collect all the operations belonging to each node

WITH n, COLLECT(op) AS Operations

// Return list

RETURN n AS Object, Operations

// WARNING: Can be a very expensive query depending on how you set it up.

// Searches for every sample ever attempted containing 'Cs' in the name. Returns the list of unique materials ever used to make such samples. Restricts search to 5 relationships in between nodes and direction must go from ingredient to sample.

MATCH (OTC: `OTC Ingredient`)-[\*0..5]->(s:Sample)

WHERE s.`Target Material` CONTAINS 'Cs'

WITH s,OTC ORDER BY s.`Target Material`

WITH s, COLLECT(DISTINCT OTC.`Description`) as `OTC Ing`

Return DISTINCT s.`Target Material` as `Target Material`, `OTC Ing`

// Return list of all precursors, and their IDs, that went straight to a mixing operation without any preceding operation.

MATCH (p:Precursor) -[:WENT\_TO]->(m:Mix)

WHERE NOT (m)<-[:THEN]-()

WITH p ORDER BY p.`Precursor ID` ASC

Return p.`Description` as Precursor, p.`Precursor ID` as ID

// Return the full path from OTC ingredient to the Cs2SbBr5 sample

MATCH path=((otc:`OTC Ingredient`) -[\*]-> (s:`Sample`))

WHERE s.`Target Material` = "Cs2SbBr5"

Return path