**Constraints & Indexes**

CREATE CONSTRAINT restaurantIdConstraint FOR (restaurant:Restaurant) REQUIRE restaurant.id IS UNIQUE

CREATE CONSTRAINT cuisineNameConstraint FOR (cuisine:Cuisine) REQUIRE cuisine.name IS UNIQUE

CREATE CONSTRAINT customerNameConstraint FOR (customer:Customer) REQUIRE customer.name IS UNIQUE

CREATE CONSTRAINT dishNameConstraint FOR (dish:Dish) REQUIRE dish.name IS UNIQUE

**Import query for restaurants in San Mateo from csv**

LOAD CSV WITH HEADERS FROM "https://raw.githubusercontent.com/VindhyaKeshav/addresses/main/restaurants\_sanmateo.csv" AS csvLine

WITH csvLine, replace(replace(csvLine.location, 'POINT (', ''), ')', '') AS cleanedLocation

WITH csvLine, split(cleanedLocation, ' ') AS coordinates

WITH csvLine, toFloat(coordinates[0]) AS x, toFloat(coordinates[1]) AS y

CREATE (r:Restaurant {id: toInteger(csvLine.id), name: csvLine.name, address:csvLine.address, city:csvLine.city, zip:csvLine.zip, location:point({x: x, y: y}) })

**Import query for customers from csv**

LOAD CSV WITH HEADERS FROM "https://raw.githubusercontent.com/VindhyaKeshav/addresses/main/customers.csv" AS csvLine

CREATE (c:Customer {id: toInteger(csvLine.id), name: csvLine.name, email:csvLine.email, totalReviews:csvLine.totalReviews, totalFollowers:csvLine.totalFollowers })

**Import query for cuisines from csv**

LOAD CSV WITH HEADERS FROM "https://raw.githubusercontent.com/VindhyaKeshav/addresses/main/cuisines.csv" AS csvLine

MERGE (c:Cuisine {id: apoc.create.uuid(), name: csvLine.name })

**Import query for “restaurant [SERVES] dishes” relationship from csv**

LOAD CSV WITH HEADERS FROM "https://raw.githubusercontent.com/VindhyaKeshav/addresses/main/restaurants\_dishes.csv" AS csvLine

MATCH(r:Restaurant{id:toInteger(csvLine.id)})

WITH r, csvLine

UNWIND range(0,99) as index

WITH r, csvLine, index

WHERE csvLine["dishes["+index+"]"] IS NOT NULL

MERGE (d:Dish{ name:csvLine["dishes["+index+"]"]})

CREATE (r)-[:SERVES {price:csvLine["prices["+index+"]"]}]->(d)

**Import query for riders from csv**

LOAD CSV WITH HEADERS FROM "https://raw.githubusercontent.com/VindhyaKeshav/addresses/main/rider.csv" AS csvLine

MERGE (r:Rider {id: toInteger(csvLine.id), name: csvLine.name,

email:csvLine.email,gender:csvLine.gender, vehicleType:csvLine.vehicleType, vehicleNumber:csvLine.vehicleNumber, location:point({x: toFloat(csvLine.x), y: toFloat(csvLine.y)}), status:csvLine.status })

**Import query for “dish [BELONGSTO] cuisine” relationship from csv**

LOAD CSV WITH HEADERS FROM "https://raw.githubusercontent.com/VindhyaKeshav/addresses/main/cuisine\_dishes.csv" AS csvLine

WITH csvLine

UNWIND range(0,19) as index

WITH  csvLine, index

MATCH(d:Dish{name:csvLine["dishes["+index+"]"]})

MATCH (c:Cuisine{ name:csvLine.name})

CREATE (d)-[:BELONGSTO]->(c)

**Import query for “customer [REVIEWED] restaurant” relationship from csv**

LOAD CSV WITH HEADERS FROM "https://raw.githubusercontent.com/VindhyaKeshav/addresses/main/restaurant\_reviews.csv" AS csvLine

WITH csvLine

MATCH (r:Restaurant{id:toInteger(csvLine.restaurantId)})

WITH csvLine, r

MATCH (c:Customer{name:csvLine.customerName})

WITH csvLine, r, c

CREATE (c)-[:REVIEWED{review:csvLine.review, rating:toInteger(csvLine.rating), costForTwo:toInteger(csvLine.cost\_for\_two), time:csvLine.time}]->(r)

**Import query for “customer [REVIEWED] restaurant” relationship from csv**

CALL gds.graph.project(

      'sanMateo',

      'Intersection',

      'ROAD\_SEGMENT',

      {

        relationshipProperties: 'length'

      }

    )

**Import query to create relationship of rider with nearest located intersection from csv**

CALL apoc.periodic.iterate(

  'MATCH (p:Rider) WHERE NOT EXISTS ((p)-[:NEAREST\_INTERSECTION]->(:Intersection)) RETURN p',

  'CALL {

  WITH p

  MATCH (i:Intersection)

  USING INDEX i:Intersection(location)

  WHERE point.distance(i.location, p.location) < 200

  WITH i

  ORDER BY point.distance(p.location, i.location) ASC

  LIMIT 1

  RETURN i

}

WITH p, i

MERGE (p)-[r:NEAREST\_INTERSECTION]->(i)

SET r.length = point.distance(p.location, i.location)

RETURN COUNT(p)',

  {batchSize:1000, parallel:false})

**Import query to create relationship of restaurant with nearest located intersection from csv**

CALL apoc.periodic.iterate(

  'MATCH (p:Restaurant) WHERE NOT EXISTS ((p)-[:NEAREST\_INTERSECTION]->(:Intersection)) RETURN p',

  'CALL {

  WITH p

  MATCH (i:Intersection)

  USING INDEX i:Intersection(location)

  WHERE point.distance(i.location, p.location) < 200

  WITH i

  ORDER BY point.distance(p.location, i.location) ASC

  LIMIT 1

  RETURN i

}

WITH p, i

MERGE (p)-[r:NEAREST\_INTERSECTION]->(i)

SET r.length = point.distance(p.location, i.location)

RETURN COUNT(p)',

  {batchSize:1000, parallel:false})