Neo4j Data Load Scripts (MVP POC)

Data Exports - from ODS

Consumer Data Export from ODS

Note: extract to consumer.csv

```
select
c.cnsmr_id cnsmr_id
,c.public_cnsmr_id public_cnsmr_id
,pe.person_pty_id person_pty_id
,mod(c.cnsmr_id, 32) +18 age
,mod(c.cnsmr_id,2) gender
from ods.cnsmr c
--note: left join allows us to get party ID for known consumer
left join person pe on pe.cnsmr_id=c.cnsmr_id
--where c.cnsmr_id=16337
;
```

Make Data Export from ODS

```
select
extractvalue(XMLTYPE(PRODUCT_XML),
'/Product/Values/Value[@AttributeID="CarsProductID"]') MakeId,
extractvalue(XMLTYPE(PRODUCT_XML),
'/Product/Values/Value[@AttributeID="CarsMakeID"]') odsId,
extractvalue(XMLTYPE(PRODUCT_XML),
'/Product/Name') name
--, product_xml
from master_product mp
where PRODUCT_TYP_CD='CarsMake'
order by 1;
```

Model Data Export from ODS

```
select
extractvalue(XMLTYPE(PRODUCT_XML),
'/Product/Values/Value[@AttributeID="CarsProductID"]') ModelId,
extractvalue(XMLTYPE(PRODUCT_XML),
'/Product/Values/Value[@AttributeID="CarsModelID"]') odsId,
extractvalue(XMLTYPE(PRODUCT_XML),
'/Product/Name') name
from master_product mp
where PRODUCT_TYP_CD='CarsModel'
order by 1;
```

Year Data Export from ODS

```
select
extractvalue(XMLTYPE(PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsProductID"]') YearId,
extractvalue(XMLTYPE(PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsYearID"]') odsId,
extractvalue(XMLTYPE(PRODUCT_XML), '/Product/Name') name
--, product_xml
from master_product mp
where PRODUCT_TYP_CD='CarsYear'
order by 1;
```

Trim Data Export from ODS

```
select
extractvalue(XMLTYPE(PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsProductID"]') TrimId,
extractvalue(XMLTYPE(PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsTrimID"]') odsId,
extractvalue(XMLTYPE(PRODUCT_XML), '/Product/Name') name
--, product_xml
from master_product mp
where PRODUCT_TYP_CD='CarsTrim'
order by 1;
```

Bodystyle Data Export from ODS

```
select
extractvalue(XMLTYPE(PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsProductID"]') bodyStyleId,
extractvalue(XMLTYPE(PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsBodyStyleID"]') odsId,
extractvalue(XMLTYPE(PRODUCT_XML), '/Product/Name') name
from master_product mp
where PRODUCT_TYP_CD='CarsBodystyle'
order by 1;
```

MakeModel Data Export from ODS

```
select distinct
--m.PRODUCT_ID,
--mm.PRODUCT_ID,
--mmy.PRODUCT_ID,
extractvalue(XMLTYPE(m.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsProductID"]') MakeId,
extractvalue(XMLTYPE(md.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsProductID"]') ModelId,
extractvalue(XMLTYPE(m.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsMakeID"]') odsMakeId,
extractvalue(XMLTYPE(md.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsModelID"]') odsModelId
from master_product m
inner join master_product mm on m.PRODUCT_ID=mm.PRODUCT_PARENT_ID
inner join master_product_ref mpr on mm.PRODUCT_ID=mpr.PRODUCT_ID
inner join master_product md on md.Product_ID=mpr.PRODUCT_REF_ID
where mm.PRODUCT_TYP_CD='CarsMM'
and md.PRODUCT_TYP_CD='CarsModel'
order by MakeId, ModelId
;
```

MakeModelYear Data Export from ODS

```
select distinct
--m.PRODUCT_ID,
--mm.PRODUCT_ID,
--mmy.PRODUCT_ID,
extractvalue(XMLTYPE(m.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsProductID"]') MakeId,
extractvalue(XMLTYPE(md.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsProductID"]') ModelId,
extractvalue(XMLTYPE(y.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsProductID"]') YearId,
extractvalue(XMLTYPE(m.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsMakeID"]') odsMakeId,
extractvalue(XMLTYPE(md.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsModelID"]') odsModelId,
extractvalue(XMLTYPE(y.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsYearID"]') odsYearId
from master_product m
inner join master_product mm on m.PRODUCT_ID=mm.PRODUCT_PARENT_ID
inner join master_product mmy on mm.PRODUCT_ID=mmy.PRODUCT_PARENT_ID
inner join master_product_ref mpr on mm.PRODUCT_ID=mpr.PRODUCT_ID
inner join master_product md on md.Product_ID=mpr.PRODUCT_REF_ID
inner join master_product_ref mpry on mmy.PRODUCT_ID=mpry.PRODUCT_ID
inner join master_product y on y.Product_ID=mpry.PRODUCT_REF_ID
where mm.PRODUCT_TYP_CD='CarsMM'
and mmy.PRODUCT_TYP_CD='CarsMMY'
and md.PRODUCT_TYP_CD='CarsModel'
and y.PRODUCT_TYP_CD='CarsYear'
order by MakeId, ModelId, YearId
;
```

MakeModelYearTrim Data Export from ODS

```
 select distinct
--m.PRODUCT_ID,
--mm.PRODUCT_ID,
--mmy.PRODUCT_ID,
extractvalue(XMLTYPE(m.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsProductID"]') MakeId,
extractvalue(XMLTYPE(md.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsProductID"]') ModelId,
extractvalue(XMLTYPE(y.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsProductID"]') YearId,
extractvalue(XMLTYPE(t.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsProductID"]') TrimId,
extractvalue(XMLTYPE(m.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsMakeID"]') odsMakeId,
extractvalue(XMLTYPE(md.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsModelID"]') odsModelId,
extractvalue(XMLTYPE(y.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsYearID"]') odsYearId,
extractvalue(XMLTYPE(t.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsTrimID"]') odsTrimId
from master_product m
inner join master_product mm on m.PRODUCT_ID=mm.PRODUCT_PARENT_ID
inner join master_product mmy on mm.PRODUCT_ID=mmy.PRODUCT_PARENT_ID
inner join master_product mmyt on mmy.PRODUCT_ID=mmyt.PRODUCT_PARENT_ID
inner join master_product_ref mpr on mm.PRODUCT_ID=mpr.PRODUCT_ID
inner join master_product md on md.Product_ID=mpr.PRODUCT_REF_ID
inner join master_product_ref mpry on mmy.PRODUCT_ID=mpry.PRODUCT_ID
inner join master_product y on y.Product_ID=mpry.PRODUCT_REF_ID
inner join master_product_ref mprt on mmyt.PRODUCT_ID=mprt.PRODUCT_ID
inner join master_product t on t.Product_ID=mprt.PRODUCT_REF_ID
where mm.PRODUCT_TYP_CD='CarsMM'
and mmy.PRODUCT_TYP_CD='CarsMMY'
and mmyt.PRODUCT_TYP_CD='CarsMMYT'
and md.PRODUCT_TYP_CD='CarsModel'
and y.PRODUCT_TYP_CD='CarsYear'
and t.PRODUCT_TYP_CD='CarsTrim'
order by MakeId, ModelId, YearId, TrimId
;
```

MakeModelYearTrimBodystyle Data Export from ODS

```
select distinct
--m.PRODUCT_ID,
--mm.PRODUCT_ID,
--mmy.PRODUCT_ID,
extractvalue(XMLTYPE(m.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsProductID"]') MakeId,
extractvalue(XMLTYPE(md.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsProductID"]') ModelId,
extractvalue(XMLTYPE(y.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsProductID"]') YearId,
extractvalue(XMLTYPE(t.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsProductID"]') TrimId,
extractvalue(XMLTYPE(b.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsProductID"]') BodyStyleId,
extractvalue(XMLTYPE(m.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsMakeID"]') odsMakeId,
extractvalue(XMLTYPE(md.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsModelID"]') odsModelId,
extractvalue(XMLTYPE(y.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsYearID"]') odsYearId,
extractvalue(XMLTYPE(t.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsTrimID"]') odsTrimId,
extractvalue(XMLTYPE(b.PRODUCT_XML), '/Product/Values/Value[@AttributeID="CarsBodyStyleID"]') odsBodyStyleId
from master_product m
inner join master_product mm on m.PRODUCT_ID=mm.PRODUCT_PARENT_ID
inner join master_product mmy on mm.PRODUCT_ID=mmy.PRODUCT_PARENT_ID
inner join master_product mmyt on mmy.PRODUCT_ID=mmyt.PRODUCT_PARENT_ID
inner join master_product mmytb on mmyt.PRODUCT_ID=mmytb.PRODUCT_PARENT_ID
inner join master_product_ref mpr on mm.PRODUCT_ID=mpr.PRODUCT_ID
inner join master_product md on md.Product_ID=mpr.PRODUCT_REF_ID
inner join master_product_ref mpry on mmy.PRODUCT_ID=mpry.PRODUCT_ID
inner join master_product y on y.Product_ID=mpry.PRODUCT_REF_ID
inner join master_product_ref mprt on mmyt.PRODUCT_ID=mprt.PRODUCT_ID
inner join master_product t on t.Product_ID=mprt.PRODUCT_REF_ID
inner join master_product_ref mprb on mmytb.PRODUCT_ID=mprb.PRODUCT_ID
inner join master_product b on b.Product_ID=mprb.PRODUCT_REF_ID
where mm.PRODUCT_TYP_CD='CarsMM'
and mmy.PRODUCT_TYP_CD='CarsMMY'
and mmyt.PRODUCT_TYP_CD='CarsMMYT'
and mmytb.PRODUCT_TYP_CD='CarsMMYTB'
and md.PRODUCT_TYP_CD='CarsModel'
and y.PRODUCT_TYP_CD='CarsYear'
and t.PRODUCT_TYP_CD='CarsTrim'
and b.PRODUCT_TYP_CD='CarsBodystyle'
order by MakeId, ModelId, YearId, TrimId, BodyStyleId
;
```

Saved Vehicle Data Export from ODS

```
--************************************************************
-- saved vehicle
--************************************************************
select
*
from
(
select
c.cnsmr_id,
cd.cnsmr_device_uid,
cpgi.cnsmr_profile_grp_instance_id,
cpv.CNSMR_PROFILE_KEY_CD,
cpv.cnsmr_prfl_val,
cpgi.meta_maint_dt
from cnsmr c
left join cnsmr_device cd on cd.cnsmr_id = c.cnsmr_id
left join cnsmr_profile_grp_instance cpgi on cpgi.cnsmr_device_uid = cd.cnsmr_device_uid
left join cnsmr_profile_grp cpg on cpg.cnsmr_profile_grp_id = cpgi.cnsmr_profile_grp_id
left join cnsmr_profile_val cpv on cpv.cnsmr_profile_grp_instance_id = cpgi.cnsmr_profile_grp_instance_id
where cpgi.cnsmr_profile_grp_id = 441 -- saved vehicle
and cpgi.meta_maint_dt > sysdate - 180
)
pivot
(max(cnsmr_prfl_val)  for (cnsmr_profile_key_cd) in ('alertDisabled' as alertDisabled,'modelYear' as Year,  'mkNm' as make_name ,'trimName'
as Trim_name, 'mdNm' as model_name));
--where alertdisabled is not null
```

Data Exports - from Teradata

VDP Impressions Extract from Teradata

note:

- requires Teradata bteq utility
- edit environment variables for TERADATA-USER-NAME and TERADATA-Password (lines 6-7)

```
#!/usr/bin/env bash

date
DATAFILE=/Users/jasonwilliams/neo4j-enterprise-3.1.0.2/import/VDP-impressions.csv
echo 'export file: ' $DATAFILE

TERADATA-USER-NAME=
TERADATA-PASSWORD=

> $DATAFILE #so data file is created new vs. appended
echo 'impressionsCount,makeName,odsMakeId,modelName,odsModelId,trimName,odsTrimId,year,odsYearId,consumerAccountId' >> $DATAFILE


'/Library/Application Support/teradata/client/16.00/bin/bteq' << EOF > ~/bteq.log
.LOGON csprtd01/$TERADATA-USER-NAME,$TERADATA-PASSWORD


.EXPORT RESET;
.SET RECORDMODE OFF;
.SET WIDTH 5000
.EXPORT report file = $DATAFILE



select
-- search_instance_id ,
-- trim(coalesce( search_instance_id  ,'-1') (char(39)) ) --39 because we need the sign bit (decimal 38,0)
trim(coalesce( sum(impressions),'-1'))
--|| ',' || trim(coalesce( date_id,cast('01/01/9999' as date format 'MM/DD/YYYY')))
--|| ',' || trim(coalesce( dma_code,'-1') (char(8)) ) --8 because we need the sign bit (decimal 37,0)
--***make
|| ',' || trim(coalesce( mm.make_name,'NULL'))
|| ',' || trim(coalesce( mm.ods_make_id,'-1'))
--***model
|| ',' || trim(coalesce( mm.model_name,'NULL'))
|| ',' || trim(coalesce( mm.ods_model_id,'-1'))
```

```
--***trim
|| ',' || trim(coalesce( t.trim_name,'NULL'))
|| ',' || trim(coalesce( t.ods_trim_id,'-1'))
--***year
|| ',' || trim(coalesce( y.model_year,'NULL'))
|| ',' || trim(coalesce( y.ods_model_year_id,'-1'))
--***consumer
|| ',' || trim(coalesce( i.consumer_account_id,'-1'))
--|| ',' || trim(coalesce( a.consumer_id,'-1') (char(11))) -- (decimal 10,0 but we need the sign bit)

(TITLE '') -- to suppress title row in BTEQ

 from DW_VW.imp_vw i
 --inner join to the dimensions, to get ODS IDs
inner join DW_VW.vehicle_trim_vw t on i.trim_id = t.trim_id
inner join dw_vw.vehicle_make_model_vw mm on i.make_model_id = mm.make_model_id
inner join dw_vw.model_year_vw y on i.model_year_id = y.model_year_id
--inner join dw_vw.consumer_account_vw a on a.consumer_account_id = i.consumer_account_id
--inner join to get just the VDP impressions
inner join dw_vw.web_page_type_vw wpt on wpt.web_page_type_id = i.web_page_type_id
--todo: need to find out what kind of impression this is?
where date_id between '2017-07-07' and '2017-07-13'
--and AUTHENTICATED_CONSUMER='Yes' -- just get the known consumers
--and dma_code=602
and bot_flag='No'
and wpt.reporting_name = 'VDP'
--and a.consumer_id = 16337
group by
--search_instance_id
-- date_id,
--dma_code,
make_name, ods_make_id, model_name, ods_model_id, trim_name,
ods_trim_id, model_year, ods_model_year_id, i.consumer_account_id
--, a.consumer_id
--sample 100000
;

.LOGOFF
.EXIT
```

```
EOF


date
```

Inventory Search Extract from Teradata

note:

- requires Teradata bteq utility
- edit environment variables for TERADATA-USER-NAME and TERADATA-Password (lines 5-6)

```
#!/usr/bin/env bash



TERADATA-USER-NAME=
TERADATA-PASSWORD=
DATAFILE=/Users/jasonwilliams/neo4j-enterprise-3.1.0.2/import/inventory-search.csv
echo 'export file: ' $DATAFILE


date

> $DATAFILE #so data file is created new vs. appended


echo 'searchInstanceId,zipCode,radius, makeName,odsMakeId,modelName,odsModelId,year,odsYearId,trimName,odsTrimId' >> $DATAFILE


'/Library/Application Support/teradata/client/16.00/bin/bteq' << EOF > ~/bteq.log
.LOGON csprtd01/$TERADATA-USER-NAME,$TERADATA-PASSWORD
```

```
.EXPORT RESET;
.SET RECORDMODE OFF;
.SET WIDTH 5000
.EXPORT report file = $DATAFILE



 select
trim(coalesce( search_instance_id  ,'-1') (char(39)) ) --39 because we need the sign bit; search_instance_id is (decimal 38,0)
|| ',' || trim(coalesce( z.zip_code,'NULL'))
|| ',' || trim(coalesce( i.radius,'-1')  (char(11)) )  --39 because we need the sign bit; radius is (decimal 10,0)
--***make
|| ',' || trim(coalesce( mm.make_name,'NULL'))
|| ',' || trim(coalesce( mm.ods_make_id,'-1'))
--***model
|| ',' || trim(coalesce( mm.model_name,'NULL'))
|| ',' || trim(coalesce( mm.ods_model_id,'-1'))
--***year
|| ',' || trim(coalesce( y.model_year,'NULL'))
|| ',' || trim(coalesce( y.ods_model_year_id,'-1'))
--***trim
|| ',' || trim(coalesce( t.trim_name,'NULL'))
|| ',' || trim(coalesce( t.ods_trim_id,'-1'))
(TITLE '') -- to suppress title row in BTEQ
from dw_vw.search_vw i
inner join DW_VW.vehicle_trim_vw t on i.trim_id = t.trim_id
inner join dw_vw.vehicle_make_model_vw mm on i.make_model_id = mm.make_model_id
inner join dw_vw.model_year_vw y on i.model_year_id = y.model_year_id
inner join search_type_vw st on i.search_type_id = st.search_type_id
inner join zip_code_vw z on i.zip_code_id = z.zip_code_id
where date_id between '2017-07-07' and '2017-07-13'
and multi_make_id <> 0 --when multi_make_id=0, mmyt is unknown
and i.bot_flag = 'No'
and st.reporting_name = 'Inventory Search'
;


.LOGOFF
.EXIT
```

```
EOF


date
```

Data Loads

Note: data loads require APOC plugin: https://neo4j.com/blog/intro-user-defined-procedures-apoc/

Make Data Load in Neo4j

```
LOAD CSV WITH HEADERS FROM "file:///make.csv" AS row
CREATE ( :Make{ makeId:toInt(row.MAKEID), odsId:toInt(row.ODSID), name:row.NAME })

CREATE INDEX ON :Make(makeId)
CREATE INDEX ON :Make(odsId)
CREATE INDEX ON :Make(name)
```

Model Data Load in Neo4j

```
LOAD CSV WITH HEADERS FROM "file:///model.csv" AS row
CREATE ( :Model{ modelId:toInt(row.MODELID), odsId:toInt(row.ODSID), name:row.NAME })

CREATE INDEX ON :Model(modelId)
CREATE INDEX ON :Model(odsId)
CREATE INDEX ON :Model(name)
```

Year Data Load in Neo4j

```
LOAD CSV WITH HEADERS FROM "file:///year.csv" AS row
CREATE ( :Year{ yearId:toInt(row.YEARID), odsId:toInt(row.ODSID), name:row.NAME, year:toInt(row.NAME) })

CREATE INDEX ON :Year(yearId)
CREATE INDEX ON :Year(odsId)
CREATE INDEX on :Year(name)
```

Trim Data Load in Neo4j

```
LOAD CSV WITH HEADERS FROM "file:///trim.csv" AS row
CREATE ( :Trim{ trimId:toInt(row.TRIMID), odsId:toInt(row.ODSID), name:row.NAME })

CREATE INDEX ON :Trim(trimId)
CREATE INDEX ON :Trim(odsId)
CREATE INDEX on :Trim(name)

//We need to create an 'unknown' trim, because the DW VDPImpressions often come back with unknown trim, which is mapped to OdsTrimID=0
CREATE (t:Trim { trimId:0, odsId:0, name: 'created to match UNKNOWN trim in DW VDP Impressions' }) return t
```

Bodystyle Data Load in Neo4j

```
LOAD CSV WITH HEADERS FROM "file:///bodyStyle.csv" AS row
CREATE ( :BodyStyle{ bodyStyleId:toInt(row.BODYSTYLEID), odsId:toInt(row.ODSID), name:row.NAME })

CREATE INDEX ON :BodyStyle(bodyStyleId)
CREATE INDEX ON :BodyStyle(odsId)
```

MakeModel Data Load in Neo4j

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///makeModel.csv" AS row
CREATE ( mm:MakeModel{ makeId:toInt(row.MAKEID), modelId:toInt(row.MODELID), odsMakeId:toInt(row.ODSMAKEID), odsModelId:toInt(row.
ODSMODELID) })

CREATE INDEX ON :MakeModel(makeId)
CREATE INDEX ON :MakeModel(modelId)
CREATE INDEX ON :MakeModel(odsMakeId)
CREATE INDEX ON :MakeModel(odsModelId)

MATCH (m:Make), (md:Model), (mm:MakeModel) WHERE m.makeId=mm.makeId and md.modelId=mm.modelId MERGE (m)<-[:OF_MAKE]-(mm)  MERGE (md)<-[:
OF_MODEL]-(mm)
```

MakeModelYear Data Load in Neo4j

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///makeModelYear.csv" AS row
CREATE ( mm:MakeModelYear{ makeId:toInt(row.MAKEID), modelId:toInt(row.MODELID), yearId:toInt(row.YEARID),
odsMakeId:toInt(row.ODSMAKEID), odsModelId:toInt(row.ODSMODELID), odsYearId:toInt(row.ODSYEARID)})

CREATE INDEX ON :MakeModelYear(makeId)
CREATE INDEX ON :MakeModelYear(modelId)
CREATE INDEX ON :MakeModelYear(yearId)
CREATE INDEX ON :MakeModelYear(odsMakeId)
CREATE INDEX ON :MakeModelYear(odsModelId)
CREATE INDEX ON :MakeModelYear(odsYearId)

MATCH (mm:MakeModel), (mmy:MakeModelYear), (mk:Make), (md:Model), (y:Year) WHERE mm.makeId=mmy.makeId AND mm.modelId=mmy.modelId AND mm.
makeId=mk.makeId AND mm.modelId=md.modelId AND md.modelId=mmy.modelId and mmy.yearId=y.yearId CREATE (mk)<-[:OF_MAKE]-(mmy) CREATE (md)<-[:
OF_MODEL]-(mmy) CREATE (mm)<-[:OF_MAKE_MODEL]-(mmy) CREATE (y)<-[:OF_YEAR]-(mmy)
```

MakeModelYearTrim Data Load in Neo4j

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///makeModelYearTrim.csv" AS row
CREATE ( mm:MakeModelYearTrim{ makeId:toInt(row.MAKEID), modelId:toInt(row.MODELID), yearId:toInt(row.YEARID), trimId:toInt(row.TRIMID),
odsMakeId:toInt(row.ODSMAKEID), odsModelId:toInt(row.ODSMODELID), odsYearId:toInt(row.ODSYEARID), odsTrimId:toInt(row.ODSTRIMID) })

CREATE INDEX ON :MakeModelYearTrim(makeId)
CREATE INDEX ON :MakeModelYearTrim(modelId)
CREATE INDEX ON :MakeModelYearTrim(yearId)
CREATE INDEX ON :MakeModelYearTrim(trimId)
CREATE INDEX ON :MakeModelYearTrim(odsMakeId)
CREATE INDEX ON :MakeModelYearTrim(odsModelId)
CREATE INDEX ON :MakeModelYearTrim(odsYearId)
CREATE INDEX ON :MakeModelYearTrim(odsTrimId)

MATCH (mmy:MakeModelYear), (mmyt:MakeModelYearTrim), (mk:Make), (md:Model), (y:Year), (t:Trim) WHERE mmy.makeId=mmyt.makeId AND mmy.
modelId=mmyt.modelId AND mmy.yearId=mmyt.yearId AND mmy.makeId=mk.makeId AND mmy.modelId=md.modelId AND mk.makeId=mmyt.makeId AND md.
modelId=mmyt.modelId AND mmy.yearId=y.yearId AND mmyt.trimId=t.trimId CREATE (mk)<-[:OF_MAKE]-(mmyt) CREATE (md)<-[:OF_MODEL]-(mmyt) CREATE
(mmy)<-[:OF_MAKE_MODEL_YEAR]-(mmyt) CREATE (t)<-[:OF_TRIM]-(mmyt) CREATE (y)<-[:OF_YEAR]-(mmyt)
```

MakeModelYearTrimBodystyle Data Load in Neo4j

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///makeModelYearTrimBodystyle.csv" AS row
CREATE ( mm:MakeModelYearTrimBodyStyle{ makeId:toInt(row.MAKEID), modelId:toInt(row.MODELID), yearId:toInt(row.YEARID), trimId:toInt(row.
TRIMID), bodyStyleId:toInt(row.BODYSTYLEID), odsMakeId:toInt(row.ODSMAKEID), odsModelId:toInt(row.ODSMODELID), odsYearId:toInt(row.
ODSYEARID), odsTrimId:toInt(row.ODSTRIMID), odsBodyStyleId:toInt(row.ODSBODYSTYLEID) })

CREATE INDEX ON :MakeModelYearTrimBodyStyle(makeId)
CREATE INDEX ON :MakeModelYearTrimBodyStyle (modelId)
CREATE INDEX ON :MakeModelYearTrimBodyStyle (yearId)
CREATE INDEX ON :MakeModelYearTrimBodyStyle (trimId)
CREATE INDEX ON :MakeModelYearTrimBodyStyle (bodyStyleId)
CREATE INDEX ON :MakeModelYearTrimBodyStyle(odsMakeId)
CREATE INDEX ON :MakeModelYearTrimBodyStyle (odsModelId)
CREATE INDEX ON :MakeModelYearTrimBodyStyle (odsYearId)
CREATE INDEX ON :MakeModelYearTrimBodyStyle (odsTrimId)
CREATE INDEX ON :MakeModelYearTrimBodyStyle (odsBodyStyleId)

MATCH (mmyt:MakeModelYearTrim), (mmytb:MakeModelYearTrimBodyStyle), (mk:Make), (md:Model), (y:Year), (t:Trim), (b:BodyStyle) WHERE mmyt.
makeId=mmytb.makeId AND mmyt.modelId=mmytb.modelId AND mmyt.yearId=mmytb.yearId AND mmyt.trimId=mmytb.trimId AND mmytb.makeId=mk.makeId AND
mmytb.modelId=md.modelId AND mmytb.yearId=y.yearId AND mmytb.trimId=t.trimId AND mmytb.bodyStyleId=b.bodyStyleId CREATE (mk)<-[:OF_MAKE]-
(mmytb) CREATE (md)<-[:OF_MODEL]-(mmytb) CREATE (t)<-[:OF_TRIM]-(mmytb) CREATE (y)<-[:OF_YEAR]-(mmytb) CREATE (mmyt)<-[:
OF_MAKE_MODEL_YEAR_TRIM]-(mmytb) CREATE (b)<-[:OF_BODYSTYLE]-(mmytb)
```

Age Data Load in Neo4J

```
LOAD CSV WITH HEADERS FROM "file:///Age.csv" as row
CREATE (:Age{ageId:toInt(row.ageId), age:toInt(row.age)})

CREATE INDEX ON :Age(ageId)
CREATE INDEX ON :Age(age)

LOAD CSV WITH HEADERS FROM "file:///YearBucket.csv" as row
CREATE (:AgeBucket{ageBucketId:toInt(row.yearBucketId), startYear:toInt(row.startYear), endYear:toInt(row.endYear)})

MATCH (a:Age), (yb:AgeBucket) WHERE a.age>=yb.startYear AND a.age<=yb.endYear MERGE (a)-[:OF_AGE_BUCKET]->(yb)
```

Gender Data Load in Neo4J

```
LOAD CSV WITH HEADERS FROM "file:///Gender.csv" as row
CREATE (:Gender{genderId:toInt(row.genderId), gender:row.gender})

CREATE INDEX ON :Gender(genderId)
```

Consumer Data Load in Neo4J

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///consumer.csv" AS row
CREATE (:Consumer {consumerId: toInt(row.CNSMR_ID), publicConsumerId: toInt(row.PUBLIC_CNSMR_ID), OdsPartyId: toInt(row.PERSON_PTY_ID), age:
toInt(row.AGE), genderId: toInt(row.GENDER)})



CREATE INDEX ON :Consumer(consumerId);
CREATE INDEX ON :Consumer(age);
CREATE INDEX ON :Consumer(genderId);



//create relationships to gender
MATCH (c:Consumer) WHERE NOT  ((c)-[:OF_GENDER]->(:Gender))
With c
MATCH (g:Gender)
WHERE c.genderId=g.genderId
CREATE (c)-[r:OF_GENDER]->(g) return count(r);

//create relationships to Age
MATCH (c:Consumer) WHERE NOT  ((c)-[:OF_AGE]->(:Age))
With c
MATCH (a:Age)
WHERE c.age=a.age
CREATE (c)-[r:OF_AGE]->(a) return count(r);
```

Searches Data Load in Neo4J

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///searches.csv" AS row
CREATE (:Search {impressionsCount: toInt(row.impressionsCount), date: row.date, dmaCode: row.dmaCode, makeName: row.makeName, odsMakeId:
toInt(row.odsMakeId), modelName: row.modelName, odsModelId: toInt(row.odsModelId), trimName: row.trimName, odsTrimId: toInt(row.odsTrimId),
year: toInt(row.year), odsYearId: toInt(row.odsYearId), consumerAccountId: toInt(row.consumerAccountId), consumerId: toInt(row.
odsConsumerId) })

CREATE INDEX ON :Search(consumerId)
CREATE INDEX ON :Search(odsMakeId)

CREATE INDEX ON :Search(odsModelId)
CREATE INDEX ON :Search(odsYearId)
CREATE INDEX ON :Search(odsTrimId)

MATCH (c:Consumer), (s:Search) WHERE c.consumerId=s.consumerId CREATE (c)<-[:BY_CONSUMER]-(s)

MATCH (m:Make), (s:Search) WHERE m.odsId=s.odsMakeId CREATE (m)<-[:OF_MAKE]-(s)

MATCH (m:Model), (s:Search) WHERE m.odsId=s.odsModelId CREATE (m)<-[:OF_MODEL]-(s)

MATCH (y:Year), (s:Search) WHERE y.odsId=s.odsYearId CREATE (y)<-[:OF_YEAR]-(s)

MATCH (t:Trim), (s:Search) WHERE t.odsId=s.odsTrimId CREATE (t)<-[:OF_TRIM]-(s)

MATCH (m:MakeModelYearTrim), (s:Search) WHERE m.odsMakeId=s.odsMakeId AND m.odsModelId=s.odsModelId AND m.odsYearId=s.odsYearId AND m.
odsTrimId=s.odsTrimId CREATE (m)<-[:OF_MAKE_MODEL_YEAR_TRIM]-(s)

//MATCH (s:Search)-->(mk:Make), (s)-->(md:Model), (s)-->(y:Year), (s)-->(t:Trim), (mk)<--(mm:MakeModel)-->(md), (mm)<--(mmy:MakeModelYear)-->
(y),(mmy)<--(mmyt:MakeModelYearTrim)-->(t) CREATE (s)-[:OF_MAKE_MODEL_YEAR_TRIM]->(mmyt)
```

VDP Impressions Data Load in Neo4J

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///VDP-impressions.csv" AS row
CREATE (:VDPImpression {
impressionsCount: toInt(row.impressionsCount)
, makeName: row.makeName
, odsMakeId: toInt(row.odsMakeId)
, modelName: row.modelName
, odsModelId: toInt(row.odsModelId)
, trimName: row.trimName
, odsTrimId: toInt(row.odsTrimId)
, year: toInt(row.year)
, odsYearId: toInt(row.odsYearId)
, consumerAccountId: toInt(row.consumerAccountId)});

//note: consumerAccountID is NOT the ODS Consumer ID

CREATE INDEX ON :VDPImpression(consumerAccountId);
CREATE INDEX ON :VDPImpression(odsMakeId);
CREATE INDEX ON :VDPImpression(odsModelId);
CREATE INDEX ON :VDPImpression(odsYearId);
CREATE INDEX ON :VDPImpression(odsTrimId);

//note: statements below are not performant with large data sets (eg, 11M VDP impressions); instead, use scripts to add relationships in
batches
MATCH (c:Consumer), (s:VDPImpression) WHERE c.consumerId=s.consumerId CREATE (c)<-[:BY_CONSUMER]-(s)

MATCH (m:Make), (v:VDPImpression) WHERE m.odsId=v.odsMakeId CREATE (m)<-[:OF_MAKE]-(v)

MATCH (m:Model), (v:VDPImpression) WHERE m.odsId=v.odsModelId CREATE (m)<-[:OF_MODEL]-(v)

MATCH (y:Year), (v:VDPImpression) WHERE y.odsId=v.odsYearId CREATE (y)<-[:OF_YEAR]-(v)

MATCH (t:Trim), (v:VDPImpression) WHERE t.odsId=v.odsTrimId CREATE (t)<-[:OF_TRIM]-(v)

MATCH (m:MakeModelYearTrim), (v:VDPImpression) WHERE m.odsMakeId=v.odsMakeId AND m.odsModelId=v.odsModelId AND m.odsYearId=v.odsYearId AND m.
odsTrimId=v.odsTrimId CREATE (m)<-[:OF_MAKE_MODEL_YEAR_TRIM]-(v)

//MATCH (v:VDPImpression)-->(mk:Make), (v)-->(md:Model), (v)-->(y:Year), (v)-->(t:Trim), (mk)<--(mm:MakeModel)-->(md), (mm)<--(mmy:
MakeModelYear)-->(y),(mmy)<--(mmyt:MakeModelYearTrim)-->(t) CREATE (v)-[:OF_MAKE_MODEL_YEAR_TRIM]->(mmyt)

// An alternative way to load VDP impressions without creating Impression nodes
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///VDP-impressions.csv" AS row
```

```
MATCH (c:Consumer {consumerId: toInt(row.consumerAccountId)})
MATCH (mmyt:MakeModelYearTrim {odsMakeId: toInt(row.odsMakeId), odsModelId: toInt(row.odsModelId), odsYearId: toInt(row.odsYearId),
odsTrimId: toInt(row.odsTrimId)})
CREATE (mmyt)-[r:VIEWED_BY {impressionsCount: toInt(row.impressionsCount)}]->(c)

//QC Queries-Make
MATCH (v:VDPImpression)-[r:OF_MAKE]->(m:Make) return count(r)
MATCH (v:VDPImpression)-[r:OF_MAKE]->(m:Make) return v,r,m limit 25

//QC Queries - Model
MATCH (v:VDPImpression)-[r:OF_MODEL]->(m:Model) return count(r)
MATCH (v:VDPImpression)-[r:OF_MODEL]->(m:Model) return v,r,m limit 25

//QC Queries - Year
MATCH (v:VDPImpression)-[r:OF_YEAR]->(y:Year) return count(r)
MATCH (v:VDPImpression)-[r:OF_YEAR]->(y:Year) return v,r,y limit 25

//QC Queries - Trim
MATCH (v:VDPImpression)-[r:OF_TRIM]->(t:Trim) return count(r)
MATCH (v:VDPImpression)-[r:OF_TRIM]->(t:Trim) return v,r,t limit 25

//QC Queries - OF_MAKE_MODEL_YEAR_TRIM
MATCH (v:VDPImpression)-[r:OF_MAKE_MODEL_YEAR_TRIM]->(mmyt:MakeModelYearTrim) return count(r)
MATCH (v:VDPImpression)-[r:OF_MAKE_MODEL_YEAR_TRIM]->(mmyt:MakeModelYearTrim) return v,r,mmyt limit 25
MATCH (v:VDPImpression) WHERE not ((v)-[:OF_TRIM]->(:Trim)) return count(v)
```

Saved Vehicles Data Load in Neo4j

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///subscription-saved-vehicle.csv" AS row
CREATE (:SubscriptionSavedVehicle {consumerId: toInt(row.CNSMR_ID), consumerDeviceUid: row.CNSMR_DEVICE_UID, consumerProfileGroupInstanceId:
```

```
row.CNSMR_PROFILE_GRP_INSTANCE_ID, consumerProfileGroupId: row.CNSMR_PROFILE_GRP_ID,year: row.YEAR, makeName: row.MAKE_NAME, modelName: row.
MODEL_NAME, trimName: row.TRIM_NAME, searchUpdateDate: row.META_MAINT_DT})

//index the subscriptionWiredSavedSearch
CREATE INDEX ON :SubscriptionSavedVehicle(consumerId);
CREATE INDEX ON :SubscriptionSavedVehicle(makeName);
CREATE INDEX ON :SubscriptionSavedVehicle(modelName);
CREATE INDEX ON :SubscriptionSavedVehicle(trimName);
CREATE INDEX ON :SubscriptionSavedVehicle(year);

//create the relationship to consumer
CALL apoc.periodic.commit(
'
MATCH (s:SubscriptionSavedVehicle) WHERE not  ((s)-[:BY_CONSUMER]->(:Consumer))
with s
match  (c:Consumer)
where s.consumerId = c.consumerId
create (s)-[r:BY_CONSUMER]->(c)
return count(r);
'
,{limit: 20000}
)

//create relationship to make
CALL apoc.periodic.commit(
'
match (s:SubscriptionSavedVehicle) where not (s)-[:OF_MAKE]->(:Make)
with s
match (m:Make)
where s.makeName=m.name
create (s)-[r:OF_MAKE]->(m)
return count(r);
'
,{limit: 20000}
)


//create relationship to model
CALL apoc.periodic.commit(
'
match (s:SubscriptionSavedVehicle) where not (s)-[:OF_MODEL]->(:Model)
with s
match (m:Model)
where s.modelName=m.name
create (s)-[r:OF_MODEL]->(m)
return count(r);
'
,{limit: 20000}
)


//create relationship to year
CALL apoc.periodic.commit(
'
match (s:SubscriptionSavedVehicle) where not (s)-[:OF_YEAR]->(:Year)
with s
match (y:Year)
where s.year=y.name
create (s)-[r:OF_YEAR]->(y)
return count(r);
'
,{limit: 20000}
)



//create relationship to trim
CALL apoc.periodic.commit(
'
match (s:SubscriptionSavedVehicle) where not (s)-[:OF_TRIM]->(:Trim)
with s
match (t:Trim)
where s.trimName=t.name
create (s)-[r:OF_TRIM]->(t)
return count(r);
'
,{limit: 20000}
)
```

```
//QC Queries
match (sv:SubscriptionSavedVehicle)-[:OF_MAKE]->(:Make) return count(sv);
match (sv:SubscriptionSavedVehicle)-[:OF_MODEL]->(:Model) return count(sv);
match (sv:SubscriptionSavedVehicle)-[:OF_YEAR]->(:Year) return count(sv);
match (sv:SubscriptionSavedVehicle)-[:OF_TRIM]->(:Trim) return count(sv);
```