# NODES

## K3S

Load csv with headers from "file:///k3S\_WK.csv" as row

//return row

Merge (n:K3S{id:row.no}) on create set

n.nama = row.nama,

n.ons\_or\_off= row.ons\_or\_off,

n.status = row.status,

n.wilayah\_kerja = row.wilayah\_kerja

## Vendor

load csv with headers from "file:///vendor.csv" as row

merge (v:Vendor{id:row.no}) on create set

v.telepon\_or\_fax = row.telepon\_or\_fax,

v.skt = row.skt,

v.nama = row.perusahaan,

v.sub\_bidang\_usaha = row.sub\_bidang\_usaha,

v.alamat = row.alamat,

v.k3s\_id = row.k3s\_id

//new script

load csv with headers from "file:///test.csv" as row

//return row

merge (v:Vendor{id: row.no}) on create set

v.address= row.address,

v.phone = row.phone,

v.company\_name =row.Company\_name,

v.domicilie = row.domicilie,

v.k3s\_id = row.k3s\_id

## Contract

load csv with headers from "file:///Contract.csv" as row

//return row

merge (c:Contract{id:row.contract\_id}) on create set

c.periode = row.periode,

c.k3s\_id = row.k3s\_id,

c.po\_number= row.po\_number,

c.effective\_date = row.effective\_date,

c.capex = row.capex,

c.opex = row.opex

## ASSET

load csv with headers from "file:///Asset.csv" as row

merge (a:Asset {id:row.Po\_number}) on create set

a.hvac= row.hvac,

a.pricehvac = row.price5,

a.well\_drill = row.well.drill,

a.pricedrill= row.price4,

a.separator= row.separator,

a.priceseparator= row.price2,

a.compressor = row.compressor,

a.pricecompressor =row.price3,

a.valve\_actuator = row.valve\_actuator,

a.pricevalveactuator= row.price1

## Wilayah\_kerja

MATCH (n:K3S)

merge (n2:Wilayah\_Kerja{wk:n.wilayah\_kerja})

merge (n)-[:LOCATED\_AT]->(n2)

## Wilayah\_kerja (with latitude longitude)

LOAD CSV WITH HEADERS FROM 'file:///wilayah\_kerja\_kordinat.csv' AS row

// RETURN row

merge (w:Wilayah\_Kerja{wk:row.nama\_wilayah\_kerja})

on create set

w.latitude = row.latitude,

w.longitude = row.longitude

## Supplier

load csv with headers from "file:///PO.csv" as row

//return row limit 10

merge (a:Supplier{id:row.Supplier}) on create set

a.name =row.Supplier\_Name

## ASSET

load csv with headers from "file:///PO.csv" as row

//return row

merge (p:PO{id:row.no}) on create set

p.originaltype = row.Original\_Type,

p.line\_descriptipion = row.Line\_Description,

p.foreign\_unit\_cost = row.Foreign\_Unit\_Cost,

p.original\_number = row.Original\_Number,

p.item = row.Item,

p.unit\_cost = row.Unit\_Cost,

p.supplier = row.Supplier,

p.UM = row.UM,

p.quantity\_to\_receive = row.Quantity\_To\_Receive

## PO

load csv with headers from "file:///PO.csv" as row

//return row

match (p:Asset{id:row.no}) set

p.order\_number = row.Order\_Number

# RELATIONSHIP

## LOCATED\_AT

MATCH (n:K3S)

merge (n2:Wilayah\_Kerja{wk:n.wilayah\_kerja})

merge (n)-[:LOCATED\_AT]->(n2)

## HAS\_CONTRACT

match (k:K3S)

match (c:Contract)

where k.id = c.k3s\_id

merge (k)-[:HAS\_CONTRACT]->(c)

## HAS\_ASSET

match (p:PO)

match (a:Asset)

where p.id= a.id

merge (P)-[:HAS\_ASSET]->(a)

## WOKRKING\_WITH

match (k:K3S)

match (v:Vendor)

where k.id = v.k3s\_id

merge (k)-[:WORKING\_WITH]->(v)

## WORKING\_IN

match (v:Vendor)<-[:WORKING\_WITH]-(:K3S)-[:LOCATED\_AT]->(w:Wilayah\_Kerja)

merge (v)-[:WORKING\_IN]->(w)

## HAS\_PO

## 

load csv with headers from "file:///rel\_po.csv" as row

match (v:Vendor{id:row.id\_vendor})

match (p:PO{id:row.id\_po})

merge (v)-[:HAS\_PO]->(p)

/// new Scirpt

match (v:Vendor)-[:HAS\_ASSSET]->(:Asset)<-[:HAS]-(p:PO)

merge (v)-[:HAS\_PO]->(p)

## SUPPLIED\_BY

match (p:ASSET)

match (s:Supplier)

where p.supplier = s.id

merge (p)-[:SUPPLIED\_BY]->(s)

## HAS

MATCH (n:PO)

MATCH (a:Asset)

where n.order\_number = a.order\_number

merge (n)-[:HAS]->(a)

# Material Management

// load from csv

load csv with headers from 'file:///po\_vendor2.csv' as row

merge (p:PO{order\_number:row.`Order Number`})

merge (s:Supplier{supplier:row.`Supplier`})

set

n.name=row.`Supplier Name`

create (m:Material{id:row.`Date Receipt`})

set

m.short\_description=row.`Short Description`,

m.order\_number=row.`Order Number`,

m.qty=row.Qty,

m.unit\_cost=row.`Unit Cost`

MERGE (p)-[:HAS\_MATERIAL]->(m)

merge (s)<-[:PO\_ISSUED]-(p);

/// nlp task from material short desc

MATCH (n:Material)

with split(n.short\_description,',') as text,n

unwind range(0,size(text)-2) as i

merge (w1:Item{text:text[i]})

merge (w2:Item{text:text[i+1]})

merge (w1)-[r:NEXT\_DESC]->(w2)

on create set r.count=1

on match set r.count=r.count+1

merge (w1)<-[:SHORT\_DESC]-(n)

merge (w2)<-[:SHORT\_DESC]-(n);

/// Fraud Use Case -

match path=(s1:Supplier)<--(p:PO)-->(m1:Material)--(i:Item)--(m2:Material)<--(p2:PO)-->(s2:Supplier)

where p<>p2 and s1<>s2

return path limit 100

**TRIAL THORIQ**

**===============**

// LOADING DATA FROM CSV

Load csv with headers from "file:///po\_vendor.csv" as row

// return row

Create (m:Material{item\_id:row.item})

set

m.receipt\_reference = row.Receipt\_Reference,

m.vendor\_name= row.VendorName,

m.qty = row.Qty,

m.pr\_id = row.PR\_Stock,

m.po\_id = row.OrderNumber,

m.vendor\_id = row.Vendor,

m.date\_receipt = row.DateReceipt,

m.uoi = row.UOI,

m.short\_description = row.ShortDescription,

m.receipt\_number = row.ReceiptNumber

// CREATING NEW NODE

match (m:Material)

merge (p:PR{pr\_id:m.pr\_id})

return p

// MAKING NEW RELATIONS

match (m:Material)

match (s:Supplier)

where m.vendor\_id = s.supplier\_id

merge (m)-[:SUPPLIED\_BY]->(s)

return m, s

// DELETE NODES

Match (p:PR)

Detach Delete p

// PATTERN RECOGNITION

match p1 = (k:K3S)--(p:PO)--(m1:Material)--(r1:Receipt)

match p2 = (k)--(p)--(m2:Material)--(r2:Receipt)

where m1.short\_description = m2.short\_description and r1.receipt\_id <> r2.receipt\_id

case

when r1.date\_receipt > r2.date\_receipt then merge (r1)-[:IS\_AFTER]->(r2)

return p1,p2