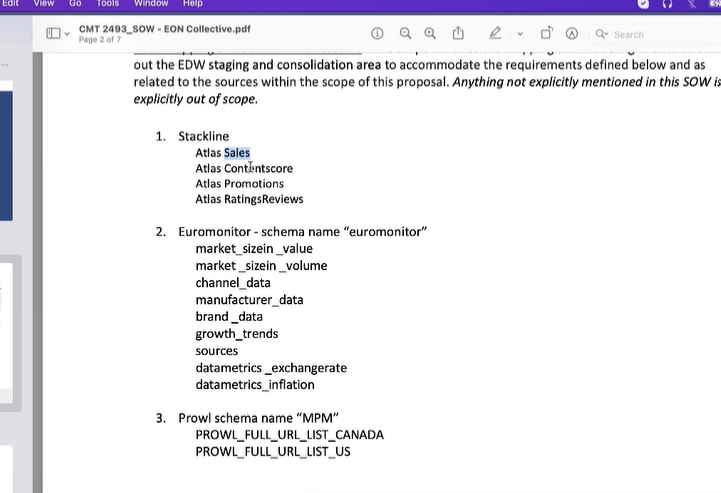
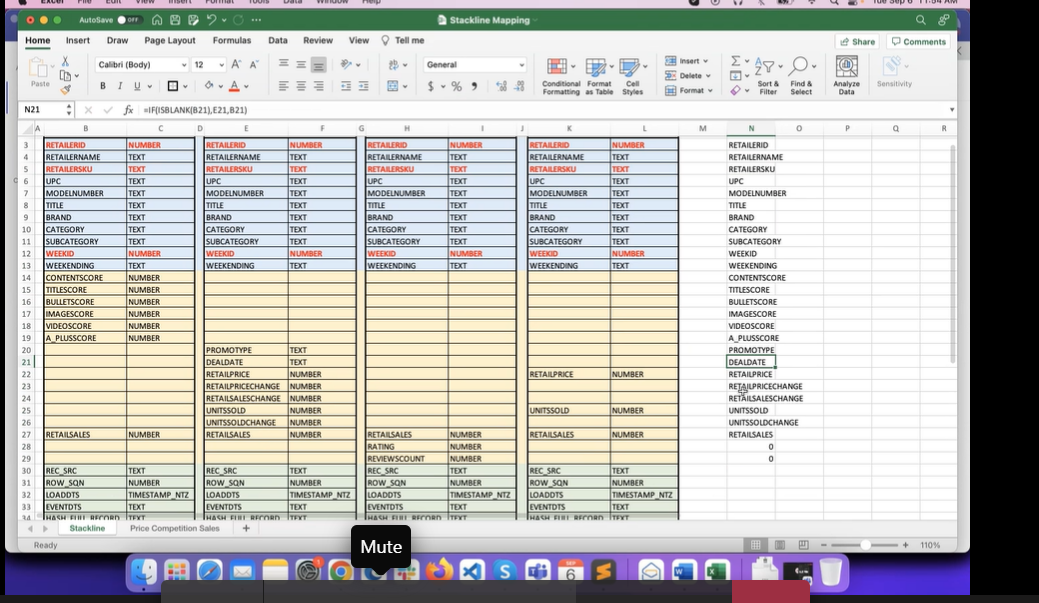
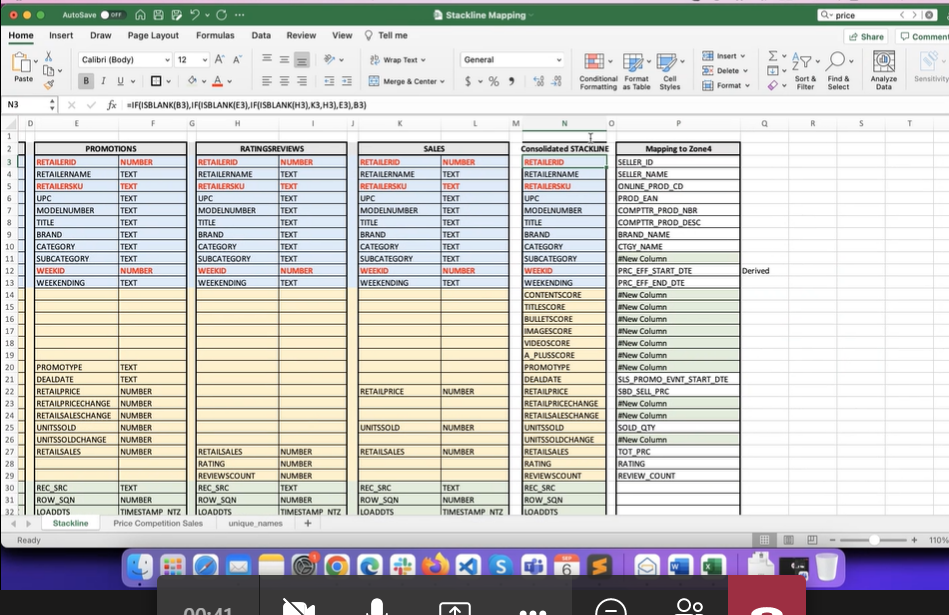
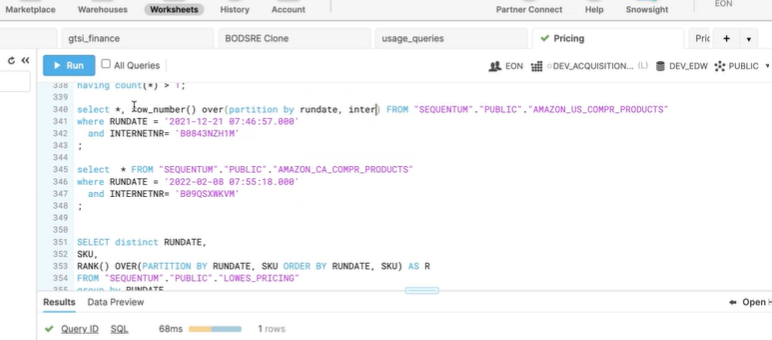
REQUIREMENTS OF PRICING PROJECT:

1. WE WILL HAVE DATA MODELS (STORING THE DATA) AND MAPPING (COLUMN FROM SOURCE TO TARGET TABLE) – PROFILING EXERCISE
2. WHAT ARE THE DEPENDENCY FOR THIS PROJECT
3. OFFICIAL COMMUNICATION IS VERY IMPORTANT FOR THIS PROJECT
4. PROWL AND STACKLINE
5. AS SOON AS WE GET THE MODELS AND TABLES (PHYSICAL IN DEVELOPMENT)
6. FOLLOW THE WORKSTREAMS
7. MAPPING DOC DONE BY DEVELOPMENT TEAM
8. WHAT IS UPSTREAM AND DOWN STREAM ?
9. Data governance , co-build person, project manager, deleiverable regarding project
10. Structures available in dev and test environment (deadline)
11. Daily stand up call for this project
12. People working on brds(source brds and consolidated brds)
13. Don’t go with mpm (projects)
14. Querying the tables is the first step of the project
15. Timeline for the next week (Tuesday)
16. url features and days (to be excluded), weekid to be stored as target









DDL OF THE TABLES IN CONSOLIDTED SCHEMA:

1.select get\_ddl('TABLE', 'US');

create or replace TABLE US (

ORIG\_SYSTEM VARCHAR(50),

FILLER VARCHAR(50),

SOLD\_TO\_CUSTOMER\_NBR VARCHAR(50),

STORE\_NUMBER VARCHAR(100),

CUSTOMER\_MATERIAL VARCHAR(50),

UPC\_NUMBER VARCHAR(50),

MATERIAL\_NUMBER VARCHAR(50),

RETAIL\_\_EXTD\_DOLLARS NUMBER(38,10),

POS\_QTY NUMBER(10,0),

BOH\_QTY NUMBER(10,0),

ON\_ORDER\_QTY NUMBER(10,0),

BEGINNING\_DATE VARCHAR(15),

END\_DATE VARCHAR(15),

LOADDTS TIMESTAMP\_NTZ(9),

EVENTDTS VARCHAR(16777216),

REC\_SRC VARCHAR(16777216),

ROW\_SQN NUMBER(38,0),

HASH\_FULL\_RECORD VARCHAR(128)

);

2. select get\_ddl('TABLE', 'EDW\_PRICING\_COMPETITOR\_SALES');

create or replace TABLE EDW\_PRICING\_COMPETITOR\_SALES (

SRC\_SYS\_KEY VARCHAR(255) NOT NULL COMMENT 'Identifies the origin of data source, in other word source system from where data is brought. The Key will be available in Source System table where one entry will exist for each participating system.\n',

PRC\_COMPTTR\_SLS\_KEY VARCHAR(255) NOT NULL COMMENT 'Key Identifier to make the Pricing Competitor Sales record Unique and This Column serves as Primary Key constraint on the Table and is generated by Concatinating (Competitor Product ID,SourceSystem Code, Store Name, RUN DATE) ',

SRC\_RCRD\_CREATE\_DTE TIMESTAMP\_NTZ(9) COMMENT 'Date and time when this record was created at source system table (driver table in case of multiple table/files are involved). NULL if not available at source. ',

SRC\_RCRD\_CREATE\_USERID VARCHAR(255) COMMENT 'The ID of the user who created the record at the at source system table (driver table in case of multiple table/files are involved). NULL if not available at source. ',

SRC\_RCRD\_UPD\_DTE TIMESTAMP\_NTZ(9) COMMENT 'Date and time when this record was updated at source system table (driver table in case of multiple table/files are involved). ',

SRC\_RCRD\_UPD\_USERID VARCHAR(255) COMMENT 'The ID of the user who updated the record at the at source system table (driver table in case of multiple table/files are involved). ',

RCRD\_HASH\_KEY VARCHAR(255) COMMENT 'The hash value (MD5/SHA256/etc.) of the qualified columns, that is considered to be different to enable versioning of record, will be stored in this column. ',

VER\_EFF\_DTE TIMESTAMP\_NTZ(9) COMMENT 'Date and time when this version of the record is created and become effective or valid ',

VER\_EXPR\_DTE TIMESTAMP\_NTZ(9) COMMENT 'Date and time when this version of the record has expired and no longer effective or valid at the target ',

CURR\_RCRD\_FLAG VARCHAR(1) COMMENT 'Default =\"Y\". This flag Indicates if the version of record is current or obsolete. “Y” indicates that the record is current and “N” indicates the record is obsolete or old. ',

ORP\_RCRD\_FLAG VARCHAR(1) COMMENT 'Default =\"N\". This flag indicates if the record is orphan (master/reference data information is missing). “Y” indicates that the record is orphan, and some program has created this record with missing/dummy key value in the Master/Reference table to maintain integrity. When the actual record arrives from its parent source, the column values are updated and this flag is set as “N”. ',

DEL\_FROM\_SRC\_FLAG VARCHAR(1) COMMENT 'Default =\"N\". When it is detected that the record has been deleted from the source table, then the corresponding record at target is marked as delete (soft delete) setting the flag as “Y”. (“U” à The status of record is unknown in terms of record deleted from source.) ',

ETL\_INS\_PID VARCHAR(255) COMMENT 'Contains information about the ETL/ELT job & package which has created the record ',

ETL\_INS\_DTE TIMESTAMP\_NTZ(9) COMMENT 'Date and time when this record is created or inserted at the target table. ',

ETL\_UPD\_PID VARCHAR(255) COMMENT 'Contains information about the ETL/ELT job & package which has updated the record last time (may be removed if ETL architect confirms) ',

ETL\_UPD\_DTE TIMESTAMP\_NTZ(9) COMMENT 'When Insert this will be NULL. Date and time when this record is updated last time at the target table. ',

ZONE3\_LOD\_DTE TIMESTAMP\_NTZ(9) COMMENT 'Zone3 Load DateTime (This will be used in ETL load from Zone 3 to Snowflake)',

PRC\_EFF\_START\_DTE TIMESTAMP\_NTZ(9) COMMENT 'Recorded date is the cut-off date established to determine the day price is updated or created. This field provides the date on which the data was recorded. ',

PRC\_EFF\_END\_DTE TIMESTAMP\_NTZ(9) COMMENT 'Business: >> This colum should update transformation as \"Effective start date - 1\"',

PROD\_KEY VARCHAR(255) DEFAULT 'N/A' COMMENT 'This field provides the SBD product model number',

COMPTTR\_PROD\_NBR VARCHAR(255) COMMENT 'This field provides the product model number of the competitor',

CRNCY\_KEY VARCHAR(255) DEFAULT 'N/A' COMMENT 'This field is an Currency to show which currency is being used. ',

BRAND\_LKEY VARCHAR(255) DEFAULT 'N/A' COMMENT 'Brand is type of product manufactured by a particular company under a particular name.This field provides the brand name of the product.',

COMPTTR\_BRAND\_LKEY VARCHAR(255) DEFAULT 'N/A' COMMENT 'Brand name of the competitor product is provided in this field',

CUST\_CHNL\_LKEY VARCHAR(255) DEFAULT 'N/A' COMMENT 'This field provides the customer channel method used.\nThe Customer Channels is the building block that describes how a company communicates with its Customer Segments to deliver a Value Proposition.',

SBU\_LKEY VARCHAR(255) DEFAULT 'N/A' COMMENT 'This field provides the Store Business Unit from which the product is sold',

SLS\_PROMO\_EVNT\_START\_DTE TIMESTAMP\_NTZ(9) COMMENT 'The date on which the event takes place',

SLS\_PROMO\_END\_MAP\_DTE TIMESTAMP\_NTZ(9) COMMENT 'This field provides the date and time on which the sales monitoring ends.',

SLS\_PROMO\_START\_DTE TIMESTAMP\_NTZ(9) COMMENT 'This field provides the date and time on which the sales monitoring starts',

SLS\_PROMO\_END\_DTE TIMESTAMP\_NTZ(9) COMMENT 'This field provides the date and time on which the sales monitoring ends',

COMPTTR\_PROD\_PRC NUMBER(28,10) COMMENT 'This field provides the price of the product\nPricing of the product is something different from its price. In simple words, pricing is the art of translating into quantitative terms the value of a product to customers at a point of time. ',

COMPTTR\_PROMO\_PRC NUMBER(28,10) COMMENT 'This field provides the price of the product after the promotion is applied.\nA price promotion is a reduced price that is intended to increase sales.',

VAT\_PRC NUMBER(28,10) COMMENT 'This field provides the Value Added Tax price\nValue Added Tax is a type of tax that is charged by the Central Government on the sale of services and goods to the consumers. It is commonly expressed as a percentage of the total cost',

PROD\_LIST\_PRC NUMBER(28,10) COMMENT 'Product Price is the price as of the time we pull the data.\nThe list price is the price that the manufacturer of an item suggests that a store should charge for it. SIMILAR WORDS: retail price',

VAT\_PROMO\_PRC NUMBER(28,10) COMMENT 'This field provides the Value Added Tax promotion price',

VAT\_LIST\_PRC NUMBER(28,10) COMMENT 'Gives the Value Added Tax list price',

SBD\_PROD\_DESC VARCHAR(2000) COMMENT 'This field give the description of the SBD product.',

COMPTTR\_STORE\_NAME VARCHAR(255) COMMENT 'The name of the store where the competitors product is sold',

ONLINE\_SELLER\_NAME VARCHAR(255) COMMENT 'The name of the store where the SBD product is sold.',

ONLINE\_PROD\_CD VARCHAR(255) COMMENT 'The price at which the SBD product is being sold',

SBD\_STORE\_NAME VARCHAR(255) COMMENT 'The name of the store where the SBD product is sold.',

SBD\_SELL\_PRC NUMBER(28,10) COMMENT 'The price at which the SBD product is being sold',

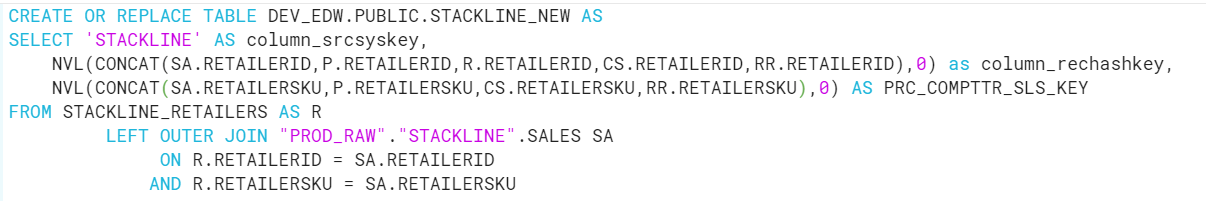
constraint EDW\_PRICING\_COMPETITOR\_SALES\_PK\_IDX primary key (SRC\_SYS\_KEY, PRC\_COMPTTR\_SLS\_KEY)

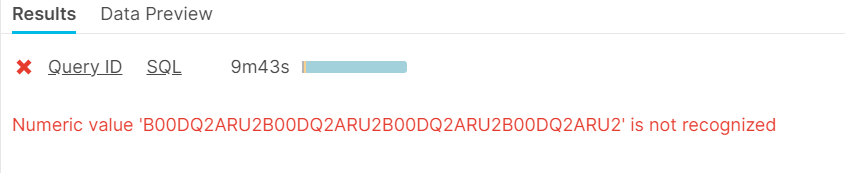
);

STEPS IN CREATING PROWL\_CANADA\_MODEL\_IN\_DBT

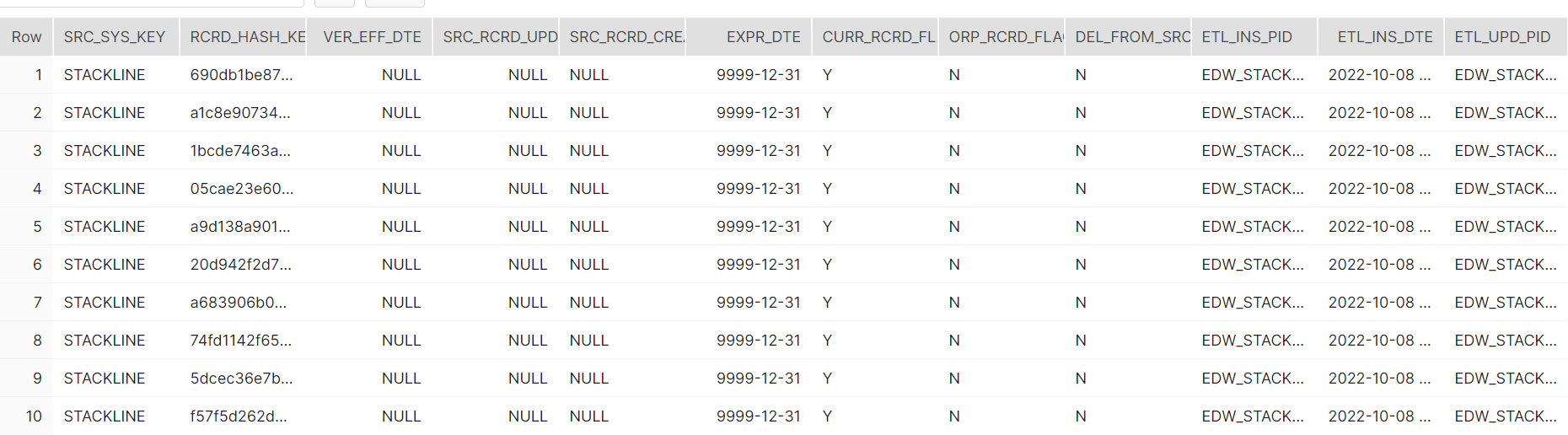
1. FIRST CREATE COMPETITOR SALES PRICE TABLE WITH NEW COLS AND WITH THAT SCRIPT WE NEED CREATE TABLE IN SNOWFLAKE
2. CLONE THE REPO FROM SBD GITHUB REPO (sbd\_edw-pricing) IN TO LOCAL CREATED FOLDER
3. (FOR CANADA) WE HAVE TO CREATE MODEL (.SQL) FILE WITH ALL THE COLS OF PROWL CANADA AND WITH CONCAT VALUE AS WELL
4. Git pull (to push any changes in the main or branch) of sbd repo
5. Git status
6. Git fetch
7. Git switch to another repo or branch
8. Git check out (file name) to take it out of the repo)
9. Finally dbt debug
10. Dbt run -m Canada view file name undr source
11. Columns name to be renamed in dbt\_project.yml (errors)
12. Source not found (git pull)
13. Dbt\_packages folder to be added in sbd-edw-pricing folder (under that sbd-edw-main)
14. Dbt\_project.yml (to be configured here as well)
15. Run dbt debug
16. Run dbt run -m Canada table name
17. Table created at snowflake

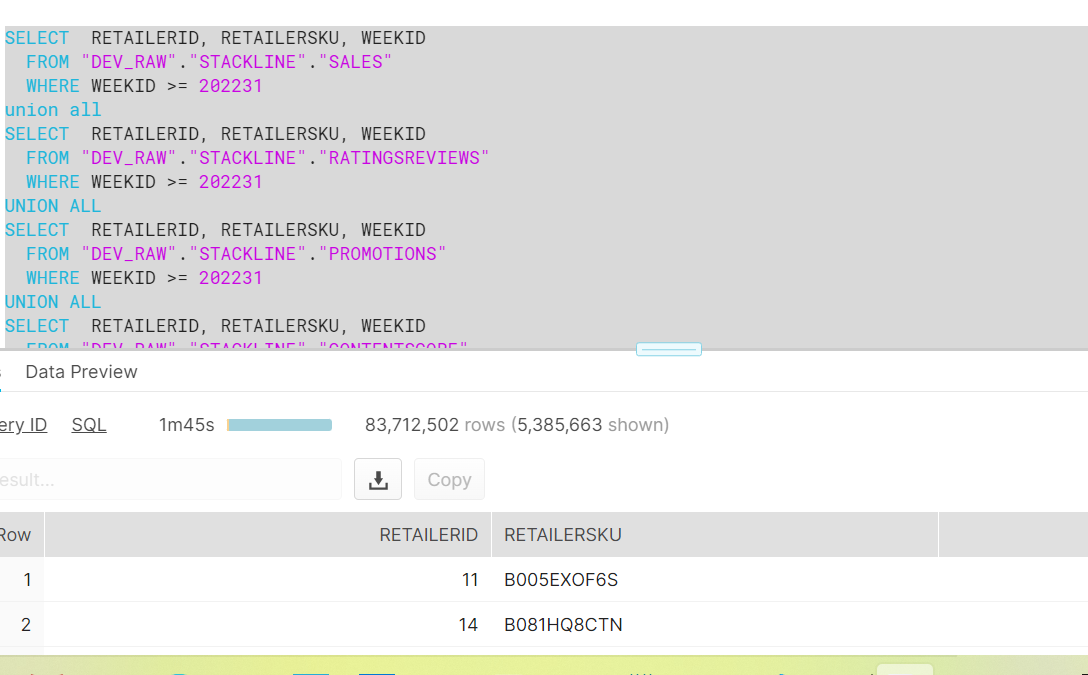
Concat of only retailersku





Output of stackline:





To test the source count (from DEV\_RAW)

-- ORIGINAL VIEW CODE

WITH STACKLINE\_RETAILERS AS

( SELECT RETAILERID,

RETAILERSKU,

WEEKID

FROM "DEV\_RAW"."STACKLINE"."SALES"

WHERE WEEKID >= 202231

UNION

SELECT RETAILERID,

RETAILERSKU,

WEEKID

FROM "DEV\_RAW"."STACKLINE"."PROMOTIONS"

WHERE WEEKID >= 202231

UNION

SELECT RETAILERID,

RETAILERSKU,

WEEKID

FROM "DEV\_RAW"."STACKLINE"."RATINGSREVIEWS"

WHERE WEEKID >= 202231

UNION

SELECT RETAILERID,

RETAILERSKU,

WEEKID

FROM "DEV\_RAW"."STACKLINE"."CONTENTSCORE"

WHERE WEEKID >= 202231

)

SELECT COUNT(\*) as SOURCE\_COUNT FROM STACKLINE\_RETAILERS AS R

-- SEQUENTUM

USE DATABASE SEQUENTUM

SELECT \* FROM "DEV\_EDW"."CONSOLIDATED"."EDW\_PRICING\_COMPETITOR\_SALES" WHERE SRC\_SYS\_KEY = 'SEQUENTUM' LIMIT 20

show primary keys in "DEV\_EDW"."CONSOLIDATED"."EDW\_PRICING\_COMPETITOR\_SALES"

select \* from "SEQUENTUM"."PUBLIC"."AMAZON\_CA\_COMPR\_PRODUCTS" limit 10

SHOW PRIMARY KEYS IN "SEQUENTUM"."PUBLIC"."AMAZON\_CA\_COMPR\_PRODUCTS"

SHOW PRIMARY KEYS IN "SEQUENTUM"."PUBLIC"."AMAZON\_US\_COMPR\_PRODUCTS"

SHOW PRIMARY KEYS IN "SEQUENTUM"."PUBLIC"."LOWES\_COMPR\_PRODUCTS"

-- target

select COUNT(\*) as "Target\_Count" from DEV\_EDW.CONSOLIDATED.EDW\_PRICING\_COMPETITOR\_SALES WHERE SRC\_SYS\_KEY = 'SEQUENTUM'

select count(\*) AS TARGET\_COUNT

from DEV\_EDW.CONSOLIDATED.EDW\_PRICING\_COMPETITOR\_SALES

where SRC\_SYS\_KEY = 'SEQUENTUM'

group by 1

limit 10;

desc table DEV\_EDW.CONSOLIDATED.EDW\_PRICING\_COMPETITOR\_SALES

-- SOURCE SOUNT

select SRC\_SYS\_KEY from DEV\_EDW.CONSOLIDATED.EDW\_PRICING\_COMPETITOR\_SALES WHERE COUNT(\*) > 1 AND SRC\_SYS\_KEY = 'SEQUENTUM' AND COUNT(\*) > 1

select 'CA', COUNT(\*) AS SOURCE\_COUNT

from "SEQUENTUM"."PUBLIC"."AMAZON\_CA\_COMPR\_PRODUCTS"

UNION ALL

select 'US', COUNT(\*) AS SOURCE\_COUNT

from "SEQUENTUM"."PUBLIC"."AMAZON\_US\_COMPR\_PRODUCTS"

UNION ALL

select 'CA', COUNT(\*) AS SOURCE\_COUNT

from "SEQUENTUM"."PUBLIC"."LOWES\_COMPR\_PRODUCTS"

UNION ALL

select 'US', COUNT(\*) AS SOURCE\_COUNT

from "SEQUENTUM"."PUBLIC"."HOMEDEPOT\_COMPR\_PRODUCTS"

UNION ALL

select 'US', COUNT(\*) AS SOURCE\_COUNT

from "SEQUENTUM"."PUBLIC"."HOMEDEPOT\_PRICING"

UNION ALL

select 'US', COUNT(\*) AS SOURCE\_COUNT

from "SEQUENTUM"."PUBLIC"."LOWES\_PRICING"

-- source count

WITH DEDUPE\_SRC\_AMZ\_US as (

SELECT \*,

ROW\_NUMBER() OVER(PARTITION BY INTERNETNR, MODELNR, ITEMMODELNUMBER, ITEMURL, RUNDATE

ORDER BY INTERNETNR, MODELNR, ITEMMODELNUMBER, ITEMURL, RUNDATE, ROW\_SQN DESC) AS ROW\_NO

FROM DEV\_RAW.SEQUENTUM.VW\_AMAZON\_US\_COMPR\_PRODUCTS

), DEDUPE\_SRC\_AMZ\_CA as (

SELECT \*,

ROW\_NUMBER() OVER(PARTITION BY INTERNETNR, MODELNR, ITEMMODELNUMBER, ITEMURL, RUNDATE

ORDER BY INTERNETNR, MODELNR, ITEMMODELNUMBER, ITEMURL, RUNDATE, ROW\_SQN DESC) AS ROW\_NO

FROM DEV\_RAW.SEQUENTUM.VW\_AMAZON\_CA\_COMPR\_PRODUCTS

), DEDUPE\_SRC\_LS\_PRC as (

SELECT \*,

ROW\_NUMBER() OVER(PARTITION BY SKU, MODELNUMBER, ITEMBRAND, ITEMURL, RUNDATE

ORDER BY SKU, MODELNUMBER, ITEMBRAND, ITEMURL, RUNDATE, ROW\_SQN DESC) AS ROW\_NO

FROM DEV\_RAW.SEQUENTUM.VW\_LOWES\_PRICING

), DEDUPE\_SRC\_HD\_PRC as (

SELECT \*,

ROW\_NUMBER() OVER(PARTITION BY SKU, MODELNUMBER, ITEMBRAND, ITEMURL, RUNDATE

ORDER BY SKU, MODELNUMBER, ITEMBRAND, ITEMURL, RUNDATE, ROW\_SQN DESC) AS ROW\_NO

FROM DEV\_RAW.SEQUENTUM.VW\_HOMEDEPOT\_PRICING

), DEDUPE\_SRC\_HD\_COMPR as (

SELECT \*,

ROW\_NUMBER() OVER(PARTITION BY INTERNETNR, MODELNR, ITEMBRAND, ITEMURL, RUNDATE

ORDER BY INTERNETNR, MODELNR, ITEMBRAND, ITEMURL, RUNDATE, ROW\_SQN DESC) AS ROW\_NO

FROM DEV\_RAW.SEQUENTUM.VW\_HOMEDEPOT\_COMPR\_PRODUCTS

), DEDUPE\_SRC\_LS\_COMPR as (

SELECT \*,

ROW\_NUMBER() OVER(PARTITION BY INTERNETNR, MODELNR, ITEMBRAND, ITEMURL, RUNDATE

ORDER BY INTERNETNR, MODELNR, ITEMBRAND, ITEMURL, RUNDATE, ROW\_SQN DESC) AS ROW\_NO

FROM DEV\_RAW.SEQUENTUM.VW\_LOWES\_COMPR\_PRODUCTS

)

SELECT 'AMZ\_US', COUNT(\*)

FROM DEDUPE\_SRC\_AMZ\_US --1

WHERE ROW\_NO = 1

UNION ALL

SELECT 'AMZ\_CA', COUNT(\*)

FROM DEDUPE\_SRC\_AMZ\_CA --2

WHERE ROW\_NO = 1

UNION ALL

SELECT 'HD\_PRC', COUNT(\*)

FROM DEDUPE\_SRC\_HD\_PRC --3

WHERE ROW\_NO = 1

UNION ALL

SELECT 'LS\_PRC', COUNT(\*)

FROM DEDUPE\_SRC\_LS\_PRC --4

WHERE ROW\_NO = 1

UNION ALL

SELECT 'HD\_COMPR', COUNT(\*)

FROM DEDUPE\_SRC\_HD\_COMPR --5

WHERE ROW\_NO = 1

UNION ALL

SELECT 'LS\_COMPR', COUNT(\*)

FROM DEDUPE\_SRC\_LS\_COMPR --6

WHERE ROW\_NO = 1;

{% macro ReleaseObjects() %}

USE SCHEMA CONSOLIDATED;

USE DATABASE DEV\_RAW

USE SCHEMA STACKLINE

SELECT \* FROM SALES LIMIT 10

SELECT \* FROM

select 'IFNULL(NULL,LOADDTS)' Condition ,IFNULL(MODELNUMBER,'') FROM SALES LIMIT 1000

create or replace temporary table EDW\_PRICING\_COMPETITOR\_SALES (

SRC\_SYS\_KEY VARCHAR(255) NOT NULL COMMENT 'Identifies the origin of data source, in other word source system from where data is brought. The Key will be available in Source System table where one entry will exist for each participating system.\n',

PRC\_COMPTTR\_SLS\_KEY VARCHAR(2000) NOT NULL COMMENT 'Key Identifier to make the Pricing Competitor Sales record Unique and This Column serves as Primary Key constraint on the Table and is generated by Concatinating (Competitor Product ID,SourceSystem Code, Store Name, RUN DATE) ',

SRC\_RCRD\_CREATE\_DTE TIMESTAMP\_NTZ(9) COMMENT 'Date and time when this record was created at source system table (driver table in case of multiple table/files are involved). NULL if not available at source. ',

SRC\_RCRD\_CREATE\_USERID VARCHAR(255) COMMENT 'The ID of the user who created the record at the at source system table (driver table in case of multiple table/files are involved). NULL if not available at source. ',

SRC\_RCRD\_UPD\_DTE TIMESTAMP\_NTZ(9) COMMENT 'Date and time when this record was updated at source system table (driver table in case of multiple table/files are involved). ',

SRC\_RCRD\_UPD\_USERID VARCHAR(255) COMMENT 'The ID of the user who updated the record at the at source system table (driver table in case of multiple table/files are involved). ',

RCRD\_HASH\_KEY VARCHAR(255) COMMENT 'The hash value (MD5/SHA256/etc.) of the qualified columns, that is considered to be different to enable versioning of record, will be stored in this column. ',

EFF\_DTE TIMESTAMP\_NTZ(9) COMMENT 'Date and time when this version of the record is created and become effective or valid ',

EXPR\_DTE TIMESTAMP\_NTZ(9) COMMENT 'Date and time when this version of the record has expired and no longer effective or valid at the target ',

CURR\_RCRD\_FLAG VARCHAR(1) COMMENT 'Default =\"Y\". This flag Indicates if the version of record is current or obsolete. “Y” indicates that the record is current and “N” indicates the record is obsolete or old. ',

ORP\_RCRD\_FLAG VARCHAR(1) COMMENT 'Default =\"N\". This flag indicates if the record is orphan (master/reference data information is missing). “Y” indicates that the record is orphan, and some program has created this record with missing/dummy key value in the Master/Reference table to maintain integrity. When the actual record arrives from its parent source, the column values are updated and this flag is set as “N”. ',

DEL\_FROM\_SRC\_FLAG VARCHAR(1) COMMENT 'Default =\"N\". When it is detected that the record has been deleted from the source table, then the corresponding record at target is marked as delete (soft delete) setting the flag as “Y”. (“U” à The status of record is unknown in terms of record deleted from source.) ',

ETL\_INS\_PID VARCHAR(255) COMMENT 'Contains information about the ETL/ELT job & package which has created the record ',

ETL\_INS\_DTE TIMESTAMP\_NTZ(9) COMMENT 'Date and time when this record is created or inserted at the target table. ',

ETL\_UPD\_PID VARCHAR(255) COMMENT 'Contains information about the ETL/ELT job & package which has updated the record last time (may be removed if ETL architect confirms) ',

ETL\_UPD\_DTE TIMESTAMP\_NTZ(9) COMMENT 'When Insert this will be NULL. Date and time when this record is updated last time at the target table. ',

ZONE3\_LOD\_DTE TIMESTAMP\_NTZ(9) COMMENT 'Zone3 Load DateTime (This will be used in ETL load from Zone 3 to Snowflake)',

PRC\_EFF\_START\_DTE TIMESTAMP\_NTZ(9) COMMENT 'Recorded date is the cut-off date established to determine the day price is updated or created. This field provides the date on which the data was recorded. ',

PRC\_EFF\_END\_DTE TIMESTAMP\_NTZ(9) COMMENT 'Business: >> This colum should update transformation as \"Effective start date - 1\"',

COMPTTR\_PROD\_NBR VARCHAR(255) COMMENT 'This field provides the product model number of the competitor',

CRNCY\_KEY VARCHAR(255) DEFAULT 'N/A' COMMENT 'This field is an Currency to show which currency is being used. ',

BRAND\_LKEY VARCHAR(255) DEFAULT 'N/A' COMMENT 'Brand is type of product manufactured by a particular company under a particular name.This field provides the brand name of the product.',

COMPTTR\_BRAND\_LKEY VARCHAR(1000) DEFAULT 'N/A' COMMENT 'Brand name of the competitor product is provided in this field',

CUST\_CHNL\_LKEY VARCHAR(255) DEFAULT 'N/A' COMMENT 'This field provides the customer channel method used.\nThe Customer Channels is the building block that describes how a company communicates with its Customer Segments to deliver a Value Proposition.',

SBU\_LKEY VARCHAR(255) DEFAULT 'N/A' COMMENT 'This field provides the Store Business Unit from which the product is sold',

SLS\_PROMO\_TYPE VARCHAR(255) COMMENT 'The method used for the promotion',

SLS\_PROMO\_EVNT\_START\_DTE TIMESTAMP\_NTZ(9) COMMENT 'The date on which the event takes place',

SLS\_PROMO\_END\_MAP\_DTE TIMESTAMP\_NTZ(9) COMMENT 'This field provides the date and time on which the sales monitoring ends.',

SLS\_PROMO\_START\_DTE TIMESTAMP\_NTZ(9) COMMENT 'This field provides the date and time on which the sales monitoring starts',

SLS\_PROMO\_END\_DTE TIMESTAMP\_NTZ(9) COMMENT 'This field provides the date and time on which the sales monitoring ends',

COMPTTR\_PROD\_PRC NUMBER(28,10) DEFAULT 0 COMMENT 'This field provides the price of the product\nPricing of the product is something different from its price. In simple words, pricing is the art of translating into quantitative terms the value of a product to customers at a point of time. ',

COMPTTR\_PROMO\_PRC NUMBER(28,10) DEFAULT 0 COMMENT 'This field provides the price of the product after the promotion is applied.\nA price promotion is a reduced price that is intended to increase sales.',

VAT\_PRC NUMBER(28,10) DEFAULT 0 COMMENT 'This field provides the Value Added Tax price\nValue Added Tax is a type of tax that is charged by the Central Government on the sale of services and goods to the consumers. It is commonly expressed as a percentage of the total cost',

PROD\_LIST\_PRC NUMBER(28,10) DEFAULT 0 COMMENT 'Product Price is the price as of the time we pull the data.\nThe list price is the price that the manufacturer of an item suggests that a store should charge for it. SIMILAR WORDS: retail price',

VAT\_PROMO\_PRC NUMBER(28,10) DEFAULT 0 COMMENT 'This field provides the Value Added Tax promotion price',

VAT\_LIST\_PRC NUMBER(28,10) DEFAULT 0 COMMENT 'Gives the Value Added Tax list price',

SBD\_PROD\_DESC VARCHAR(2000) COMMENT 'This field give the description of the SBD product.',

RTL\_SLS\_CHNG NUMBER(28,10) COMMENT 'Gives the value of change in retail sales effected by the promotion',

RTL\_PRC\_CHNG NUMBER(28,10) COMMENT 'Gives the value of change in retail price effected by the promotion',

STORE\_NAME VARCHAR(255) COMMENT 'The name of the store where the competitors product is sold',

ONLINE\_SELLER\_NAME VARCHAR(255) COMMENT 'The name of the store where the SBD product is sold.',

ONLINE\_PROD\_CD VARCHAR(255) COMMENT 'The price at which the SBD product is being sold',

SBD\_SELL\_PRC NUMBER(28,10) DEFAULT 0 COMMENT 'The price at which the SBD product is being sold',

COMPTTR\_PROD\_DESC VARCHAR(2000),

TOT\_PRC NUMBER(28,10) DEFAULT 0,

TOT\_USD\_PRC NUMBER(28,10) DEFAULT 0,

SELLER\_ID VARCHAR(255),

SELLER\_NAME VARCHAR(255),

SELLER\_CITY VARCHAR(255),

SELLER\_STATE VARCHAR(255),

STORE\_ID VARCHAR(255),

SELLER\_TYP\_ID VARCHAR(255),

SELLER\_TYP\_NAME VARCHAR(255),

CTGY\_ID VARCHAR(255),

CTGY\_NAME VARCHAR(255),

SUB\_CTGY VARCHAR(255) COMMENT 'Sub Category Name of the Product',

ORIG\_QTY NUMBER(28,10),

AVAIL\_QTY NUMBER(28,10),

SOLD\_QTY NUMBER(28,10),

UNITS\_SOLD\_CHNG NUMBER(28,10) COMMENT 'Gives the value of change in the total number of units sold effected by the Sales promotion',

MAP\_PRICE NUMBER(28,10) COMMENT 'This field provides the minimum advertisement Price of the product by the Seller',

MAPP\_POLICY VARCHAR(255) COMMENT 'This field defines the policy governing the MAP of the product by the Seller',

CONV\_RATE NUMBER(28,10),

COMPTTR\_PROD\_BASE\_PRC NUMBER(28,10) DEFAULT 0 COMMENT 'This field provides the price of the product',

PROD\_BASE\_PRC NUMBER(28,10) DEFAULT 0 COMMENT 'Product Base Price ',

COMPTTR\_PROD\_USD\_PRC NUMBER(28,10) COMMENT 'This field provides the price of the product\r\nPricing of the product is something different from its price. In simple words, pricing is the art of translating into quantitative terms the value of a product to customers at a point of time.',

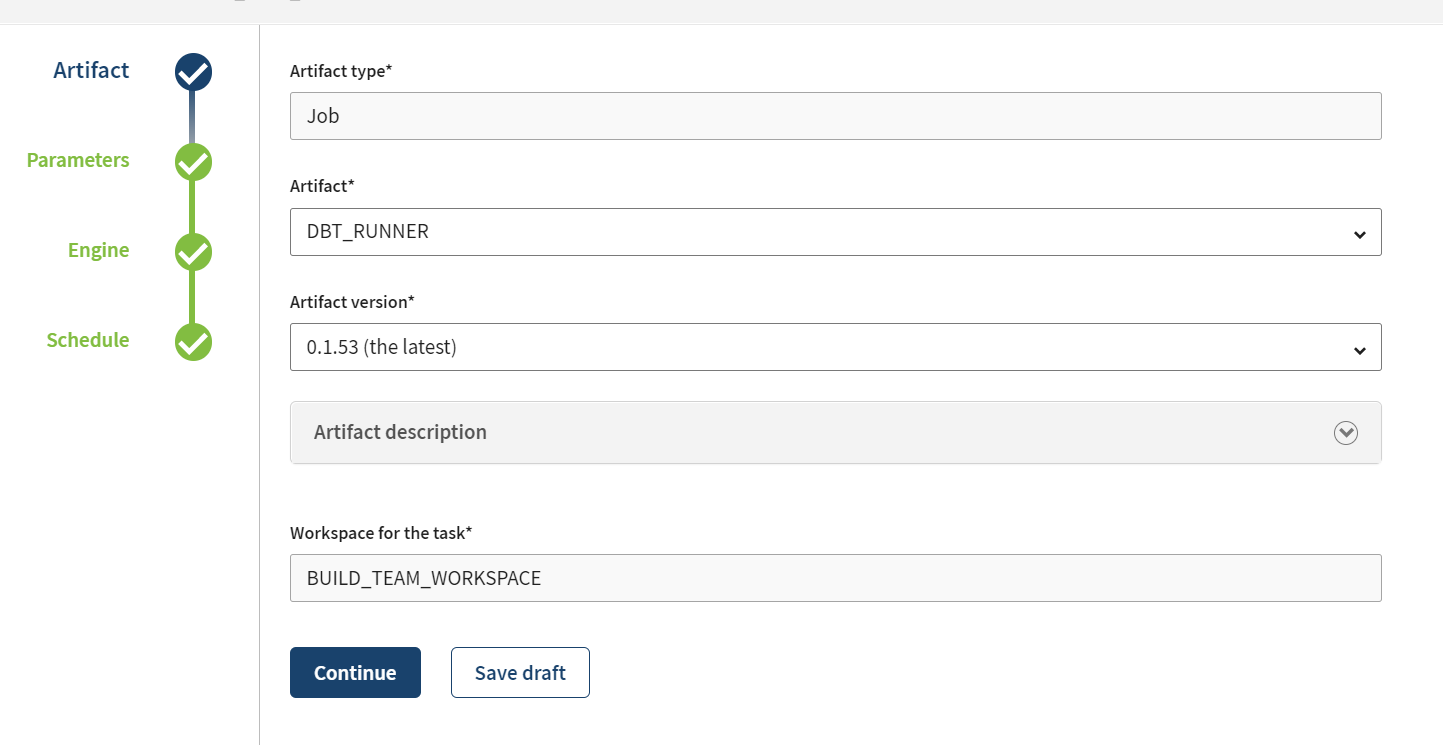
PROD\_LIST\_USD\_PRC NUMBER(28,10) COMMENT 'Product Price is the price as of the time we pull the data.',

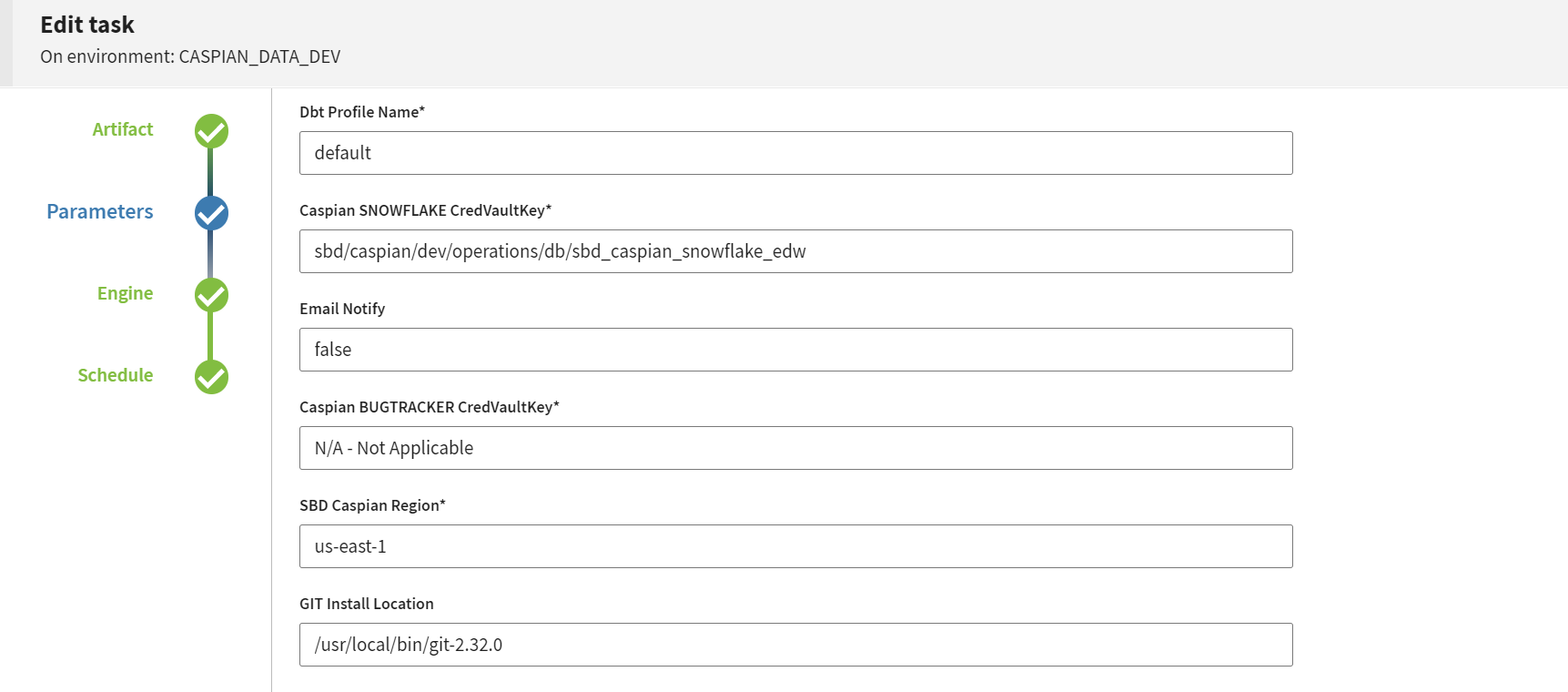
BRAND\_NAME VARCHAR(1000),

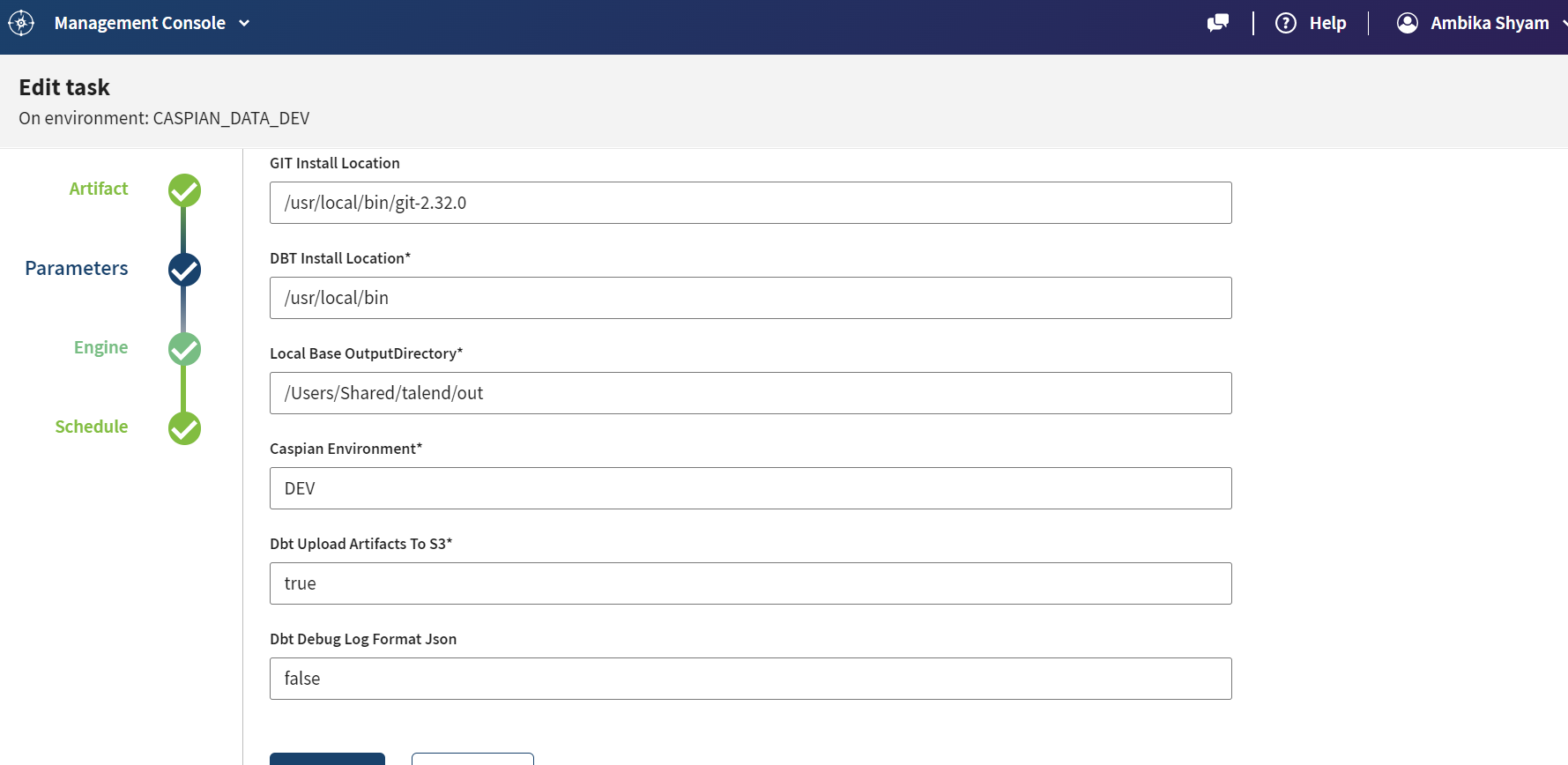
PROD\_KEY VARCHAR(1000) DEFAULT 'N/A',

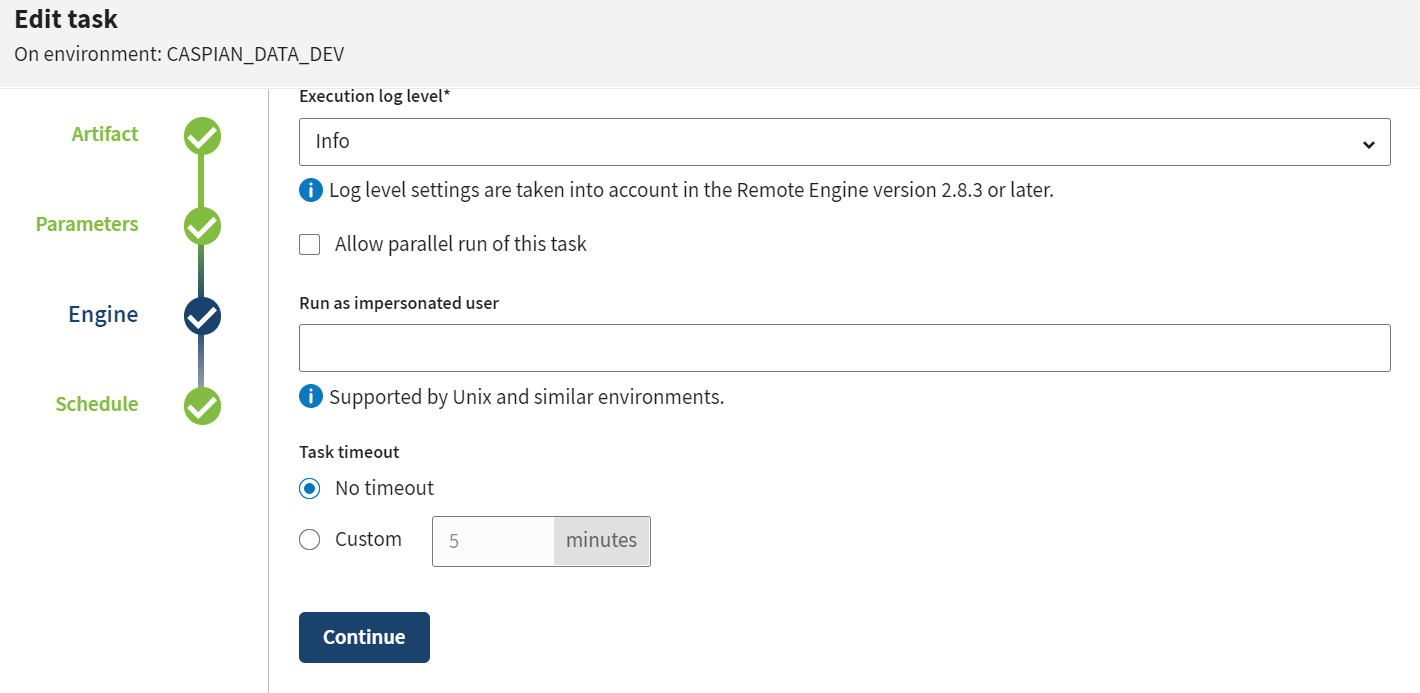
constraint EDW\_PRICING\_COMPETITOR\_SALES\_PK\_IDX primary key (SRC\_SYS\_KEY, PRC\_COMPTTR\_SLS\_KEY)

);









Demerits☹risk in stackline

1. Sales table don’t have data but ratings table have data for that
2. For which week sales is not done

Things done by bala (after model creation)

1. He developed the code for stackline (in loaddts – least())
2. he changed all the cols as (varchar) like (:: varchar)
3. he took out weekending as weekid (in stackline) as it was creating issue
4. he included currency (with clause) to compare with retailer id()
5. he created # for other tables (promo\_type) and promotype already in promotions)
6. after this he changed the code (to compare the source and target)
7. he helped me to get git status for capsian folder and git commit and git switch to feature/prowl-stackline-seq and git pull to point to feature branch
8. now he has updated master branch as empty and kept the branches ready for migration.

Things done by Ambika (after model creation)

1. before (setting the least(loaddts) and individual col ::varchar)
2. I merged the view (given by bala and put in the stackline) and build the model
3. I was trying to keep all the cols in the view and try building the model and got the col values in SF
4. Wrote the Sf date code for weekid
5. Went through the unit test results and run book of pricing team and

And produced the same in pricing competitor sales

Done with cut over plan

1. Handled the queries given by the sheetal (out of 3, 2 was correct)
2. A. Target count greater than source (because they failed to check union query)

b. src\_create\_date (was varchar) and change it to timestamp (by using to\_timestanmp (suggested) and rerun the stackline and brought the unit test results

c. brought the code to the code (checking inner join of the key cols) and retailsprice <> p.retailprice (this alone missed)

1. Tried the source and target count query (using rec\_hash\_key) from test\_edw and concat(keys cols) in prod\_raw

How to create talend task :

Step 1:

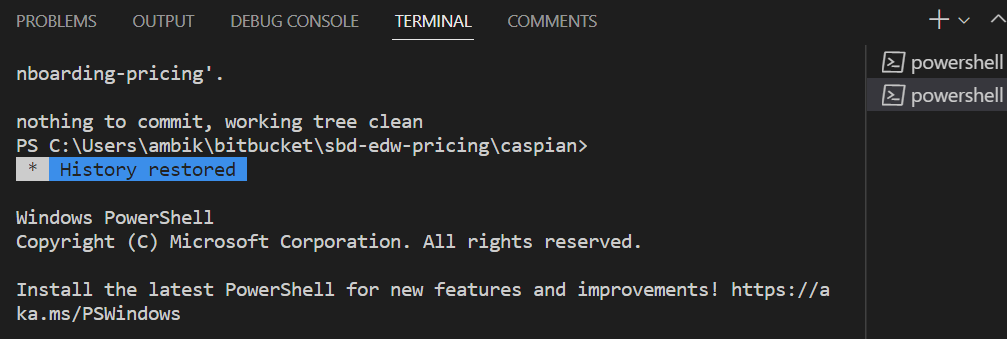
In production (branch) we are going to run the TMC task in master or

If we can run the TMC task in feature branch and then into master

(repo branch)

Repo change:

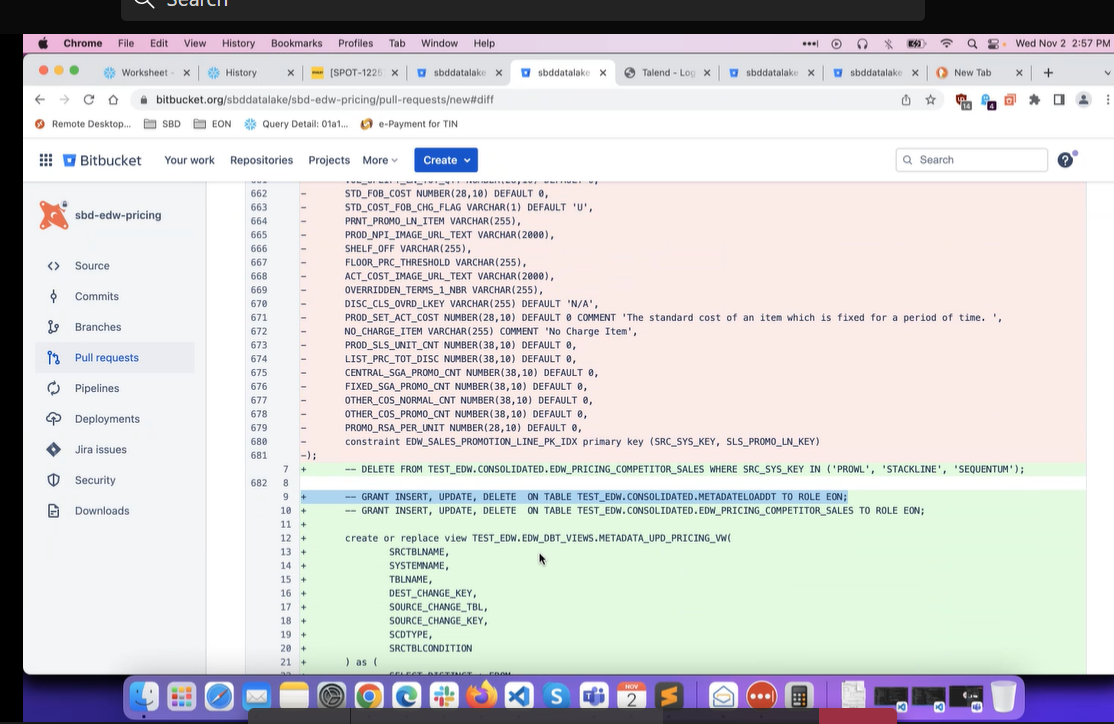
1. (through source tree keeping master branch clean and branches ready)

2. 

This screen should be maintained in VS ( for edw-pricing-main) and capsian folder pointing to

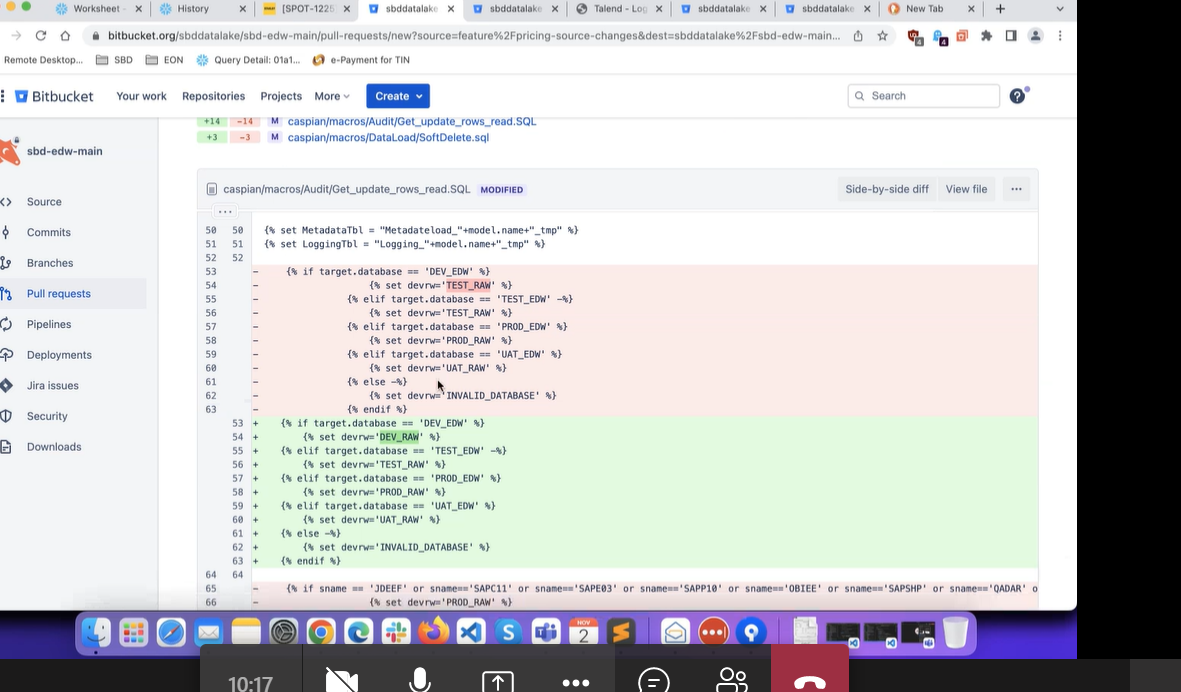
GIT current (feature branch) and up to date

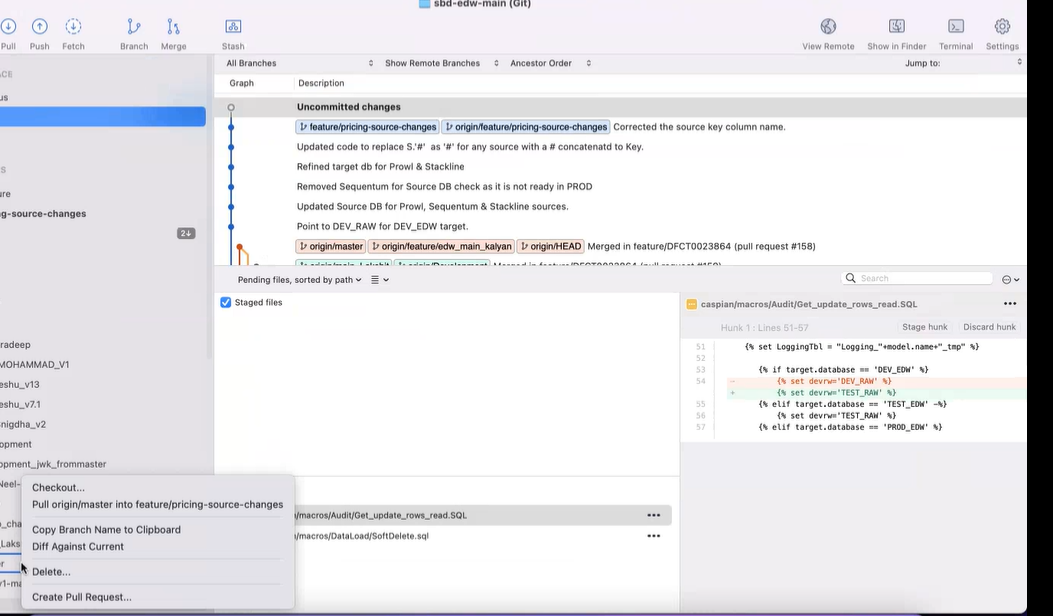
1. When we are doing migration to prod\_edw (whatever the new cols we have introduced in test\_edw that should not be there
2. desc table "PROD\_EDW"."CONSOLIDATED"."EDW\_PRICING\_COMPETITOR\_SALES"
3. DESC TABLE TEST\_EDW.CONSOLIDATED.EDW\_PRICING\_COMPETITOR\_SALES
4. We need to check the above two query to confirm it

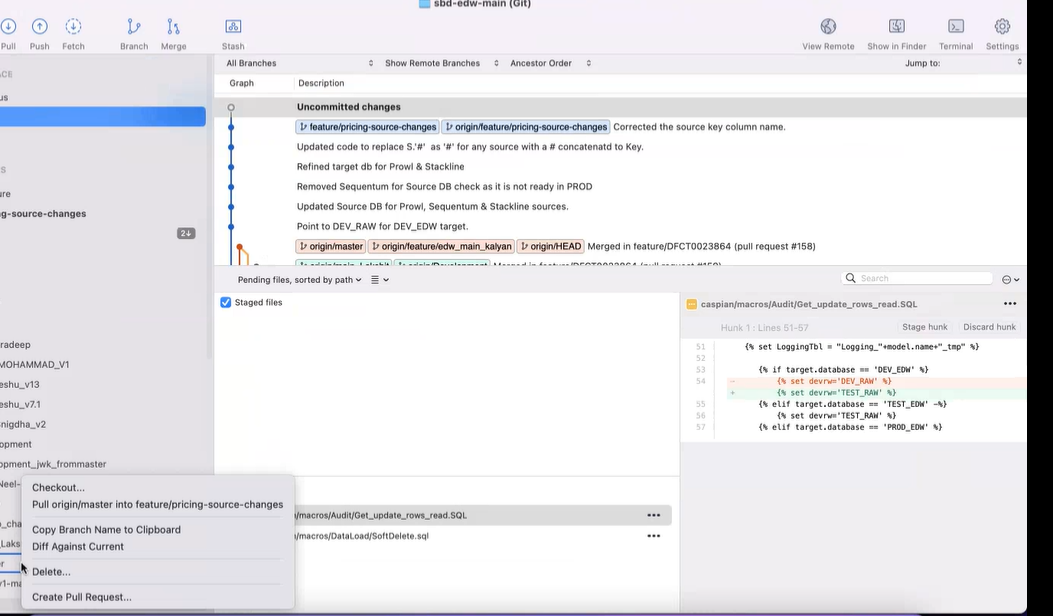


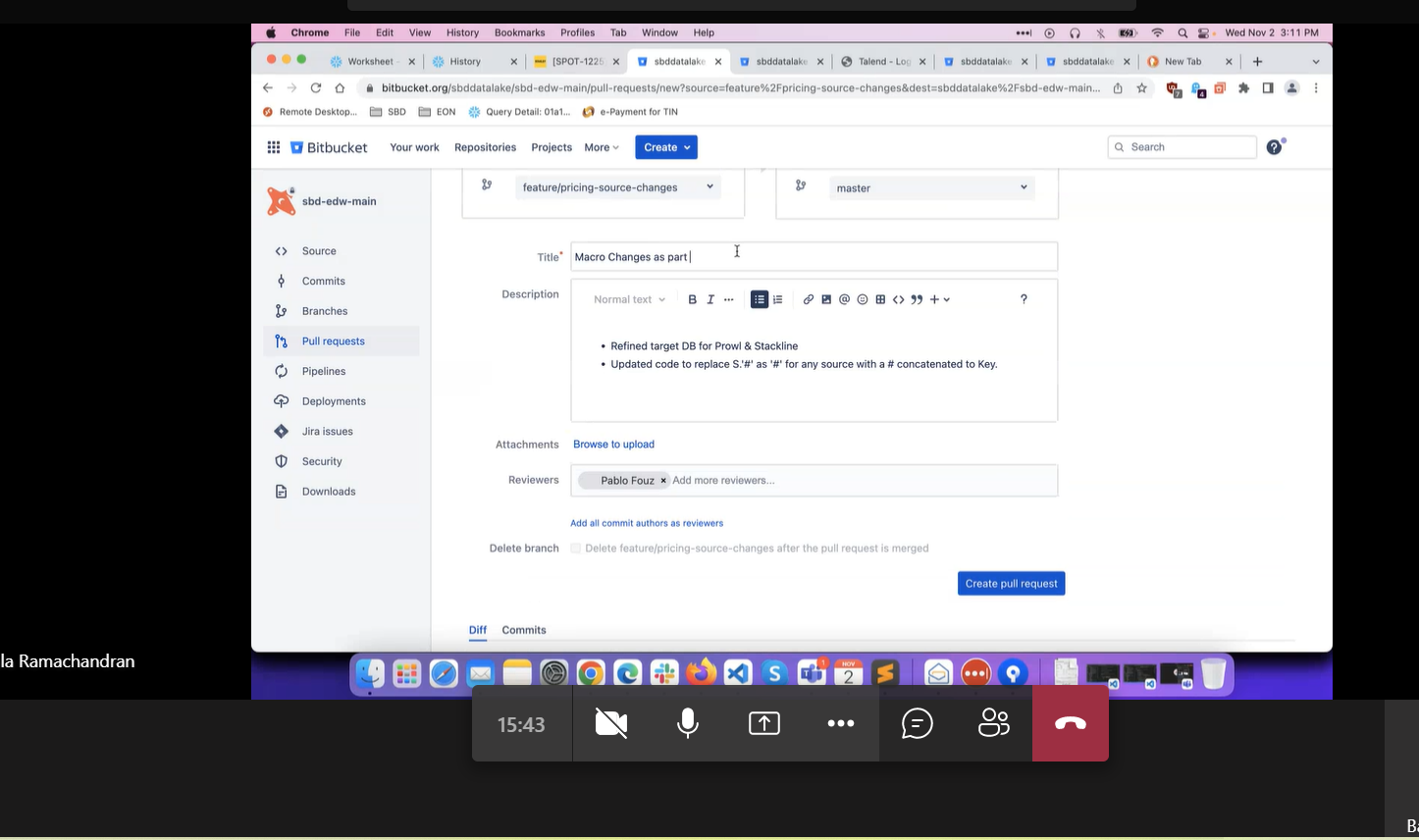
Go to source tree

Fetch from master









How to clean the master branch in git

git clone <https://your_repo_url/>

git status #check if you are in master branch.

git branch clean\_up

git checkout clean\_up

git rm . #remove all files from current folder.

git rm -r \* #remove all folder and their data.

git checkout master

git merge clean\_up

git branch -d clean\_up

git push



PRODUCTION QUERIES:

--production queries

use database prod\_edw

use schema consolidated

use database prod\_raw

desc table "PROD\_EDW"."CONSOLIDATED"."EDW\_PRICING\_COMPETITOR\_SALES"

DESC TABLE TEST\_EDW.CONSOLIDATED.EDW\_PRICING\_COMPETITOR\_SALES

--target count

select COUNT(\*) from "PROD\_EDW"."CONSOLIDATED"."EDW\_PRICING\_COMPETITOR\_SALES" where SRC\_SYS\_KEY = 'PROWL'

select COUNT(\*) from "PROD\_EDW"."CONSOLIDATED"."EDW\_PRICING\_COMPETITOR\_SALES" where SRC\_SYS\_KEY = 'SEQUENTUM'

select COUNT(\*) from "PROD\_EDW"."CONSOLIDATED"."EDW\_PRICING\_COMPETITOR\_SALES" where SRC\_SYS\_KEY = 'STACKLINE'

--prowl (source count)

select COUNT(\*) from "PROD\_RAW"."PROWL"."PROWL\_FULL\_URL\_LIST\_CANADA" where reports = 'Y'

union

select COUNT(\*) from "PROD\_RAW"."PROWL"."PROWL\_FULL\_URL\_LIST\_US" where reports = 'Y'

--unit test query

select MODEL\_NAME, LOADSTATUS, ROWREAD, ROWINSERTED, ROWUPDATED, ERROR\_DESCRIPTION, job\_time\_start, job\_time\_end

from "PROD\_EDW"."CONSOLIDATED"."LOGGING" where LID in

(Select SID from "PROD\_EDW"."CONSOLIDATED"."METADATELOADDT" where SYSTEM\_NAME LIKE '%PROWL%')

and MODEL\_NAME like '%PRICING%' and ERROR\_DESCRIPTION IS NULL and LOADSTATUS='END'

order by job\_time\_start desc

--sequentum (source sount)

WITH DEDUPE\_SRC\_AMZ\_US as (

SELECT \*,

ROW\_NUMBER() OVER(PARTITION BY INTERNETNR, MODELNR, ITEMMODELNUMBER, ITEMURL, RUNDATE

ORDER BY INTERNETNR, MODELNR, ITEMMODELNUMBER, ITEMURL, RUNDATE, ROW\_SQN DESC) AS ROW\_NO

FROM PROD\_RAW.SEQUENTUM.VW\_AMAZON\_US\_COMPR\_PRODUCTS

), DEDUPE\_SRC\_AMZ\_CA as (

SELECT \*,

ROW\_NUMBER() OVER(PARTITION BY INTERNETNR, MODELNR, ITEMMODELNUMBER, ITEMURL, RUNDATE

ORDER BY INTERNETNR, MODELNR, ITEMMODELNUMBER, ITEMURL, RUNDATE, ROW\_SQN DESC) AS ROW\_NO

FROM PROD\_RAW.SEQUENTUM.VW\_AMAZON\_CA\_COMPR\_PRODUCTS

), DEDUPE\_SRC\_LS\_PRC as (

SELECT \*,

ROW\_NUMBER() OVER(PARTITION BY SKU, MODELNUMBER, ITEMBRAND, ITEMURL, RUNDATE

ORDER BY SKU, MODELNUMBER, ITEMBRAND, ITEMURL, RUNDATE, ROW\_SQN DESC) AS ROW\_NO

FROM PROD\_RAW.SEQUENTUM.VW\_LOWES\_PRICING

), DEDUPE\_SRC\_HD\_PRC as (

SELECT \*,

ROW\_NUMBER() OVER(PARTITION BY SKU, MODELNUMBER, ITEMBRAND, ITEMURL, RUNDATE

ORDER BY SKU, MODELNUMBER, ITEMBRAND, ITEMURL, RUNDATE, ROW\_SQN DESC) AS ROW\_NO

FROM PROD\_RAW.SEQUENTUM.VW\_HOMEDEPOT\_PRICING

), DEDUPE\_SRC\_HD\_COMPR as (

SELECT \*,

ROW\_NUMBER() OVER(PARTITION BY INTERNETNR, MODELNR, ITEMBRAND, ITEMURL, RUNDATE

ORDER BY INTERNETNR, MODELNR, ITEMBRAND, ITEMURL, RUNDATE, ROW\_SQN DESC) AS ROW\_NO

FROM PROD\_RAW.SEQUENTUM.VW\_HOMEDEPOT\_COMPR\_PRODUCTS

), DEDUPE\_SRC\_LS\_COMPR as (

SELECT \*,

ROW\_NUMBER() OVER(PARTITION BY INTERNETNR, MODELNR, ITEMBRAND, ITEMURL, RUNDATE

ORDER BY INTERNETNR, MODELNR, ITEMBRAND, ITEMURL, RUNDATE, ROW\_SQN DESC) AS ROW\_NO

FROM PROD\_RAW.SEQUENTUM.VW\_LOWES\_COMPR\_PRODUCTS

)

SELECT 'AMZ\_US', COUNT(\*) AS SOURCECOUNT

FROM DEDUPE\_SRC\_AMZ\_US --1

WHERE ROW\_NO = 1

UNION ALL

SELECT 'AMZ\_CA', COUNT(\*)

FROM DEDUPE\_SRC\_AMZ\_CA --2

WHERE ROW\_NO = 1

UNION ALL

SELECT 'HD\_PRC', COUNT(\*)

FROM DEDUPE\_SRC\_HD\_PRC --3

WHERE ROW\_NO = 1

UNION ALL

SELECT 'LS\_PRC', COUNT(\*)

FROM DEDUPE\_SRC\_LS\_PRC --4

WHERE ROW\_NO = 1

UNION ALL

SELECT 'HD\_COMPR', COUNT(\*)

FROM DEDUPE\_SRC\_HD\_COMPR --5

WHERE ROW\_NO = 1

UNION ALL

SELECT 'LS\_COMPR', COUNT(\*)

FROM DEDUPE\_SRC\_LS\_COMPR --6

WHERE ROW\_NO = 1;

--unit test query

select MODEL\_NAME, LOADSTATUS, ROWREAD, ROWINSERTED, ROWUPDATED, ERROR\_DESCRIPTION, job\_time\_start, job\_time\_end

from "PROD\_EDW"."CONSOLIDATED"."LOGGING" where LID in

(Select SID from "PROD\_EDW"."CONSOLIDATED"."METADATELOADDT" where SYSTEM\_NAME LIKE '%SEQUENTUM%')

and MODEL\_NAME like '%PRICING%' and ERROR\_DESCRIPTION IS NULL and LOADSTATUS='END'

order by job\_time\_start desc

-- stackline (source count)

SELECT COUNT(\*) FROM "PROD\_RAW"."STACKLINE"."CONTENTSCORE"

UNION

SELECT COUNT(\*) FROM "PROD\_RAW"."STACKLINE"."SALES"

UNION

SELECT COUNT(\*) FROM "PROD\_RAW"."STACKLINE"."PROMOTIONS"

UNION

SELECT COUNT(\*) FROM "PROD\_RAW"."STACKLINE"."RATINGSREVIEWS"

with output as (

select RCRD\_HASH\_KEY from PROD\_EDW.CONSOLIDATED.EDW\_PRICING\_COMPETITOR\_SALES

WHERE SRC\_SYS\_KEY = 'STACKLINE' AND RCRD\_HASH\_KEY IN

(SELECT(md5(CONCAT(COALESCE(S.RETAILERSKU::VARCHAR,''),

'~',COALESCE(S.RETAILERID::VARCHAR,''),

'~',COALESCE(S.WEEKID::VARCHAR,''),

'~',COALESCE('#'::VARCHAR,''))))

FROM "PROD\_RAW"."STACKLINE"."SALES" AS S)

UNION

select RCRD\_HASH\_KEY from PROD\_EDW.CONSOLIDATED.EDW\_PRICING\_COMPETITOR\_SALES

WHERE SRC\_SYS\_KEY = 'STACKLINE' AND RCRD\_HASH\_KEY IN

(SELECT(md5(CONCAT(COALESCE(P.RETAILERSKU::VARCHAR,''),

'~',COALESCE(P.RETAILERID::VARCHAR,''),

'~',COALESCE(P.WEEKID::VARCHAR,''),

'~',COALESCE(P.PROMOTYPE::VARCHAR,''))))

FROM "PROD\_RAW"."STACKLINE"."PROMOTIONS" AS P)

UNION

select RCRD\_HASH\_KEY from PROD\_EDW.CONSOLIDATED.EDW\_PRICING\_COMPETITOR\_SALES

WHERE SRC\_SYS\_KEY = 'STACKLINE' AND RCRD\_HASH\_KEY IN

(SELECT(md5(CONCAT(COALESCE(R.RETAILERSKU::VARCHAR,''),

'~',COALESCE(R.RETAILERID::VARCHAR,''),

'~',COALESCE(R.WEEKID::VARCHAR,''),

'~',COALESCE('#'::VARCHAR,''))))

FROM "PROD\_RAW"."STACKLINE"."RATINGSREVIEWS" AS R)

UNION

select RCRD\_HASH\_KEY from PROD\_EDW.CONSOLIDATED.EDW\_PRICING\_COMPETITOR\_SALES

WHERE SRC\_SYS\_KEY = 'STACKLINE' AND RCRD\_HASH\_KEY IN

(SELECT(md5(CONCAT(COALESCE(CS.RETAILERSKU::VARCHAR,''),

'~',COALESCE(CS.RETAILERID::VARCHAR,''),

'~',COALESCE(CS.WEEKID::VARCHAR,''),

'~',COALESCE('#'::VARCHAR,''))))

FROM "PROD\_RAW"."STACKLINE"."CONTENTSCORE" AS CS)

)

SELECT COUNT(\*) as SOURCE\_COUNT, sum(0) as TARGET\_COUNT FROM output AS R

UNION ALL

SELECT SUM(0) as SOURCE\_COUNT, COUNT(\*) as TARGET\_COUNT FROM PROD\_EDW.CONSOLIDATED.EDW\_PRICING\_COMPETITOR\_SALES

where SRC\_SYS\_KEY = 'STACKLINE'

and CURR\_RCRD\_FLAG = 'Y'

--unit test query

select MODEL\_NAME, LOADSTATUS, ROWREAD, ROWINSERTED, ROWUPDATED, ERROR\_DESCRIPTION, job\_time\_start, job\_time\_end

from "PROD\_EDW"."CONSOLIDATED"."LOGGING" where LID in

(Select SID from "PROD\_EDW"."CONSOLIDATED"."METADATELOADDT" where SYSTEM\_NAME LIKE '%STACKLINE%')

and MODEL\_NAME like '%PRICING%' and ERROR\_DESCRIPTION IS NULL and LOADSTATUS='END'

order by job\_time\_start desc

BALA’S QUERY:

WITH STACKLINE\_RETAILERS AS

( SELECT RETAILERID,

RETAILERSKU,

WEEKID,

'#' as SLS\_PROMO\_TYPE

FROM "PROD\_RAW"."STACKLINE"."SALES"

-- WHERE WEEKID >= 202231

UNION

SELECT RETAILERID,

RETAILERSKU,

WEEKID,

PROMOTYPE as SLS\_PROMO\_TYPE

FROM "PROD\_RAW"."STACKLINE"."PROMOTIONS"

--WHERE WEEKID >= 202231

UNION

SELECT RETAILERID,

RETAILERSKU,

WEEKID,

'#' as SLS\_PROMO\_TYPE

FROM "PROD\_RAW"."STACKLINE"."RATINGSREVIEWS"

--WHERE WEEKID >= 202231

UNION

SELECT RETAILERID,

RETAILERSKU,

WEEKID,

'#' as SLS\_PROMO\_TYPE

FROM "PROD\_RAW"."STACKLINE"."CONTENTSCORE"

-- WHERE WEEKID >= 202230

)

SELECT SUM(SOURCE\_COUNT) AS PROD\_RAW\_SOURCE\_COUNT,

SUM(TARGET\_COUNT) AS TEST\_EDW\_TARGET\_COUNT

FROM (

SELECT COUNT(\*) as SOURCE\_COUNT, sum(0) as TARGET\_COUNT FROM STACKLINE\_RETAILERS AS R

UNION ALL

SELECT SUM(0) as SOURCE\_COUNT, COUNT(\*) as TARGET\_COUNT FROM test\_EDW.CONSOLIDATED.EDW\_PRICING\_COMPETITOR\_SALES

where SRC\_SYS\_KEY = 'STACKLINE'

and CURR\_RCRD\_FLAG = 'Y'

);