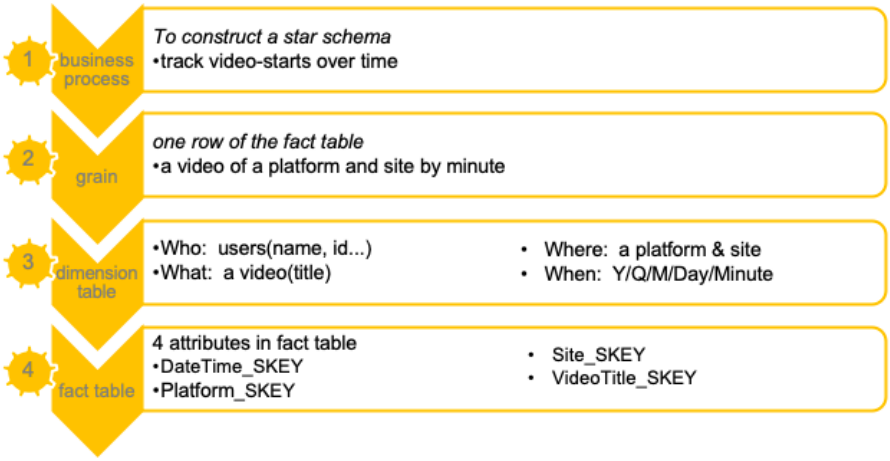
Data pipeline ETL

Table of Contents

Data Warehouse Design

Four phases



Tables design

video_data (raw data)

COLUMN_NAME	DATA_TYPE	PK	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
DATETIME	VARCHAR(30)	N	Yes	null	1	Data from raw file
VIDEOTITLE	VARCHAR(200)	N	Yes	null	2	Data from raw file
EVENTS	VARCHAR(150)	N	Yes	null	3	Data from raw file

staging(staging fact table)

COLUMN_NAME	DATA_TYPE	PK	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
DATETIME	VARCHAR(20)	N	NO	null	1	Data reformatted from video_data. DATETIME
PLATFORM	VARCHAR(20)	N	NO	null	2	Data derived from video_data. VIDEOTITLE
SITE	VARCHAR(10)	N	NO	null	3	Data derived from video_data. VIDEOTITLE
TITLE	VARCHAR(200)	N	NO	null	4	Data derived from video_data. VIDEOTITLE

FACTVIDEOSTART (Fact table)

COLUMN_NAME	DATA_TYPE	PK	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
DATETIME_SKEY	int	N	NO	null	1	Data derived from DIMDATE. DATETIME_SKEY
PLATFORM_SKEY	int	N	NO	null	2	Data derived from DIMPLATFORM. PLATFORM_SKEY
SITE_SKEY	int	N	NO	null	3	Data derived from DIMSITE. SITE_SKEY
TITLE_SKEY	int	N	NO	null	4	Data derived from DIMTITLE. TITLE_SKEY

dim_time(staging dimension table)

COLUMN_NAME	DATA_TYPE	PK	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
DATETIME	VARCHAR(20)	N	NO	null	1	Data reformatted from fact_dlt. DataTime
YEAR	int	N	NO	null	2	Data reformatted from fact_dlt. DataTime
MONTH	int	N	NO	null	3	Data reformatted from fact_dlt. DataTime
DAY	int	N	NO	null	4	Data reformatted from fact_dlt. DataTime
HOUR	int	N	NO	null	5	Data reformatted from fact_dlt. DataTime
MINU	int	N	NO	null	6	Data reformatted from fact_dlt. DataTime

dim_platform(staging dimension table)

COLUMN_NAME	DATA_TYPE	PK	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
PLATFORM	VARCHAR(20)	N	NO	null	1	Data derived from fact_dlt. platform

dim_site (staging dimension table)

COLUMN_NAME	DATA_TYPE	PK	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
SITE	VARCHAR(10)	N	NO	null	1	Data derived from fact_dlt. site

dim_title(staging dimension table)

COLUMN_NAME	DATA_TYPE	PK	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
TITLE	VARCHAR(200)	N	NO	null	1	Data derived from fact_dlt. title

DIMDATE (dimension table)

COLUMN_NAME	DATA_TYPE	PK	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
DATETIME_SKEY	int	Y	NO		1	
DATETIME	VARCHAR(12)	N	NO	null	2	Data derived from dim_time. DataTime

DIMPLATFORM (dimension table)

COLUMN_NAME	DATA_TYPE	PK	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
PLATFORM_SKEY	int	Y	NO		1	
PLATFORM	VARCHAR(20)	N	NO	null	2	Data derived from staging. PLATFORM

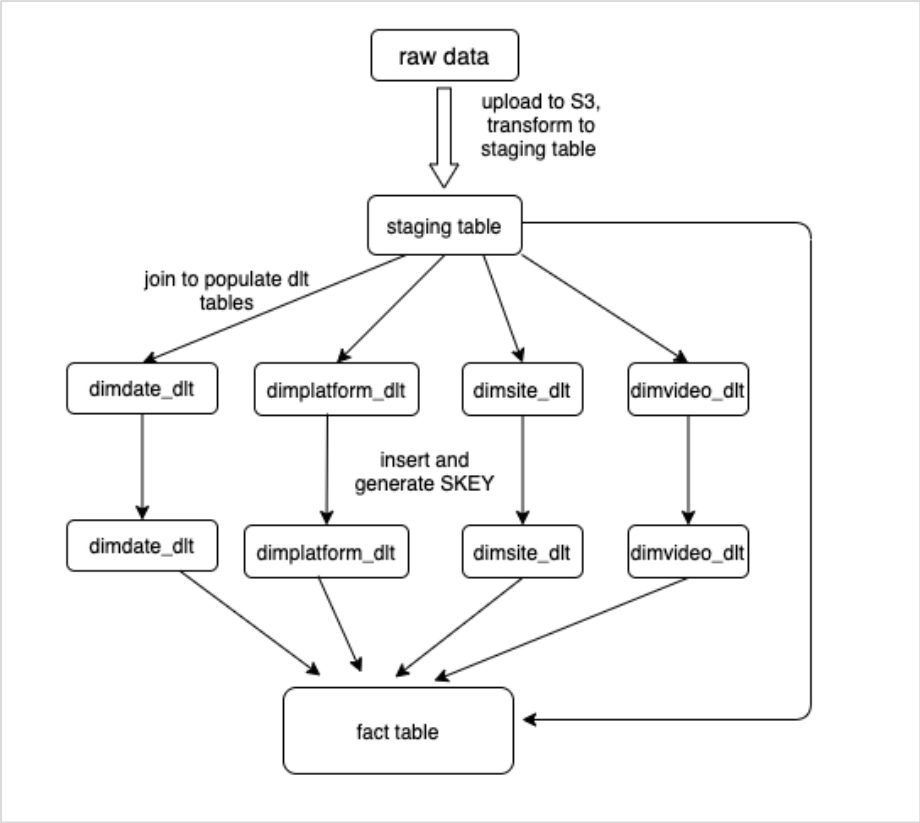
DIMSITE (dimension table)

COLUMN_NAME	DATA_TYPE	PK	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
SITE_SKEY	int	Y	NO		1	
SITE	VARCHAR(10)	N	NO	null	2	Data derived from staging. SITE

DIMTITLE(dimension table)

COLUMN_NAME	DATA_TYPE	PK	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
TITLE_SKEY	int	Y	NO		1	
TITLE	VARCHAR(200)	N	NO	null	2	Data derived from dim_title. TITLE

Process Design flow chart



Automation ETL Process(load to Redshift)

once upload raw data into S3, it will invoke lambda(python script) to extract data and transform data, generate dimension delta and fact delta csv file, store back to S3 and then invokes lambda(python script) to load processed data into database redshift.

Extraction, validation, clean and transformation(please find relative detail code)

1. Change encoding into UTF-8, otherwise the special characters will show.
2. Upload raw data video_data.csv into S3
3. Lambda will be invoked (Python script) to do data auditing

```
#data auditing: count total #records
    print(f"total recieved records number is {len(raw)}")
```

```
▶  2020-09-14T00:33:55.481+10:0...  total recieved records number is 1333658
```

CloudWatch Log Events:

```
    #data auditing: find max length of all field in fact
for col in fact.columns:
    print(f"{col}  max len is {max([len(i) for i in fact[col] if i])}")
```

CloudWatch Log Events:

▶	2020-09-14T00:56:29.870+10:0...	DateTime max len is 16
▶	2020-09-14T00:56:30.212+10:0...	title max len is 106
▶	2020-09-14T00:56:30.551+10:0...	platform max len is 7
▶	2020-09-14T00:56:30.851+10:0...	site max len is 4

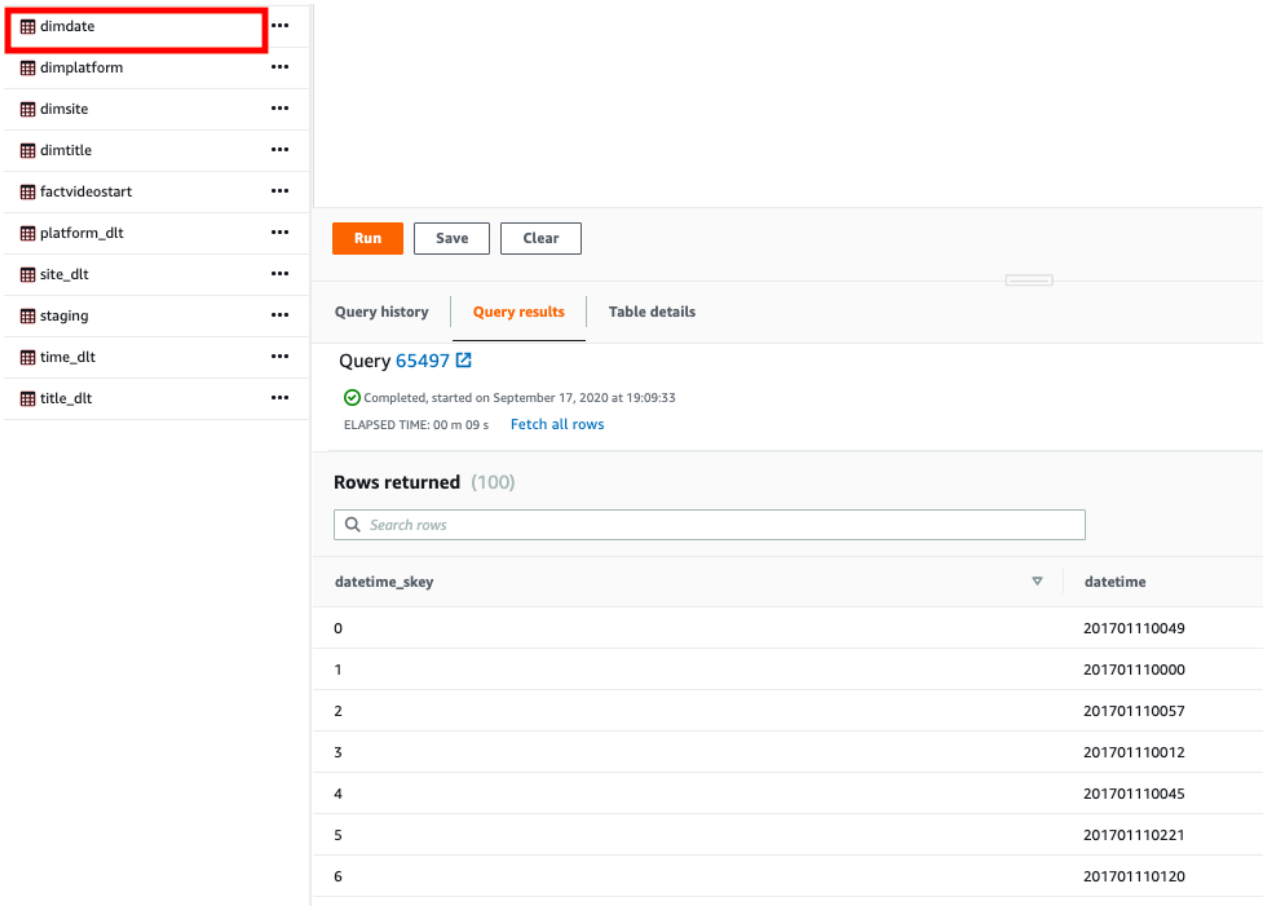
Use the result to adjust the length of columns in tables in Redshift.

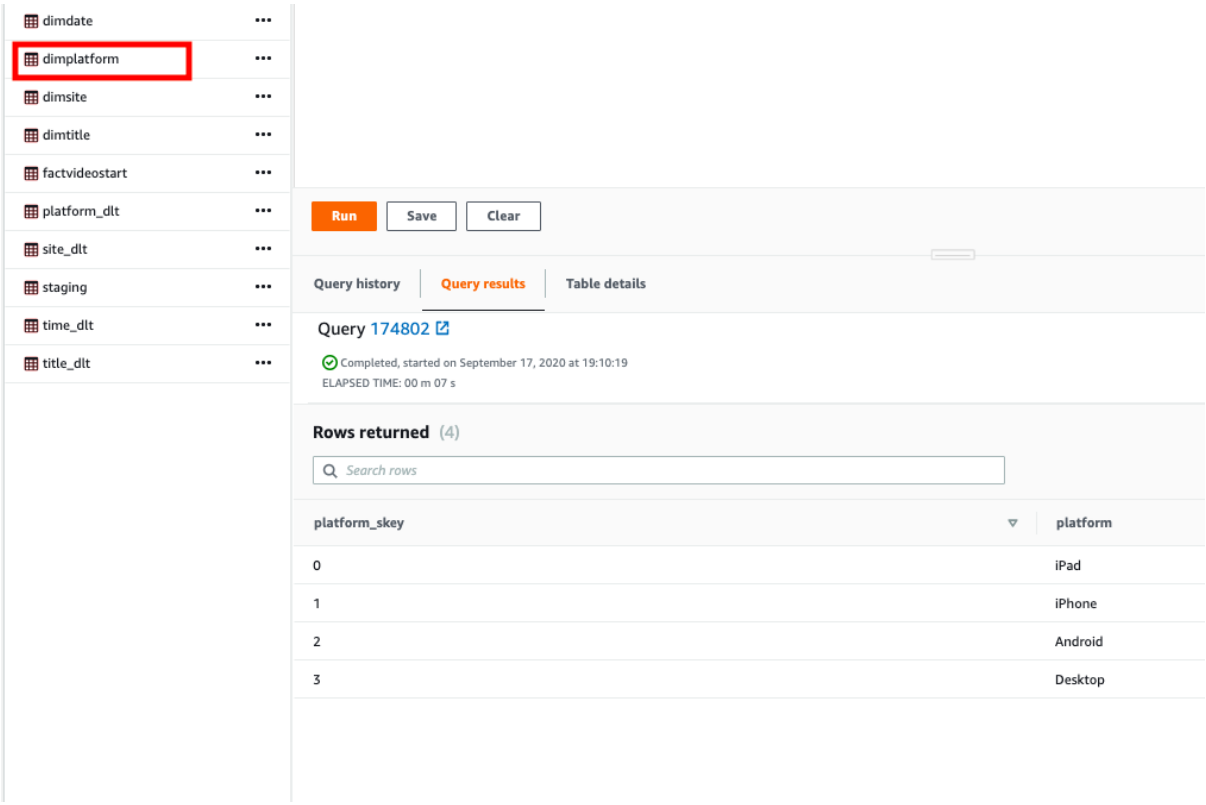
4. Then wash raw data and load into fact delta and dimension delta tables.
   Only keep events which contain '206';
   Only keep records that len(split(',',))>1.

5. The lambda will populate all dimension delta tables and fact delta table as following separately: dim_time.csv, dim_platform.csv, dim_site.csv, dim_title.csv, fact_dlt.csv.

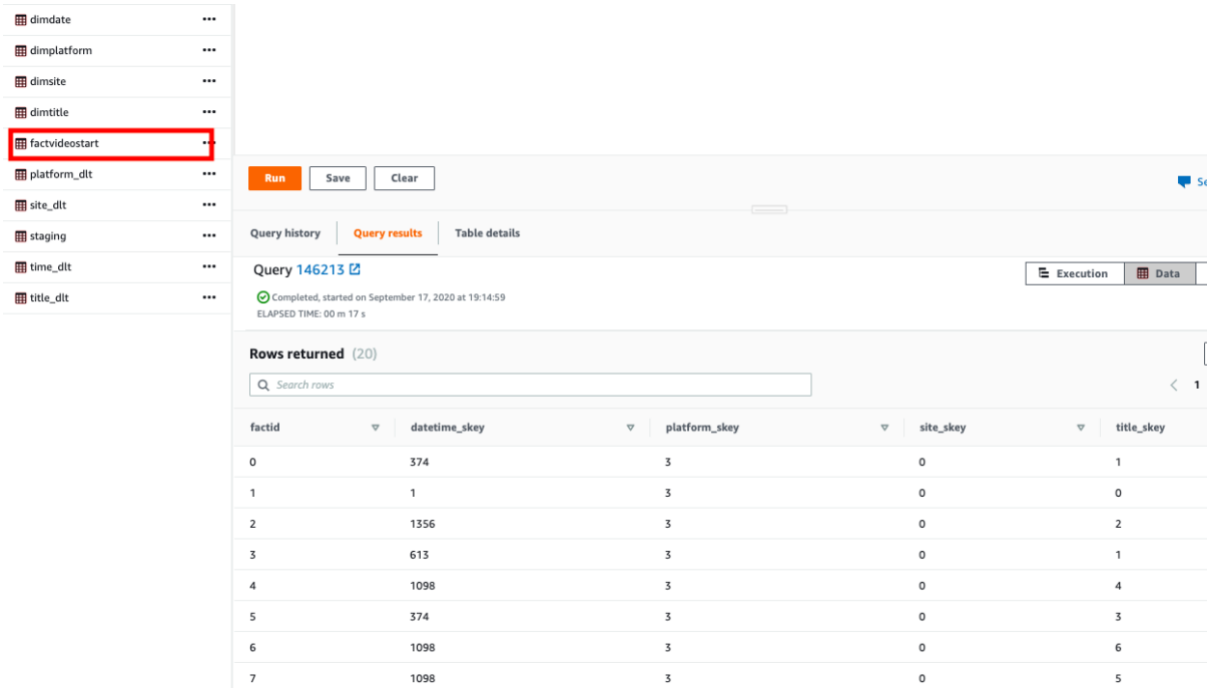
Load data from S3 to redshift (detailed code please find relative code)

1. SQL create dimension tables in Redshift if those dimension tables not exist.
2. SQL create dimension delta tables and staging table if not exist in Redshift
3. Redshift SQL query to COPY data in processed folder in S3 into delta tables and staging table.
4. Insert and generate SKEY for DIM tables DIMDATE, DIMPLATFORM, DIMSITE and DIMTITLE.





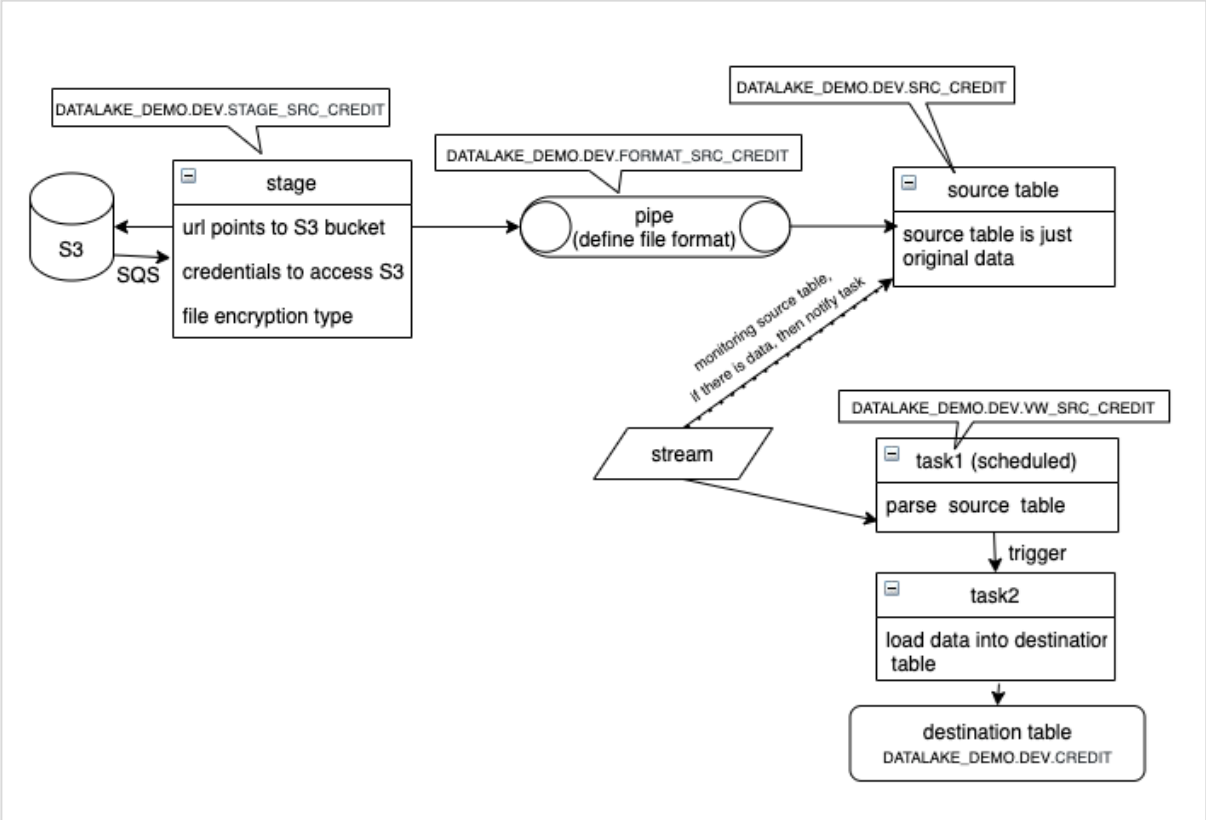
5. Use staging table, DIMDATE, DIMPLATFORM, DIMSITE and DIMTITLE to generate output data and append the data into fact table FACTVIDEOSTART.



6. Truncate staging, time_dlt, site_dlt, platform_dlt, title_dlt.

Automation ETL Process with Snow pipe (load to Snowflake)

1. Change encoding into UTF-8, otherwise the special characters will show.
2. Create a snow pipe. snow pipe.sql
3. Upload raw data into S3, and then SQS will trigger the pipe to load data into Snowflake as below process figure shows.



4. Similar with what have done in Automation ETL process with loading to Redshift: after staging table generated, updated dimension delta tables and then populated dimension tables with those delta tables, and then get SKEY for fact table by joining staging table and dimension tables. Finally, don't remember to truncate delta tables. populate_dimension_fact_tables.sql

NOTE:

1. **SKEY stands for surrogate key.**
2. **The current design is Dimension Type One.**
3. **If the source dimension data contains other attributes other than just PK, and we want to track changes of those attributes, eg. sometimes we want to track the product's price changing, then we should use Dimension Type Two.**
   One sample of Dimension Type Two
   Delta Data from 06/04/2017:

Product_ID	Product_Name	Price	Location
P001	Iphone6	750	Townhall Shop
P003	Iphone7	1000	Townhall Shop

Data in dimension table:

Product_SKEY	Product_ID	Product_Name	Price	Location	Current_Flag	Start_Date	End_Date
111	P001	Iphone6	800	Townhall Shop	Y	31/12/2016	31/12/9999
112	P002	Iphone6Plus	900	Townhall Shop	Y	20/01/2017	31/12/9999

Add new product (P003) and update product (P001) in dimension table:

Product_SKEY	Product_ID	Product_Name	Price	Location	Current_Flag	Start_Date	End_Date
111	P001	Iphone6	800	Townhall Shop	N	31/12/2016	05/04/2017
112	P002	Iphone6Plus	900	Townhall Shop	Y	20/01/2017	31/12/9999
113	P003	Iphone7	1000	Townhall Shop	Y	06/04/2017	31/12/9999
114	P001	Iphone6	800	Townhall Shop	Y	06/04/2017	31/12/9999

Yellow part is updated, red part is insertion.
When there is a new record coming, we generate new record with new SKEY, Current_Flag='Y',Start_Date = Current_Date, End_Date=31/12/9999
When there is a updated record coming, we generate new record with new SKEY, Current_Flag='Y',Start_Date = Current_Date, End_Date=31/12/9999, and at the same time, we update the old record with Current_Flag='N', End_Date= Current_Date-1.

Therefore, when we populate new records into fact table, we need a filter eg. Current_Flag='Y' in order to get the correct SKEY;
If we want to track the history data for certain days, we use a time filter between Start_Date and End_Date.

For example, if in fact table we see a transaction like customer purchased product(P001) on 01/04/2017, by looking at product dimension table, we could find the price that customer paid at that moment was 800 not 750, although 750 is the current price of P001