**DOCUMENTATION**

1. **Data tables overview**
2. Physical model, indexation, PK’s, FK’s *(Answer for task 8a as well as 2b, 2c)*

Each table has its own key ID; however, this is not unique because the data is logged to the database in different times. Therefore, primary key is a composite key including ‘TABLE\_ID’ and ‘BUCKET’. To be more specific, the primary key of each table is as below:

CLIENT: (CLIENT\_ID, BUCKET) (a)

INCOME: (INCOME\_ID, BUCKET) (b)

LOAN: (LOAN\_ID, BUCKET) (c)

HOUSEHOLD: (HOUSEHOLD\_ID, BUCKET) (d)

For foreign keys, table INCOME has a relation with table client so it has foreign key as (a). The same happens with table LOAN. Between income and household, table HOUSEHOLD has a foreign key as (b).

Regarding indexes, as the primary keys and foreign keys are all composite keys, the indexes are set using the 2 columns of each key. It helps improving the productivity of querying in the database as this process always look up the two columns as a combination.

1. Table customization

Due to the need of the reports, some columns need to be created in LOAN table in addition to the original data from the csv file. Below is the list of new columns and their description.

* DPD: This column indicates the DPD buckets, it includes 4 values
* 0 – if number of days past due is smaller than 30 days or there is no amount past due
* 1 - if number of days past due is equal or greater than 30 days and smaller than 60 days
* 2 - if number of days past due is equal or greater than 60 days and smaller than 90 days
* 3 - if number of days past due is equal or greater than 90 days
* INTODEFAULT\_New: This column indicates whether a record is default or not. It includes “Y” or “N” and it follows the logic suggested in the task requirement. This column is created because the current column INTODEFAULT is suspected to be wrong. It might happen that the definition in the task doesn’t match with the definition of the column INTODEFAULT when it was created.
* INTODEFAULT\_Mismatch: This is “N” when INTODEFAULT\_New matches INTODEFAULT and “Y” when they don’t match.
* BUCKET\_PAST\_DUE\_AMT: When checking the data, it seems that the PAST\_DUE\_AMT is the accumulate past due amount through time, because there are records in which PAST\_DUE\_AMT is greater than INSTALLMENT\_AMT. As there is no past due amount for each period (the delta amount generated in specific period), it is possible to calculate the amount of money paid each period. Therefore, BUCKET\_PAST\_DUE\_AMT is created for that purpose and it is the amount of past due in each period. It is calculated by taking PAST\_DUE\_AMT in that period, minus the PAST\_DUE\_AMT in previous PERIOD. In case it is the first period, it is equal to PAST\_DUE\_AMT. This value can be negative when people pay more than the INSTALLMENT\_AMT in that period to reduce the PAST\_DUE\_AMT.
* PAID\_AMT: This column indicates the amount paid each period. It is the amount people paid each period, it is calculated by taking INSTALLMENT\_AMT minus BUCKET\_PAST\_DUE\_AMT.

1. **Data quality metrics and data cleaning**
2. Summary for data quality metrics *(Answer for task 8b)*

In this case, I suggest 2 data quality metrics: % Data Import; % Data Join

% Data Import \* 100%

% Data Join \* 100%

% Data Import indicate the quality of the source data. If it is 100%, it means all records are correct, there is no NULL, no invalid values in any table and there are no problems happening when importing the data to the data base.

% Data Join indicate the complete match among the tables. The higher the metric, the completer and more consistent it is among four tables, in other words, there are no missing records or there are no problem happening in the joining process. In this project, the relationships among tables are all one – one.

Using it in this task requirement:

% Data Import table CLIENT = \*100% = 100%

% Data Import table HOUSEHOLD = \*100% = 100%

% Data Import table INCOME = \*100% = 100%

% Data Import table LOAN = \*100% ~ 100%

% Data Join = \*100% = 100%

For the metrics, I visualize them and put in the report file.

*(The above results and comments are answer for point 3b, 3c)*

In addition to the above metrics, I also check the inconsistency in some columns across reporting periods. There are some columns that I think they should remain unchanged along time.

Table CLIENT: EDUCATION, AGE.

|  |  |  |
| --- | --- | --- |
| no\_affected\_id | no\_incon\_education | no\_incon\_age |
| Number of ID that have at least one inconsistent column | Number of ID that have inconsistent in EDUCATION | Number of ID that have inconsistent in AGE |
| 1851 | 510 | 1711 |

Table HOUSEHOLD: MARRIED, HOUSE\_OWNER, CHILD\_NO, HH\_MEMBERS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| no\_affected\_id | no\_incon\_ married | no\_incon\_ house\_owner | no\_incon\_ child\_no | no\_incon\_ hh\_members |
| 17 | 0 | 17 | 0 | 0 |

Table INCOME: FIRST\_JOB, INCOME

|  |  |  |
| --- | --- | --- |
| no\_affected\_id | no\_incon\_first\_job | no\_incon\_income |
| 1711 | 0 | 1711 |

Table LOAN: INSTALLMENT\_NM, LOAN\_AMT, INSTALLMENT\_AMT

|  |  |  |  |
| --- | --- | --- | --- |
| no\_affected\_id | no\_incon\_ INSTALLMENT\_NM | no\_incon\_ LOAN\_AMT | no\_incon\_ INSTALLMENT\_AMT |
| 17 | 0 | 17 | 0 |

For the inconsistency, it depends on the policy of each institution, for some institutions, the data declared initially remain unchanged while in other institutions, the data can be updated through time. Therefore, in this step of checking, I just point out that there are inconsistencies. Yet, in order to take action, more information needs to be provided.

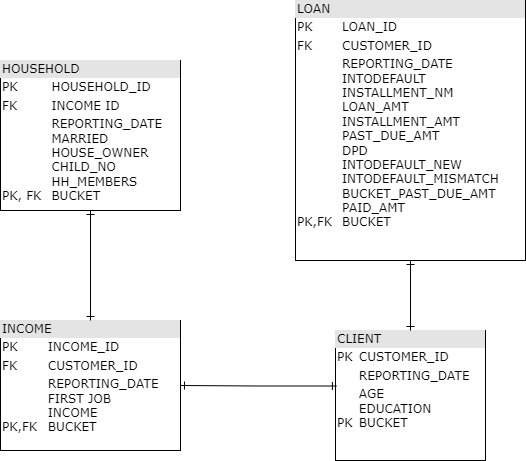
1. Data checking and cleaning *(Answer for 3a, 3d)*

Regarding data checking, the codes are stored in the folder (Accenture\_project\_job\_application\_Khon Nguyen\Deliverables\SQL script\data check).

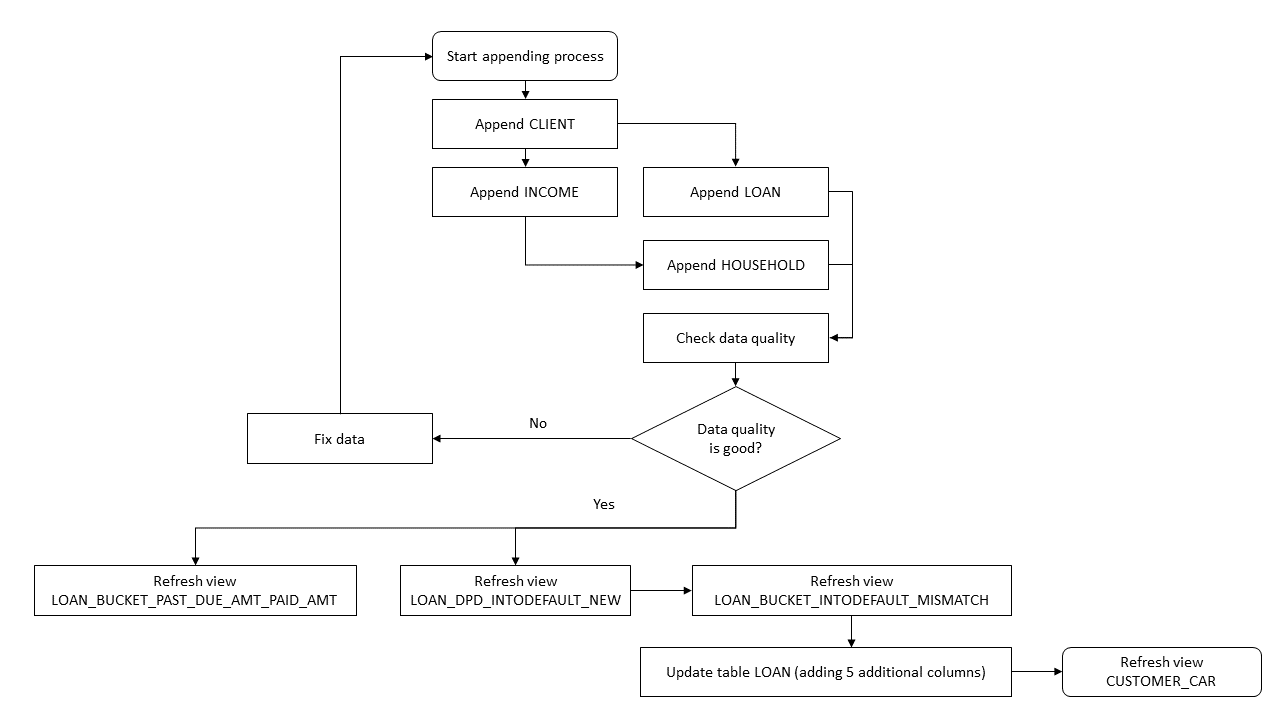
The first type of checking is validity check (the files “check table [table name] on validity”). The goal is to identify which record(s) contain the value that are not in the possible range. The codes are used to define the combination of ID and BUCKET. After these composite keys are defined, I run the code from the file “clean all tables” to remove all the invalid records. In practical approach, the record can also be double checked and fixed in the source rather than be removed.

Also, in order to define the inconsistencies, the codes “check table [table name] on consistency” are used. No action is taken for now due the lack of information.

1. **Entity relation diagram and data pipeline process**
2. Entity relation diagram *(Answer for 8c)*



1. Summary for data pipeline process *(Answer for 5a)*

The data pipeline process to append data for CUSTOMER\_CAR is shown in the above flowchart. Firstly, data is appended in CLIENT table, then INCOME and LOAN, then finally HOUSEHOLD. After that, the quality check takes place and in case the data quality is not good, it is fixed and the appending process restarts. In case the data quality is good, the views relating 5 additional columns for table LOAN are refreshed in which LOAN\_BUCKET\_INTODEFAULT\_MISMATCH is done after LOAN\_DPD\_INTODEFAULT\_NEW. Then the new LOAN table is updated and finally CUSTOMER\_CAR is refreshed.

Regarding the data appending frequency, as it is updated monthly in the end of the month, the appending process should happen monthly.

There are 2 approaches for appending data. For the 1st one, if the history data don’t change through time, we can append the data using the code INSERT INTO TABLE and we should only append the row that has the maximum value in column BUCKET. For the 2nd approach, if the history data change through time, for each month, we will remove all records currently stored on the database and import the new data. This approach is not as efficient. I assume the history data don’t change in the future; therefore the 1st approach is preferred.

To handle date correction, in case there are problem at the appending process, we can navigate the new data by filtering the records with greatest value in column ‘BUCKET’. In my opinion, the best way is to find the composite key that have problems to double check, and repair them in the source data. After that, new records are removed and the new repaired records will be appended again.

In the future, in case the data become bigger and there is a need to append more often than monthly, I suggest including one more column “DATE IMPORT” to each table. It indicates the date the data is imported to the database. With that column, it can play the row of the ‘BUCKET’ column when we append the data monthly. Using it will make it easier to identify the batch that have problem so that it’s easier to remove, repair and import again.

1. **Recommendation about RDBMS**

When doing this task, I encounter some problems with SQLite. Firstly, SQLite doesn’t support various data types like other RDBMS dialects. There are no DATETIME datatype and no DECIMAL datatype, so in this task, I have to use TEXT for DATETIME and REAL for FLOAT and DECIMAL. Secondly, when importing data, SQLite allows importing even though there is constrain in the data type. For example, in table ‘Loan’, there is one record that has text value in column PAST\_DUE\_AMT which was set as REAL but that record is still successfully imported to the database. Thirdly, SQLite lack some common function to validate data. For example, in the below codes in bold.

CREATE VIEW Loan\_invalid\_rows AS

select loan\_id, bucket from loan

where loan\_id <=0

or customer\_id <= 0

or intodefault not in ("Y", "N")

or installment\_nm < 12

or installment\_nm > 72

or loan\_amt < 500

or loan\_amt > 100000

**or loan\_amt NOT GLOB '\*[!0-9]\*'**

**or loan\_amt NOT GLOB '\*[!0-9].[!0-9]\*'**

or installment\_amt < 10

or installment\_amt > 100000

**or installment\_amt NOT GLOB '\*[!0-9]\*'**

**or installment\_amt NOT GLOB '\*[!0-9].[!0-9]\*'**

or past\_due\_amt < 0

**or past\_due\_amt NOT GLOB '\*[!0-9]\*'**

**or past\_due\_amt NOT GLOB '\*[!0-9].[!0-9]\*'**

or bucket <= 0

I need to validate the column in REAL datatype using Regex because SQLite lack of ISNUMERIC() function which is a very common one.

Therefore, I recommend using another RDBMS, such as Oracle. Because later, when the database get bigger significantly, Oracle is the dialect that are the most powerful on the market so it can adapt the need of the business. In addition, it can tackle all of the issues listed above.