InBank Assignment Solution

Title: Building a Data Warehouse Solution for Netflix Customer Metrics

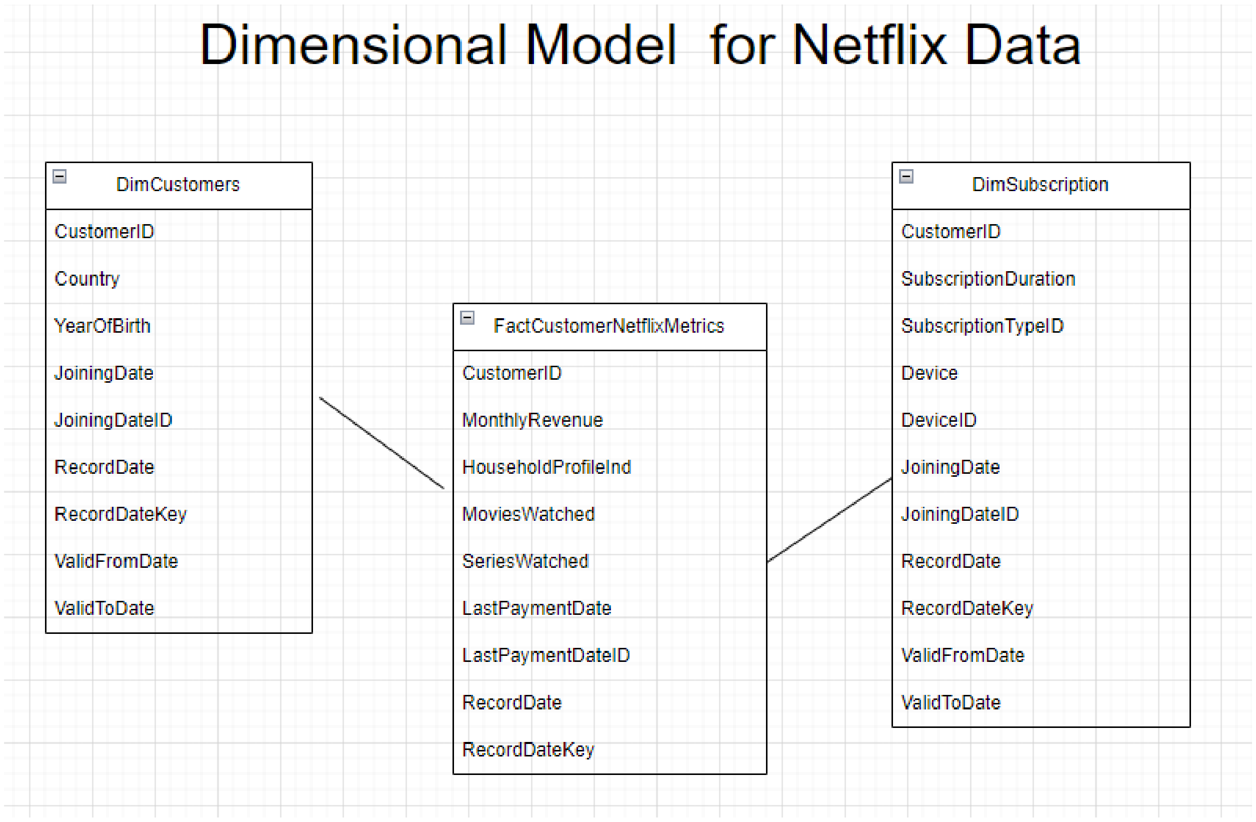
Deployment Details:
Python 3.9
MySQl 8.0.33

To run the solution a Python environment with an active MySQL server connection would be needed.

Note: All the files are in the git repo project. For ease of use, please download the complete project.

File Execution Sequence:
TableCreationsInbank.sql
DataExtractionLoading.py
DataStaging.py
CNFTableLoading.py
DimDateFeeder.py
DimSurrogateKeys.py
DataConform.py
DataPresentation.py
Queries_inbank.sql

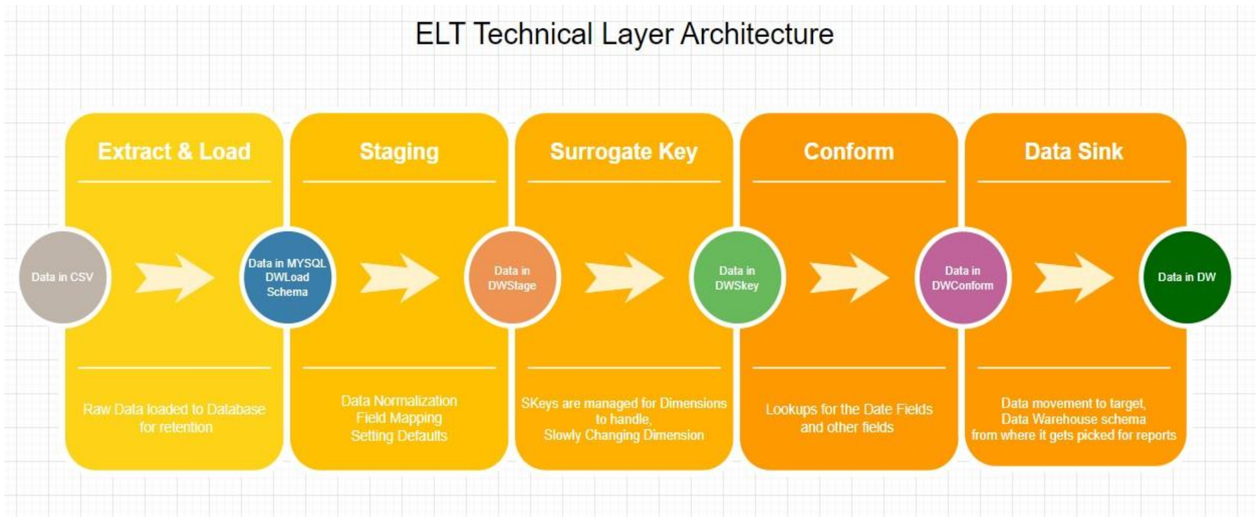
Introduction: This document presents an overview of the data warehouse solution designed and implemented for Netflix's customer metrics. The goal of this solution is to provide a robust ETL to get a good quality data for insightful analytics and reporting on customer behavior.



The provided diagram represents a high-level dimensional model of the data. A STAR schema for the model was chosen due to its advantages in enabling fast and straightforward data querying and analytics. It offers efficient information retrieval and serves well for the team's specific needs. However, the team acknowledges that the Snowflake schema could be a viable alternative in certain scenarios, depending on the data usage and requirements.

Solution Architecture:

ELT has been implemented in different stages. For each stage tables have created in a different schema.


ELT Technical Layer Architecture
Extract & Load
Staging
Surrogate Key
Conform
Data Sink
Data in CSV
Data in MYSQL DWLoad Schema
Data in DWStage
Data in DWSkey
Data in DWConform
Data in DW
Raw Data loaded to Database for retention
Data Normalization Field Mapping Setting Defaults
SKeys are managed for Dimensions to handle, Slowly Changing Dimension
Lookups for the Date Fields and other fields
Data movement to target, Data Warehouse schema from where it gets picked for reports

Extract & Load:

Data is read from the provided CSV file and pushed to MySql database within DWLoad Schema. Data is loaded for the retention in its raw form. This table is supposed to take all the data that comes in, so no rules for handling duplicity has been implemented. How we schedule the job to be run is extremely important here, since it will depend on the arrival time of the source files. If daily, parameters to pick the file would have to be changed.

Implementation - Last file from the mentioned directory is picked. All the records are read in data frame and the records without CustomerID are dropped. Dates are converted in a format handled by MySql.

Table name:
DWLoad.NetflixSubscription

Staging:

Data is taken from DWLoad and is loaded to DWStage schema. Data is normalized from one table to three tables. In this stage, the tables are truncated each time the data is loaded because no history is required for this step. It is at this stage all the mapping for the filed names take place according to the requirements for the target table.

Implementation – Data reading has been set to take the data loaded in current month. If the ETL is run monthly, parameters must be changed to take the data from last month. Default values are implemented for each field. Age has been changed to YearOfBirth, so it would be more convenient to handle the field in Skey. Bulk insertion has been implemented to maintain the data Integrity and make it fast. A new filed RecordDate has been added to each table to mark their loading date.

Tables:
DWStage.DimCustomers , DWStage.DimSubscription , DWStage.FactCustomerNetflixMetrics

Surrogate Key:

In this stage, data is read from DWStage and pushed to DWSkey tables. This stage is only for the Dimensions. At this stage slowly changing dimensions are handled.

Implementation – SCD Type II has been implemented to maintain the maximum history. Four new fields are added in each table, which tells, if record about the customer is outdated or still valid. Newly added fields are as follows:
RecordKey,ValidFromDate, ValidToDate, IsCurrent.

Here RecordKey is auto incremented and set as primary. Whenever SKey job is run for appending the data in dimensions. For each incoming record it is checked if record for this CustomerID with IsCurrent status 1 exists or not. If record does not exist, incoming row is inserted keeping ValidFromDate as joining date, default infinite value for ValidToDate and default flag 1 for IsCurrent. If record exists and has different values, then the new incoming record is inserted. If record exists but the values are same, nothing happens. While the record is inserted, ValidToDate for the old record is set to the current date - 1 and IsCurrent is to 0. For the new record, ValidFromDate is set to current date, ValidToDate is set to infinite and IsCurrent is set to 1. Tables have been constrained not to accept duplication. Indexing have been applied to the table for running the queries faster.

Tables:
DWSkey.DimCustomers, DWSkey.DimSubscription

Conform:

At this stage, dimensions take data from SKey and Facts take data from DWStage. Here, data is finalized for the target tables. None of the change should go beyond this step.

Implementation – Truncation for each run is implemented. New fields are added for each date field to store dates in integer form for better reporting in the end. The field which are coming from source have suffix ID and the fields generated
by system have a suffix Key. For ValidFromDate and ValidToDate, now integer field are not implemented but can easily be implemented later as per the need.
In this one CNF schema is also created, this hold the date dimension, dimension for subscriptionType, DeviceType and infuture, may be for the Country names. Tables for CNF are loaded separately. They are loaded to be joined with main tables and get integer values for their corresponding fields. If the table goes huge, then to save the resources only integer countervalues for many of the can be used.

Tables:
DWConform.DimCustomers, DWConform.DimSubscription, DWConform.FactCustomerNetflixMetrics

Data Presentation/Sink :

At this stage data is read from DWConform and feed to DW. DW is our aret schema, from where the data can be picked for further analytics.

Implementation – For dimensions all the records with new CustomerID are inserted. If there are some records with CustomerID matching with existing one then the CustomerID is checked to get the maximum value for ValidFromDate, Record with maximum value is updated for the existing one. Tables have been constrained not to accept duplication. Indexing have been applied to the table for running the queries faster.

Tables :
DW.DimCustomers, DW.DimSubscription, DW.FactCustomerNetflixMetrics

How would you make such loading repeatable on a regular basis?
Answer: ELT can be scheduled using CRON jobs. With minor configurational change, this ELT can be run for data loading on a regular basis. Only thing which would matter is when the job is run as per the arrival of new files and from what time we have scheduled to pick the data from DWLoad in staging.

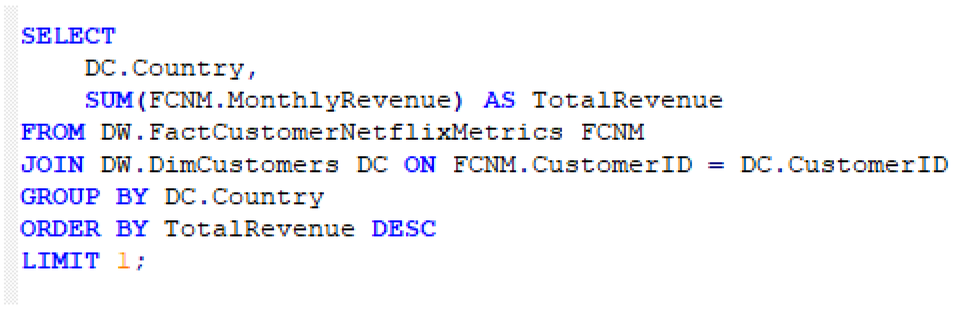
What control measures could be deployed here to ensure the correctness of data? Do you notice something odd about the dataset provided?
Answer: To ensure the correctness of data in the system, several control measures can be deployed:

1. Data Validation: Implement data validation checks at different stages of the ETL process. This includes validating data types, ranges, setting defaults formats, and ensuring data conforms to expected patterns.
2. Data Integrity Constraints: Set up appropriate primary key, foreign key, unique, and check constraints on database tables to maintain data integrity and prevent inconsistencies.
3. Data Quality Rules: Data quality rules can be implemented which would check the daily row count for each table. W.R.T to some pre-defined threshold if the row count decreases or increases suddenly, a red flag could be raised
4. Error Handling: Implement comprehensive error handling and logging mechanisms to capture and track any issues that arise during data processing. This will facilitate timely resolution and prevent data inconsistencies.
5. Automated Testing: Regularly perform automated testing and reconciliation of data between source systems and the data warehouse to identify discrepancies or data quality issues.
6. Data Profiling: Use data profiling techniques to analyze data quality, identify anomalies, and understand data patterns. This will help in addressing data quality issues proactively.
7. Data Cleansing: Implement data cleansing processes to identify and correct errors, inconsistencies, and missing values in the data.

Yes, something odd has been noted down, there are two customers with age 100+. One is from United States aging, 107 and one from UK, declared age 904. To handle such things ranges would have to be defined it staging.

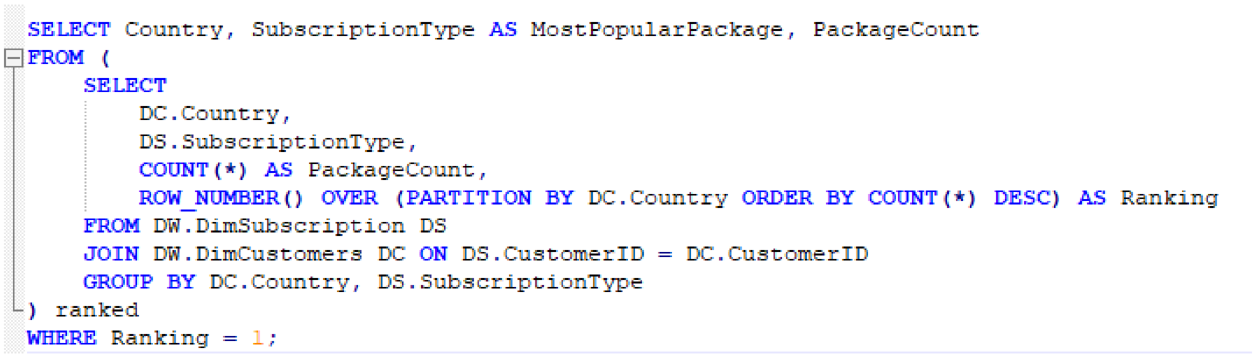
Write queries for answering the following questions:

A. The most profitable country for Netflix.



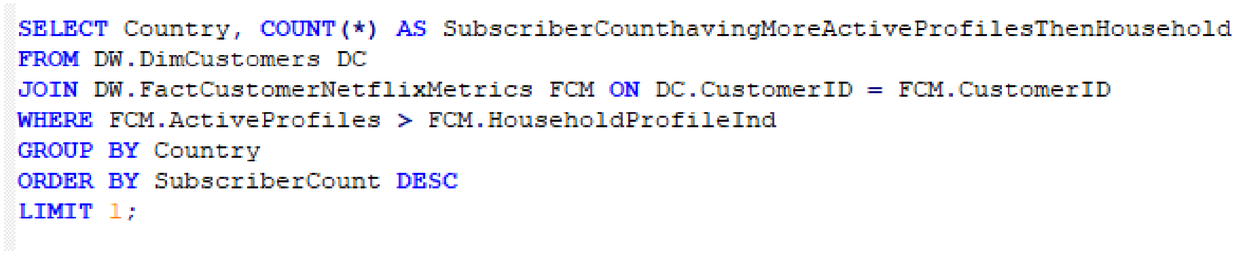
```sql
SELECT
    DC.Country,
    SUM(FCNM.MonthlyRevenue) AS TotalRevenue
FROM DW.FactCustomerNetflixMetrics FCNM
JOIN DW.DimCustomers DC ON FCNM.CustomerID = DC.CustomerID
GROUP BY DC.Country
ORDER BY TotalRevenue DESC
LIMIT 1;
```

B. The most popular packages per country.



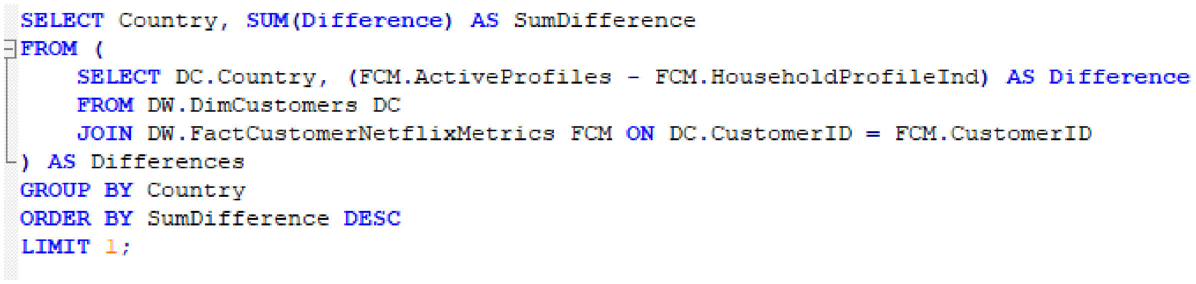
```sql
SELECT Country, SubscriptionType AS MostPopularPackage, PackageCount
FROM (
    SELECT
        DC.Country,
        DS.SubscriptionType,
        COUNT(*) AS PackageCount,
        ROW_NUMBER() OVER (PARTITION BY DC.Country ORDER BY COUNT(*) DESC) AS Ranking
    FROM DW.DimSubscription DS
    JOIN DW.DimCustomers DC ON DS.CustomerID = DC.CustomerID
    GROUP BY DC.Country, DS.SubscriptionType
) ranked
WHERE Ranking = 1;
```

C. Which country has the potential for improving earnings if Netflix starts charging subscribers an additional fee for sharing Netflix households outside of their own?



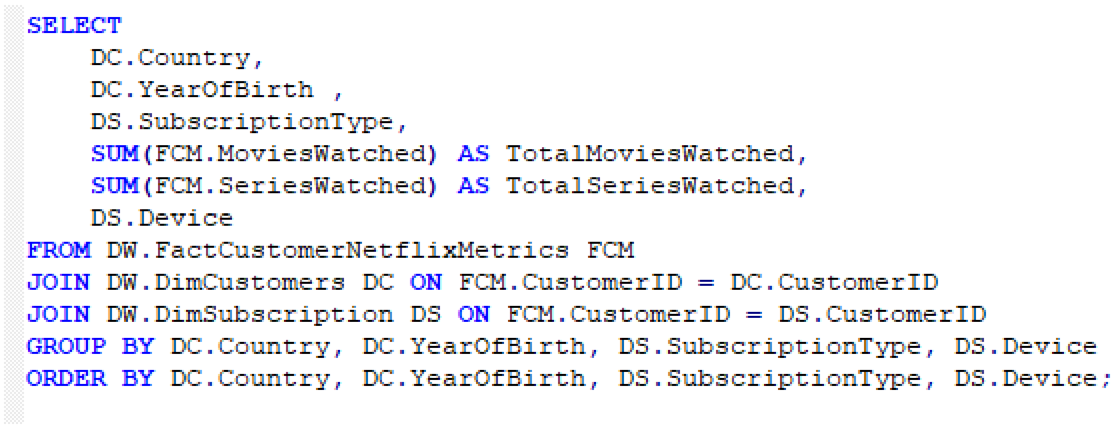
```sql
SELECT Country, COUNT(*) AS SubscriberCounthavingMoreActiveProfilesThenHousehold
FROM DW.DimCustomers DC
JOIN DW.FactCustomerNetflixMetrics FCM ON DC.CustomerID = FCM.CustomerID
WHERE FCM.ActiveProfiles > FCM.HouseholdProfileInd
GROUP BY Country
ORDER BY SubscriberCount DESC
LIMIT 1;
```

Another way to find it would be, finding the country having maximum difference between active profiles vs household profiles.



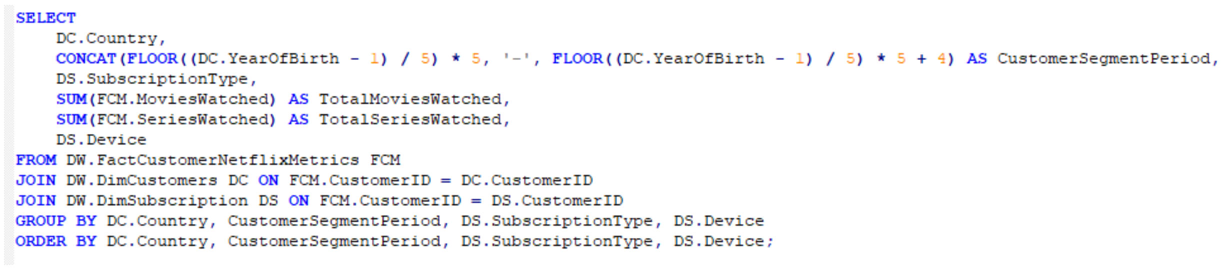
```
SELECT Country, SUM(Difference) AS SumDifference
FROM (
    SELECT DC.Country, (FCM.ActiveProfiles - FCM.HouseholdProfileInd) AS Difference
    FROM DW.DimCustomers DC
    JOIN DW.FactCustomerNetflixMetrics FCM ON DC.CustomerID = FCM.CustomerID
) AS Differences
GROUP BY Country
ORDER BY SumDifference DESC
LIMIT 1;
```

D. A report showing the popularity of Movies and Series in different customer segments and the device used to consume, across the different markets the company operates in.



```
SELECT
    DC.Country,
    DC.YearOfBirth ,
    DS.SubscriptionType,
    SUM(FCM.MoviesWatched) AS TotalMoviesWatched,
    SUM(FCM.SeriesWatched) AS TotalSeriesWatched,
    DS.Device
FROM DW.FactCustomerNetflixMetrics FCM
JOIN DW.DimCustomers DC ON FCM.CustomerID = DC.CustomerID
JOIN DW.DimSubscription DS ON FCM.CustomerID = DS.CustomerID
GROUP BY DC.Country, DC.YearOfBirth, DS.SubscriptionType, DS.Device
ORDER BY DC.Country, DC.YearOfBirth, DS.SubscriptionType, DS.Device;
```

Another way of showing the segments as a period of 5 years range



```
SELECT
    DC.Country,
    CONCAT(FLOOR((DC.YearOfBirth - 1) / 5) * 5, '-', FLOOR((DC.YearOfBirth - 1) / 5) * 5 + 4) AS CustomerSegmentPeriod,
    DS.SubscriptionType,
    SUM(FCM.MoviesWatched) AS TotalMoviesWatched,
    SUM(FCM.SeriesWatched) AS TotalSeriesWatched,
    DS.Device
FROM DW.FactCustomerNetflixMetrics FCM
JOIN DW.DimCustomers DC ON FCM.CustomerID = DC.CustomerID
JOIN DW.DimSubscription DS ON FCM.CustomerID = DS.CustomerID
GROUP BY DC.Country, CustomerSegmentPeriod, DS.SubscriptionType, DS.Device
ORDER BY DC.Country, CustomerSegmentPeriod, DS.SubscriptionType, DS.Device;
```

Choices:

- Chose Python and MySql because of experience in working with them.
- Implemented SCD type II, because of its capability to store history as much as want.
- Have tried to follow good practices in ETL such as all the mappings and major changes in staging, surrogate keys in skey and lookups in conform step.
- For Data quality, have implemented “not null” for all the fields in target tables.
- CNF table for DeviceType and Subscription type can be done manually as well and could have implemented some if else while fulfilling the values but make it universal implemented lookups.

Things I would have done better:

- If I would have more time – I would have implemented DQ rules, Data Range validation parameters.
- I would have implemented some monitoring system, with the alerts.
- I would have implemented checks for late arriving dimensions as well.
- I would have deployed the whole solution in a virtual machine and would have shared credential, So it would be easy to evaluate.
- I would have represented data using some visualization tool.