**Neo4j — Load Sample Car Data & Run the CQ (Paste‑and‑Go)**

Generated: 2025-08-31 22:03 UTC

This guide shows three ways to load the tiny car dataset into Neo4j and run the competency question (CQ). Use Option A for the fastest paste‑and‑go.

**Dataset (RDBMS-style tables, two rows each)**

vehicle\_model: M1 FALCON\_X 'Falcon X' | M2 ROADSTER 'Roadster'  
trim: T101 M1 2023 | T102 M1 2023  
feature: F1 ACC 'Adaptive Cruise Control' | F2 HUD 'Head-Up Display'  
trim\_feature: (T101,F1) | (T102,F2)  
region: CA 'California' | TX 'Texas'  
sale\_record: S1 T101 CA 2023-03-15 120 | S2 T102 CA 2023-03-15 55

**Option A — Paste this in Neo4j Browser (fastest)**

Open http://localhost:7474/browser and paste each block in order. The statements are idempotent (MERGE), so you can re-run safely.

A1) (Optional) Clear your test graph:

MATCH (n) DETACH DELETE n;

A2) Create uniqueness constraints (recommended for quality & speed):

CREATE CONSTRAINT vehicle\_model\_code IF NOT EXISTS FOR (n:VehicleModel) REQUIRE n.code IS UNIQUE;  
CREATE CONSTRAINT trim\_code IF NOT EXISTS FOR (n:Trim) REQUIRE n.code IS UNIQUE;  
CREATE CONSTRAINT feature\_code IF NOT EXISTS FOR (n:Feature) REQUIRE n.code IS UNIQUE;  
CREATE CONSTRAINT region\_code IF NOT EXISTS FOR (n:Region) REQUIRE n.code IS UNIQUE;  
CREATE CONSTRAINT sale\_id IF NOT EXISTS FOR (n:SaleRecord) REQUIRE n.id IS UNIQUE;

A3) Load nodes and relationships:

// Vehicle models  
UNWIND [  
 {code:'FALCON\_X', name:'Falcon X'},  
 {code:'ROADSTER', name:'Roadster'}  
] AS m  
MERGE (vm:VehicleModel {code:m.code})  
SET vm.name = m.name;  
  
// Trims + BELONGS\_TO  
UNWIND [  
 {code:'T101', model:'FALCON\_X', modelYear:2023},  
 {code:'T102', model:'FALCON\_X', modelYear:2023}  
] AS t  
MERGE (tr:Trim {code:t.code})  
SET tr.modelYear = t.modelYear  
WITH tr, t  
MATCH (vm:VehicleModel {code:t.model})  
MERGE (tr)-[:BELONGS\_TO]->(vm);  
  
// Features  
UNWIND [  
 {code:'ACC', prefLabel:'Adaptive Cruise Control'},  
 {code:'HUD', prefLabel:'Head-Up Display'}  
] AS f  
MERGE (fe:Feature {code:f.code})  
SET fe.prefLabel = f.prefLabel;  
  
// Trim—HAS\_FEATURE  
UNWIND [  
 {trim:'T101', feature:'ACC'},  
 {trim:'T102', feature:'HUD'}  
] AS tf  
MATCH (t:Trim {code:tf.trim})  
MATCH (f:Feature {code:tf.feature})  
MERGE (t)-[:HAS\_FEATURE]->(f);  
  
// Regions  
UNWIND [  
 {code:'CA', name:'California'},  
 {code:'TX', name:'Texas'}  
] AS r  
MERGE (rg:Region {code:r.code})  
SET rg.name = r.name;  
  
// Sale records + FOR\_TRIM + SOLD\_IN  
UNWIND [  
 {id:'S1', trim:'T101', region:'CA', saleDate:date('2023-03-15'), quantity:120},  
 {id:'S2', trim:'T102', region:'CA', saleDate:date('2023-03-15'), quantity:55}  
] AS s  
MERGE (sr:SaleRecord {id:s.id})  
SET sr.saleDate = s.saleDate, sr.quantity = s.quantity  
WITH sr, s  
MATCH (t:Trim {code:s.trim})  
MATCH (r:Region {code:s.region})  
MERGE (sr)-[:FOR\_TRIM]->(t)  
MERGE (sr)-[:SOLD\_IN]->(r);

A4) Sanity checks:

MATCH (n) RETURN labels(n) AS label, count(\*) AS cnt ORDER BY cnt DESC;  
MATCH ()-[r]->() RETURN type(r) AS rel, count(\*) AS cnt;

A5) Visualize in the Graph tab:

MATCH p=(t:Trim)-[\*1..2]->() RETURN p LIMIT 50;

A6) Run the CQ (typed, directed traversal + aggregate):

MATCH (:VehicleModel {code:'FALCON\_X'})<-[:BELONGS\_TO]-(t:Trim {modelYear:2023})  
MATCH (t)-[:HAS\_FEATURE]->(:Feature {code:'ACC'})  
MATCH (sr:SaleRecord)-[:FOR\_TRIM]->(t)  
MATCH (sr)-[:SOLD\_IN]->(:Region {code:'CA'})  
WHERE sr.saleDate = date('2023-03-15')  
RETURN t.code AS trim, sum(sr.quantity) AS units  
ORDER BY units DESC;

**Option B — LOAD CSV (files in Neo4j /import)**

Create CSV files under Neo4j’s import directory and run these. Example contents:

-- vehicle\_model.csv  
id,code,name  
M1,FALCON\_X,Falcon X  
M2,ROADSTER,Roadster  
  
-- trim.csv  
id,model\_id,model\_year,code  
T101,M1,2023,T101  
T102,M1,2023,T102  
  
-- feature.csv  
id,code,pref\_label  
F1,ACC,Adaptive Cruise Control  
F2,HUD,Head-Up Display  
  
-- trim\_feature.csv  
trim\_id,feature\_id  
T101,F1  
T102,F2  
  
-- region.csv  
id,code,name  
CA,CA,California  
TX,TX,Texas  
  
-- sale\_record.csv  
id,trim\_id,region\_id,sale\_date,quantity  
S1,T101,CA,2023-03-15,120  
S2,T102,CA,2023-03-15,55

Then run LOAD CSV:

USING PERIODIC COMMIT  
  
LOAD CSV WITH HEADERS FROM 'file:///vehicle\_model.csv' AS row  
MERGE (m:VehicleModel {code: row.code})  
SET m.name = row.name;  
  
LOAD CSV WITH HEADERS FROM 'file:///trim.csv' AS row  
MERGE (t:Trim {code: row.code})  
SET t.modelYear = toInteger(row.model\_year)  
WITH t, row  
MATCH (m:VehicleModel {code: 'FALCON\_X'}) WHERE row.model\_id = 'M1'  
MERGE (t)-[:BELONGS\_TO]->(m);  
  
LOAD CSV WITH HEADERS FROM 'file:///feature.csv' AS row  
MERGE (f:Feature {code: row.code})  
SET f.prefLabel = row.pref\_label;  
  
LOAD CSV WITH HEADERS FROM 'file:///trim\_feature.csv' AS row  
MATCH (t:Trim {code: row.trim\_id}), (f:Feature {code: row.feature\_id})  
MERGE (t)-[:HAS\_FEATURE]->(f);  
  
LOAD CSV WITH HEADERS FROM 'file:///region.csv' AS row  
MERGE (r:Region {code: row.code})  
SET r.name = row.name;  
  
LOAD CSV WITH HEADERS FROM 'file:///sale\_record.csv' AS row  
MERGE (sr:SaleRecord {id: row.id})  
SET sr.saleDate = date(row.sale\_date),  
 sr.quantity = toInteger(row.quantity)  
WITH sr, row  
MATCH (t:Trim {code: row.trim\_id})  
MATCH (r:Region {code: row.region\_id})  
MERGE (sr)-[:FOR\_TRIM]->(t)  
MERGE (sr)-[:SOLD\_IN]->(r);

**Option C — RDF/Turtle (n10s plugin)**

If you enabled the n10s plugin, you can import the Turtle and still query with Cypher.

CALL n10s.graphconfig.init();  
  
:param ttl => "  
@prefix ex: <http://example.com/auto#> .  
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .  
  
ex:M1 a ex:VehicleModel ; ex:code \"FALCON\_X\" .  
  
ex:T101 a ex:Trim ; ex:belongsTo ex:M1 ; ex:modelYear \"2023\"^^xsd:gYear ; ex:hasFeature ex:F\_ACC .  
ex:T102 a ex:Trim ; ex:belongsTo ex:M1 ; ex:modelYear \"2023\"^^xsd:gYear ; ex:hasFeature ex:F\_HUD .  
  
ex:F\_ACC a ex:Feature ; ex:prefLabel \"Adaptive Cruise Control\" .  
ex:F\_HUD a ex:Feature ; ex:prefLabel \"Head-Up Display\" .  
  
ex:S1 a ex:SaleRecord ; ex:forTrim ex:T101 ; ex:soldIn ex:CA ;  
 ex:saleDate \"2023-03-15\"^^xsd:date ; ex:quantity 120 .  
  
ex:S2 a ex:SaleRecord ; ex:forTrim ex:T102 ; ex:soldIn ex:CA ;  
 ex:saleDate \"2023-03-15\"^^xsd:date ; ex:quantity 55 .  
";  
CALL n10s.rdf.import.inline($ttl, 'Turtle');

Run the same CQ Cypher query from Option A step A6 to verify results.

Fix Data steps

MATCH (n:SaleRecord|Trim|VehicleModel|Feature|Region)

DETACH DELETE n;

* Deleted 10 nodes, deleted 6 relationships

### 2) (Optional) keep or drop constraints

If your **unique keys are unchanged** (code on VehicleModel/Trim/Feature/Region; id on SaleRecord), **keep constraints**.  
If you changed keys and need to recreate constraints:

DROP CONSTRAINT vehicle\_model\_code IF EXISTS;

DROP CONSTRAINT trim\_code IF EXISTS;

DROP CONSTRAINT feature\_code IF EXISTS;

DROP CONSTRAINT region\_code IF EXISTS;

DROP CONSTRAINT sale\_id IF EXISTS;

CREATE CONSTRAINT vehicle\_model\_code IF NOT EXISTS FOR (n:VehicleModel) REQUIRE n.code IS UNIQUE;

CREATE CONSTRAINT trim\_code IF NOT EXISTS FOR (n:Trim) REQUIRE n.code IS UNIQUE;

CREATE CONSTRAINT feature\_code IF NOT EXISTS FOR (n:Feature) REQUIRE n.code IS UNIQUE;

CREATE CONSTRAINT region\_code IF NOT EXISTS FOR (n:Region) REQUIRE n.code IS UNIQUE;

CREATE CONSTRAINT sale\_id IF NOT EXISTS FOR (n:SaleRecord) REQUIRE n.id IS UNIQUE;

**3) Re-load the corrected data (idempotent MERGE)**

You can paste the whole block. I’ve added a small tip: tag nodes with dataset:'car\_demo' so future cleanups can target only this data.

// Vehicle models

UNWIND [

{code:'FALCON\_X', name:'Falcon X'},

{code:'ROADSTER', name:'Roadster'}

] AS m

MERGE (vm:VehicleModel {code:m.code})

SET vm.name = m.name,

vm.dataset = 'car\_demo';

// Trims + BELONGS\_TO

UNWIND [

{code:'T101', model:'FALCON\_X', modelYear:2023},

{code:'T102', model:'FALCON\_X', modelYear:2023}

] AS t

MERGE (tr:Trim {code:t.code})

SET tr.modelYear = toInteger(t.modelYear),

tr.dataset = 'car\_demo'

WITH tr, t

MATCH (vm:VehicleModel {code:t.model})

MERGE (tr)-[:BELONGS\_TO]->(vm);

// Features

UNWIND [

{code:'ACC', prefLabel:'Adaptive Cruise Control'},

{code:'HUD', prefLabel:'Head-Up Display'}

] AS f

MERGE (fe:Feature {code:f.code})

SET fe.prefLabel = f.prefLabel,

fe.dataset = 'car\_demo';

// Trim—HAS\_FEATURE

UNWIND [

{trim:'T101', feature:'ACC'},

{trim:'T102', feature:'HUD'}

] AS tf

MATCH (t:Trim {code:tf.trim})

MATCH (f:Feature {code:tf.feature})

MERGE (t)-[:HAS\_FEATURE]->(f);

// Regions

UNWIND [

{code:'CA', name:'California'},

{code:'TX', name:'Texas'}

] AS r

MERGE (rg:Region {code:r.code})

SET rg.name = r.name,

rg.dataset = 'car\_demo';

// Sale records + FOR\_TRIM + SOLD\_IN

UNWIND [

{id:'S1', trim:'T101', region:'CA', saleDate:date('2023-03-15'), quantity:120},

{id:'S2', trim:'T102', region:'CA', saleDate:date('2023-03-15'), quantity:55}

] AS s

MERGE (sr:SaleRecord {id:s.id})

SET sr.saleDate = date(s.saleDate),

sr.quantity = toInteger(s.quantity),

sr.dataset = 'car\_demo'

WITH sr, s

MATCH (t:Trim {code:s.trim})

MATCH (r:Region {code:s.region})

MERGE (sr)-[:FOR\_TRIM]->(t)

MERGE (sr)-[:SOLD\_IN]->(r);

### 4) Verify & run your CQ

// sanity counts

MATCH (n) RETURN labels(n) AS label, count(\*) AS cnt ORDER BY cnt DESC;

MATCH ()-[r]->() RETURN type(r) AS rel, count(\*) AS cnt;

// the CQ

MATCH (:VehicleModel {code:'FALCON\_X'})<-[:BELONGS\_TO]-(t:Trim {modelYear:2023})

MATCH (t)-[:HAS\_FEATURE]->(:Feature {code:'ACC'})

MATCH (sr:SaleRecord)-[:FOR\_TRIM]->(t)

MATCH (sr)-[:SOLD\_IN]->(:Region {code:'CA'})

WHERE date(sr.saleDate) = date('2023-03-15') // robust if saleDate was a string

RETURN t.code AS trim, sum(sr.quantity) AS units

ORDER BY units DESC;

**Option B — One-shot reset + reload in a single transaction**

If you want the delete + reload to be **atomic**:

:begin

MATCH (n:SaleRecord|Trim|VehicleModel|Feature|Region) DETACH DELETE n;

CREATE CONSTRAINT vehicle\_model\_code IF NOT EXISTS FOR (n:VehicleModel) REQUIRE n.code IS UNIQUE;

CREATE CONSTRAINT trim\_code IF NOT EXISTS FOR (n:Trim) REQUIRE n.code IS UNIQUE;

CREATE CONSTRAINT feature\_code IF NOT EXISTS FOR (n:Feature) REQUIRE n.code IS UNIQUE;

CREATE CONSTRAINT region\_code IF NOT EXISTS FOR (n:Region) REQUIRE n.code IS UNIQUE;

CREATE CONSTRAINT sale\_id IF NOT EXISTS FOR (n:SaleRecord) REQUIRE n.id IS UNIQUE;

// (paste the entire reload block from Option A step 3 here)

:commit