# FINANCIAL SERVICES CASE STUDY

Accurate, Controlled Test Data for Credit Card Testing

A major financial services company, managing billions of dollars in assets, provides a variety of services including consumer and commercial banking, credit cards, auto loans and savings accounts. Their credit card business alone represents millions of cardholders transacting billions of dollars in annual purchase volume.

The credit card market has become the largest U.S. consumer lending market and continues to expand by every measure of growth.  Credit card providers are under intense competitive pressure to deliver new and innovative products in the form of loyalty and reward programs, balance transfers, cash advances and other perks.

This places a big demand on developers and testers to manage a continuous pipeline of software enhancements and revisions. These applications interface with multiple systems as they execute a massive volume of credit transactions and manage user accounts. As part of a regulated industry, QA organizations are keenly focused on the accuracy and security of their information systems. The financial services company profiled in this case study provides an example of the rigorous testing and data provisioning requirements that must be followed when testing financial application software.

## THE NEED FOR SYNTHETIC TEST DATA

QA leadership for this organization wanted to eliminate a potential breach of sensitive customer information, such as social security and credit card account numbers, during their testing process. For this reason, they decided to evaluate the use of synthetic test data. They needed test data in a variety of formats for testing multiple systems and databases, including ASCII, EBCDIC, Parquet, JSON, and AVRO. The database environments include DB2, Oracle, SQL Server, Postgres, MySQL and Snowflake.

Previously, the QA team pulled data from production and manually scrubbed it to remove sensitive information and to create appropriate data combinations for testing. Now their goal is to automate the process of provisioning secure, synthetic test data with full control over data volume and variation with an ability to output the data in the multiple file formats described above. The QA team was introduced to GenRocket by one of the company's system integration partners and decided to evaluate GenRocket's *Test Data Automation* capabilities as part of a formal Proof of Concept (POC).

## MEETING THE POC REQUIREMENTS

The POC team identified two use cases, or stories, to conduct the POC. These stories describe test data challenges and the criteria for a successful POC. Test data must be output in multiple file formats. Some of these files require the control characters \x001 and ^A to act as delimiters between nested data segments or to mark an end of file. For both stories, sample data files were provided to GenRocket containing DDL to describe the table columns, primary keys, field attributes and record counts.

**Story 1: GenRocket populates the CreditCard Product File**

The following CreditCard_Product attributes are to be populated

- id (unique)
- Product Description
- ACE
- AcquisitionStrategyCode
- CMID
- CPC
- TPC
- Type
- ProductID
- All Values must come from Provided File named "UseCase1"

For each credit card description, the corresponding data attributes must be generated in all specified formats. The ASCII format requires a \x001 or Ctrl character ^A field separator.

**Story 2: GenRocket populates the CrossReference File**

The following CrossReference attributes are to be populated

- Possible_Plastic_Num_0 (unique)
- Non_NPI_1
- Possible_Driverlicense_2
- Possible_Plastic_Num_3 (Each row should be the same value as Possible_Plastic_Num_0)
- Possible_Acnt_ID\\Acnt_Num_4
- Non_NPI_5
- Non_NPI_6
- Non_NPI_7
- Possible_Loan_ID_Num\\Acnt_Num_8
- Possible_Loan_ID_Num\\Acnt_Num_9
- Non_NPI_10
- Non_NPI_11
- The details for each attribute should be obtained from the Data Profiler output which should be used as the input to GenRocket.

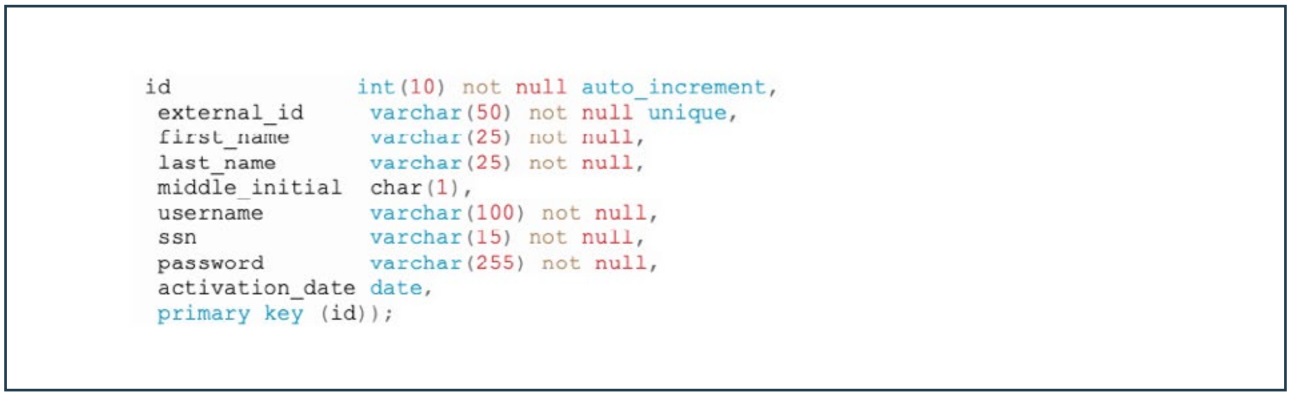
Synthetic data must represent valid production data and contain name, address, social security number and credit card number. Credit card numbers must have realistic values for Visa, Mastercard, American Express and Discover formats and pass the Mod-10 algorithm check. Test data files should be generated in all specified formats. ASCII formats must have \x001 or Ctrl character ^A field separators.

## THE GENROCKET SOLUTION

The GenRocket *Test Data Automation* platform is a perfect match for these requirements. Its component architecture allows testers to define data structures, select the required data generators for each field, and match them with data receivers for the desired output format.

The GenRocket platform allows testers to import any data model using a database schema or Data Definition Language (DDL) to automate the process of provisioning synthetic test data. This ensures test data is generated with the correct data structure and parent/child/sibling relationships between data tables.

Here is a sample DDL:



```
id              int(10) not null auto_increment,
 external_id     varchar(50) not null unique,
 first_name      varchar(25) not null,
 last_name       varchar(25) not null,
 middle_initial  char(1),
 username        varchar(100) not null,
 ssn             varchar(15) not null,
 password        varchar(255) not null,
 activation_date date,
 primary key (id));
```

GenRocket has more than 250 data generators including 7 different data generators just for credit cards. Generators have parameters that can be configured to accurately control the data values that are generated. They are part of a *Test Data Scenario* that controls the quantity of test data that is to be generated.

To support a wide variety of output formats, GenRocket has more than 50 data receivers including two new receivers created for this POC – IBM EBCDIC and Parquet file formats. Any receiver can be paired with any combination of generators to provide flexibility over the data values used for testing and consistency of data used across multiple environments.

GenRocket worked with the QA team at this financial services company to automate the process of importing the data model, configuring the data generators, controlling the record counts, and generating test data on-demand and in the desired output format.

## THE IMPACT OF TEST DATA AUTOMATION

This level of control, flexibility, security and efficiency of test data provisioning was not possible prior to GenRocket. As a result, the POC was highly successful. The implementation of GenRocket's *Test Data Automation* promises to bring a new level of data quality and test coverage to any QA process. At the same time, the accelerated pace of data delivery will improve the efficiency of test operations. Now software testing teams at financial services organizations can test at the speed of development as they ensure the quality of testing and the security of data.