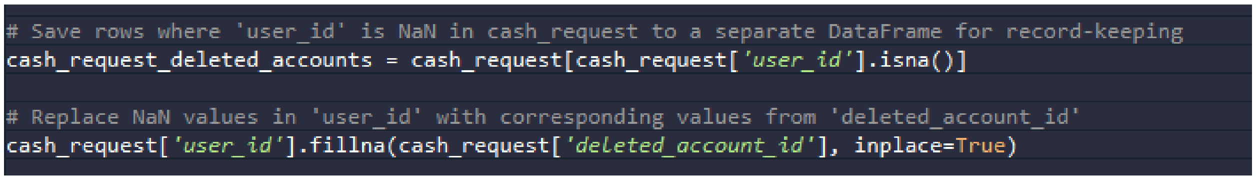
Introduction

This section documents the results of the data quality analysis conducted on the cash request and fee datasets, outlining issues and resolutions.

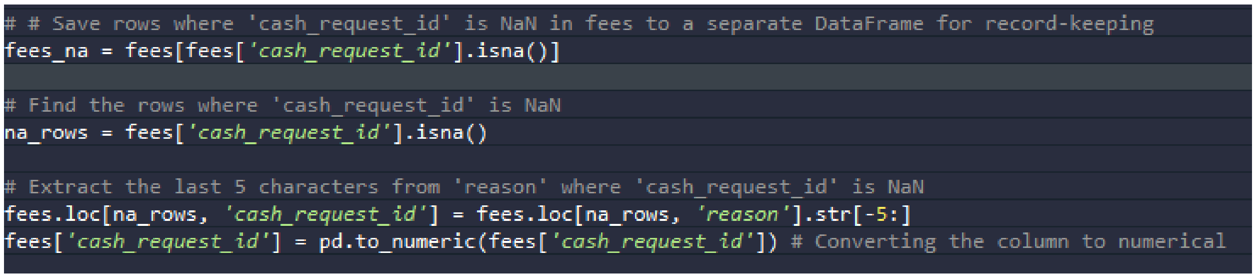
Missing Values

- In the **cash_request** dataset:
    - Found missing values in the user_id and deleted_account_id columns.
    - Steps taken: Replaced NaN user_id values with deleted_account_id where applicable.



```
# Save rows where 'user_id' is NaN in cash_request to a separate DataFrame for record-keeping
cash_request_deleted_accounts = cash_request[cash_request['user_id'].isna()]

# Replace NaN values in 'user_id' with corresponding values from 'deleted_account_id'
cash_request['user_id'].fillna(cash_request['deleted_account_id'], inplace=True)
```

- In the **fees** dataset:
    - Identified missing values in cash_request_id.
    - Steps taken: Extracted values from the reason column to populate missing cash_request_id values and converted the column to numeric type.



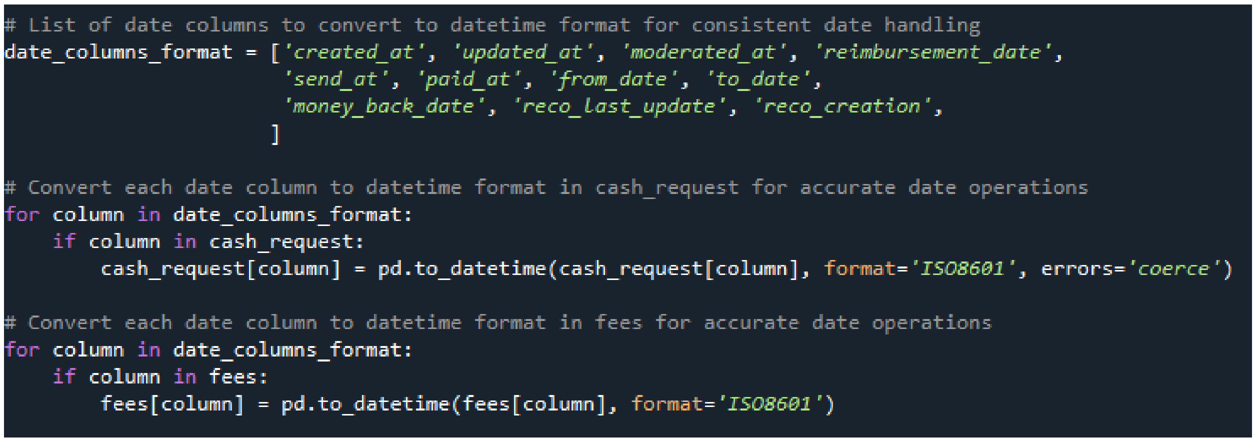
```
# # Save rows where 'cash_request_id' is NaN in fees to a separate DataFrame for record-keeping
fees_na = fees[fees['cash_request_id'].isna()]

# Find the rows where 'cash_request_id' is NaN
na_rows = fees['cash_request_id'].isna()

# Extract the last 5 characters from 'reason' where 'cash_request_id' is NaN
fees.loc[na_rows, 'cash_request_id'] = fees.loc[na_rows, 'reason'].str[-5:]
fees['cash_request_id'] = pd.to_numeric(fees['cash_request_id']) # Converting the column to numerical
```

Duplicates

- Found:
    - *0* duplicated rows in the **cash_request** dataset based on id.
    - *12177* duplicated rows based on user_id. This number of clients returned for another cash request.
    - *0* duplicated rows in the **fees** dataset based on id.
- Steps taken: No steps were taken to remove the duplicate values in user_id column.

Data Types

- Checked and confirmed that:
    - All columns have appropriate data types (see exploratory data report).
    - Converted date columns to datetime format for accurate date handling.



```python
# List of date columns to convert to datetime format for consistent date handling
date_columns_format = ['created_at', 'updated_at', 'moderated_at', 'reimbursement_date',
                       'send_at', 'paid_at', 'from_date', 'to_date',
                       'money_back_date', 'reco_last_update', 'reco_creation',
                      ]

# Convert each date column to datetime format in cash_request for accurate date operations
for column in date_columns_format:
    if column in cash_request:
        cash_request[column] = pd.to_datetime(cash_request[column], format='ISO8601', errors='coerce')

# Convert each date column to datetime format in fees for accurate date operations
for column in date_columns_format:
    if column in fees:
        fees[column] = pd.to_datetime(fees[column], format='ISO8601')
```