Technical Analysis of SAS Import and Data Exploration Procedures

NYC Airbnb Data Processing Pipeline

NYCBnB Analytics Project
Miron Alin & Nitu Tudor

May 26, 2025

Contents

1 Introduction

This document provides a comprehensive technical analysis of the `import_explore.sas` script used in the NYCBnB Analytics project. The script implements a sophisticated data pipeline for importing, processing, and analyzing Airbnb data from New York City. This analysis examines the technical implementation details, procedural choices, and analytical methodologies employed throughout the script.

1.1 Dataset Context

The analysis focuses on the Airbnb listings dataset for New York City (featured_AB_NYC_2019.csv), which contains detailed information about available properties, including:

- Listing identifiers and metadata (ID, name, host information)
- Geographic information (neighbourhood, neighbourhood_group, latitude, longitude)
- Property characteristics (room_type)
- Pricing information (price)
- Availability metrics (minimum_nights, availability_365)
- Review metrics (number_of_reviews, reviews_per_month, last_review)
- Host metrics (calculated_host_listings_count)

1.2 Technical Objectives

The script's primary technical objectives include:

1. Implementing efficient data import procedures for CSV-formatted data
2. Executing comprehensive data validation and quality assessment
3. Performing data transformation and feature engineering
4. Creating specialized data subsets for targeted analysis
5. Implementing descriptive, exploratory, and predictive analytics
6. Generating visualization outputs for data interpretation

2 Program Architecture and SAS Implementation Details

The SAS program is structured as a complete analytical pipeline that fulfills multiple SAS programming requirements while maintaining a logical workflow. The script demonstrates advanced proficiency in the following SAS functionality domains:

- **Data Access and Import:** Efficient file handling and CSV import techniques
- **Data Transformation:** Advanced formatting, conditional logic, and variable creation
- **Data Quality:** Validation procedures and statistical assessment of data properties
- **Data Subsetting:** WHERE clause implementation and targeted dataset creation
- **SQL Integration:** Utilizing PROC SQL for advanced aggregation and cross-tabulation
- **Reporting:** Customized output formatting with PROC REPORT
- **Visualization:** Statistical graphics generation with PROC SGPLOT
- **Statistical Analysis:** Regression modeling with PROC REG
- **Machine Learning:** Predictive modeling with PROC LOGISTIC

3 Detailed Technical Analysis of Program Components

3.1 Data Import Procedure Implementation

The program begins with a import procedure, incorporating multiple efficiency and error-handling parameters:

```
* SAS Program: import_explore.sas ;
* Purpose: Import Airbnb data and perform initial exploration & formatting.;

* Set a libname for where to store the SAS dataset (optional, can use WORK
    library too) ;
* For this example, we'll use the WORK library which is temporary. ;
* LIBNAME project "." ; * This would create a permanent library in the current
    directory. For now, WORK is fine.;

* 1. Import the CSV file using DATA step with explicit variables ;
FILENAME ABDATA "/home/u64208646/Project/featured_AB_NYC_2019.csv";

DATA WORK.nyc_airbnb;
    /* Use efficient INFILE options */
    INFILE ABDATA DLM=',' MISSOVER DSD FIRSTOBS=2 TRUNCOVER;

    /* INPUT statement with explicit variables */
    INPUT
        id $
        name $
        host_id $
        host_name $
        neighbourhood_group $
        neighbourhood $
        latitude
        longitude
        room_type $
        price
        minimum_nights
        number_of_reviews
        last_review $
        reviews_per_month
        calculated_host_listings_count
        availability_365
        has_reviews
        days_since_last_review;
RUN;

FILENAME ABDATA CLEAR;
```

Listing 1: CSV Import Configuration

Technical Implementation Details:

- `FILENAME` statement creates a file reference (`ABDATA`) to the CSV file, enabling access through a logical name rather than a physical path
- `INFILE` statement employs multiple optimization parameters:
  - `DLM=','` - Specifies comma as the delimiter character
  - `MISSOVER` - Prevents SAS from reading past the end of the current line when a specified variable is not found

  - `DSD` - Handles delimited data with embedded delimiters, allowing for quotes around text fields that may contain commas
  - `FIRSTOBS=2` - Skips the header row of the CSV file
  - `TRUNCOVER` - Reads records that are shorter than expected without generating errors

- `INPUT` statement explicitly defines variable names and types:
  - `$` notation indicates character variables (e.g., `id $`)
  - Numeric variables are specified without type indicators (e.g., `price`)

- `FILENAME ABDATA CLEAR;` properly releases the file reference after import

This import configuration demonstrates advanced knowledge of SAS data handling, particularly regarding efficiency optimizations for large datasets and defensive programming practices to handle potential data quality issues.

3.2 Data Validation and Exploratory Analysis

Following import, the script implements a systematic approach to data validation and exploration through multiple complementary procedures:

```
TITLE "Contents of Imported Data (Check Column Types)";
PROC CONTENTS DATA=WORK.nyc_airbnb;
RUN;

TITLE "First 20 Rows of Imported Data (Check Data Alignment)";
PROC PRINT DATA=WORK.nyc_airbnb (OBS=20);
    VAR id name host_id host_name neighbourhood_group neighbourhood
        latitude longitude room_type price minimum_nights number_of_reviews
        last_review reviews_per_month calculated_host_listings_count
        availability_365 has_reviews days_since_last_review;
RUN;

TITLE "Descriptive Statistics for Numerical Variables";
PROC MEANS DATA=WORK.nyc_airbnb N MEAN STD MIN MAX Q1 MEDIAN Q3;
    VAR price minimum_nights number_of_reviews reviews_per_month
        calculated_host_listings_count availability_365 days_since_last_review;
RUN;

TITLE "Frequency Counts for Key Categorical Variables (Original Values)";
PROC FREQ DATA=WORK.nyc_airbnb ORDER=FREQ;
    TABLES neighbourhood_group room_type has_reviews / MISSING;
RUN;
```

Listing 2: Data Validation and Initial Exploration

Technical Implementation Analysis:

- `PROC CONTENTS` provides metadata verification:
  - Confirms correct variable types (character vs. numeric)
  - Validates dataset structure and properties
  - Reports on memory allocation and dataset characteristics
  - Identifies any automatic type conversions performed during import

- `PROC PRINT` enables visual inspection of data:
  - Limited to 20 observations for efficient review

  - Includes all variables to verify alignment and structure
  - Allows for detection of obvious data quality issues or import errors

- `PROC MEANS` performs comprehensive univariate analysis:
  - Calculates count of non-missing values (`N`)
  - Computes central tendency measures (`MEAN`, `MEDIAN`)
  - Determines dispersion metrics (`STD`)
  - Reports range statistics (`MIN`, `MAX`)
  - Calculates quartile boundaries (`Q1`, `Q3`) for distribution analysis

- `PROC FREQ` analyzes categorical variable distributions:
  - `ORDER=FREQ` sorts categories by frequency for immediate identification of dominant categories
  - `MISSING` option explicitly includes missing values in frequency counts
  - Focused on key categorical variables to understand data composition

This sequential validation approach demonstrates best practices in data quality assessment, providing both technical validation of the import process and initial exploratory insights into dataset characteristics.

3.3 Data Transformation and Feature Engineering

The script implements sophisticated data transformation techniques through custom formats and conditional processing:

```
* 3. Create User-Defined Formats ;
TITLE "Creating and Applying User-Defined Formats";
PROC FORMAT;
    VALUE $neigh_group_fmt
        'Bronx'         = 'Bronx'
        'Brooklyn'      = 'Brooklyn'
        'Manhattan'     = 'Manhattan'
        'Queens'        = 'Queens'
        'Staten Island' = 'Staten Island'
        OTHER           = 'Other/Unknown';

    VALUE $room_type_fmt
        'Entire home/apt' = 'Entire Home/Apt'
        'Private room'    = 'Private Room'
        'Shared room'     = 'Shared Room'
        OTHER             = 'Other/Unknown';

    VALUE review_fmt /* Numeric format for has_reviews */
        0 = 'No Reviews'
        1 = 'Has Reviews'
        . = 'Missing'
        OTHER = 'Error';
RUN;

* Apply formats in a new data step for clarity and further processing ;
DATA WORK.airbnb_processed;
    SET WORK.nyc_airbnb;

    FORMAT neighbourhood_group $neigh_group_fmt.
           room_type $room_type_fmt.
           has_reviews review_fmt.;

```

```
    * 4. Data Step Enhancements: Conditional Processing & New Variables ;
    * Create price_category ;
    IF price <= 75 THEN price_category = 'Budget';
    ELSE IF 75 < price <= 200 THEN price_category = 'Mid-Range';
    ELSE IF price > 200 THEN price_category = 'Premium';
    ELSE price_category = 'Unknown'; /* Should not happen if price is not
    missing */
    LABEL price_category = "Price Category";

    * Extract year_last_review (ensure last_review is handled correctly) ;
    * Using a more straightforward approach to extract year from last_review;
    year_last_review = .;
    IF NOT MISSING(last_review) THEN
        year_last_review = YEAR(INPUT(SUBSTR(last_review,1,10), B8601DA10.));
    LABEL year_last_review = "Year of Last Review";
    FORMAT year_last_review 4.;
RUN;

TITLE "Frequency of New Price Categories";
PROC FREQ DATA=WORK.airbnb_processed ORDER=FREQ;
    TABLES price_category;
RUN;

TITLE "Frequency of Year of Last Review";
PROC FREQ DATA=WORK.airbnb_processed ORDER=FREQ;
    TABLES year_last_review / MISSING; /* See distribution of review years */
RUN;
```

Listing 3: Format Definition and Variable Creation

Technical Implementation Analysis:

- `PROC FORMAT` creates custom data presentation formats:
  - Character formats (`$neigh_group_fmt`, `$room_type_fmt`) standardize text display
  - Numeric format (`review_fmt`) converts binary indicators to descriptive text
  - `OTHER` specifications provide defensive handling for unexpected values
  - Missing value specification (`.`) ensures proper handling of NULL values
- Feature engineering in the DATA step:
  - Format application enhances data interpretability while preserving original values
  - Conditional price categorization creates a derived business-relevant feature
  - Date string parsing extracts meaningful temporal information:
    - `NOT MISSING()` check prevents errors on null date values
    - `SUBSTR()` extracts the date portion of the string
    - `INPUT()` with `B8601DA10.` informat converts ISO-format date strings
    - `YEAR()` extracts only the year component for analysis
  - `LABEL` statements provide descriptive metadata for derived variables
  - `FORMAT` statements control display properties for numeric output
- Validation of transformations:
  - `PROC FREQ` provides immediate verification of derived variable distributions
  - `ORDER=FREQ` prioritizes most common categories for quick assessment

- `MISSING` option explicitly shows null values in the year variable

This transformation section demonstrates advanced SAS data manipulation techniques, combining format-based display enhancements with true data transformations to derive business-relevant features from raw data.

3.4 Strategic Data Subsetting for Targeted Analysis

The script implements targeted data subsetting to focus analysis on high-value segments:

```
* Create a subset: Manhattan listings with high availability ;
DATA WORK.manhattan_high_avail;
    SET WORK.airbnb_processed;
    WHERE neighbourhood_group = 'Manhattan' AND availability_365 > 180;
RUN;

TITLE "Manhattan Listings with Availability > 180 days";
PROC PRINT DATA=WORK.manhattan_high_avail (OBS=10);
    VAR name neighbourhood_group room_type price availability_365;
RUN;
PROC MEANS DATA=WORK.manhattan_high_avail NOPRINT;
    OUTPUT OUT=WORK.manhattan_stats (DROP=_TYPE_ _FREQ_) N=count_manhattan_high_
    avail;
RUN;
PROC PRINT DATA=WORK.manhattan_stats;
    TITLE "Count of Manhattan Listings with Availability > 180 days";
RUN;
```

Listing 4: Specialized Data Subset Creation

Technical Implementation Analysis:

- Subset creation leverages the `WHERE` clause for optimized filtering:
  - Multiple criteria combined with `AND` operator
  - Geographic filtering on `neighbourhood_group`
  - Availability threshold filtering (`>180` days) to identify high-availability listings
- Subset validation and exploration:
  - `PROC PRINT` with `OBS=10` provides a sample view of subset contents
  - Selected variables focus on key properties for this analysis
  - `PROC MEANS` with `NOPRINT` option suppresses detailed statistics
  - `OUTPUT` statement creates a derived dataset with only the count statistic
  - `DROP=_TYPE_ _FREQ_` removes automatically generated variables
  - Final `PROC PRINT` provides a clean report of the subset size

This subsetting approach demonstrates advanced analytical strategy by focusing on a specific market segment (Manhattan properties) with a particular characteristic (high availability) that may indicate investment opportunities or under-utilized inventory.

3.5 Advanced SQL-Based Aggregation Analysis

The script leverages PROC SQL for sophisticated aggregation and cross-tabulation:

```
* 5. PROC SQL Query ;
TITLE "Average Price and Count by Neighbourhood Group and Room Type";
PROC SQL;
    SELECT neighbourhood_group,
           room_type,
           COUNT(*) AS listing_count FORMAT=COMMA10.,
           AVG(price) AS average_price FORMAT=DOLLAR8.2
    FROM WORK.airbnb_processed
    GROUP BY neighbourhood_group, room_type
    ORDER BY neighbourhood_group, listing_count DESC;
QUIT;
```

Listing 5: SQL-Based Aggregation and Cross-Tabulation

Technical Implementation Analysis:

- SQL implementation provides enhanced analytical capabilities:
  - Multi-dimensional grouping with `GROUP BY` on two categorical variables
  - Aggregation functions `COUNT(*)` and `AVG(price)` generate summary statistics
  - Column aliases improve output readability
  - Format specifications enhance display (`FORMAT=COMMA10.` and `FORMAT=DOLLAR8.2`)
  - `ORDER BY` with multiple keys creates logical grouping by neighborhood with descending count

This SQL implementation demonstrates the complementary use of SQL syntax within SAS, leveraging SQL's concise syntax for multi-dimensional aggregation while maintaining SAS's output formatting capabilities.

3.6 Customized Reporting with PROC REPORT

The script implements sophisticated reporting capabilities:

```
* 6. PROC REPORT ;
TITLE "Listings Report for Brooklyn (Sample)";
PROC REPORT DATA=WORK.airbnb_processed(OBS=100) NOWD HEADSKIP;
    COLUMN name neighbourhood_group room_type price number_of_reviews
    availability_365;
    DEFINE name / DISPLAY "Listing Name" WIDTH=40;
    DEFINE neighbourhood_group / DISPLAY "Borough";
    DEFINE room_type / ORDER "Room Type";
    DEFINE price / ANALYSIS MEAN "Avg Price" FORMAT=DOLLAR8.2; /* Will show
    overall mean if no group */
    DEFINE number_of_reviews / ANALYSIS SUM "Total Reviews";
    DEFINE availability_365 / DISPLAY "Days Available";

    WHERE neighbourhood_group = 'Brooklyn'; /* Filter for Brooklyn */

    /* Example of breaking by room_type within Brooklyn and showing summary
    stats */
    BREAK AFTER room_type / SUMMARIZE STYLE=Header{font_weight=bold};
    RBREAK AFTER / SUMMARIZE STYLE=Header{font_weight=bold textalign=right}; /*
    Grand total summary */

    COMPUTE AFTER room_type;
        name = "Subtotal for " || room_type;
    ENDCOMP;
    COMPUTE AFTER;
```

```
        name = "Grand Total for Brooklyn";
    ENDCOMP;
RUN;
TITLE;
```

Listing 6: Customized Reporting Implementation

Technical Implementation Analysis:

- `PROC REPORT` configuration demonstrates advanced reporting techniques:
  - `NOWD` option prevents windowing display for batch processing
  - `HEADSKIP` adds space after column headers for readability
  - `COLUMN` statement defines report structure and variable order
  - `DEFINE` statements customize each column's behavior and appearance:
    - `DISPLAY` type for character data and direct numeric display
    - `ORDER` type for grouping variables (room_type)
    - `ANALYSIS` type with statistical functions (`MEAN`, `SUM`)
    - Custom headers with quoted strings
    - Width specification for text columns
    - Format specifications for numeric display
  - `BREAK` and `RBREAK` statements create summary rows:
    - `AFTER room_type` creates subtotals for each room type
    - `SUMMARIZE` generates summary statistics for analysis columns
    - `STYLE` options control the visual presentation of summary rows
  - `COMPUTE` blocks provide dynamic text for summary rows:
    - `COMPUTE AFTER room_type` generates subtotal labels
    - `COMPUTE AFTER` generates grand total labels
    - String concatenation with `||` creates dynamic text

This implementation demonstrates PROC REPORT's powerful capabilities for creating production-ready reports with complex grouping, summary statistics, and custom formatting.

3.7 Data Visualization with PROC SGPLOT

The script implements statistical graphics for visual data exploration:

```
* 7. PROC SGPLOT - Graphics ;
TITLE "Average Price by Neighbourhood Group (SGPLOT)";
PROC SGPLOT DATA=WORK.airbnb_processed;
    VBAR neighbourhood_group / RESPONSE=price STAT=MEAN
        DATALABEL DATALABELATTRS=(color=gray) GROUPDISPLAY=CLUSTER;
    YAXIS LABEL="Average Price ($)";
    XAXIS LABEL="Neighbourhood Group";
RUN;
TITLE;

TITLE "Price vs. Number of Reviews (SGPLOT)";
PROC SGPLOT DATA=WORK.airbnb_processed;
    SCATTER X=number_of_reviews Y=price / GROUP=neighbourhood_group TRANSPARENCY
    =0.5;
    YAXIS LABEL="Price ($)";
    XAXIS LABEL="Number of Reviews";
    KEYLEGEND / LOCATION=OUTSIDE POSITION=BOTTOM TITLE=""; /* Adjust legend */
```

```
    /* Add a regression line for overall trend */
    REG X=number_of_reviews Y=price / NOMARKERS LINEATTRS=(color=red thickness
    =2);
RUN;
TITLE;
```

Listing 7: Statistical Graphics Implementation

Technical Implementation Analysis:

- Bar chart implementation:
  - `VBAR` creates vertical bar chart of categorical variable
  - `RESPONSE=price` specifies the analysis variable
  - `STAT=MEAN` calculates average price by neighborhood
  - `DATALABEL` adds value labels to bars
  - `DATALABELATTRS` customizes label appearance
  - `GROUPDISPLAY=CLUSTER` specifies bar arrangement
  - Axis labels provide context for the visualization
- Scatter plot with regression line:
  - `SCATTER` creates point plot of two continuous variables
  - `X=` and `Y=` specify the analysis variables
  - `GROUP=neighbourhood_group` creates color-coded points by area
  - `TRANSPARENCY=0.5` reduces overplotting in dense regions
  - `KEYLEGEND` customizes the group legend placement
  - `REG` statement adds regression line to show overall relationship:
    - `NOMARKERS` suppresses regression data points
    - `LINEATTRS` customizes line appearance

These visualizations demonstrate effective data communication techniques, with the bar chart providing categorical comparisons and the scatter plot revealing relationships between continuous variables while controlling for a categorical factor.

3.8 Statistical Analysis with PROC REG

The script implements regression analysis for relationship modeling:

```
* 8. PROC REG - Simple Linear Regression ;
TITLE "Simple Linear Regression: Price vs. Number of Reviews";
PROC REG DATA=WORK.airbnb_processed PLOTS(MAXPOINTS=NONE);
    MODEL price = number_of_reviews;
    /* Add more predictors here if desired, e.g., number_of_reviews minimum_
    nights */
    /* PLOT R.*P.; */ /* Example for residual plots */
RUN;
QUIT;
```

Listing 8: Linear Regression Implementation

Technical Implementation Analysis:

- Regression analysis configuration:
  - `PLOTS(MAXPOINTS=NONE)` disables point limiting for comprehensive visualization

- MODEL statement specifies dependent and independent variables:
  - price as the dependent (response) variable
  - number_of_reviews as the independent (predictor) variable
- Commented options indicate potential enhancements:
  - Additional predictors could extend the model
  - Residual plots could evaluate model fit and assumptions

This regression analysis provides quantitative assessment of the relationship between property price and review count, potentially revealing insights about pricing dynamics and market perception.

3.9 Machine Learning Implementation with PROC LOGISTIC

The script implements logistic regression for binary classification:

```
* 9. PROC LOGISTIC - Predicting 'has_reviews' (SAS ML Example) ;
TITLE "Logistic Regression: Predicting if a Listing Has Reviews";
PROC LOGISTIC DATA=WORK.airbnb_processed DESCENDING; /* DESCENDING to model P(
    has_reviews=1) */
     CLASS neighbourhood_group room_type / PARAM=REF REF=FIRST;
     MODEL has_reviews = price minimum_nights calculated_host_listings_count
    availability_365
                         neighbourhood_group room_type / SELECTION=FORWARD LINK=
    LOGIT TECHNIQUE=FISHER RSQUARE;
     /* SELECTION=FORWARD for variable selection example */
     /* TECHNIQUE=FISHER can be more stable for some datasets */
     /* RSQUARE for pseudo R-squared measures */
RUN;
QUIT;
```

Listing 9: Logistic Regression Implementation

Technical Implementation Analysis:

- Logistic regression configuration demonstrates advanced modeling techniques:
  - DESCENDING option models probability of has_reviews=1 (SAS defaults to modeling the lower value)
  - CLASS statement handles categorical predictors:
    - PARAM=REF specifies reference parameterization for categorical variables
    - REF=FIRST uses the first level as the reference category
  - MODEL statement specifies multiple predictors:
    - Numeric predictors: price, minimum_nights, calculated_host_listings_count, availability_365
    - Categorical predictors: neighbourhood_group, room_type
  - Advanced modeling options:
    - SELECTION=FORWARD implements automated variable selection
    - LINK=LOGIT specifies the logistic function as the link function
    - TECHNIQUE=FISHER uses Fisher's scoring method for optimization
    - RSQUARE requests pseudo-R^2 statistics to assess model fit

This machine learning implementation demonstrates advanced predictive modeling, using logistic regression to identify factors that influence whether a listing receives reviews. The model combines continuous and categorical predictors with automated variable selection to create an optimized prediction model.

4 Conclusion and Technical Significance

This `SAS` script exemplifies sophisticated SAS programming techniques across multiple domains of data processing and analysis. Key technical achievements include:

- **Comprehensive Data Pipeline**: Implements a complete analytical workflow from raw data import through advanced predictive modeling
- **Efficient Data Handling**: Demonstrates optimized techniques for processing complex text-based datasets
- **Advanced Data Transformation**: Implements sophisticated feature engineering for business-relevant derived variables
- **Multi-faceted Analysis**: Combines descriptive, exploratory, and predictive techniques for comprehensive insights
- **Production-Ready Reporting**: Creates formatted, summarized outputs suitable for business consumption
- **Statistical Modeling**: Implements both exploratory and confirmatory statistical techniques

The implementation satisfies all required SAS programming functionalities while maintaining a coherent analytical narrative focused on understanding and extracting value from the NYC Airbnb market data. The technical approaches demonstrated in this script establish a foundation for further advanced analytics in both the SAS and Python components of the NYCBnB Analytics project.