**C:\Program Files\PostgreSQL\14\pgAdmin 4\bin>"C:\Program Files\PostgreSQL\14\bin\psql.exe" "host='localhost' port='5432' dbname='miso' user='armando' password='Epfy44yqE\_WEB' sslmode='prefer' sslcompression='False' " 2>>&1**

**CREATE ROLE armando WITH**

**LOGIN**

**SUPERUSER**

**INHERIT**

**CREATEDB**

**CREATEROLE**

**REPLICATION**

**ENCRYPTED PASSWORD 'SCRAM-SHA-256$4096:Pa7w4AU3BxcvdkfKUDcuaQ==$0CCnYu9kmuN20liUSvSFVDgaoOX0EKPuHN8Um2Gs/WA=:F221zcAkXoWRC6mGTGfbwrL0mDwc4aVtrg1JoP+ulYk=';**

**CREATE DATABASE miso**

**WITH**

**OWNER = postgres**

**ENCODING = 'UTF8'**

**LC\_COLLATE = 'English\_United States.1252'**

**LC\_CTYPE = 'English\_United States.1252'**

**TABLESPACE = pg\_default**

**CONNECTION LIMIT = -1;**

**CREATE SCHEMA IF NOT EXISTS "MISO\_CUST\_PORTAL"**

**AUTHORIZATION postgres;**

**GRANT ALL ON SCHEMA "MISO\_CUST\_PORTAL" TO pg\_database\_owner WITH GRANT OPTION;**

**GRANT ALL ON SCHEMA "MISO\_CUST\_PORTAL" TO pg\_execute\_server\_program WITH GRANT OPTION;**

**GRANT ALL ON SCHEMA "MISO\_CUST\_PORTAL" TO pg\_write\_all\_data WITH GRANT OPTION;**

**GRANT ALL ON SCHEMA "MISO\_CUST\_PORTAL" TO pg\_write\_server\_files WITH GRANT OPTION;**

**GRANT ALL ON SCHEMA "MISO\_CUST\_PORTAL" TO postgres WITH GRANT OPTION;**

**ALTER DEFAULT PRIVILEGES IN SCHEMA "MISO\_CUST\_PORTAL"**

**GRANT ALL ON TABLES TO pg\_database\_owner WITH GRANT OPTION;**

**CREATE SCHEMA IF NOT EXISTS meta\_001 AUTHORIZATION postgres;**

**GRANT ALL ON SCHEMA meta\_001 TO armando WITH GRANT OPTION;**

**GRANT ALL ON SCHEMA meta\_001 TO postgres;**

|  |
| --- |
| DROP TABLE if exists d\_date; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| CREATE TABLE d\_date |
|  |

|  |
| --- |
| ( |
|  |

|  |
| --- |
| date\_dim\_id INT NOT NULL, |
|  |

|  |
| --- |
| date\_actual DATE NOT NULL, |
|  |

|  |
| --- |
| epoch BIGINT NOT NULL, |
|  |

|  |
| --- |
| day\_suffix VARCHAR(4) NOT NULL, |
|  |

|  |
| --- |
| day\_name VARCHAR(9) NOT NULL, |
|  |

|  |
| --- |
| day\_of\_week INT NOT NULL, |
|  |

|  |
| --- |
| day\_of\_month INT NOT NULL, |
|  |

|  |
| --- |
| day\_of\_quarter INT NOT NULL, |
|  |

|  |
| --- |
| day\_of\_year INT NOT NULL, |
|  |

|  |
| --- |
| week\_of\_month INT NOT NULL, |
|  |

|  |
| --- |
| week\_of\_year INT NOT NULL, |
|  |

|  |
| --- |
| week\_of\_year\_iso CHAR(10) NOT NULL, |
|  |

|  |
| --- |
| month\_actual INT NOT NULL, |
|  |

|  |
| --- |
| month\_name VARCHAR(9) NOT NULL, |
|  |

|  |
| --- |
| month\_name\_abbreviated CHAR(3) NOT NULL, |
|  |

|  |
| --- |
| quarter\_actual INT NOT NULL, |
|  |

|  |
| --- |
| quarter\_name VARCHAR(9) NOT NULL, |
|  |

|  |
| --- |
| year\_actual INT NOT NULL, |
|  |

|  |
| --- |
| first\_day\_of\_week DATE NOT NULL, |
|  |

|  |
| --- |
| last\_day\_of\_week DATE NOT NULL, |
|  |

|  |
| --- |
| first\_day\_of\_month DATE NOT NULL, |
|  |

|  |
| --- |
| last\_day\_of\_month DATE NOT NULL, |
|  |

|  |
| --- |
| first\_day\_of\_quarter DATE NOT NULL, |
|  |

|  |
| --- |
| last\_day\_of\_quarter DATE NOT NULL, |
|  |

|  |
| --- |
| first\_day\_of\_year DATE NOT NULL, |
|  |

|  |
| --- |
| last\_day\_of\_year DATE NOT NULL, |
|  |

|  |
| --- |
| mmyyyy CHAR(6) NOT NULL, |
|  |

|  |
| --- |
| mmddyyyy CHAR(10) NOT NULL, |
|  |

|  |
| --- |
| weekend\_indr BOOLEAN NOT NULL |
|  |

|  |
| --- |
| ); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| ALTER TABLE public.d\_date ADD CONSTRAINT d\_date\_date\_dim\_id\_pk PRIMARY KEY (date\_dim\_id); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| CREATE INDEX d\_date\_date\_actual\_idx |
|  |

|  |
| --- |
| ON d\_date(date\_actual); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| COMMIT; |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| INSERT INTO d\_date |
|  |

|  |
| --- |
| SELECT TO\_CHAR(datum, 'yyyymmdd')::INT AS date\_dim\_id, |
|  |

|  |
| --- |
| datum AS date\_actual, |
|  |

|  |
| --- |
| EXTRACT(EPOCH FROM datum) AS epoch, |
|  |

|  |
| --- |
| TO\_CHAR(datum, 'fmDDth') AS day\_suffix, |
|  |

|  |
| --- |
| TO\_CHAR(datum, 'TMDay') AS day\_name, |
|  |

|  |
| --- |
| EXTRACT(ISODOW FROM datum) AS day\_of\_week, |
|  |

|  |
| --- |
| EXTRACT(DAY FROM datum) AS day\_of\_month, |
|  |

|  |
| --- |
| datum - DATE\_TRUNC('quarter', datum)::DATE + 1 AS day\_of\_quarter, |
|  |

|  |
| --- |
| EXTRACT(DOY FROM datum) AS day\_of\_year, |
|  |

|  |
| --- |
| TO\_CHAR(datum, 'W')::INT AS week\_of\_month, |
|  |

|  |
| --- |
| EXTRACT(WEEK FROM datum) AS week\_of\_year, |
|  |

|  |
| --- |
| EXTRACT(ISOYEAR FROM datum) || TO\_CHAR(datum, '"-W"IW-') || EXTRACT(ISODOW FROM datum) AS week\_of\_year\_iso, |
|  |

|  |
| --- |
| EXTRACT(MONTH FROM datum) AS month\_actual, |
|  |

|  |
| --- |
| TO\_CHAR(datum, 'TMMonth') AS month\_name, |
|  |

|  |
| --- |
| TO\_CHAR(datum, 'Mon') AS month\_name\_abbreviated, |
|  |

|  |
| --- |
| EXTRACT(QUARTER FROM datum) AS quarter\_actual, |
|  |

|  |
| --- |
| CASE |
|  |

|  |
| --- |
| WHEN EXTRACT(QUARTER FROM datum) = 1 THEN 'First' |
|  |

|  |
| --- |
| WHEN EXTRACT(QUARTER FROM datum) = 2 THEN 'Second' |
|  |

|  |
| --- |
| WHEN EXTRACT(QUARTER FROM datum) = 3 THEN 'Third' |
|  |

|  |
| --- |
| WHEN EXTRACT(QUARTER FROM datum) = 4 THEN 'Fourth' |
|  |

|  |
| --- |
| END AS quarter\_name, |
|  |

|  |
| --- |
| EXTRACT(YEAR FROM datum) AS year\_actual, |
|  |

|  |
| --- |
| datum + (1 - EXTRACT(ISODOW FROM datum))::INT AS first\_day\_of\_week, |
|  |

|  |
| --- |
| datum + (7 - EXTRACT(ISODOW FROM datum))::INT AS last\_day\_of\_week, |
|  |

|  |
| --- |
| datum + (1 - EXTRACT(DAY FROM datum))::INT AS first\_day\_of\_month, |
|  |

|  |
| --- |
| (DATE\_TRUNC('MONTH', datum) + INTERVAL '1 MONTH - 1 day')::DATE AS last\_day\_of\_month, |
|  |

|  |
| --- |
| DATE\_TRUNC('quarter', datum)::DATE AS first\_day\_of\_quarter, |
|  |

|  |
| --- |
| (DATE\_TRUNC('quarter', datum) + INTERVAL '3 MONTH - 1 day')::DATE AS last\_day\_of\_quarter, |
|  |

|  |
| --- |
| TO\_DATE(EXTRACT(YEAR FROM datum) || '-01-01', 'YYYY-MM-DD') AS first\_day\_of\_year, |
|  |

|  |
| --- |
| TO\_DATE(EXTRACT(YEAR FROM datum) || '-12-31', 'YYYY-MM-DD') AS last\_day\_of\_year, |
|  |

|  |
| --- |
| TO\_CHAR(datum, 'mmyyyy') AS mmyyyy, |
|  |

|  |
| --- |
| TO\_CHAR(datum, 'mmddyyyy') AS mmddyyyy, |
|  |

|  |
| --- |
| CASE |
|  |

|  |
| --- |
| WHEN EXTRACT(ISODOW FROM datum) IN (6, 7) THEN TRUE |
|  |

|  |
| --- |
| ELSE FALSE |
|  |

|  |
| --- |
| END AS weekend\_indr |
|  |

|  |
| --- |
| FROM (SELECT '1970-01-01'::DATE + SEQUENCE.DAY AS datum |
|  |

|  |
| --- |
| FROM GENERATE\_SERIES(0, 29219) AS SEQUENCE (DAY) |
|  |

|  |
| --- |
| GROUP BY SEQUENCE.DAY) DQ |
|  |

|  |
| --- |
| ORDER BY 1; |
|  |

|  |
| --- |
|  |
|  |

COMMIT;

**CREATE** **TABLE** meta\_001.meta\_table (

meta\_table\_id serial **NOT** **NULL**,

s\_name **name** **NOT** **NULL**,

t\_name **name** **NOT** **NULL**,

data\_source **text** **NULL**,

"sensitive" **bool** **NOT** **NULL** **DEFAULT** **false**,

**CONSTRAINT** pk\_dd\_meta\_table\_id **PRIMARY** **KEY** (meta\_table\_id),

**CONSTRAINT** uq\_dd\_meta\_table\_schema\_table **UNIQUE** (s\_name, t\_name)

);

**COMMENT** **ON** **TABLE** meta\_001.meta\_table **IS** 'User definable meta-data at the schema + table level.';

INSERT INTO meta\_001.meta\_table (s\_name,t\_name,data\_source,"sensitive") VALUES

('meta\_001','meta\_schema','Manually maintained',false),

('meta\_001','meta\_table','Manually maintained',false),

('meta\_001','meta\_column','Manually maintained',false);

**CREATE** **OR** **REPLACE** **VIEW** meta\_001.tables1

**AS** **SELECT** (n.nspname::**text** || '.'::**text**) || c.relname::**text** **AS** full\_name,

n.nspname **AS** s\_name,

c.relname **AS** t\_name,

**CASE**

**WHEN** c.relkind = **ANY** (**ARRAY**['r'::"char", 'p'::"char"]) **THEN** 'table'::**text**

**WHEN** c.relkind = 's'::"char" **THEN** 'special'::**text**

**WHEN** c.relkind = 'f'::"char" **THEN** 'foreign table'::**text**

**ELSE** **NULL**::**text**

**END** **AS** **type**,

pg\_get\_userbyid(c.relowner) **AS** **owner**,

pg\_size\_pretty(pg\_table\_size(c.**oid**::**regclass**)) **AS** size\_pretty,

pg\_table\_size(c.**oid**::**regclass**) **AS** size\_bytes,

c.reltuples **AS** **rows**,

**CASE**

**WHEN** c.reltuples > 0::**double** **precision** **THEN** pg\_table\_size(c.**oid**::**regclass**)::**double** **precision** / c.reltuples

**ELSE** **NULL**::**double** **precision**

**END** **AS** bytes\_per\_row,

pg\_size\_pretty(pg\_total\_relation\_size(c.**oid**::**regclass**)) **AS** size\_plus\_indexes,

**obj\_description**(c.**oid**, 'pg\_class'::**name**) **AS** description,

**CASE**

**WHEN** n.nspname !~ '^pg\_toast'::**text** **AND** (n.nspname <> **ALL** (**ARRAY**['pg\_catalog'::**name**, 'information\_schema'::**name**])) **THEN** **false**

**ELSE** **true**

**END** **AS** system\_object,

mt.data\_source,

mt.**sensitive**

**FROM** pg\_class c

**LEFT** **JOIN** pg\_namespace n **ON** n.**oid** = c.relnamespace

**LEFT** **JOIN** meta\_001.meta\_table mt **ON** n.nspname = mt.s\_name **AND** c.relname = mt.t\_name

**WHERE** (c.relkind = **ANY** (**ARRAY**['r'::"char", 'p'::"char", 's'::"char", 'f'::"char"])) **AND** (n.nspname::**text** <> **ALL** (**ARRAY**['pg\_catalog'::**text**, 'information\_schema'::**text**]));

Commit;

CREATE OR REPLACE VIEW meta\_001.v\_dba\_get\_tbl\_sch\_owner

AS SELECT inet\_server\_addr() AS ipaddress,

inet\_server\_port() AS postgres\_port,

q.tablename,

s.nspname AS table\_schema,

s.oid AS schema\_id,

u.usename AS schema\_owner

FROM pg\_namespace s,

pg\_tables q,

pg\_user u

WHERE q.schemaname = s.nspname AND u.usesysid = s.nspowner AND s.nspname <> 'information\_schema'::name AND q.tablename !~~ 'sql%'::text AND s.nspname <> 'pg\_catalog'::name

ORDER BY u.usename, s.nspname;

commit;

CREATE OR REPLACE VIEW meta\_001.v\_dba\_get\_tbl\_ddl

AS WITH thistbls AS (

SELECT (v\_dba\_get\_tbl\_sch\_owner.table\_schema::text || '.'::text) || v\_dba\_get\_tbl\_sch\_owner.tablename::text AS thesetables

FROM meta\_001.v\_dba\_get\_tbl\_sch\_owner

), pkey AS (

SELECT cc.conrelid,

format(',

constraint %I primary key(%s)'::text, cc.conname, string\_agg(a\_1.attname::text, ', '::text ORDER BY (array\_position(cc.conkey, a\_1.attnum)))) AS pkey

FROM pg\_constraint cc

JOIN pg\_class c\_1 ON c\_1.oid = cc.conrelid

JOIN pg\_attribute a\_1 ON a\_1.attrelid = cc.conrelid AND (a\_1.attnum = ANY (cc.conkey))

WHERE cc.contype = 'p'::"char"

GROUP BY cc.conrelid, cc.conname

)

SELECT format('create %stable %s%I

(

%s%s

);

'::text,

CASE c.relpersistence

WHEN 't'::"char" THEN 'temporary '::text

ELSE ''::text

END,

CASE c.relpersistence

WHEN 't'::"char" THEN ''::text

ELSE n.nspname::text || '.'::text

END, c.relname, string\_agg(format(' %I %s%s'::text, a.attname, format\_type(a.atttypid, a.atttypmod),

CASE

WHEN a.attnotnull THEN ' not null'::text

ELSE ''::text

END), ',

'::text ORDER BY a.attnum), ( SELECT pkey.pkey

FROM pkey

WHERE pkey.conrelid = c.oid)) AS sql,

c.relname,

n.nspname

FROM pg\_class c

JOIN pg\_namespace n ON n.oid = c.relnamespace

JOIN pg\_attribute a ON a.attrelid = c.oid AND a.attnum > 0

JOIN pg\_type t ON a.atttypid = t.oid

WHERE ((n.nspname::text || '.'::text) || c.relname::text IN ( SELECT thistbls.thesetables

FROM thistbls))

GROUP BY c.oid, c.relname, c.relpersistence, n.nspname

ORDER BY n.nspname, c.relname;

CREATE OR REPLACE VIEW meta\_001."functions"

AS SELECT n.nspname AS s\_name,

p.proname AS f\_name,

pg\_get\_function\_result(p.oid) AS result\_data\_types,

pg\_get\_function\_arguments(p.oid) AS argument\_data\_types,

pg\_get\_userbyid(p.proowner) AS "Owner",

CASE

WHEN p.prosecdef THEN 'definer'::text

ELSE 'invoker'::text

END AS proc\_security,

array\_to\_string(p.proacl, ''::text) AS access\_privileges,

l.lanname AS proc\_language,

p.prosrc AS source\_code,

obj\_description(p.oid, 'pg\_proc'::name) AS description,

CASE

WHEN n.nspname <> ALL (ARRAY['pg\_catalog'::name, 'information\_schema'::name]) THEN false

ELSE true

END AS system\_object

FROM pg\_proc p

LEFT JOIN pg\_namespace n ON n.oid = p.pronamespace

LEFT JOIN pg\_language l ON l.oid = p.prolang;

CREATE OR REPLACE VIEW meta\_001."functions"

AS SELECT n.nspname AS s\_name,

p.proname AS f\_name,

pg\_get\_function\_result(p.oid) AS result\_data\_types,

pg\_get\_function\_arguments(p.oid) AS argument\_data\_types,

pg\_get\_userbyid(p.proowner) AS "Owner",

CASE

WHEN p.prosecdef THEN 'definer'::text

ELSE 'invoker'::text

END AS proc\_security,

array\_to\_string(p.proacl, ''::text) AS access\_privileges,

l.lanname AS proc\_language,

p.prosrc AS source\_code,

obj\_description(p.oid, 'pg\_proc'::name) AS description,

CASE

WHEN n.nspname <> ALL (ARRAY['pg\_catalog'::name, 'information\_schema'::name]) THEN false

ELSE true

END AS system\_object

FROM pg\_proc p

LEFT JOIN pg\_namespace n ON n.oid = p.pronamespace

LEFT JOIN pg\_language l ON l.oid = p.prolang;

COMMENT ON VIEW meta\_001."functions" IS 'Data dictionary view: Lists functions (procedures)';

COMMENT ON COLUMN meta\_001."functions".system\_object IS 'Allows to easily show/hide system objects.';

CREATE OR REPLACE VIEW meta\_001."tables"

AS SELECT n.nspname AS s\_name,

c.relname AS t\_name,

CASE

WHEN c.relkind = ANY (ARRAY['r'::"char", 'p'::"char"]) THEN 'table'::text

WHEN c.relkind = 's'::"char" THEN 'special'::text

WHEN c.relkind = 'f'::"char" THEN 'foreign table'::text

ELSE NULL::text

END AS type,

pg\_get\_userbyid(c.relowner) AS owner,

pg\_size\_pretty(pg\_table\_size(c.oid::regclass)) AS size\_pretty,

pg\_table\_size(c.oid::regclass) AS size\_bytes,

c.reltuples AS rows,

CASE

WHEN c.reltuples > 0::double precision THEN pg\_table\_size(c.oid::regclass)::double precision / c.reltuples

ELSE NULL::double precision

END AS bytes\_per\_row,

pg\_size\_pretty(pg\_total\_relation\_size(c.oid::regclass)) AS size\_plus\_indexes,

obj\_description(c.oid, 'pg\_class'::name) AS description,

CASE

WHEN n.nspname !~ '^pg\_toast'::text AND (n.nspname <> ALL (ARRAY['pg\_catalog'::name, 'information\_schema'::name])) THEN false

ELSE true

END AS system\_object,

mt.data\_source,

mt.sensitive

FROM pg\_class c

LEFT JOIN pg\_namespace n ON n.oid = c.relnamespace

LEFT JOIN meta\_001.meta\_table mt ON n.nspname = mt.s\_name AND c.relname = mt.t\_name

WHERE c.relkind = ANY (ARRAY['r'::"char", 'p'::"char", 's'::"char", 'f'::"char"]);

COMMENT ON VIEW meta\_001."tables" IS 'Data dictionary view: Lists tables';

COMMENT ON COLUMN meta\_001."tables".size\_pretty IS 'Size (pretty) of data and TOAST. Does not include indexes. Suitable for display';

COMMENT ON COLUMN meta\_001."tables".size\_bytes IS 'Size (bytes) of data and TOAST. Does not include indexes. Suitable for sorting. ';

COMMENT ON COLUMN meta\_001."tables".size\_plus\_indexes IS 'Total size (pretty) of data, TOAST, and indexes. Suitable for display';

COMMENT ON COLUMN meta\_001."tables".system\_object IS 'Allows to easily show/hide system objects.';

CREATE OR REPLACE VIEW meta\_001."views"

AS SELECT n.nspname AS s\_name,

c.relname AS v\_name,

CASE c.relkind

WHEN 'v'::"char" THEN 'view'::text

WHEN 'm'::"char" THEN 'materialized view'::text

ELSE NULL::text

END AS view\_type,

pg\_get\_userbyid(c.relowner) AS owner,

c.reltuples AS rows,

pg\_size\_pretty(pg\_table\_size(c.oid::regclass)) AS size\_pretty,

pg\_table\_size(c.oid::regclass) AS size\_bytes,

obj\_description(c.oid, 'pg\_class'::name) AS description,

CASE

WHEN n.nspname !~ '^pg\_toast'::text AND (n.nspname <> ALL (ARRAY['pg\_catalog'::name, 'information\_schema'::name])) THEN false

ELSE true

END AS system\_object

FROM pg\_class c

LEFT JOIN pg\_namespace n ON n.oid = c.relnamespace

WHERE (c.relkind = ANY (ARRAY['v'::"char", 'm'::"char"])) AND n.nspname !~ '^pg\_toast'::text;

COMMENT ON VIEW meta\_001."views" IS 'Data dictionary view: Lists views and materialized views';

COMMENT ON COLUMN meta\_001."views".system\_object IS 'Allows to easily show/hide system objects.';

CREATE OR REPLACE VIEW meta\_001.v\_dba\_get\_fk\_insert\_order

AS WITH RECURSIVE fkeys AS (

SELECT pg\_constraint.conrelid AS source,

pg\_constraint.confrelid AS target

FROM pg\_constraint

WHERE pg\_constraint.contype = 'f'::"char"

), tables AS (

(

SELECT pg\_class.oid AS table\_name,

1 AS level,

ARRAY[pg\_class.oid] AS trail,

false AS circular

FROM pg\_class

WHERE pg\_class.relkind = 'r'::"char" AND NOT (pg\_class.relnamespace::regnamespace::text ~~ ANY (ARRAY['pg\_catalog'::text, 'information\_schema'::text, 'pg\_temp\_%'::text]))

EXCEPT

SELECT fkeys.source,

1,

ARRAY[fkeys.source] AS "array",

false AS bool

FROM fkeys

) UNION ALL

SELECT fkeys.source,

tables.level + 1,

tables.trail || fkeys.source,

tables.trail @> ARRAY[fkeys.source]

FROM fkeys

JOIN tables ON tables.table\_name = fkeys.target

WHERE cardinality(array\_positions(tables.trail, fkeys.source)) < 2

), ordered\_tables AS (

SELECT DISTINCT ON (tables.table\_name) tables.table\_name,

tables.level,

tables.circular

FROM tables

ORDER BY tables.table\_name, tables.level DESC

)

SELECT ordered\_tables.table\_name::regclass AS table\_name,

ordered\_tables.level

FROM ordered\_tables

WHERE NOT ordered\_tables.circular

ORDER BY ordered\_tables.level, (ordered\_tables.table\_name::regclass);

commit;

CREATE OR REPLACE VIEW meta\_001.v\_dba\_aaa\_gold\_fk\_finder

AS WITH unnested\_confkey AS (

SELECT pg\_constraint.oid,

unnest(pg\_constraint.confkey) AS confkey

FROM pg\_constraint

), unnested\_conkey AS (

SELECT pg\_constraint.oid,

unnest(pg\_constraint.conkey) AS conkey

FROM pg\_constraint

), abc AS (

SELECT c.conname AS constraint\_name,

c.contype AS constraint\_type,

tbl.relname AS constraint\_table,

col.attname AS constraint\_column,

referenced\_tbl.relname AS referenced\_table,

referenced\_field.attname AS referenced\_column,

pg\_get\_constraintdef(c.oid) AS definition,

tbl.relnamespace::regnamespace::text AS parent\_schema,

referenced\_tbl.relnamespace::regnamespace::text AS fk\_schema

FROM pg\_constraint c

LEFT JOIN unnested\_conkey con ON c.oid = con.oid

LEFT JOIN pg\_class tbl ON tbl.oid = c.conrelid

LEFT JOIN pg\_attribute col ON col.attrelid = tbl.oid AND col.attnum = con.conkey

LEFT JOIN pg\_class referenced\_tbl ON c.confrelid = referenced\_tbl.oid

LEFT JOIN unnested\_confkey conf ON c.oid = conf.oid

LEFT JOIN pg\_attribute referenced\_field ON referenced\_field.attrelid = c.confrelid AND referenced\_field.attnum = conf.confkey

WHERE c.contype = 'f'::"char"

), find\_order AS (

SELECT

CASE

WHEN "position"(v\_dba\_get\_fk\_insert\_order.table\_name::text, '.'::text) = 0 THEN 'public.'::text

ELSE ''::text

END || v\_dba\_get\_fk\_insert\_order.table\_name::text AS full\_table\_name,

v\_dba\_get\_fk\_insert\_order.level

FROM meta\_001.v\_dba\_get\_fk\_insert\_order

)

SELECT abc.constraint\_name,

abc.constraint\_type,

abc.parent\_schema,

abc.constraint\_table,

( SELECT find\_order\_1.level

FROM find\_order find\_order\_1

WHERE find\_order\_1.full\_table\_name = ((abc.parent\_schema || '.'::text) || abc.constraint\_table::text)) AS parent\_insert\_order,

abc.constraint\_column,

abc.fk\_schema,

abc.referenced\_table,

COALESCE(( SELECT find\_order\_1.level

FROM find\_order find\_order\_1

WHERE find\_order\_1.full\_table\_name = ((abc.fk\_schema || '.'::text) || abc.referenced\_table::text)), 99, 100) AS fk\_insert\_order,

abc.referenced\_column,

abc.definition,

((('select count(1) from '::text || abc.parent\_schema) || '.'::text) || abc.constraint\_table::text) || ';'::text AS base\_sql,

((('select count(1) from '::text || abc.fk\_schema) || '.'::text) || abc.referenced\_table::text) || ';'::text AS ref\_sql

FROM abc,

find\_order

WHERE find\_order.full\_table\_name = ((abc.parent\_schema || '.'::text) || abc.constraint\_table::text) OR find\_order.full\_table\_name = ((abc.fk\_schema || '.'::text) || abc.referenced\_table::text)

ORDER BY (COALESCE(( SELECT find\_order\_1.level

FROM find\_order find\_order\_1

WHERE find\_order\_1.full\_table\_name = ((abc.fk\_schema || '.'::text) || abc.referenced\_table::text)), 99, 100)), (( SELECT find\_order\_1.level

FROM find\_order find\_order\_1

WHERE find\_order\_1.full\_table\_name = ((abc.parent\_schema || '.'::text) || abc.constraint\_table::text))), abc.fk\_schema, abc.parent\_schema;

/

CREATE OR REPLACE VIEW meta\_001.v\_dba\_migration\_reload\_copy\_order

AS WITH tbl\_cnts AS (

SELECT DISTINCT 0 AS parent\_insert\_order,

t.table\_schema,

t.table\_name,

(xpath('/row/cnt/text()'::text, t.xml\_count))[1]::text::integer AS row\_count,

0 AS fk\_insert\_order,

t.explain\_it

FROM ( SELECT tables.table\_name,

tables.table\_schema,

query\_to\_xml(format('select count(\*) as cnt from %I.%I'::text, tables.table\_schema, tables.table\_name), false, true, ''::text) AS xml\_count,

'get\_row\_cnts'::text AS explain\_it

FROM information\_schema.tables

WHERE tables.table\_schema::text !~~ 'pg%'::text AND tables.table\_schema::text !~~ 'information\_%'::text AND tables.table\_type::text = 'BASE TABLE'::text) t

ORDER BY t.table\_schema, t.table\_name

), fks\_in AS (

SELECT DISTINCT v\_dba\_aaa\_gold\_fk\_finder.parent\_insert\_order,

v\_dba\_aaa\_gold\_fk\_finder.parent\_schema,

v\_dba\_aaa\_gold\_fk\_finder.constraint\_table,

'-1'::integer someval,

v\_dba\_aaa\_gold\_fk\_finder.fk\_insert\_order,

(((v\_dba\_aaa\_gold\_fk\_finder.constraint\_name::text || '\_\_'::text) || v\_dba\_aaa\_gold\_fk\_finder.constraint\_column::text) || '\_\_'::text) || v\_dba\_aaa\_gold\_fk\_finder.definition AS explain\_it

FROM meta\_001.v\_dba\_aaa\_gold\_fk\_finder

)

SELECT xyz.parent\_insert\_order,

xyz.table\_schema,

xyz.table\_name,

xyz.row\_count,

xyz.fk\_insert\_order,

xyz.explain\_it,

CASE

WHEN xyz.fk\_insert\_order = 0 THEN 'ANY TIME'::text

ELSE substr(substr(xyz.explain\_it, "position"(xyz.explain\_it, 'REFERENCES '::text) + 11, 999), 1, "position"(substr(xyz.explain\_it, "position"(xyz.explain\_it, 'REFERENCES '::text) + 11, 999), '('::text) - 1)

END AS depends\_on\_tbl

FROM ( SELECT tbl\_cnts.parent\_insert\_order,

tbl\_cnts.table\_schema,

tbl\_cnts.table\_name,

tbl\_cnts.row\_count,

tbl\_cnts.fk\_insert\_order,

tbl\_cnts.explain\_it

FROM tbl\_cnts

UNION ALL

SELECT fks\_in.parent\_insert\_order,

fks\_in.parent\_schema,

fks\_in.constraint\_table,

--fks\_in."?column?",

fks\_in.someval,

fks\_in.fk\_insert\_order,

fks\_in.explain\_it

FROM fks\_in

ORDER BY 5, 1, 2, 3) xyz;

commit;

CREATE OR REPLACE VIEW meta\_001.dependency

AS WITH RECURSIVE preference AS (

SELECT 10 AS max\_depth,

16384 AS min\_oid,

'^(londiste|pgq|pg\_toast)'::text AS schema\_exclusion,

'^pg\_(conversion|language|ts\_(dict|template))'::text AS class\_exclusion,

'{"SCHEMA":"00", "TABLE":"01", "TABLE CONSTRAINT":"02", "DEFAULT VALUE":"03",

"INDEX":"05", "SEQUENCE":"06", "TRIGGER":"07", "FUNCTION":"08",

"VIEW":"10", "MATERIALIZED VIEW":"11", "FOREIGN TABLE":"12"}'::json AS type\_sort\_orders

), dependency\_pair AS (

SELECT dep.objid,

array\_agg(dep.objsubid ORDER BY dep.objsubid) AS objsubids,

upper(obj.type) AS object\_type,

COALESCE(obj.schema, "substring"(obj.identity, '(\w+?)\.'::text), ''::text) AS object\_schema,

obj.name AS object\_name,

obj.identity AS object\_identity,

dep.refobjid,

array\_agg(dep.refobjsubid ORDER BY dep.refobjsubid) AS refobjsubids,

upper(refobj.type) AS refobj\_type,

COALESCE(

CASE

WHEN refobj.type = 'schema'::text THEN refobj.identity

ELSE refobj.schema

END, "substring"(refobj.identity, '(\w+?)\.'::text), ''::text) AS refobj\_schema,

refobj.name AS refobj\_name,

refobj.identity AS refobj\_identity,

CASE dep.deptype

WHEN 'n'::"char" THEN 'normal'::text

WHEN 'a'::"char" THEN 'automatic'::text

WHEN 'i'::"char" THEN 'internal'::text

WHEN 'e'::"char" THEN 'extension'::text

WHEN 'p'::"char" THEN 'pinned'::text

ELSE NULL::text

END AS dependency\_type

FROM pg\_depend dep,

LATERAL pg\_identify\_object(dep.classid, dep.objid, 0) obj(type, schema, name, identity),

LATERAL pg\_identify\_object(dep.refclassid, dep.refobjid, 0) refobj(type, schema, name, identity),

preference

WHERE (dep.deptype = ANY ('{n,a}'::"char"[])) AND dep.objid >= preference.min\_oid::oid AND (dep.refobjid >= preference.min\_oid::oid OR dep.refobjid = 2200::oid) AND COALESCE(obj.schema, "substring"(obj.identity, '(\w+?)\.'::text), ''::text) !~ preference.schema\_exclusion AND COALESCE(

CASE

WHEN refobj.type = 'schema'::text THEN refobj.identity

ELSE refobj.schema

END, "substring"(refobj.identity, '(\w+?)\.'::text), ''::text) !~ preference.schema\_exclusion

GROUP BY dep.objid, obj.type, obj.schema, obj.name, obj.identity, dep.refobjid, refobj.type, refobj.schema, refobj.name, refobj.identity, dep.deptype

), dependency\_hierarchy AS (

SELECT DISTINCT 0 AS level,

root.refobjid AS objid,

root.refobj\_type AS object\_type,

root.refobj\_identity AS object\_identity,

NULL::text AS dependency\_type,

ARRAY[root.refobjid] AS dependency\_chain,

ARRAY[concat(preference.type\_sort\_orders ->> root.refobj\_type, root.refobj\_type, ':', root.refobj\_identity)] AS dependency\_sort\_chain

FROM dependency\_pair root,

preference

WHERE NOT (EXISTS ( SELECT 'x'::text AS text

FROM dependency\_pair branch

WHERE branch.objid = root.refobjid)) AND root.refobj\_schema !~ preference.schema\_exclusion

UNION ALL

SELECT parent.level + 1 AS level,

child.objid,

child.object\_type,

child.object\_identity,

child.dependency\_type,

parent.dependency\_chain || child.objid,

parent.dependency\_sort\_chain || concat(preference.type\_sort\_orders ->> child.object\_type, child.object\_type, ':', child.object\_identity)

FROM dependency\_pair child

JOIN dependency\_hierarchy parent ON parent.objid = child.refobjid,

preference

WHERE parent.level < preference.max\_depth AND child.object\_schema !~ preference.schema\_exclusion AND child.refobj\_schema !~ preference.schema\_exclusion AND NOT (child.objid = ANY (parent.dependency\_chain))

)

SELECT dependency\_hierarchy.level,

dependency\_hierarchy.objid,

dependency\_hierarchy.object\_type,

dependency\_hierarchy.object\_identity,

dependency\_hierarchy.dependency\_type,

dependency\_hierarchy.dependency\_chain,

dependency\_hierarchy.dependency\_sort\_chain

FROM dependency\_hierarchy

ORDER BY dependency\_hierarchy.dependency\_chain;

CREATE OR REPLACE FUNCTION meta\_001.dependency\_tree(search\_pattern text)

RETURNS TABLE(dependency\_tree text)

LANGUAGE sql

SECURITY DEFINER

AS $function$

-- Procedure to report depedency tree using regexp search pattern (relation-only)

WITH target AS (

SELECT objid, dependency\_chain

FROM meta\_001.dependency

WHERE object\_identity ~ search\_pattern

)

, list AS (

SELECT

format('%\*s%s %s', -4\*level

, CASE WHEN object\_identity ~ search\_pattern THEN '\*' END

, object\_type, object\_identity

) AS dependency\_tree

, dependency\_sort\_chain

FROM target

JOIN meta\_001.dependency report

ON report.objid = ANY(target.dependency\_chain) -- root-bound chain

OR target.objid = ANY(report.dependency\_chain) -- leaf-bound chain

WHERE length(search\_pattern) > 0

-- Do NOT waste search time on blank/null search\_pattern.

UNION

-- Query the entire dependencies instead.

SELECT

format('%\*s%s %s', 4\*level, '', object\_type, object\_identity) AS depedency\_tree

, dependency\_sort\_chain

FROM meta\_001.dependency

WHERE length(coalesce(search\_pattern,'')) = 0

)

SELECT dependency\_tree FROM list

ORDER BY dependency\_sort\_chain;

$function$

;

commit;

CREATE OR REPLACE VIEW meta\_001.v\_dba\_newdev\_migration\_deps

AS WITH abc AS (

SELECT views.s\_name,

views.v\_name AS objofname,

(views.s\_name::text || '.'::text) || views.v\_name::text AS full\_name\_of

FROM meta\_001.views

WHERE views.s\_name <> ALL (ARRAY['pg\_catalog'::name, 'information\_schema'::name])

), xyz AS (

SELECT functions.s\_name,

functions.f\_name AS objofname,

(functions.s\_name::text || '.'::text) || functions.f\_name::text AS full\_name\_of

FROM meta\_001.functions

WHERE functions.s\_name <> ALL (ARRAY['pg\_catalog'::name, 'information\_schema'::name])

), tab AS (

SELECT tables.s\_name,

tables.t\_name AS objofname,

(tables.s\_name::text || '.'::text) || tables.t\_name::text AS full\_name\_of

FROM meta\_001.tables

WHERE tables.s\_name <> ALL (ARRAY['pg\_catalog'::name, 'information\_schema'::name])

)

SELECT qwerty.typeofobj,

qwerty.objname,

qwerty.full\_name\_of,

replace(qwerty.xxz, 'RULE "\_RETURN" '::text, 'VIEW '::text) AS xxz

FROM ( SELECT DISTINCT 'views'::text AS typeofobj,

abc.objofname AS objname,

abc.full\_name\_of,

dependency\_tree.dependency\_tree AS xxz

FROM abc,

LATERAL meta\_001.dependency\_tree(abc.full\_name\_of) dependency\_tree(dependency\_tree)

UNION ALL

SELECT DISTINCT 'function'::text AS typeofobj,

xyz.objofname AS objname,

xyz.full\_name\_of,

dependency\_tree.dependency\_tree AS xxz

FROM xyz,

LATERAL meta\_001.dependency\_tree(xyz.full\_name\_of) dependency\_tree(dependency\_tree)

UNION ALL

SELECT DISTINCT 'table'::text AS typeofobj,

tab.objofname AS objname,

tab.full\_name\_of,

dependency\_tree.dependency\_tree AS xxz

FROM tab,

LATERAL meta\_001.dependency\_tree(tab.full\_name\_of) dependency\_tree(dependency\_tree)) qwerty

ORDER BY qwerty.full\_name\_of, (replace(qwerty.xxz, 'RULE "\_RETURN" '::text, 'VIEW '::text));

CREATE OR REPLACE VIEW meta\_001.v\_dba\_newdev\_migration\_table\_\_restore1

AS WITH wip\_dir AS (

SELECT chr(39) || '/opt/oracle/ora\_ext\_data/build\_dev/'::text AS bnr\_loc

), abc AS (

SELECT v\_dba\_get\_tbl\_ddl.nspname,

current\_database() AS current\_db,

v\_dba\_get\_tbl\_ddl.relname,

wip\_dir.bnr\_loc

FROM meta\_001.v\_dba\_get\_tbl\_ddl,

wip\_dir

ORDER BY (current\_database()), v\_dba\_get\_tbl\_ddl.nspname, v\_dba\_get\_tbl\_ddl.relname

), xyz AS (

SELECT (((((((((((((((((((((((('copy '::text || lpad((abc.nspname::text || '.'::text) || abc.relname::text, 60)) || ' FROM '::text) || abc.bnr\_loc) || abc.nspname::text) || '.'::text) || abc.relname::text) || '\_loaded\_cnts'::text) ||

CASE

WHEN inet\_client\_addr() = '12.21.38.37'::inet THEN 'STAGE'::text

WHEN inet\_client\_addr() = '3.86.2.150'::inet THEN 'DEV'::text

ELSE 'UNKNOWN'::text

END) || '\_'::text) || abc.current\_db::text) || '\_\_'::text) || abc.nspname::text) || '\_\_'::text) || abc.relname::text) || '.csv'::text) || chr(39)) || ' with (format csv,header true, delimiter '::text) || chr(39)) || ','::text) || chr(39)) || ');" > '::text) || abc.nspname::text) || '.'::text) || abc.relname::text) || '\_\_insert\_results.log '::text AS sqltext,

abc.nspname,

abc.current\_db,

abc.relname

FROM abc

WHERE ((abc.nspname::text || '.'::text) || abc.relname::text) <> 'log\_001.t\_audit'::text

)

SELECT zzaa.sqltext,

zzaa.nspname,

zzaa.current\_db,

zzaa.relname,

zzaa.orderby

FROM ( SELECT '#! we need to use the postgres account with superuser to load this data into --dbname=wfma

export WHICH\_ENV=DEV

export DEV\_WFMA\_URL=jdbc:postgresql://aus0devpostgres.c3znzejcnw82.us-east-1.rds.amazonaws.com:5432/wfma

export DEV\_WFMA\_USERNAME=xpTwj05YAVoE

export DEV\_WFMA\_USER\_PASSWORD=<not provided>

export DEV\_DB=wfma

export DEV\_DB2=identity

export DEV\_PORT=5432

export DEV\_HOST=aus0devpostgres.c3znzejcnw82.us-east-1.rds.amazonaws.com

export PGPASSWORD=${DEV\_WFMA\_USER\_PASSWORD}

#!

echo "/usr/pgsql-13/bin/psql --host=${DEV\_HOST} --username=${DEV\_WFMA\_USERNAME} --dbname=${DEV\_DB} --port=${DEV\_PORT} -w "

/usr/pgsql-13/bin/psql --host=${DEV\_HOST} --username=${DEV\_WFMA\_USERNAME} --dbname=${DEV\_DB} --port=${DEV\_PORT} -w <<EOF'::text AS sqltext,

' '::name AS nspname,

' '::name AS current\_db,

' '::name AS relname,

1 AS orderby

UNION ALL

SELECT '/usr/pgsql-13/bin/psql --host=${DEV\_HOST} --username=${DEV\_WFMA\_USERNAME} --dbname=${DEV\_DB} --port=${DEV\_PORT} -c "\'::text || xyz.sqltext AS sqltext,

xyz.nspname,

xyz.current\_db,

xyz.relname,

2 AS orderby

FROM xyz

UNION ALL

SELECT 'EOF'::text AS sqltext,

' '::name AS nspname,

' '::name AS current\_db,

' '::name AS relname,

3 AS orderby) zzaa

ORDER BY zzaa.orderby;

CREATE OR REPLACE VIEW meta\_001.v\_dba\_newdev\_migration\_table\_\_restore2

AS WITH wip\_dir AS (

SELECT chr(39) || '/opt/oracle/ora\_ext\_data/build\_dev/'::text AS bnr\_loc

), abc AS (

SELECT v\_dba\_get\_tbl\_ddl.nspname,

current\_database() AS current\_db,

v\_dba\_get\_tbl\_ddl.relname,

wip\_dir.bnr\_loc

FROM meta\_001.v\_dba\_get\_tbl\_ddl,

wip\_dir

ORDER BY (current\_database()), v\_dba\_get\_tbl\_ddl.nspname, v\_dba\_get\_tbl\_ddl.relname

), xyz AS (

SELECT ((((((((((((((((((((((((('copy '::text || lpad((abc.nspname::text || '.'::text) || abc.relname::text, 60)) || ' FROM '::text) || abc.bnr\_loc) || to\_char(CURRENT\_TIMESTAMP, 'DY'::text)) || date\_part('year'::text, CURRENT\_TIMESTAMP)) || to\_char(date\_part('month'::text, CURRENT\_TIMESTAMP), 'fm00'::text)) || to\_char(date\_part('day'::text, CURRENT\_TIMESTAMP), 'fm00'::text)) || '\_'::text) || 'STG'::text) || '\_'::text) || abc.current\_db::text) || '\_\_'::text) || abc.nspname::text) || '\_\_'::text) || abc.relname::text) || '.csv'::text) || chr(39)) || ' with (format csv,header true, delimiter '::text) || chr(39)) || ','::text) || chr(39)) || ');" > '::text) || abc.nspname::text) || '.'::text) || abc.relname::text) || '\_\_insert\_results.log '::text AS sqltext,

abc.nspname,

abc.current\_db,

abc.relname

FROM abc

WHERE ((abc.nspname::text || '.'::text) || abc.relname::text) <> 'log\_001.t\_audit'::text

)

SELECT zzaa.sqltext,

zzaa.nspname,

zzaa.current\_db,

zzaa.relname,

zzaa.orderby

FROM ( SELECT '#! we need to use the postgres account with superuser to load this data into --dbname=wfma

export WHICH\_ENV=DEV

export DEV\_WFMA\_URL=jdbc:postgresql://aus0devpostgres.c3znzejcnw82.us-east-1.rds.amazonaws.com:5432/wfma

export DEV\_WFMA\_USERNAME=xpTwj05YAVoE

export DEV\_WFMA\_USER\_PASSWORD=<not provided>

export DEV\_DB=wfma

export DEV\_DB2=identity

export DEV\_PORT=5432

export DEV\_HOST=aus0devpostgres.c3znzejcnw82.us-east-1.rds.amazonaws.com

export PGPASSWORD=${DEV\_WFMA\_USER\_PASSWORD}

#!

echo "/usr/pgsql-13/bin/psql --host=${DEV\_HOST} --username=${DEV\_WFMA\_USERNAME} --dbname=${DEV\_DB} --port=${DEV\_PORT} -w "

/usr/pgsql-13/bin/psql --host=${DEV\_HOST} --username=${DEV\_WFMA\_USERNAME} --dbname=${DEV\_DB} --port=${DEV\_PORT} -w <<EOF'::text AS sqltext,

' '::name AS nspname,

' '::name AS current\_db,

' '::name AS relname,

1 AS orderby

UNION ALL

SELECT '/usr/pgsql-13/bin/psql --host=${DEV\_HOST} --username=${DEV\_WFMA\_USERNAME} --dbname=${DEV\_DB} --port=${DEV\_PORT} -c "\'::text || xyz.sqltext AS sqltext,

xyz.nspname,

xyz.current\_db,

xyz.relname,

2 AS orderby

FROM xyz

UNION ALL

SELECT 'EOF'::text AS sqltext,

' '::name AS nspname,

' '::name AS current\_db,

' '::name AS relname,

3 AS orderby) zzaa

ORDER BY zzaa.orderby;

CREATE OR REPLACE VIEW meta\_001.v\_dba\_newdev\_migration\_table\_stg\_copy

AS WITH wip\_dir AS (

SELECT ' /opt/oracle/ora\_ext\_data/build\_dev/'::text AS bnr\_loc

), abc AS (

SELECT v\_dba\_get\_tbl\_ddl.nspname,

current\_database() AS current\_db,

v\_dba\_get\_tbl\_ddl.relname,

wip\_dir.bnr\_loc

FROM meta\_001.v\_dba\_get\_tbl\_ddl,

wip\_dir

ORDER BY (current\_database()), v\_dba\_get\_tbl\_ddl.nspname, v\_dba\_get\_tbl\_ddl.relname

)

SELECT xxxhz.sqltext,

xxxhz.nspname,

xxxhz.current\_db,

xxxhz.relname,

xxxhz.orderof

FROM ( SELECT ((((((((((((((((((('\copy (SELECT \* FROM '::text || lpad((abc.nspname::text || '.'::text) || abc.relname::text, 60)) || ' ) TO '::text) || abc.bnr\_loc) || to\_char(CURRENT\_TIMESTAMP, 'DY'::text)) || date\_part('year'::text, CURRENT\_TIMESTAMP)) || to\_char(date\_part('month'::text, CURRENT\_TIMESTAMP), 'fm00'::text)) || to\_char(date\_part('day'::text, CURRENT\_TIMESTAMP), 'fm00'::text)) || '\_'::text) || 'STG'::text) || '\_'::text) || abc.current\_db::text) || '\_\_'::text) || abc.nspname::text) || '\_\_'::text) || abc.relname::text) || '.csv WITH (FORMAT csv, HEADER, ENCODING '::text) || chr(39)) || 'UTF8'::text) || chr(39)) || ') '::text AS sqltext,

abc.nspname,

abc.current\_db,

abc.relname,

2 AS orderof

FROM abc

WHERE 1 = 1

UNION ALL

SELECT '#! we need to use the postgres account with dbuser to extract this data

export WHICH\_ENV=STAGE

export STAGE\_WFMA\_URL=jdbc:postgresql://aus0stage0postgres.c3znzejcnw82.us-east-1.rds.amazonaws.com:5432/wfma

export STAGE\_WFMA\_USERNAME=IHSLaMltbFn1

export STAGE\_WFMA\_USER\_PASSWORD=<not provided>

export STAGE\_DB=wfma

export STAGE\_DB2=identity

export STAGE\_PORT=5432

export STAGE\_HOST=aus0stage0postgres.c3znzejcnw82.us-east-1.rds.amazonaws.com

export PGPASSWORD=${STAGE\_WFMA\_USER\_PASSWORD}

#!

/usr/pgsql-13/bin/psql --host=${WHICH\_ENV}\_HOST --username=${WHICH\_ENV}\_WFMA\_USERNAME --dbname=${WHICH\_ENV}\_DB --port=${WHICH\_ENV}\_PORT -w <<EOF'::text AS sqltext,

' '::name AS nspname,

' '::name AS current\_db,

' '::name AS relname,

1 AS orderof

UNION ALL

SELECT ('exit;'::text || chr(10)) || 'EOF'::text AS sqltext,

' '::name AS nspname,

' '::name AS current\_db,

' '::name AS relname,

3 AS orderof) xxxhz

ORDER BY xxxhz.orderof;

CREATE OR REPLACE VIEW meta\_001.v\_dba\_newdev\_migration\_table\_prd\_copy

AS WITH wip\_dir AS (

SELECT ' /opt/oracle/ora\_ext\_data/build\_dev/'::text AS bnr\_loc

), abc AS (

SELECT v\_dba\_get\_tbl\_ddl.nspname,

current\_database() AS current\_db,

v\_dba\_get\_tbl\_ddl.relname,

wip\_dir.bnr\_loc

FROM meta\_001.v\_dba\_get\_tbl\_ddl,

wip\_dir

ORDER BY (current\_database()), v\_dba\_get\_tbl\_ddl.nspname, v\_dba\_get\_tbl\_ddl.relname

)

SELECT xxxhz.sqltext,

xxxhz.nspname,

xxxhz.current\_db,

xxxhz.relname,

xxxhz.orderof

FROM ( SELECT ((((((((((((((((((('\copy (SELECT \* FROM '::text || lpad((abc.nspname::text || '.'::text) || abc.relname::text, 60)) || ' ) TO '::text) || abc.bnr\_loc) || to\_char(CURRENT\_TIMESTAMP, 'DY'::text)) || date\_part('year'::text, CURRENT\_TIMESTAMP)) || to\_char(date\_part('month'::text, CURRENT\_TIMESTAMP), 'fm00'::text)) || to\_char(date\_part('day'::text, CURRENT\_TIMESTAMP), 'fm00'::text)) || '\_'::text) || 'STG'::text) || '\_'::text) || abc.current\_db::text) || '\_\_'::text) || abc.nspname::text) || '\_\_'::text) || abc.relname::text) || '.csv WITH (FORMAT csv, HEADER, ENCODING '::text) || chr(39)) || 'UTF8'::text) || chr(39)) || ') '::text AS sqltext,

abc.nspname,

abc.current\_db,

abc.relname,

2 AS orderof

FROM abc

WHERE 1 = 1

UNION ALL

SELECT '#! we need to use the postgres account with dbuser to extract this data

export WHICH\_ENV=STAGE

export STAGE\_WFMA\_URL=jdbc:postgresql://aus0stage0postgres.c3znzejcnw82.us-east-1.rds.amazonaws.com:5432/wfma

export STAGE\_WFMA\_USERNAME=IHSLaMltbFn1

export STAGE\_WFMA\_USER\_PASSWORD=<not provided>

export STAGE\_DB=wfma

export STAGE\_DB2=identity

export STAGE\_PORT=5432

export STAGE\_HOST=aus0stage0postgres.c3znzejcnw82.us-east-1.rds.amazonaws.com

export PGPASSWORD=${STAGE\_WFMA\_USER\_PASSWORD}

#!

/usr/pgsql-13/bin/psql --host=${WHICH\_ENV}\_HOST --username=${WHICH\_ENV}\_WFMA\_USERNAME --dbname=${WHICH\_ENV}\_DB --port=${WHICH\_ENV}\_PORT -w <<EOF'::text AS sqltext,

' '::name AS nspname,

' '::name AS current\_db,

' '::name AS relname,

1 AS orderof

UNION ALL

SELECT ('exit;'::text || chr(10)) || 'EOF'::text AS sqltext,

' '::name AS nspname,

' '::name AS current\_db,

' '::name AS relname,

3 AS orderof) xxxhz

ORDER BY xxxhz.orderof;

/

**SELECT** sqltext **FROM** meta\_001.v\_dba\_newdev\_migration\_table\_stg\_copy;

SELECT x.\* FROM meta\_001.v\_dba\_newdev\_migration\_table\_\_restore2 x WHERE relname = 't\_login\_fails' or orderby in (1,3);

**If we have referential integrity issues ie dependencies betweeen tables , then a insert order must be implemented.:**

**SELECT** parent\_insert\_order, table\_schema, table\_name, row\_count, fk\_insert\_order, explain\_it, depends\_on\_tbl **FROM** meta\_001.v\_dba\_migration\_reload\_copy\_order where

parent\_insert\_order = 0 and fk\_insert\_order = 0;

**SELECT** parent\_insert\_order, table\_schema, table\_name, row\_count, fk\_insert\_order, explain\_it, depends\_on\_tbl **FROM** meta\_001.v\_dba\_migration\_reload\_copy\_order where

parent\_insert\_order <> 0 and fk\_insert\_order <> 0; /\* can be inserted only in order, where there are dependencies \*/

/\* Like below \*/

All columns with fk\_insert\_order must be completed first then the parent\_insert\_order matters thereafter.

The specifics of how tables are inter-related can also be determined with this query

**SELECT** x.\* **FROM** meta\_001.v\_dba\_newdev\_migration\_deps x

**WHERE** objname = 'v\_somename'

**ORDER** **BY** x.xxz **DESC;**

**/\*create** **or** **replace** **view** meta\_001.v\_dba\_migration\_functions\_out\_bkup **as**

**with** abc **as**

( **SELECT** n.nspname ,

**replace**( substr( pg\_get\_functiondef(f.**oid**) , 1, (**position**('(' **in** pg\_get\_functiondef(f.**oid**) ) -1)) , 'CREATE OR REPLACE FUNCTION '||n.nspname||'.' ,'')||'\_'||f.**oid** function\_name

,f.pronamespace , f.**oid**,

**replace**(pg\_get\_functiondef(f.**oid**), E'\n','')||';' function\_sql

**-----** pg\_get\_functiondef(f.**oid**) function\_sql

**FROM** pg\_catalog.pg\_proc f

**INNER** **JOIN** pg\_catalog.pg\_namespace n **ON** (f.pronamespace = n.**oid**)

**WHERE** n.nspname **not** **in** ( 'information\_schema','pg\_catalog')

) , xxo **as** (

**select** 'DEV\_\_'||nspname||'\_\_'||function\_name||'\_\_FUNCTION.sql' file\_name,

abc.\*

**from** abc )

**select** xxo.\*,

'/usr/pgsql-13/bin/psql -t --host=${DEV\_HOST} --port=${DEV\_PORT} --username=${DEV\_WFMA\_USERNAME} --dbname=wf -c "select function\_sql from meta\_001.v\_dba\_migration\_functions\_out where file\_name = '||**chr**(39)||file\_name||**chr**(39)||'; " > '||file\_name sqltext

**from** xxo;

**commit**;

\*/

**CREATE** **OR** **REPLACE** **VIEW** meta\_001.v\_dba\_migration\_functions\_out

**AS** **WITH** abc **AS** (

**SELECT** n.nspname,

(**replace**(substr(pg\_get\_functiondef(f.**oid**), 1, "position"(pg\_get\_functiondef(f.**oid**), '('::**text**) - 1), ('CREATE OR REPLACE FUNCTION '::**text** || n.nspname::**text**) || '.'::**text**, ''::**text**) || '\_'::**text**) || f.**oid** **AS** function\_name,

f.pronamespace,

f.**oid**,

**replace**(pg\_get\_functiondef(f.**oid**), '

'::**text**, ''::**text**) || ';'::**text** **AS** function\_sql

**FROM** pg\_proc f

**JOIN** pg\_namespace n **ON** f.pronamespace = n.**oid**

**WHERE** n.nspname <> **ALL** (**ARRAY**['information\_schema'::**name**, 'pg\_catalog'::**name**])

), xxo **AS** (

**SELECT** ((('DEV\_\_'::**text** || abc.nspname::**text**) || '\_\_'::**text**) || abc.function\_name) || '\_\_FUNCTION.sql'::**text** **AS** file\_name,

abc.nspname,

abc.function\_name,

abc.pronamespace,

abc.**oid**,

**replace**(**replace**(**replace**(**replace**(**replace**(**replace**(**replace**(abc.function\_sql, 'STRICTAS'::**text**, ' STRICT AS '::**text**), 'STABLEAS'::**text**, ' STABLE AS '::**text**), 'DEFINERAS'::**text**, ' DEFINER AS '::**text**), 'plpgsqlAS'::**text**, ' plpgsql AS '::**text**), 'SAFEAS'::**text**, ' SAFE AS '::**text**), ((**chr**(39) || 'iso'::**text**) || **chr**(39)) || 'AS'::**text**, (((' '::**text** || **chr**(39)) || 'iso'::**text**) || **chr**(39)) || ' AS '::**text**), 'sqlAS'::**text**, ' sql AS '::**text**) **AS** function\_sql

**FROM** abc

)

**SELECT** xxo.file\_name,

xxo.nspname,

xxo.function\_name,

xxo.pronamespace,

xxo.**oid**,

xxo.function\_sql,

(((('/usr/pgsql-13/bin/psql -t --host=${DEV\_HOST} --port=${DEV\_PORT} --username=${DEV\_WFMA\_USERNAME} --dbname=${DEV\_DB} -c "select function\_sql from meta\_001.v\_dba\_migration\_functions\_out where file\_name = '::**text** || **chr**(39)) || xxo.file\_name) || **chr**(39)) || '; " > '::**text**) || xxo.file\_name **AS** sqltext

-- (((('/usr/pgsql-13/bin/psql -t --host=${STAGE\_HOST} --port=${STAGE\_PORT} --username=${STAGE\_WFMA\_USERNAME} --dbname=${STAGE\_DB} -c "select function\_sql from meta\_001.v\_dba\_migration\_functions\_out where file\_name = '::text || chr(39)) || xxo.file\_name) || chr(39)) || '; " > '::text) || xxo.file\_name AS sqltext

-- (((('/usr/pgsql-13/bin/psql -t --host=${PROD\_HOST} --port=${PROD\_PORT} --username=${PROD\_WFMA\_USERNAME} --dbname=${PROD\_DB} -c "select function\_sql from meta\_001.v\_dba\_migration\_functions\_out where file\_name = '::text || chr(39)) || xxo.file\_name) || chr(39)) || '; " > '::text) || xxo.file\_name AS sqltext

**FROM** xxo;

**commit**;

**select trim( trailing '+' from**

**'**

**if [[ 1 -eq 2 ]]**

**then**

**export WHICH\_ENV=DEV**

**export DEV\_WFMA\_URL=jdbc:postgresql://aus0devpostgres.c3znzejcnw82.us-east-1.rds.amazonaws.com:5432/wfma**

**export DEV\_WFMA\_USERNAME=xpTwj05YAVoE**

**export DEV\_WFMA\_USER\_PASSWORD=<replace>**

**export DEV\_DB=wfma**

**export DEV\_DB2=identity**

**export DEV\_PORT=5432**

**export DEV\_HOST=aus0devpostgres.c3znzejcnw82.us-east-1.rds.amazonaws.com**

**export PGPASSWORD=${DEV\_WFMA\_USER\_PASSWORD}**

**export PATH=${PATH}:.:${PWD}**

**#!export psqlcmd\_wfma=`/usr/pgsql-13/bin/pg\_dump --host=${DEV\_HOST} --port=${DEV\_PORT} --username=${DEV\_WFMA\_USERNAME} --dbname=wfma --schema-only `**

**#!export psqlcmd\_identity=`/usr/pgsql-13/bin/pg\_dump --host=${DEV\_HOST} --port=${DEV\_PORT} --username=${DEV\_WFMA\_USERNAME} --dbname=identity --schema-only `**

**#!echo "/usr/pgsql-13/bin/pg\_dump --host=${DEV\_HOST} --port=${DEV\_PORT} --username=${DEV\_WFMA\_USERNAME} --dbname=wfma --schema-only -t '||chr(39)||'"${ij}"'||chr(39)||' > ${this\_output\_file} "**

**#! /usr/pgsql-13/bin/pg\_dump --host=${DEV\_HOST} --port=${DEV\_PORT} --username=${DEV\_WFMA\_USERNAME} --dbname=wfma --schema-only -t ${ij} > ${this\_output\_file}**

**#!echo "/usr/pgsql-13/bin/psql --host=${DEV\_HOST} --username=${DEV\_WFMA\_USERNAME} --dbname=${DEV\_DB} --port=${DEV\_PORT} -w "**

**#! /usr/pgsql-13/bin/psql --host=${DEV\_HOST} --username=${DEV\_WFMA\_USERNAME} --dbname=${DEV\_DB} --port=${DEV\_PORT} -w**

**fi**

**#! ====================================================================**

**if [[ 1 -eq 1 ]]**

**then**

**export WHICH\_ENV=STAGE**

**export STAGE\_WFMA\_URL=jdbc:postgresql://aus0stage0postgres.c3znzejcnw82.us-east-1.rds.amazonaws.com:5432/wfma**

**export STAGE\_WFMA\_USERNAME=IHSLaMltbFn1**

**export STAGE\_WFMA\_USER\_PASSWORD=<replace>**

**export STAGE\_DB=wfma**

**export STAGE\_DB2=identity**

**export STAGE\_PORT=5432**

**export STAGE\_HOST=aus0stage0postgres.c3znzejcnw82.us-east-1.rds.amazonaws.com**

**export PGPASSWORD=${STAGE\_WFMA\_USER\_PASSWORD}**

**export PATH=${PATH}:.:${PWD}**

**#!export psqlcmd\_wfma=`/usr/pgsql-13/bin/pg\_dump --host=${STAGE\_HOST} --port=${STAGE\_PORT} --username=${STAGE\_WFMA\_USERNAME} --dbname=wfma --schema-only `**

**#!export psqlcmd\_identity=`/usr/pgsql-13/bin/pg\_dump --host=${STAGE\_HOST} --port=${STAGE\_PORT} --username=${STAGE\_WFMA\_USERNAME} --dbname=identity --schema-only `**

**#!echo "/usr/pgsql-13/bin/pg\_dump --host=${STAGE\_HOST} --port=${STAGE\_PORT} --username=${STAGE\_WFMA\_USERNAME} --dbname=wfma --schema-only -t '||chr(39)||'"${ij}"'||chr(39)||' > ${this\_output\_file} "**

**#! /usr/pgsql-13/bin/pg\_dump --host=${STAGE\_HOST} --port=${STAGE\_PORT} --username=${STAGE\_WFMA\_USERNAME} --dbname=wfma --schema-only -t ${ij} > ${this\_output\_file}**

**echo "/usr/pgsql-13/bin/psql --host=${STAGE\_HOST} --username=${STAGE\_WFMA\_USERNAME} --dbname=${STAGE\_DB} --port=${STAGE\_PORT} -w "**

**#! /usr/pgsql-13/bin/psql --host=${STAGE\_HOST} --username=${STAGE\_WFMA\_USERNAME} --dbname=${STAGE\_DB} --port=${STAGE\_PORT} -w**

**fi**

**#! ====================================================================**

**if [[ 1 -eq 2 ]]**

**then**

**export WHICH\_ENV=PROD**

**export PROD\_WFMA\_URL=jdbc:postgresql://aus0beta0postgres.c3znzejcnw82.us-east-1.rds.amazonaws.com:5432/wfma**

**export PROD\_WFMA\_USERNAME=xNgjdHF5ZeOd**

**export PROD\_WFMA\_USER\_PASSWORD=<replace>**

**export PROD\_DB=wfma**

**export PROD\_DB2=identity**

**export PROD\_PORT=5432**

**export PROD\_HOST=aus0beta0postgres.c3znzejcnw82.us-east-1.rds.amazonaws.com**

**export PGPASSWORD=${PROD\_WFMA\_USER\_PASSWORD}**

**export PATH=${PATH}:.:${PWD}**

**#!export psqlcmd\_wfma=`/usr/pgsql-13/bin/pg\_dump --host=${PROD\_HOST} --port=${PROD\_PORT} --username=${PROD\_WFMA\_USERNAME} --dbname=wfma --schema-only `**

**#!export psqlcmd\_identity=`/usr/pgsql-13/bin/pg\_dump --host=${PROD\_HOST} --port=${PROD\_PORT} --username=${PROD\_WFMA\_USERNAME} --dbname=identity --schema-only `**

**#!echo "/usr/pgsql-13/bin/pg\_dump --host=${PROD\_HOST} --username=${PROD\_WFMA\_USERNAME} --dbname=wfma --port=${PROD\_PORT} --schema-only -t '||chr(39)||'"${ij}"'||chr(39)||' > ${this\_output\_file} "**

**#! /usr/pgsql-13/bin/pg\_dump --host=${PROD\_HOST} --username=${PROD\_WFMA\_USERNAME} --dbname=wfma --port=${PROD\_PORT} --schema-only -t ${ij} > ${this\_output\_file}**

**echo "/usr/pgsql-13/bin/psql --host=${PROD\_HOST} --username=${PROD\_WFMA\_USERNAME} --dbname=${PROD\_DB} --port=${PROD\_PORT} -w "**

**#! /usr/pgsql-13/bin/psql --host=${PROD\_HOST} --username=${PROD\_WFMA\_USERNAME} --dbname=${PROD\_DB} --port=${PROD\_PORT} -w**

**fi**

**cd /opt/oracle/ora\_ext\_data/build\_dev**

**' ) sqltext**

**union** **all**

**select** sqltext sqltext **from** meta\_001.v\_dba\_migration\_functions\_out;

create or replace function meta\_001.f\_dba\_ddlx\_get\_dependants(

in oid,

out depth int, out classid regclass, out objid oid

)

returns setof record as $$

with recursive

tree(depth,classid,objid,objsubid,refclassid,refobjid,refobjsubid,deptype,edges)

as (

select 1,

case when r.oid is not null then 'pg\_class'::regclass

else d.classid::regclass

end as classid,

coalesce(r.ev\_class,d.objid) as objid,

d.objsubid, d.refclassid, d.refobjid,d.refobjsubid, d.deptype,

array[array[d.refobjid::int,d.objid::int]]

from pg\_depend d

left join pg\_rewrite r on

(r.oid = d.objid and r.ev\_type = '1' and r.rulename = '\_RETURN')

where d.refobjid = $1 and r.ev\_class is distinct from d.refobjid

union all

select depth+1,

case when r.oid is not null then 'pg\_class'::regclass

else d.classid::regclass

end as classid,

coalesce(r.ev\_class,d.objid) as objid,

d.objsubid, d.refclassid, d.refobjid, d.refobjsubid, d.deptype,

t.edges || array[array[d.refobjid::int,d.objid::int]]

from tree t

join pg\_depend d on (d.refobjid=t.objid)

left join pg\_rewrite r on

(r.oid = d.objid and r.ev\_type = '1' and r.rulename = '\_RETURN')

where r.ev\_class is distinct from d.refobjid

and not ( t.edges @> array[array[d.refobjid::int,d.objid::int]] )

),

ddlx\_get\_dependants\_recursive as (

select distinct

depth,

classid,objid,objsubid,

refclassid,refobjid,refobjsubid,

deptype

from tree

),

q as (

select distinct depth,classid,objid

from ddlx\_get\_dependants\_recursive

where deptype = 'n'

)

select depth,classid,objid

from q

where (objid,depth) in (select objid,max(depth) from q group by objid)

order by depth,objid

$$ language sql;

commit;

create or replace view meta\_001.v\_dba\_RE\_type\_definition as

with abc as (

SELECT

--n.oid pg\_namespace\_oid,

t.oid pg\_type\_oid ,

n.nspname AS schema\_of,

t.typname ,

string\_agg(e.enumlabel, chr(39)||chr(10)||' '||','||chr(39) ORDER BY e.enumsortorder) AS enum\_labels

FROM pg\_catalog.pg\_type t

JOIN pg\_catalog.pg\_namespace n ON n.oid = t.typnamespace

JOIN pg\_catalog.pg\_enum e ON t.oid = e.enumtypid

group by n.nspname , t.typname , n.oid, t.oid),

xyz as (

select pg\_type\_oid, schema\_of, typname, enum\_labels , 1 sourceof , 'drop type '||schema\_of||'.'||typname||';'||chr(10)||

'CREATE TYPE ' ||schema\_of||'.'|| format\_type(pg\_type\_oid,null) || ' AS ENUM ('||chr(39)||enum\_labels||chr(39)||');'||chr(10) cr8\_sql from abc union all

select pg\_type.oid, null , pg\_type.typname, pg\_enum.enumlabel , 2 sourceof , null

from pg\_type

join pg\_enum ON pg\_enum.enumtypid = pg\_type.oid

order by pg\_type\_oid, sourceof, typname, enum\_labels )

select pg\_type\_oid, schema\_of, typname, enum\_labels , sourceof , cr8\_sql , null dependson from xyz

union all

select pg\_type\_oid,schema\_of, typname, enum\_labels , 3 sourceof , null cr8\_sql , null dependson from xyz

commit;

create or replace view meta\_001.v\_dba\_RE\_type\_definition\_deps as

with get\_deps as (

select replace(split\_part( cast( meta\_001.f\_dba\_ddlx\_get\_dependants(pg\_type\_oid) as text ) , ',',3),')','') as f\_dba\_ddlx\_get\_dependants\_out,

pg\_type\_oid ,

cast(pg\_type\_oid as text) as objid\_text,

meta\_001.v\_dba\_RE\_type\_definition.schema\_of,

meta\_001.v\_dba\_RE\_type\_definition.typname

from meta\_001.v\_dba\_RE\_type\_definition where sourceof = 1 )

select distinct pg\_describe\_object( pg\_depend.classid, pg\_depend.objid, pg\_depend.objsubid) pg\_describe\_object\_of,

pg\_depend.classid,

pg\_depend.objid,

pg\_depend.objsubid,

get\_deps.pg\_type\_oid,

get\_deps.schema\_of,

get\_deps.typname

from pg\_depend ,

get\_deps

where cast( pg\_depend.objid as text) = get\_deps.f\_dba\_ddlx\_get\_dependants\_out order by schema\_of, typname, pg\_describe\_object\_of;

commit;

**SELECT** schema\_of, typname, x.cr8\_SQL **FROM** meta\_001.v\_dba\_re\_type\_definition x **WHERE** sourceof = 1;

**CREATE** **OR** **REPLACE** **VIEW** meta\_001.v\_dba\_re\_type\_definition

**AS** **WITH** abc **AS** (

**SELECT** t.**oid** **AS** pg\_type\_oid,

n.nspname **AS** schema\_of,

t.typname,

**string\_agg**(e.enumlabel::**text**, (((**chr**(39) || **chr**(10)) || ' '::**text**) || ','::**text**) || **chr**(39) **ORDER** **BY** e.enumsortorder) **AS** enum\_labels

**FROM** pg\_type t

**JOIN** pg\_namespace n **ON** n.**oid** = t.typnamespace

**JOIN** pg\_enum e **ON** t.**oid** = e.enumtypid

**GROUP** **BY** n.nspname, t.typname, n.**oid**, t.**oid**

), xyz **AS** (

**SELECT** abc.pg\_type\_oid,

abc.schema\_of,

abc.typname,

abc.enum\_labels,

1 **AS** sourceof,

(((((((((((((('drop type '::**text** || abc.schema\_of::**text**) || '.'::**text**) || abc.typname::**text**) || ';'::**text**) || **chr**(10)) || 'CREATE TYPE '::**text**) || abc.schema\_of::**text**) || '.'::**text**) || **format\_type**(abc.pg\_type\_oid, **NULL**::**integer**)) || ' AS ENUM ('::**text**) || **chr**(39)) || abc.enum\_labels) || **chr**(39)) || ');'::**text**) || **chr**(10) **AS** cr8\_sql

**FROM** abc

**UNION** **ALL**

**SELECT** pg\_type.**oid**,

**NULL**::**name** **AS** **name**,

pg\_type.typname,

pg\_enum.enumlabel,

2 **AS** sourceof,

**NULL**::**text** **AS** **text**

**FROM** pg\_type

**JOIN** pg\_enum **ON** pg\_enum.enumtypid = pg\_type.**oid**

**ORDER** **BY** 1, 5, 3, 4

)

**SELECT** xyz.pg\_type\_oid,

xyz.schema\_of,

xyz.typname,

xyz.enum\_labels,

xyz.sourceof,

xyz.cr8\_sql,

**NULL**::**text** **AS** dependson

**FROM** xyz

**UNION** **ALL**

**SELECT** **commit**.pg\_type\_oid,

**commit**.schema\_of,

**commit**.typname,

**commit**.enum\_labels,

3 **AS** sourceof,

**NULL**::**text** **AS** cr8\_sql,

**NULL**::**text** **AS** dependson

**FROM** xyz **commit**;

**select** **distinct** 'begin;'||**chr**(10)||' set constraints '||constraint\_name||' deferred;'||**chr**(10)||'commit;'||**chr**(10)||'end;'||**chr**(10) sqlof

**from** information\_schema.table\_constraints **where** constraint\_type = 'FOREIGN KEY';

SET session\_replication\_role = 'replica';

SET session\_replication\_role = 'origin';

CREATE OR REPLACE FUNCTION meta\_001.f\_dba\_loader\_disable\_triggers(a boolean, nsp character varying)

RETURNS void AS

$BODY$

declare

act character varying;

r record;

begin

if(a is true) then

act = 'disable';

else

act = 'enable';

end if;

for r in

select

n.nspname, c.relname

from pg\_namespace n

join pg\_class c on c.relnamespace = n.oid and c.relhastriggers = true

where n.nspname = nsp

loop

execute format('alter table %I.%I %s trigger all', r.nspname , r.relname, act);

end loop;

end;

$BODY$

LANGUAGE plpgsql VOLATILE

COST 100;

with abc as ( select distinct n.nspname from pg\_namespace n join pg\_class c on c.relnamespace = n.oid and c.relhastriggers = true )

select 'select meta\_001.f\_dba\_loader\_disable\_triggers(true,'||chr(39)||abc.nspname||chr(39)||');' from abc;

select meta\_001.f\_dba\_loader\_disable\_triggers(true,'domo\_001');

select meta\_001.f\_dba\_loader\_disable\_triggers(true,'io\_001');

select meta\_001.f\_dba\_loader\_disable\_triggers(true,'admin\_001');

select meta\_001.f\_dba\_loader\_disable\_triggers(true,'redcell\_001');

select meta\_001.f\_dba\_loader\_disable\_triggers(true,'log\_001');

select meta\_001.f\_dba\_loader\_disable\_triggers(true,'public');

select meta\_001.f\_dba\_loader\_disable\_triggers(true,'core\_001');