MuckRock: US Military Base Slot Machine Revenue Explorer

By
Team B
Cameron Moore (CDS'25 jmijares@bu.edu)
Ching Hsuan Lin (QST'25 sjmanua@bu.edu)
Jyun-Ru Huang (QST'25 jhuangbp@bu.edu)
Nithya Priya Jayakumar (CDS'25 hannahsh@bu.edu)
Pin-Hao Pan (QST'25 pp0126@bu.edu)
Quoc Dat Nguyen (CDS'25 hannahsh@bu.edu)

Analysis Paper

MuckRock: US Military Base Slot Machine Revenue Explorer

Table of Contents

Introduction

Project Objective---2

Data Cleaning, and Processing-- 2

1) Overview--7
2) Asset Report - 2020, 2023, 2024---7
3) Asset Report - 2021---7
4) Asset Report - 2022---7
5) Marine Corps Revenue---7
6) Navy Revenue---7
7) District Revenues--- 7
8) Financial Statements--7

Preliminary Analysis-- 3

1) Geographic Distribution--- 7
2) Service Branch Summary---7
3) Top 10 Revenue-Generating Bases---7
4) Temporal Trends--- 7
5) Financial Trends-- 7
6) Revenue Efficiency (Per Machine)---7

Base Questions---7

1) What is the total and per-base slot machine revenue by military branch and by region?------------- 7
2) Which bases generate the highest total revenue, and how does that rank change by year and by branch?--- 7
3) How have the types of games or machines installed at bases changed over time?---------------------7

Datasette Dashboard Visualization---8

1) Datasette UI Deployment Process--- 8
2) Slot Machine Revenue Dashboard---8

Conclusion--- 10

Individual Contribution--10

Introduction

The **MuckRock US Military Base Slot Machine Revenue Explorer** project is a data liberation initiative in partnership with MuckRock, a nonprofit news organization that helps journalists, researchers, and the public request, analyze, and share government information.

The project centers on the **Army Recreation Machine Program (ARMP)**, an obscure initiative that operates approximately **1,889 slot machines across 79 overseas US military bases**. In fiscal year 2024 alone, these machines generated **$70.9 million in net revenue**, almost entirely from enlisted service members and their families.

Despite the program's significant scale and financial impact, there has been virtually no public transparency about the gambling addiction risks it poses to military personnel. To expose this issue, MuckRock obtained official ARMP records from the military through the Freedom of Information Act. These documents include financial statements, revenue reports, and asset inventories. However, due to the sensitive nature of military data, all records were released only as unstructured PDFs with inconsistent layouts, tiny fonts, merged cells, and damaged text, making the original data effectively inaccessible.

In collaboration with MuckRock, this project's core mission is to convert these opaque and unusable records into a clean, standardized, and publicly accessible dataset. Through reproducible extraction pipelines and in-depth analysis, the initiative reveals the true scope of military gambling operations, identifies the highest-risk installations, and supports calls for greater oversight and more responsible recreation practices for service members.

Project Objective

The project has two major goals: data engineering (extracting and cleaning data from messy raw PDF files) and analytical insights (exploratory data analysis and visualization). The detailed objectives are as follows:

- **Data Extraction and Cleaning**: Develop automated, reproducible Python pipelines to extract tabular data from 14 highly unstructured, multi-format, and complex PDF reports and transform them into clean, standardized, analysis-ready CSV files.
- **Cloud Deployment**: Build and deploy a cloud-based Datasette instance, a powerful SQLite-powered interactive platform that enables journalists, researchers, and the public to easily query, browse, filter, and visualize the liberated data.
- **Gambling Risk Analysis**: Conduct in-depth analysis to reveal revenue distribution across branches, regions, and individual bases, identify temporal trends (including COVID-19 impact and recovery patterns), highlight seasonal peaks, and pinpoint the

specific installations that generate the highest slot machine revenue and therefore represent the greatest potential risk for gambling-related harm among service members.

- **Documentation and Reproducibility**: Thoroughly document every dataset, column definition, extraction decision, cleaning step, and analytical finding so that future teams, journalists, or researchers can seamlessly understand, validate, reproduce, and extend the work. The final report serves as the comprehensive synthesis of the entire project.

Data Collection, Cleaning, and Processing

1) Overview

The main technical challenge of this project was converting "human readable" PDF reports into structured data that a computer can process. The source documents provided by Muckrock did not follow a single consistent format and often contained problems such as very small fonts, complex table layouts, merged cells, footnotes inside tables, and changes introduced during the COVID period. Because of these issues, the extraction process required the use of multiple tools and a significant amount of custom code. The following table presents the specific strategies applied to each document:

Report Type	Fiscal Years	Primary Tools Used	Key Challenges & Solutions	Main Output Files
Asset Reports	2020–2024	pdfplumber, pdftotext -layout, Camelot/Tabula (FY2021)	8 different table layouts in each report required a dedicated parser for each layout. COVID-specific categories appeared only in FY2021, so custom logic was needed to handle them.	assets_by_region_service.csv, asset_details.csv, floor_asset_details.csv, site_operational_status.csv, etc.
Marine Corps Revenue	2020–2024	pdfplumber (custom page-by-page with tunable tolerances)	Misaligned columns and tables spanning multiple pages were addressed through automated table detection with adjustable parameters. Tables that spanned multiple pages required custom rules.	Marine_Revenue_FY20-FY24_summary table.csv, Marine_Revenue_FY20-FY24_detail.csv

Navy Revenue	2016–2024	pdftotext -layout, state-machine regex parser	Extremely small fonts (size one) were processed using a state-machine approach combined with regex to track line-by-line context.	Navy_Revenue_Table.csv, Navy_Monthly_Revenue_Report.csv
District Revenues	2020–2024	PyMuPDF (coordinate-based block extraction), state-machine regex parser	Multi-line installation names and inconsistent fiscal-year headers were handled through coordinate-based filtering with a regex state machine.	District_Revenue_filtered_FY20-FY24_final.csv
Financial Statements	2020–2024	poppler-utils + Adobe Express OCR (chunked)	Large file sizes and low scan quality were managed by splitting documents into smaller sections, processing them with Adobe Express OCR, and then recombining the results.	FinancialStatement.csv
Revenue Comparison	2023–2024	Manual extraction via Adobe Express	Pages containing multiple unrelated tables and visual noise required manual extraction of relevant sections using Adobe Express, followed by consolidation in Excel.	Revenue Comparison_C.xlsx

2) Asset Report - 2020, 2023, 2024

a) Overview of the Asset Report Dataset

The Asset Report processing pipeline handles a series of PDF documents (FY2020, FY2023, FY2024) containing inventory and operational status data. Unlike the Marine report which used geometric parsing, this pipeline relies on pdftotext with the -layout flag to preserve physical column spacing as whitespace-separated text. The script processes the text stream to extract three distinct levels of data:

- High-Level Summaries: Aggregated counts by Region and Field Office.
- Installed Assets Matrix: A wide-format table breaking down machine counts by manufacturer (IGT, WMS, etc.).
- Detailed Inventory: Granular, row-by-row listings of individual assets, their lifecycle dates, and storage status.

b) Data Dictionary

The pipeline generates up to 8 CSV files per year in the output directory:

Summary Tables

File Name	**Description**
assets_by_region_service.csv	Aggregated machine counts grouped by Region (e.g., Europe, Japan) and Service branch (Army, Navy, etc.).
assets_by_field_office.csv	Counts grouped by Field Office (FO#) and Location name. Breakdown includes Slots, ACM, and ITC.
installed_assets_location_manufacture.csv	A matrix showing the number of machines per location by manufacturer (NOV, AIN, IGT, WMS, BAL, KON, ITE).

Detailed Inventory Tables

File Name	**Description**
asset_details.csv	**The primary inventory list. Contains Region, FO#, Asset ID, Class, Description, Serial Number, and key lifecycle dates (Acquire, Effective, Disposed).**
floor_asset_details.csv	**Specific extraction for sections labeled "Floor". Captures Asset ID, Machine Type, Manufacture Class, and Age.**
floor_summary_details.csv	**Captures summary lines embedded within the Floor sections, containing aggregate counts (CountA, CountB) and site codes.**
site_operational_status.csv	**Tracks the operational timeline of locations, including Open Date, Closed Date, Community Number (CmtyNum), and Service.**

Derived Analysis

File Name	**Description**
years_in_storage.csv	**A generated pivot table that cross-tabulates Asset Age (rows) against Years in Storage (columns) to visualize inventory aging.**

c) Processing Logic

Text Layout Extraction

The script strictly requires pdftotext (from the poppler-utils package). It executes a system command pdftotext -layout <PDF> - to stream text directly into Python. This method is chosen to convert visual table columns into predictable whitespace patterns, which are then split using the regex \s{2,} (two or more spaces).

Section Detection and Parsing

The code iterates through the document page by page, using specific keywords to trigger different parsing logic:

- **Region/Field Office Parsing:**
  Detects headers like "ASSETS BY REGION". It parses lines by splitting on multiple spaces. Special handling is applied to numeric fields to strip percent signs and convert parenthesized numbers (e.g., (5)) into negative values.
- **Installed Assets (Matrix) Parsing:**
  - Identifies the "Installed Assets by Location" header.
  - The parser dynamically identifies the Region header (e.g., "Europe") and applies it to subsequent rows.
  - It handles the variable number of manufacturer columns by checking if tokens are numeric. It includes logic to calculate a Total_Computed field (sum of columns) to validate against the printed Total_PDF.
- **Asset Details Parsing (State-Machine):**
  - This parser handles the most irregular data. It identifies rows starting with a valid Region.
  - Date Anchoring: Because the "Description" field varies in word count, the parser looks for date patterns (MM/DD/YYYY) or numeric strings to anchor the end of the row. Text preceding the dates is reconstructed as the Description/Type.
  - Serial Number logic: It uses a regex (^[a-z0-9-]+$) to identify serial numbers appearing immediately after dates or at the end of the description block.
- **Floor Section Logic:**
  - The code separates "Floor" pages from standard pages.
  - It distinguishes between a Detail Row (individual machine) and a Summary Row (location totals) by checking if the second column contains a known region name. Summary rows are diverted to floor_summary_details.csv, while machine rows go to floor_asset_details.csv.
- **Years in Storage Logic:**

After extracting the detailed asset list, the script casts "Age" and "Years_in_Storage" columns to integers. It then uses pandas pivot_table to generate a frequency matrix, helping analysts identify aging equipment concentrations.

d) Output Quality and Limitations

- **Date Sensitivity:** The parsing relies heavily on DATE_RE (regex for dates). Lines with malformed dates or dates merged with other text due to kerning issues may fail to parse correctly or be skipped.
- **Column Alignment:** The logic relies on \s{2,} to separate columns. If the PDF generation places columns too close together (single space), the parser may merge two columns (e.g., merging "IGT" and "WMS" counts), though the script attempts to mitigate this by strictly typing numeric tokens.
- **Calculated vs. Printed Totals:** In the "Installed Assets" table, the script captures both the printed total from the PDF (Total_PDF) and calculates a sum of the extracted parts (Total_Computed). Analysts should compare these columns to verify extraction accuracy.

3) Asset Report - 2021

a) Overview of the Dataset

For FY2021, the Asset Report extraction pipeline is the **same core design** as for 2020 / 2023 / 2024: same PDF → text strategy, same family of tables, same overall CSV outputs. Here's the twist - some tables, especially Site Operational Status & Asset Details - carry **extra COVID-related cells and new fields.** The 2021 notebook (FY2021_Asset Report_Extraction.ipynb) is basically the "pandemic-focused" version of the standard pipeline.

b) Data Dictionary

The 2021 extractor writes the same logical tables as the other years, just with year-specific filenames:

- assets_by_region_service_FY2021.csv
- assets_by_field_office_FY2021.csv
- installed_assets_location_manufacture_FY2021.csv
- asset_details_FY2021.csv
- floor_asset_details_FY2021.csv
- site_operational_status_FY2021.csv
- years_in_storage_FY2021.csv

So structurally it aligns with the multi-year data dictionary you already defined, with **two main differences:**

1. **site_operational_status_FY2021.csv** includes **COVID-specific** and SMS-related columns:
    - MESSAGE, RptGrp, Shortname, Split
    - COVID19, TVLREST, CMMTY
    - SMSSection, SMSBank
    - ParseOK (flag showing whether the FOSHORT parse succeeded)
2. **asset_details_FY2021.csv** includes extra derived and geo columns:
    - Age_int, Years_in_Storage_float, Years_in_Storage_int
    - BaseName_match, match_score, match_country, Latitude, Longitude (from bases CSV via function *attach_coordinates_from_bases*)

All the other core fields (Region, FO#, location, counts, lifecycle dates, etc.) are the same as in the 2020/2023/2024 definitions.

c) **Processing Logic**

1 Text Layout Extraction: PDF -> Text with layout preserved using pdftotext-layout on the full 2021 Asset Report file; splitting on form-feed (*\f*) into pages; using **whitespace patterns** and **regexes** to reconstruct columns.

2: Section Detection and Parsing - the script walks pages and dispatches to different parsers based on header keywords and detected month contex:

a. **Month detection & page-to-month map -** scans pages for key headers like "Assets by Region, Service for month of <Month Year>", "Assets by Field Office" or "Installed Assets by Location". Then it builds a month_map from page index → <Month Year>. All parsed rows receive the correct Month value via this mapping. Extracted rows are validated, cleaned and safely merged using *safe_concat()*.

Assets by Region, Service
for month of October 2020

Figure 1: header with specific month & year

b. **Region/Field Office Parsing -** uses parsing with specific key headers like "Assets by Region", "Assets by Field Office", "Installed Assets"; skips header lines, then splits each row on 2 or more spaces to separate columns; normalizes numeric fields by stripping percent signs and converts parenthesized negatives. For the region summary and field office it builds specific rows related to them. Finally it adds Month before concatenating into the final region and field CSVs.

c. **Installed Assets Parsing** - pages containing "Installed Assets by Location" are fed into an installed-assets parser; detects the **Region header** on the page and applies it to all rows until the next region header; splits each data line on multi-space gaps by extracting location identifiers plus manufacturer counts; computes row-wise total as the sum of the manufacturer columns; captures total_pdf from the text so analysts can compare the printed total to the computed total. Finally it adds Month and writes to the year 2021 CSV file *installed_assets_location_manufacture*.

d. **Asset Details Parsing -** pages belonging to the detailed asset list are parsed line-by-line using a state-machine style parser centered on date patterns (MM/DD/YYYY). Left Side: identifies core identifiers like **Region, FONUM, FOSHORT, Loc, LNAME, and Asset;** Middle Section: uses the location of the dates to find and reassemble **Class, Desc, and Type (**located between "class code" and the first date**)**; Date Fields: locates the Acquire date and Effective date, and potentially Disposed data. Right Side: extracts the **Serial Name** (using a regex) and picks up **Age** and **Years_in_Storage** at the far right. Overall the function applies this line-by-line parsing and then attaches the relevant **Month** to the collected data. After parsing all pages, it writes into the Asset Details CSV file.

e. **Floor Section -** floor pages are identified by header patterns and **"white 'Place' floor"** formatting. Firstly it identifies valid lines by checking for **numeric Loc and floor Place** in the initial columns. The rest of the line is then split into various asset attributes. Now a key part of the process is to **distinguish individual asset rows** from embedded **summary lines** by analyzing specific patterns. Finally, it attaches the Month and writes all the parsed data to the output file.

f. **Site Operational Status Logic (COVID-era Enhancements) -** pages containing site status by grouping lines that share as leading **numeric Loc** into a single logical row. For each row, a basic parser extracts primary operational data, specifically containing the FOSHORT field, which had to be decoded with two helper functions: one of them, split_tail() interprets the "tail" string based on the date, as the format changed in May 2021:

    i. Standardizes punctuation and find the Split percentage
    ii. Old Format (before May 2021): it specifically identifies and assigns the **TVLREST** and **COVID19** field
    iii. New Format (after May 2021): it extracts and assigns the CMMTY field
    iv. Breaks the left part of the tail into MESSAGE and Shortname

All the parsed and decoded components are merged and written into Site Operational Status CSV file Year 2021. ParseOK flags whether FOSHORT parsing went successfully. Finally all rows retain Month so that temporal changes in site status during COVID can be analyzed.

g. **Years in Storage Logic -** process starts with Asset Details CSV file and focuses only on rows where the Age and Years in Storage are numeric. For each Month field, we build a pandas pivot table with index Asset Age, columns of Years in Storage and values; concatenates monthly pivots into a single table with columns of storages. Finally it writes this into Years in Storage in CSV file Year 2021.
h. **Base Matching Enrichment** - the notebook optionally loads an external bases CSV containing the name of the base with latitude and longitude. For each unique site identified PDF file, it normalizes the name to match with the base name. These fields are written into asset details CSV file Year 2021, with the underlying parsing of the PDF being the same as other years.

d) Output Quality & Limitations

The FY2021 extraction pipeline inherits the same general structure as the 2020/2023/2024 process with some extra specific reformations:

- **Data sensitivity:** asset detail parsing is based on date patterns. Any line where dates are distorted or visually merged (ex. broken PDF text) can fail to parse or be skipped.
- **Whitespace/column alignment:** the parser depends on 2+ spaces as column separators. If the FY2021 PDF compresses certain columns into single space, the script may merge adjacent fields. Numeric typing and checks help catch some of these issues.
- **FOSHORT parsing & COVID-era tails:** the FOSHORT field in Site Operational Status file is packed with COVID-related codes: so the parser supports both old and new tail formats, with unexpected wording or abbreviations can cause partial parses. ParseOK flag should be used to filter rows where COVID fields are reliable.
- **Geocoding/fuzzy matching:** coordinate fields in Asset Details File depend on string similarity- typos or abbreviations in site names can lead to lower match scores. These fields should be treated as best effort matching rather than real truth.

4) Asset Report - 2022

a) Overview of the FY2022 Asset Report Dataset

The FY2022 Asset Report pipeline processes FY2022 Asset Reports.pdf and converts it into a set of structured CSVs. As with the other Asset Report years, it uses pdftotext -layout to convert the PDF into a whitespace-aligned text stream, and then applies section-specific parsers.

The FY2022 script is implemented as a single master Python script that calls seven parsers, each targeting a specific table or section in the report:

- EGMs by Region, Service
- EGMs by Field Office
- Installed Assets by Location, Manufacture (wide manufacturer matrix)
- Asset Details (Installed Assets by Location – row-level inventory)
- Years in Storage (EGMs Only) grid
- Site Operational Status
- Floor Asset Details

b) Data Dictionary

assets_by_region_service_FY2022.csv

Aggregated EGM counts grouped by region and service for each month.

Column	Description
Region	Europe, Japan, Korea, Total, or special labels like "Locations by Service", "% by Service".
#Location	Number of locations in the region (blank for summary rows).
Army	Count of EGMs assigned to the Army.
Navy	Count of EGMs assigned to the Navy.
Marine_Corps	Count of EGMs assigned to the Marine Corps.
Airforce	Count of EGMs assigned to the Air Force.
Total	Total EGMs for the row.
Percent	Percent of total, when printed (blank otherwise).
Month	Reporting month in MMM-YY format (e.g., Oct-21, Feb-22).

assets_by_field_office_FY2022.csv

EGM counts per Field Office (FO) within each region, including regional and ARMP totals.

Column	Description
Region	Europe, Japan, Korea, or "ARMP" for the ARMP total row.
FO#	Field Office number (blank for region and ARMP totals).
FOSHORT	FO short name, or labels like "Europe Total", "%", "ARMP Total".
Slots	Slot machine counts.
ACM_CountR	ACM-related count.
ITC	ITC machine counts.
FRS	FRS machine counts.
Total	Total EGMs for that FO or total row.
Month	Reporting month in MMM-YY format.

installed_assets_by_location_manufacturer_FY2022.csv
Wide-format table showing manufacturer counts and totals for each site and subtotal.

Column	Description
region	Region name for the block (Europe, Japan, Korea); NaN for ARMP total rows.
group_location	Group label / FO short name (FOSHORT) for the site or subtotal group.
site_name	Site name (e.g., "Club"), or synthetic values like "Subtotal" or "ARMP Total".
FO #	Field Office number (float); NaN for subtotal and ARMP rows.
Loc	Location number (float) for the site; NaN for pure subtotals.

Svc	Service branch at the site (Army, Navy, Air Force, Marine Corps, Airforce).
NOV, AIN, IGT, WMS, BAL, KON, ITE	Counts of EGMs by manufacturer.
Tot/EGMs	Total EGMs at the site or subtotal row.
FRS, ACM, ITC	Program counts derived from the Installed Assets table.
Total	Overall total (should match Tot/EGMs or the printed total column).
Month	Reporting month in MMM-YY format.

asset_details_FY2022.csv
The detailed asset inventory: one row per asset, with lifecycle dates and storage information.

Column	**Description**
REGION	Region (Europe, Japan, Korea).
FONUM	Field Office number.
FOSHORT	Field Office short name (may be empty on some rows).
Loc	Location number.
LNAME	Location name.
Asset	Asset ID/code.
Class	Class code.
Desc	Asset description text.
Type	Type code (3–5 digit numeric).
Aquire	Acquisition date (MM/DD/YYYY or converted Excel serial).

Effective	Effective date (MM/DD/YYYY or converted Excel serial).
SerialNum	Asset serial number (text with alphanumerics and dashes).
PLACE	Place code.
Age	Asset age (often numeric, in years).
Years in Storage	Number of years in storage (string/numeric representation).
Months	Additional months in storage.
Month	Reporting month in MMM-YY format, based on the header date in the asset-list section.

years_in_storage_FY2022.csv
Direct capture of the printed "Years in Storage (EGMs Only)" grid.

Column	Description
EGM Age	Age bucket (0–25), plus special rows: Totals_By, Mini_Header, Mini_Values.
0–14	Fifteen columns representing years-in-storage buckets (fixed-width grid parsed from the PDF).
Total by Age	Total number of EGMs for that age bucket (row sum).
Month	Reporting month in MMM-YY format.

site_operational_status_FY2022.csv
Operational status information for each location.

Column	**Description**
Loc	Location number (3–5 digits, optional letter).
LNAME	Location name.
PLACE	Place code.
Open	Open date (normalized to MM/DD/YYYY).

Closed	Closed date, if present.
KSI	KSI value.
CmtyNum	Community number.
SVC	Service branch.
FONUM	Field Office number.
FOSHORT	Field Office short name.
REGNUM	Region number.
Region	Region name (Europe, Japan, Korea).
SMSSection / SMSBank / Split / CMMTY / MESSAGE	SMS or message-related fields, when present in the header.
Month	Reporting month in MMM-YY format.

floor_asset_details_FY2022.csv
Machine-level inventory for "Floor" sections.

Column	**Description**
Loc	Location number.
Place	Place code.
Region	Region name.
SVC	Service branch (including "Marine Corps" as a single value).
Asset	Asset ID/code.
SerialNum	Machine serial number.
Type	Machine type code.
Desc	Asset description.

Acquire	Acquisition date.
Effective	Effective date.
Disposed	Disposed date, if present.
Class	Class number.
MFG	Manufacturer.
LNAME	Location name.
FONUM	Field Office number.
FOSHORT	Field Office short name.
Cat	Category field from the floor table.
Year	Year attribute on the floor line (e.g., install year).
Age	Machine age.
Month	Reporting month in MMM-YY format.

c) Processing Logic

For the FY2022 Asset Report, all parsers share the same underlying design principles:

- PDF → Text via pdftotext -layout so that columns remain aligned as whitespace.
- Section detection via headers (e.g., "EGMs by Field Office", "Installed Assets by Location").
- Month context: most sections are tagged with a Month field in MMM-YY form (e.g., Oct-21, Sep-22), derived from header phrases like "for month of October 2021" and propagated to all rows in that block.
- Defensive parsing & deduplication: the report often repeats monthly sections; parsers track seen_months or (Month, Region) keys and skip exact repeats to avoid duplicate rows.

Logically, the pipeline extracts three layers of data. First, summary tables: “EGMs by Region, Service” and “EGMs by Field Office” are parsed into region-level and field-office-level counts by service branch (Army, Navy, Marine Corps, Air Force), producing assets_by_region_service_FY2022.csv and assets_by_field_office_FY2022.csv. Second, the installed-assets structure: the “Installed

Assets by Location, Manufacture" section is split into a wide manufacturer matrix (installed_assets_by_location_manufacturer_FY2022.csv) and a detailed, row-level asset list (asset_details_FY2022.csv). The matrix parser reconstructs counts by manufacturer (NOV, AIN, IGT, WMS, BAL, KON, ITE) for each site and subtotal, while the asset-details parser uses date patterns and numeric type codes to recover variable-length text fields (Class, Desc, Type) and lifecycle dates (Acquire, Effective), including conversion of Excel-serial dates when necessary.

Finally, the script handles specialized operational views. The "Years in Storage (EGMs Only)" grid is parsed directly from its printed matrix on selected pages, using fixed column offsets to build an age-by-years-in-storage table (years_in_storage_FY2022.csv) instead of computing it from pivots. The "Site Operational Status" section is parsed with position-based column slicing to reconstruct location identifiers, open/closed dates, region/SMS metadata, and status fields into site_operational_status_FY2022.csv, with multi-line rows merged into single logical records. A separate parser scans "Floor" pages to extract detailed floor-level machine rows (asset IDs, serials, manufacturer class, age) into floor_asset_details_FY2022.csv, keyed by location and month.

Together, these outputs provide a consistent, month-resolved view of FY2022 assets that aligns with the structure used for other Asset Report years.

d) Output Quality and Limitations

Dependence on pdftotext Layout:
All parsers assume that columns are separated by stable whitespace. If the FY2022 PDF compresses or misaligns columns more severely than other years, some fields may merge or split incorrectly. The code mitigates this with:

- Multi-space regex splits.
- Type-aware parsing (numeric vs. text).
- Custom regexes for known patterns.

Month Deduplication:
Several parsers rely on seen_months or (Month, Region) sets to skip duplicate reprints of the same tables. If a "true" second version of a month appears with corrections but shares the same header text, only the first instance will be kept.

Irregular Lines Skipped Silently:
The asset detail parser (extract_asset_details) intentionally wraps row parsing in try/except and skips malformed lines without raising. This avoids crashes but means a small number of badly formatted rows will be omitted.

Subtotals vs. Details:
In the installed assets matrix and floor sections, subtotal and ARMP rows are included as rows with special site_name or group_location values (e.g., "Subtotal", "ARMP Total").

5) Marine Corps Revenue

Overview of the Marine Revenue Report Dataset

The Marine revenue report processing pipeline extracts data from Marine_Revenue_FY20-FY24.pdf. Unlike the text-stream approach used for the Navy dataset, this pipeline utilizes a geometric, coordinate-based extraction method (pdfplumber) to handle the complex, dense columnar layouts found in the Marine reports. The pipeline produces two distinct output files:

- **Marine_Revenue_FY20-FY24_summary_table.csv**: A high-level annual summary of slot revenue extracted from specific overview pages (Pages 1, 35, 73, 115, 157).
- **Marine_Revenue_FY20-FY24_detail.csv**: A detailed, monthly transaction-level panel for every machine location, extracted from the bulk of the document (Pages 2–202).

Data Dictionary

Marine_Revenue_FY20-FY24_summary_table.csv

Column	Description
Country	The country where the Marine installation is located (e.g., Japan, Korea).
Installation	The specific base or camp (e.g., Camp Fuji, Camp Hansen).
FY16-FY19	Annual revenue totals for the respective fiscal years.
FY20-FY24	Annual revenue totals for the respective fiscal years.
Annualized FY20	The annualized revenue projection based on the YTD figures.

Marine_Revenue_FY20-FY24_detail.csv

Column	Description

Loc #	4-digit location identifier.
Location	Human-readable name of the venue (e.g., "BeachHead", "Tavern").
Month	The reporting month (MMM-YY format).
Revenue	Monthly slot machine revenue.
NAFI Amt	Monthly Non-Appropriated Fund Instrumentality reimbursement amount.
Annual Revenue	The printed fiscal year total for revenue (usually appearing on the September row).
Annual NAFI	The printed fiscal year total for NAFI reimbursement.

Processing Logic

The Marine extraction code is split into two distinct routines: one for the summary table and one for the detailed monthly tables.

Table Parsing Logic

- **Targeted Page Extraction:** The script targets specific pages known to contain the "Slot Revenue" summary tables (1-based indices: 1, 35, 73, 115, 157).
- **Header-Based Cut Detection:** The parser detects the header row containing "Country", "Installation", and "FY". It calculates vertical "cut" lines based on the horizontal center of these header keywords.
- **Merged Text separation:** A specific function (fix_country_installation) handles a common OCR/layout issue where the Country and Installation columns merge (e.g., "JapanCampFuji"). It uses a dictionary of known mappings and regex patterns to split these strings back into distinct columns.

Detail Table Parsing Logic (Geometric & Adaptive)

- **Adaptive Page Parameters:** The code utilizes a SPECIAL_PARAMS dictionary to apply different extraction tolerances based on page ranges.
    - *Standard Pages:* Use base tolerances for vertical merging and horizontal joining.
    - *Dense Pages (e.g., 178–190):* Tighter spacing requires larger "Edge Padding" and specific "Left Shifts" for column boundaries to prevent data from being clipped into the wrong column.

- **Type-Aware Column Refinement:**
    - **Initial Cuts:** Column boundaries are first estimated based on the position of the headers (Loc #, Location, Month, Revenue, etc.).
    - **Refinement:** The script scans the actual data rows. It detects the edges of "Month" text tokens and "Money" numeric tokens. It then dynamically adjusts the cut lines to ensure they sit exactly in the whitespace between columns, preventing numbers from being split or assigned to the wrong field.
- **Row Construction & Repair:**
    - **Binning:** Text tokens are assigned to columns based on the refined cut lines.
    - **Loc/Location Repair:** The script fixes multi-line rows where the Location name wraps. It also handles cases where the "Loc #" and "Location" fields are merged. Logic is applied to append floating 6-digit site codes to the previous row's Location field.
    - **Month/Revenue Swapping:** A heuristic check identifies rows where the Month and Revenue columns are misaligned (swapped) or merged into a single cell, using regex to separate date formats from currency formats.
- **Data Imputation**
    - The code identifies September rows (end of Fiscal Year). If the PDF does not explicitly print the "Annual Revenue" or "Annual NAFI" on this row, the script calculates these values by summing the monthly "Revenue" and "NAFI Amt" figures for the preceding 12 months relative to that location.
- **Continuation Handling:**
    - The PDF often omits the "Loc #" and "Location" for subsequent months within the same block. The script uses a "Seeded Forward Fill" to propagate location metadata down to these orphan rows and across page breaks to ensure every row has a valid identifier.

Output Quality and Limitations

- **FY Row Segregation:** The detailed extraction logic detects rows containing "FY" (Fiscal Year) totals that are interleaved with monthly data. These are separated into a specific `_FY_rows.csv` output to prevent them from corrupting time-series analyses performed on the main detail file.
- **Imputed vs. Printed Totals:** Users should be aware that some "Annual" values in the detail file are calculated sums (imputed) rather than OCR'd values. This is necessary because the PDF layout is inconsistent in printing these totals.
- **Parameter Sensitivity:** The extraction relies on hard-coded page ranges for parameter tuning (e.g., `range(178, 191)`). If the pagination of the source PDF

changes significantly in future years, these ranges will need to be manually updated to maintain extraction accuracy.

4) Navy Revenue

a) Overview of the Navy Revenue Report Dataset

The Navy revenue report is delivered as a 138 pages PDF that combines several different table layouts:

- Navy_Revenue_Reimburse_Summary.csv: annual slot-machine revenue and NAFI reimbursement at the installation × country level.
- Navy Revenue Report FY20-FY24-2_monthly_summary_master.csv: a monthly, location-level panel showing how revenue and reimbursements evolve over time within each installation.

Both CSVs are generated from the same Python scripts which use pdftotext -layout plus regular expressions and small "state-machine" parser to reconstruct the tables.

b) Data Dictionary

Navy_Revenue_Reimburse_Summary.csv

Column	**Description**
Country	Country where the Navy installation is located
Installation	Name of the Navy base or facility
Category	Type of amount: Slot Revenue – gross slot-machine revenue. NAFI Reimbursement – Non-Appropriated Fund Instrumentality reimbursements from ARMP.
FY16-FY24	Annual totals in USD for each fiscal year
FY23 thru SEP	Year-to-date FY23 amount through September
ANNUALIZED FY23	FY23 amount annualized from the YTD figure reported in the PDF
FY24 thru APR	Year-to-date FY24 amounts through April (partial fiscal year)
ANNUALIZED FY24	Annualized FY24 estimate based on the YTD April totals.

Each row represents one installation–category combination, such as “Okinawa – Slot Revenue” or “Okinawa – NAFI Reimbursement.”

Navy Revenue Report FY20-FY24-2_monthly_summary_master.csv

Column	**Description**
Installation	Navy installation name
Loc#	4-digit ARMP location number within the installation.
Location	Human-readable location label, often with an appended 6-digit site code.
Month	Month of activity in MMM-YY format
Revenue	Slot-machine revenue for this Installation–Loc#–Month.
Annual Revenue	In the PDF, September rows sometimes include a printed fiscal-year total; those values are captured here when present.
Annual NAFI	Analogous to Annual Revenue, but for NAFI reimbursements.
NAFI Amt	NAFI reimbursement amount for the same location and month.
Status	Special flags for non-standard months.

Each row represents a single location (Loc#) within an installation for a specific month.

c) Converting PDF to Text

Both Navy extraction scripts use the same initial conversion step, which begins by calling pdftotext -layout <PDF> - to transform the PDF into structured text; the -layout option preserves the original column alignment, which is essential for accurately reconstructing multi-column tables. In the code, the extract_pdf_text helper wraps this system call and returns the converted text to Python. After extraction, the normalize_dashes function standardizes all dash variants (such as en dashes and em dashes) into a single hyphen so that downstream regular expressions—especially those detecting fiscal-year labels and numeric fields—operate consistently throughout the document. Details in each document are listed below:

Slot Results and NAFI Reimbursements

- **Overview**: For the high-level summary tables ("Slot Machine Results – Navy" and "NAFI Reimbursement from ARMP"), we built a parser that identifies each section, discovers the fiscal-year columns dynamically, and reconstructs a clean installation-level panel of revenue and reimbursement across multiple reports and year ranges.

- **Section Detection**: We first perform section detection by scanning the PDF text line by line for headers that contain any of the following phrases: Slot Machine Results - Navy, NAFI Reimbursement from ARMP, or ARMP Navy Slot Report. Once a matching header is found, we treat the subsequent lines as belonging to that particular summary table until the next section header or a terminating pattern is encountered.

- **Dynamic Fiscal-Year Header Parsing**: Within approximately 25 lines after each section header, we search for the table header that contains the columns Country and Installation. We then merge header continuation lines and apply the FY_TOKEN regular expression to detect all fiscal-year style column labels. This includes standard annual columns such as FY16, FY17, …, as well as partial and derived columns such as FY23 thru SEP, ANNUALIZED FY23, and ANNUALIZED FY24. Because the year labels are discovered dynamically rather than hard-coded, the same logic works across different report ranges (e.g., FY16–FY22 vs. FY18–FY24) without modifying the code.

- **Row Parsing ("State Machine")**: For each body line in the table, we use a simple state-machine style row parser. The first token on the line is interpreted as the Country. We then accumulate subsequent tokens into the Installation name until we encounter a token that looks like a money field or a dash (-), which signals the start of numeric data. All remaining tokens are treated as numeric values aligned with the previously detected fiscal-year columns. The parse_value helper function removes dollar signs and commas, interprets parentheses as negative values, and returns either a floating-point number or an empty string when a valid number cannot be parsed.

- **Multi-Block Merge Across Pages**: Some installations are split across multiple blocks in the PDF, often due to page breaks or layout constraints. To handle this, we attach an internal _cols marker to each parsed row that records which fiscal-year columns were present in that block. After parsing all blocks, we perform a multi-block merge keyed by (Country, Installation), combining the columns from all related blocks so that each installation appears as a single consolidated row with its full set of fiscal-year values.
- **Output Files:** For each input PDF, the script generates two intermediate CSV files: one named slot_results.csv, which stores the parsed slot-machine revenue by

country and installation, and another named nafi_reimbursements.csv, which contains the parsed NAFI reimbursement amounts for the same regions. A downstream processing step then merges these outputs into a consolidated file, Navy_Revenue_Reimburse_Summary.csv, which integrates both revenue and reimbursement records and standardizes all fiscal-year columns to support consistent analysis and reporting.

Monthly Summary by Location Parsing Logic

- **Overview:** The Monthly Summary by Location section in the Navy report is more irregular than the high-level summary tables and therefore required a more detailed parsing strategy. Our goal was to reconstruct a clean, panel-style dataset at the installation–location–month level that can be used for time-series analysis and downstream aggregation.

- **Installation Block Detection:** We first perform installation block detection by scanning for the main header line ARMP Navy Slot Report followed by Monthly Summary by Location. The next non-blank line after this header is treated as the installation name (e.g., *Atsugi*, *Naples*), and that installation context remains active until the next header block is encountered. This ensures that every row is correctly tagged with its parent installation even when the PDF spans multiple pages.

- **Loc#, Location, and Site Code Parsing:** Within each installation block, we parse the Loc# / Location / site code information. A new row typically begins with a 4-digit Loc# (matching the regex ^\d{4}$). Everything that appears after this Loc# and before either the first month token or a 6-digit site code is treated as the human-readable location name. If a 6-digit site code is present, it is captured separately and appended to the location string (for example, "Corner Pocket 401401"), so that each row can be uniquely identified at the location–site level.

- **Month Detection and Numeric Field Extraction:** Next, we extract the month and numeric fields. Month tokens follow the MMM-YY pattern (e.g., Oct-12), and everything to the right of the month is scanned for numeric tokens using the re_nums pattern, which handles commas, parentheses, and decimals. The map_tokens function then assigns the last one to four numeric values on the line to the corresponding fields: Revenue, NAFI Amt, Annual Revenue, and Annual NAFI. This function includes special logic for September rows, where the PDF sometimes prints only the annual totals rather than monthly figures.

- **Continuation Lines and "Temp Closed" Handling:** The parser also handles continuation lines and "Temp Closed" rows. Some lines contain only a month and numeric amounts (or a 6-digit site code plus a month); these are treated as continuations of the previously seen Loc# and Location. We use a simple state-machine approach together with forward-fill (ffill) on Installation, Loc#, and Location to propagate the correct context to these continuation lines. Any line containing the phrase "Temp Closed" is captured with Status = "Temp Closed" and its numeric fields left blank, so that closures are explicitly recorded without generating misleading zero values.

- **Cleaning, Deduplication, and Installation-Specific Fixes:** After building the raw DataFrame, we apply cleaning and deduplication steps. Month strings are converted to dates so that records can be sorted chronologically, and we drop duplicate rows based on (Installation, Loc#, Location, Month), keeping the last occurrence. We then apply installation-specific cleanup rules for known problem cases: dropping corrupted Yokosuka rows with broken "Oct 15" labels, removing Souda Bay graffiti/shipmate rows that lack valid site codes, cleaning Sasebo “BC” lines without proper codes, and fixing Atsugi location 3079 rows where Location appears only as "None 401401" or as duplicate “Club Trilogy” records.
- **Final Output:** The cleaned table is saved as Navy_Revenue_Report FY20-FY24-2_monthly_summary_master.csv. This file gives a consistent, analysis-ready view of monthly Navy slot-machine revenue and NAFI reimbursements by location. It supports trend analysis, installation comparisons, and higher-level reporting.

d) Output Quality and Limitations

- **Tiny fonts and partially corrupted numeric fields**: The source PDF uses extremely small fonts and partially damaged numeric fields. Even with our custom cleanups, a few Yokosuka rows remain noisy or missing; this limitation has already been discussed with the client.
- **Partial FY24:** All FY24 figures represent data only through April and are then annualized. When comparing FY24 with previous years, analysts should either use the “thru APR” year-to-date values consistently or rely on the annualized columns while noting the limitations of doing so.
- **Meaningful Parentheses:** For monthly data, we intentionally keep some numeric strings (especially parentheses) as text to avoid silently changing the meaning; analysts can convert them to signed numbers later if needed.

6) District Revenues

a) Overview of the District Revenues Dataset

The District Revenues report is a multi-year PDF that summarizes slot-machine revenue and NAFI reimbursements by region (Europe, Korea, Japan), service branch (Army, Navy, USMC), base, and more granular location lines within each base. For this project, we built a dedicated parser in Python that converts the "Slot Revenue & NAFI Reimbursement by Month" tables into a long, analysis-ready panel covering fiscal years 2020–2024.

The main output is District_Revenue_filtered_FY20-FY24_final.csv, which standardizes all pages into a consistent schema with each row representing one month of revenue or reimbursement for a specific Service, Region, Base, and Location. This dataset is the primary backbone for our revenue analysis and for the Datasette dashboard used in the project.

b) Data Dictionary

District_Revenue_filtered_FY20-FY24_final.csv
Each row in the final table represents one monthly observation of either slot-machine revenue or NAFI reimbursement at a specific base or sub-location. The fields are summarized in the table below:

Column	Description
Service	Army, Navy, or USMC
Category	Slot Revenue or NAFI Reimbursement
Region	Europe, Korea, or Japan
Base	Standardized installation name
Location	Optional sub-location within a base (e.g., a club or facility)
Month	Month name (e.g., Oct, Jan)
Year	Calendar year assigned based on fiscal-year mapping (Oct–Dec → FY−1; Jan–Sep → FY)
Amount	Numeric value parsed from the PDF, normalized for currency formatting and parentheses

PDF Page Filtering and Coordinate-Based Extraction
Unlike the Navy revenue report (which was handled with pdftotext -layout and a text-only regex parser), the District Revenues document required a **coordinate-based approach** because of its layout: multi-line base names, inconsistent fiscal-year headers, and dense month columns made it difficult to rely solely on plain text. For this reason, the District pipeline uses **PyMuPDF** (fitz) to work directly with the PDF's text boxes and coordinates. The parsing workflow is Building a "filtered" PDF for auditability that includes recognizable headers (e.g., "Slot Revenue … Europe/Korea/Japan" or "Slot Revenue & NAFI Reimbursement by Month – Far East") and total lines that clearly reference each region and service, such as "Total Europe – Army Slot Revenue."

These selected pages are copied into a smaller file, **District_Revenues__filtered.pdf**, which contains only the relevant table content used for the final dataset. This makes it easy to audit any row in the CSV by going back to a much shorter PDF. The parser identifies the twelve fiscal-year month columns by analyzing horizontal text alignment and token positions on each page:

- Using page.get_text("words"), all word boxes — including their coordinates (x0, y0, x1, y1) — are extracted and grouped into rows based on rounded y-coordinates.
- Rows containing at least eight recognizable month labels (e.g., "October," "Nov," "Dec 2020," "Jan-21") are used to infer the x-centers and spacing pattern of the month columns.
- Twelve evenly spaced bins from October through September are constructed, each with a small tolerance so that slightly misaligned numeric tokens still map to the correct month.

After the month columns are defined, every line on the page is re-examined:

- Tokens whose x-center lies to the left of the October column are treated as part of the descriptive "slot name" area (region/base/location text).
- Tokens whose x-center lies within a month bin are candidate numeric values for that month. If multiple numeric tokens fall into the same bin, the right-most is taken as the authoritative value (this handles cases where both revenue and reimbursement appear in a dense cluster).
- Anything far to the right of the September column is ignored as likely noise (e.g., page numbers or footnotes).
- **Fiscal Year Detection**:
- The script looks for "Fiscal Year 20XX" strings in the page text and uses that fiscal year as context for all rows on the page.

- When the fiscal-year header is missing on a given page, the parser carries forward the last known fiscal year from previous pages.

c) Record Construction and Service/Region Assignment

Once month columns and text blocks are identified, the parser converts each logical line into structured records:

Building the Base/Location string:

The parser uses a “slot name builder” to assemble the descriptive part of each line:

- Short tokens like K, dashes, and numbers (e.g., “K”, “-”, “16”) are merged into more human-readable strings like K-16.
- Stand-alone punctuation or pure symbol tokens are discarded.
- This ensures that complex names and line prefixes do not get split into multiple pseudo-bases.

Detecting totals lines and inferring Service/Region:

Special regex patterns detect lines like “Total Europe Army Slot Revenue” or truncated variants where “revenue” is partially cut off. From these totals lines, the parser extracts:

- **Region** (Europe, Korea, or Japan)
- **Service** (Army, Navy, USMC)
- **Category** (Revenue vs. Reimbursement)

The parser then retro-fills this information: all lines immediately above a totals line (within the same page block) inherit the corresponding Service, Region, and Category. This is crucial because the underlying table layout often prints the branch and region only once at the top or bottom of each block.

Expanding each row into a month-level panel:

- For each base/line, the parser collects the numeric values mapped into the 12 fiscal-year month bins.
- It then melts these wide rows into long form: for every non-blank month value, it creates a record: Service, Category, Region, Base, Location, Month, Year, and Amount.

- Months **October–December** are assigned Year = FiscalYear - 1, while **January–September** are assigned Year = FiscalYear. This preserves the correct calendar timeline while retaining the fiscal-year context.

d) Cleaning, Canonicalization, and Manual Patches

After parsing all pages, the pipeline applies several cleaning and normalization steps before writing the final CSV:

Base Name Canonicalization:

Because the PDF contains inconsistent spellings and spacing, base names are standardized to prevent downstream analysis from splitting the same installation into multiple entries. For example, "Camp Hanson" is corrected to "Camp Hansen," "Tori Station" becomes "Torii Station," "Garmisch -AFRC" is standardized to "Garmisch-AFRC," and multiple variants of "Hohenfels" are unified under a single consistent name.

Numeric parsing and missing values:

The parse_number helper removes dollar signs, commas, and whitespace, interprets parentheses as negative values (e.g., "(1,234.56)" → -1234.56), and leaves any tokens that cannot be parsed cleanly as NaN rather than forcing them to zero, ensuring that unreadable source values are not artificially fabricated.

Manual patch for Sasebo Navy – Hario (FY2020):

One known problem area in the District Revenues PDF is the **Sasebo Navy – Hario** line for FY2020, where the February–September monthly values were partially corrupted and could not be reliably read by the automated parser. To preserve this important installation while remaining transparent, we:

- reconstructed the missing values from a combination of nearby lines and earlier versions of the report as provided by the client, and
- inserted them as a small manual patch at the end of the pipeline.

The patched rows are clearly documented in the notebook so that future auditors can see exactly which values were adjusted and why.

Final Dataset Export:

All cleaned records are assembled into a single pandas DataFrame with the columns described above and written to **District_Revenue_filtered_FY20-FY24_final.csv**. This file is used throughout the analysis and is the primary table exposed in our Datasette instance and dashboard.

e) Output Quality and Limitations

- **Layout and font constraints**: The District Revenues report suffers from dense month columns and occasional truncation of words (e.g., "nue" instead of "Revenue"), which motivated the use of coordinate-based extraction instead of pure text parsing. Some borderline cases may still be misaligned, but the month-binning strategy and totals-line inference significantly reduce these errors.
- **Partially corrupted rows**: A small number of lines exhibit damaged text or overlapping characters. Except for the explicitly documented **Sasebo Navy – Hario** patch, such rows are either left with missing values or excluded when they cannot be interpreted safely.
- **Fiscal-year assumptions**: The mapping from fiscal months (Oct–Sep) to calendar years assumes the standard U.S. federal fiscal-year structure. Analysts comparing District Revenues with other datasets should ensure they use consistent fiscal vs. calendar conventions.

Overall, the District Revenues pipeline converts a complex, multi-year PDF into a reproducible, machine-readable panel that integrates cleanly with our other datasets (Assets, Marine Revenue, Navy Revenue). It enables branch-by-region, base-level, and time-series analyses that would be extremely difficult to perform directly on the original District Revenues report.

7) Financial Statements

a) Overview of the Financial Statements Dataset

The Financial Statements report contains monthly Financial data from December 2019 to March 2024. The report is split into four table types,

- **Statement of Financial Condition**, reporting Asset/Liabilities for a given month
- **Actual Vs Budget Operating Results**, comparing actual revenues and expenses to budget amounts
- **Branch of Service Operating Results**, listing revenue/expenses for a given reigon split by military branch
- **Statement of Gaming Revenue**, listing gambling revenue by base for a given month/YTD

The pipeline processes the pdf, outputting to four CSV files (FinancialStatement.csv, ActualVsBudget.csv, BranchBudget.csv, GamingRevenue.csv respective to the order of the list above).

b) Data Dictionary

FinancialStatement.csv

Column Name	Description
Date	Date associated with data entry
AssetType	Lists the category of asset associated with data entry (for example, Cash, Fixed Assets, Accounts Payable)
Balance	Current balance for a given data entry
Category	Lists whether the entry falls under Assets, Liabilities or Equity

ActualVsBudget.csv

Column Name	Description
Date	Date associated with data entry
Location	Region associated with data entry (Europe, Korea, Japan, or Consolidated)
AssetType	Lists whether the entry falls under Revenue, Expenses or Net category
Category	Lists the category of asset associated with data entry (for example, Gaming Machine Revenue, Communications, Training, Distributions)
Month_Actual	Actual balance for a given category over the last month
Month_Budget	Amount budgeted for a given category over the last month
Month_Variance	Month_Actual - Month_Budget
YTD_Actual	Actual balance for a given category year-to-date
YTD_Budget	Amount budgeted for a given category year-to-date

YTD_Variance	YTD_Actual - YTD_Budget

BranchBudget.csv

Column Name	**Description**
Date	Date associated with data entry
Location	Region associated with data entry (Europe, Korea, Japan, or Consolidated)
AssetType	Lists whether the entry falls under Revenue, Expenses or Net category
Category	Lists the category of asset associated with data entry (for example, Gaming Machine Revenue, Communications, Training, Distributions)
ARMP	Combined balance for data entry across all branches of the military
Army	Balance for data entry for Army bases only
Navy	Balance for data entry for Navy bases only
USMC	Balance for data entry for Marine Corp bases only
MonthCount	How many months prior to the date value is this data covering (report includes two versions of these tables, one that is prior month, and one that is YTD starting in October)

GamingRevenue.csv

Column Name	**Description**
Date	Date associated with data entry
Base_Location	Name of base associated with data entry
Europe	Revenue for a the data entries base this month (if it is located in Europe)
Korea	Revenue for a the data entries base this month (if it is located in Korea)

Japan	Revenue for a the data entries base this month (if it is located in Japan)
YTD Europe	Revenue for a the data entries base this year-to-date (if it is located in Europe)
YTD Korea	Revenue for a the data entries base this year-to-date (if it is located in Korea)
YTD Japan	Revenue for a the data entries base this year-to-date (if it is located in Japan)

c) Processing Logic

OCR: Prior to running the script, the FinancialStatements.pdf file was run through Adobe's OCR tool to extract text from scanned pages

Text Layout Extraction: Using pdftotext-layout from the poppler-utils library, the parser extracts the text while maintaining table formatting

Determining Page Type: Iterating through the document, the parser determines which table type a page is by scanning the header to find title phrases unique to each table

Building the dataset: From here, the parser splits into four separate functions (one for each CSV output), parseFinancials, parseTotalBudget, parseBranchBudget, and parseRevenue. Each one of these functions will process all pages containing their specific table type, cleaning data to ensure a fiscally accurate output

parseFinancials: Each page is separated into lines, with blank lines removed. The date is parsed from the header, and the start of table data is determined. From there, the parser iterates through each line, splitting each one into columns using multiple spaces ([\s\{2,} in regex) as a delimiter. If the line has valid data, the asset category is determined and the columns are sent to the buildFinancialRow function, which determines the asset type and cleans the balance associated with it using the numCleanup function to fix any errors in the OCR process. This row is added to a 2D array holding pdf data, and once all pages are processed, the array is exported to the FinancialStatement.csv file.

parseTotalBudget: Like parseFinancials (and for the following functions), each page is separated into lines, with blank lines being removed. The date and location are parsed from the header, and the parser begins to iterate through each line. The line is split into columns using the 2+ whitespaces regex, and the asset type for the data entry is extracted. From there, date, location, assetType and remaining columns are sent to buildBudgetRow. This function determines the asset category, and checks to see that the column count is correct (occasionally the table will have entries with only a single space between columns). From there, each column entry is run through numCleanup to fix transcription errors, and a new row for the 2D array is returned. Once all pages have been processed, the array is exported as the ActualVsBudget.csv file

parseBranchBudget: After processing/splitting into lines, the date, location, and monthCount are determined from the header (monthCount using a dictionary mapping from word to number). Each line is split into columns by multiple spaces, and then attempts to determine asset category by matching against a dictionary containing key words that indicate the table is in a new asset category. Next, the data is sent to buildBudgetRow to be processed in the same manner as individual row data in parseTotalBudget. Once all pages have been processed, the array is exported as the BranchBudget.csv file

parseRevenue: Each page is split into lines, and the date is pulled from the header. For 1/2020 and 12/2019, manual fixes are applied to specific broken lines. Each line is split into columns and run through buildRevenueRow. Like buildBudgetRow, this function will determine if there are columns that weren't split properly, or if the row's asset category got split into more than one column. From there, the base location is run through a dictionary that imports the categoryMap.csv file, which maps broken base location names to their correct spelling. Each balance is run through numCleanup, and the row is returned. Once all pages have been processed, the array is exported as the GamingRevenue.csv file.

d) Output Quality and Limitations

Text Quality: While some of the report tables are digital records with accurate OCR text, a decent percentage are scans of physical records. Due to this, transposed letters/numbers are common, and in select instances required manual adjustment to keep the balance sheet accurate.

Table Structure: Additionally, select tables that were scanned from a physical copy had the structure of the table completely break upon OCR. These pages were excluded from the final output.

Preliminary Analysis

1) Geographic Distribution

Region	% of Total Slot Machines	% of [1]Cumulative Revenue

[1] Cumulative means from Fiscal year 2016 to 2024

Europe	39.4%	~58%
Japan	37.8%	~$42%
Korea	22.8%	

2) Service Branch Summary

Branch	% of Total Slot Machines	% of Cumulative Revenue	Highest-Revenue Installation
Army	55.7%	~60%	Camp Humphreys, Korea ($60.9M cumulative)
Navy	25.4%	~20%	Yokosuka, Japan ($36.5M cumulative)
USMC	19.0%	~20%	Camp Butler/Foster, Japan ($70.6M cumulative)

3) Top 10 Revenue-Generating Bases

Rank	Base Name	Branch	Location	Cumulative Revenue
1	Camp Butler / Foster	USMC	Japan	$70.6M
2	Camp Humphreys	Army	Korea	$60.9M
3	Yokosuka Navy	Navy	Japan	$36.5M
4	AFRC Dragon Hill Lodge	Army	Korea	$24.3M
5	Kaiserslautern	Army	Germany	$22M
6	Wiesbaden	Army	Germany	$18M
7	Sasebo Navy	Navy	Japan	$16.2M
8	Daegu / Casey / Hovey	Army	Korea	$15M
9	Stuttgart	Army	Germany	$12M+
10	Iwakuni	USMC	Japan	$10M

4) Temporal Trends

- **COVID-19 Impact**: The COVID-19 pandemic caused a sharp decline in revenue during FY2020 across all service branches. Revenue then recovered steadily from FY2021 through FY2023, and FY2023 became the strongest year for most installations.
- **March and May Seasonality**: Marine Corps bases display a clear seasonal pattern, with noticeably higher gambling activity in March and in May.

5) Financial Trends

- **Restricted Cash Reserves:** Total cash increased steadily until 2023, then declined sharply due to the complete depletion of restricted cash reserves. This pattern indicates weakening financial health.
- **Static Equity Anomaly:** Total equity has remained exactly unchanged, down to the cent, since October 31, 2021. This unusual stability suggests possible reporting or aggregation issues in the Financial Statement Report files.
- **Asset Aging during the COVID-19 period**: Asset values decreased from sixteen million dollars in 2020 to less than seven million dollars in 2022, reflecting significant machine aging. Most aging machines were produced by Novomatic and Bally. These older units were replaced quickly after the pandemic, and by April 2024 the total asset value had returned to nearly twenty million dollars.

6) Revenue Efficiency (Per Machine)

Branch–Region	Avg Annual Revenue per Base	Avg Daily Revenue per Machine
Navy – Japan	$6.91M	$118.40
USMC – Japan	$8.29M	~$100–110
Army – Korea	$20.11M	~$90–100
Army – Europe	$4.73M	$80.06
Navy – Europe	$2.88M	~$60–70

Base Questions

1) What is the total and per-base slot machine revenue by military branch and by region?

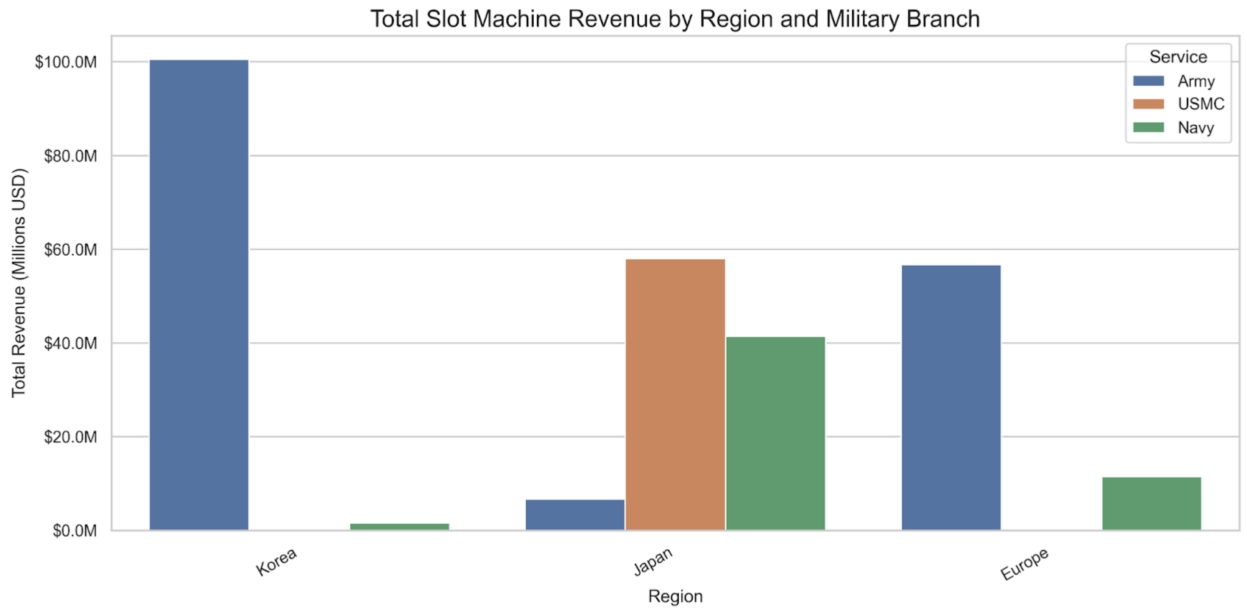


Figure 2: Total Slot Machine Revenue by Region and Military Branch

Figure 1 summarizes total slot-machine revenue by military branch and region, along with the number of bases and the implied per-base revenue. The Army operates slot machines in Korea, Japan, and Europe. Its five bases in Korea generate about $100.6 million in total revenue, which is both the largest branch–region total and the highest per-base average at roughly $20.1 million per base. In contrast, the Army's presence in Japan is relatively small: two bases there earn about $6.7 million, or $3.35 million per base. The Army's 12 bases in Europe collectively bring in approximately $56.7 million, averaging $4.7 million per base.

The Navy also operates across the three regions but at a smaller scale. Six Navy bases in Japan generate about $41.4 million in total slot-machine revenue, averaging $6.9 million per base, while four European Navy bases produce $11.5 million (about $2.9 million per base). Navy activity in Korea is limited to a single installation, which earns roughly $1.6 million; because there is only one base, the per-base figure is identical to the total. Finally, the Marine Corps' slot-machine operations are concentrated entirely in Japan. Seven USMC bases there generate about $58.1 million in total revenue, corresponding to a relatively high per-base average of $8.3 million. Taken together, these figures show that the Army in Korea is the single largest revenue driver, while Marine Corps and Navy revenues are heavily focused in Japan, and neither service operates slot machines in Europe for the Marine Corps or in large numbers in Korea for the Navy.

The first bar chart, *"Total Slot Machine Revenue by Region and Military Branch,"* visualizes these patterns. The x-axis compares Korea, Japan, and Europe, while the y-axis expresses total revenue in millions of U.S. dollars. Army bars (blue) dominate the Korea and Europe regions, especially in Korea where the bar reaches the $100 million mark. In Japan, the Marine Corps (orange) and Navy (green) together account for nearly $100 million, with the Marine Corps slightly ahead of the Navy, whereas the Army's contribution is visibly smaller. The chart makes clear at a glance that the Army's Korean operations and the Marine Corps' and Navy's Japanese operations are the principal sources of slot-machine revenue, and that Marine Corps revenue is geographically confined to Japan.

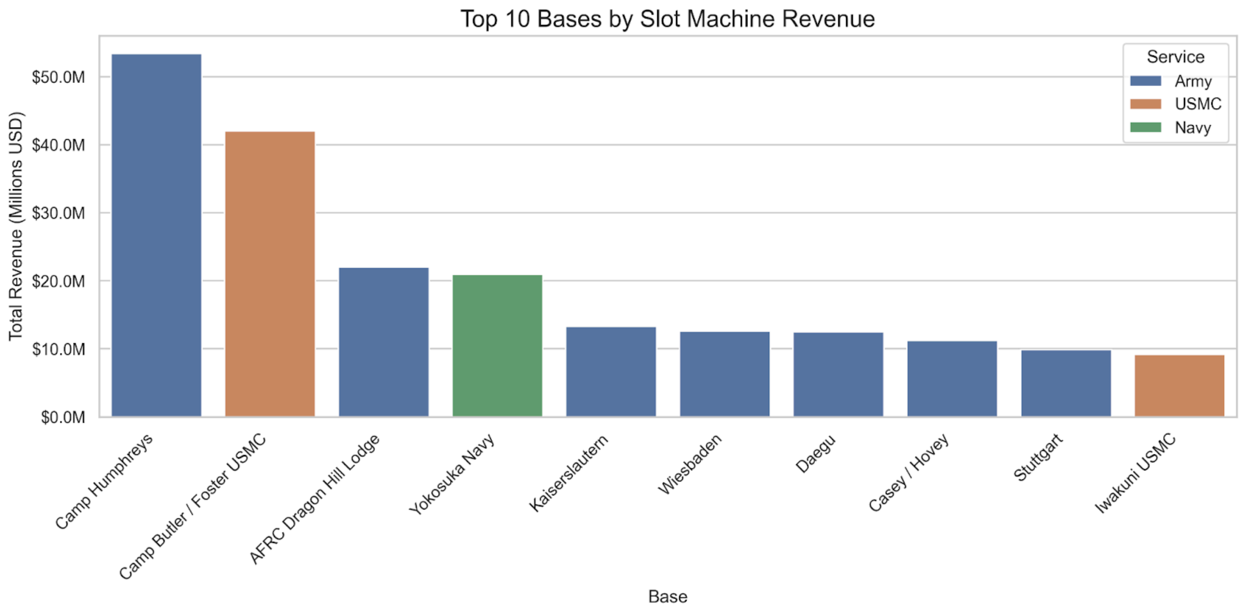


Figure 3: Top 10 Bases by Slot Machine Revenue

In **figure 2**, it ranks the top 10 individual bases by total slot-machine revenue across all branches and regions. Each bar represents a single installation, colour-coded by service branch (blue = Army, orange = Marine Corps, green = Navy), with the y-axis again showing revenue in millions of U.S. dollars. This chart highlights how a small number of large installations account for a disproportionate share of overall revenue.

At the top of the ranking is Camp Humphreys (Army, Korea), which earns just over $50 million, making it the single highest-grossing slot-machine location in the dataset. The second-largest contributor is Camp Butler / Foster (Marine Corps, Japan), with roughly $41 million in revenue. Several other Army installations follow, including AFRC Dragon Hill Lodge, Kaiserslautern, Wiesbaden, Daegu, Casey / Hovey, and Stuttgart, each generating between about $10 million and

$22 million. The Navy's primary representative in the top ten is Yokosuka (Japan), which contributes around $21 million. The list is rounded out by Iwakuni USMC (Japan) at approximately $10 million.

Overall, this ranking shows that seven of the top ten revenue-generating bases belong to the Army, reinforcing the Army's central role in slot-machine operations, especially in Korea and Europe. It also underscores the importance of a handful of flagship installations—Camp Humphreys, Camp Butler / Foster, AFRC Dragon Hill Lodge, and Yokosuka—in driving branch-level and region-level revenue totals. Together with the branch-by-region summary, these charts demonstrate that slot-machine revenue is not evenly distributed across all bases, but is instead highly concentrated in a few large, high-traffic locations.

Answer for Question 1:
For Base Question 1, the charts show that the Army in Korea is the dominant source of slot-machine revenue, with about $100M generated across five bases (roughly $20M per base on average), making it both the largest branch–region total and the highest per-base earner. In Japan, Marine Corps bases collectively bring in around $58M (7 bases, ~$8M per base) and Navy bases about $41M (6 bases, ~$7M per base), while the Army's two Japanese bases contribute a much smaller $7M total. In Europe, the Army's 12 bases generate roughly $57M (about $5M per base) and the Navy's European sites about $15M; the Marine Corps has no European slot-machine revenue. The per-base ranking mirrors this pattern: Camp Humphreys (Army, Korea) is the top individual base at over $50M, followed by Camp Butler / Foster USMC (Japan) at about $41M, and then a mix of high-earning Army and Navy bases such as AFRC Dragon Hill Lodge, Yokosuka Navy, Kaiserslautern, Wiesbaden, Daegu, Casey/Hovey, Stuttgart and Iwakuni. Overall, Army installations,especially in Korea,dominate both total and per-base revenue, and seven of the top ten bases by revenue belong to the Army, indicating that slot-machine income is highly concentrated in a small set of large overseas hubs.

2) Which bases generate the highest total revenue, and how does that rank change by year and by branch?

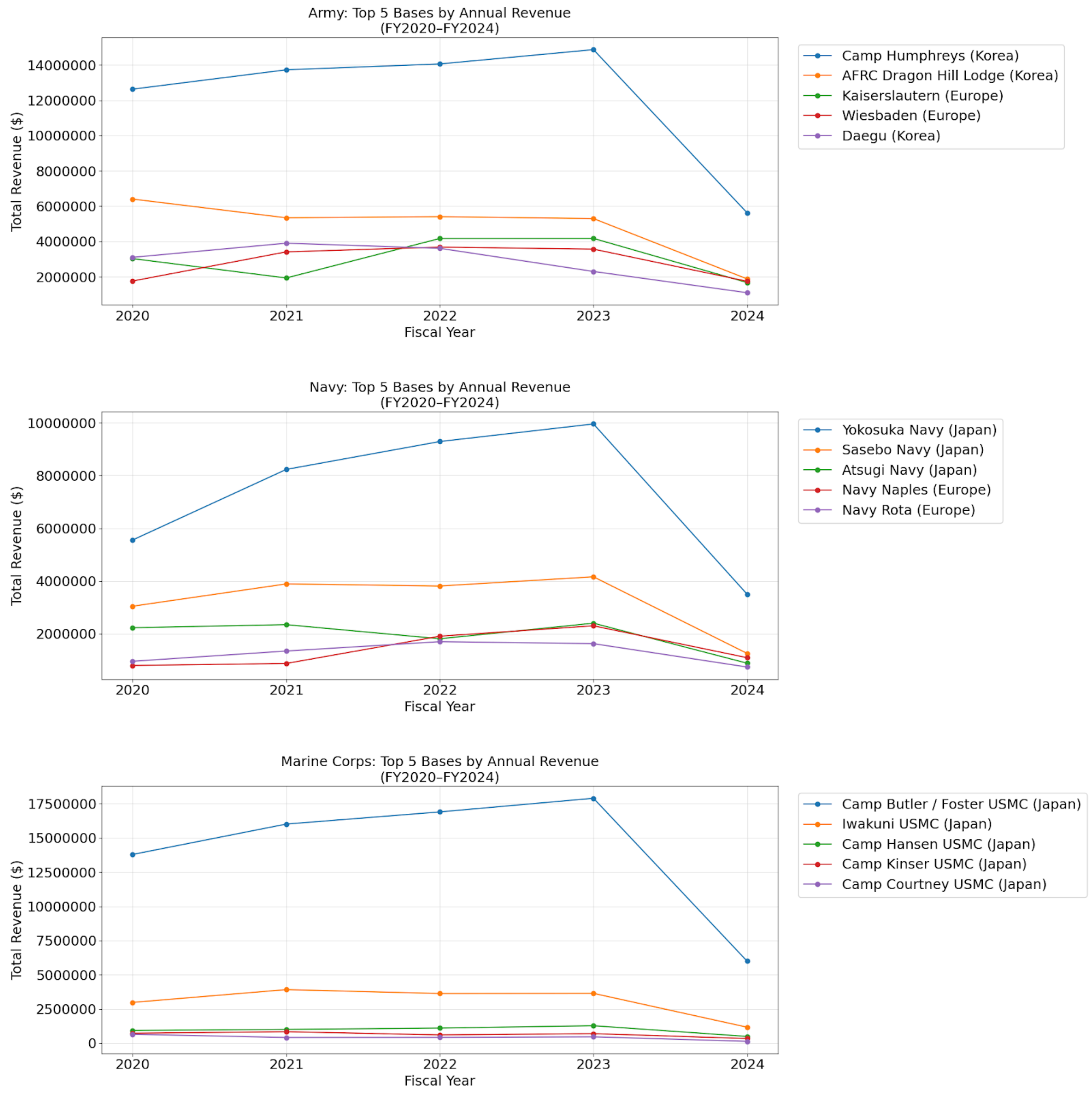


Figure 4: Top 5 Revenue generating Bases by branches

This **figure 3** compares the top five overseas bases by annual slot-machine revenue for the Army, Navy, and Marine Corps from FY2020 to FY2024. For the Army, Camp Humphreys (Korea) is consistently the top-earning installation, rising from about $12–15M between FY2020 and FY2023 before a sharp drop in FY2024. The Navy's clear leader is Yokosuka Navy Base (Japan), which grows from roughly $5–10M through FY2023 and then also declines in FY2024, while Sasebo and the other Japan/Europe bases stay in the mid-single-million range. For the Marine Corps, Camp Butler / Foster (Japan) dominates its branch and is the single largest revenue-generating base in the dataset, increasing from about $14M to nearly $18M by FY2023 before falling back in FY2024. Overall, the chart shows that a small number of large Japanese and Korean hubs account for the majority of revenue in each branch, with strong growth up to FY2023 followed by a broad pullback in FY2024.

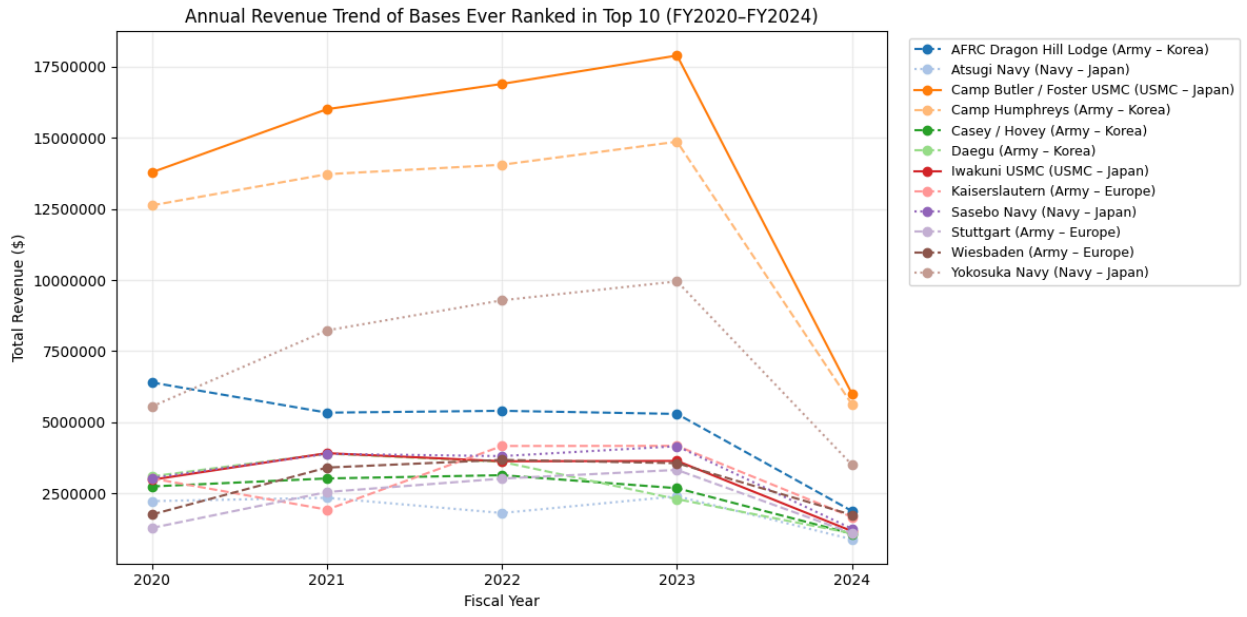


Figure 5: Annual Revenue Trend of Bases Ever Ranked in Top 10

Figure 4 tracks the annual slot-machine revenue from FY2020 to FY2024 for every base that ever ranked in the overall top 10. It highlights how a small set of large overseas hubs account for most of the revenue across all branches. Camp Butler / Foster USMC (Japan) is the clear top performer, rising from about $13.8M in FY2020 to roughly $18M in FY2023 before dropping sharply in FY2024. Camp Humphreys (Army – Korea) is consistently the second-highest base, with steady growth from around $12.6M to $15M over FY2020–FY2023 and a similar pullback in FY2024. A second tier of bases—including Yokosuka Navy (Japan) and Wiesbaden (Army – Europe)—shows moderate growth into the $8–10M range by FY2023, while mid-tier sites such as AFRC Dragon Hill Lodge, Iwakuni, Sasebo, Kaiserslautern, and others peak around $3–5M. All bases experience a noticeable revenue decline in FY2024, but the relative ordering between bases and branches remains largely unchanged.

Answer for Question 2:

Across FY2020–FY2024, the highest-revenue base in the entire dataset is Camp Butler / Foster USMC (Japan), which consistently ranks #1 overall and within the Marine Corps, followed by Camp Humphreys (Army – Korea) as the steady #2 base and top Army site, and Yokosuka Navy (Japan) as the leading Navy base that generally sits in the #3 overall position. A second tier of Army and Navy installations such as Wiesbaden (Europe), AFRC Dragon Hill Lodge and Casey/Hovey/Daegu (Korea), Iwakuni USMC, Sasebo and Atsugi Navy, Kaiserslautern and Stuttgart, contributes substantially but never surpasses these three leaders. While revenues for most top bases rise from FY2020 to FY2023 and then drop sharply in FY2024, this volatility mainly affects the magnitude of revenue rather than the ordering, so the cross-branch ranking of the top-earning bases remains remarkably stable over time.

3) How have the types of games or machines installed at bases changed over time?

To answer this question, the Asset Reports from **FY2020 through FY2024** were combined and analyzed to classify each machine into one of four major categories. Because the raw asset descriptions vary widely across vendors, bases, and years, a keyword-based classification system was developed. This system scans each machine's description and assigns it to one of the following groups:

- **SLOT** – Slot machines, video poker, multi-game cabinets, keno machines, reel/stepper games, and equipment from major gaming vendors such as IGT, Bally, Konami, Aristocrat, and others.
- **FRS** – Financial and point-of-sale systems such as IBM SurePOS units, cash registers, redemption kiosks, TITO (ticket-in/ticket-out) systems, and cashiering hardware.
- **ITC** – IT and communications equipment, including servers, PCs, laptops, printers, routers, switches, UPS devices, and other network or computing assets.
- **ACM** – Automated cash machines: ATMs, cash dispensers, and smart card dispensers.

This classification allows all machines across the five fiscal years to be compared consistently, even when naming conventions differ.

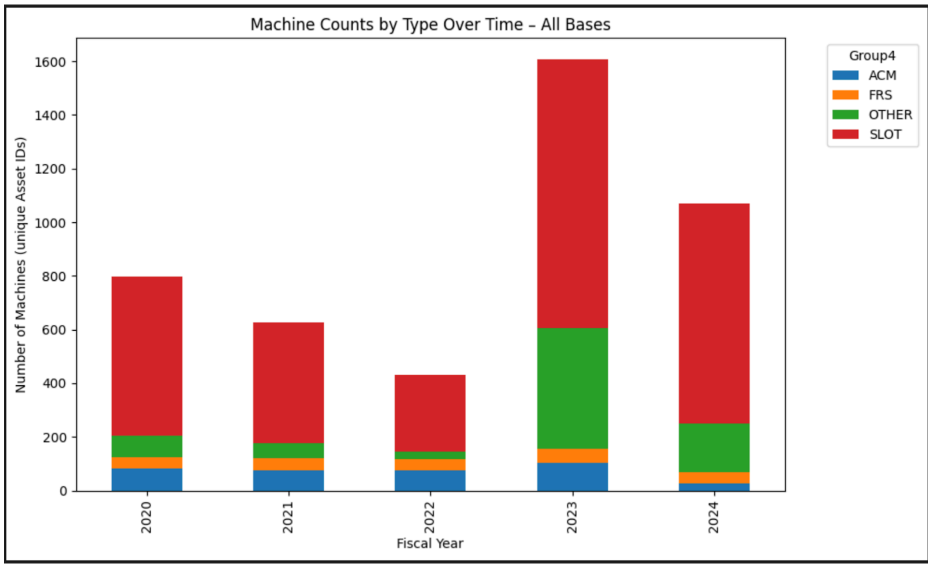


Figure 6: Machine Counts by Type Over Time in all bases

Figure 5 displays the **number of machines of each type installed across all bases** from FY2020 to FY2024. Across all years, **SLOT machines** remain the dominant machine type, consistently representing the largest share of equipment installed at bases. Total machine counts gradually decline across all categories, showing a contraction in available equipment. However, in FY2023 there is a **major increase in total machines**, driven by more SLOT machines and

ITC-classified equipment. This suggests a significant technology refresh during that year—large deployments of servers, PCs, networking gear, and other IT infrastructure. By FY2024, machine counts decline from the FY2023 peak but remain above early-year levels. SLOT machines continue to be the dominant type, while ITC/OTHER equipment remains elevated relative to the pre-2023 baseline.

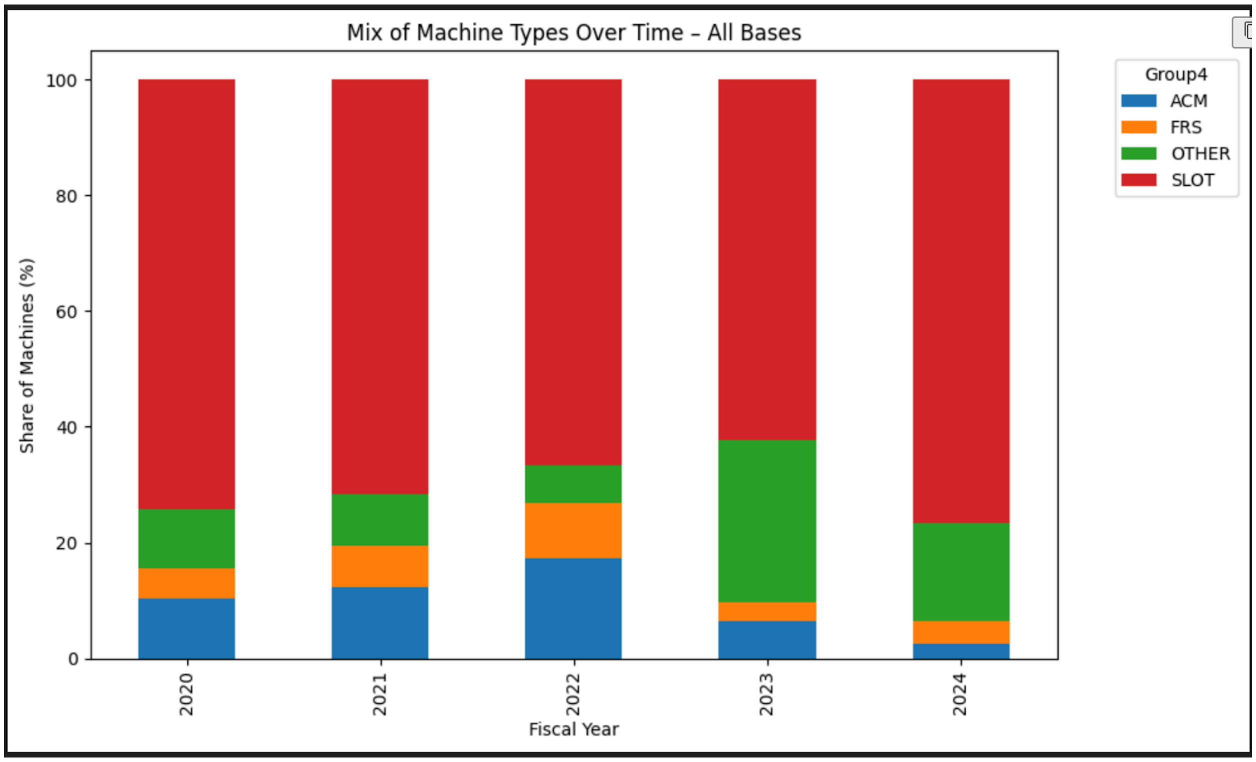


Figure 7: Mix of Machine Types Over Time in all Bases

Figure 6 shows how the proportion of each machine type has shifted over time. Instead of total counts, this view highlights the relative mix of SLOT, FRS, ITC, and ACM equipment at bases year by year. Across the entire period, SLOT machines consistently make up the vast majority of all installed equipment, usually accounting for around 70–80% of the machine inventory. This confirms their dominant operational ground, even when overall machine counts fluctuate. From FY2020 through FY2022, the mix of non-slot equipment—ACM, FRS, and ITC—remains relatively stable. The composition shifts noticeably in FY2023, as the **share of ITC equipment increases significantly**, even though SLOT machines still represent most of the equipment. This suggests that 2023 wasn't just a year of adding machines overall, but one in which IT-related assets became a larger part of the operational ecosystem. These assets include servers, computers, networking devices, and other IT infrastructure identified through the classification model.

Answer for Question 3:

The overall mix of machines at bases remains dominated by SLOT machines every year, but the composition underneath that dominance does shift, especially in 2023. From 2020–2022, the non-slot categories (ACM, FRS, ITC/OTHER) stay relatively steady and modest in size. Then 2023 produces a striking spike: both the total number of machines and the share attributed to ITC/OTHER jump sharply. By 2024, that spike relaxes—SLOT machines once again take up a very large majority of all equipment, but ITC/OTHER remains elevated compared to earlier years.

Datasette Dashboard Visualization

1) Datasette UI Deployment Process

The deployment process for the Military Slot Machine Revenue Explorer follows a streamlined pipeline that transforms raw CSV data into an interactive, publicly accessible web application hosted on Render. The workflow begins with data preparation, where multiple cleaned CSV files containing slot machine revenue data from Army, Navy, and Marine Corps installations are processed through a Python conversion script (`convert_csv_to_db.py`). This script normalizes column names, calculates fiscal year fields from calendar dates, enriches records with geocoded coordinates, and builds optimized SQLite database tables with strategic indexes for query performance.

Once the SQLite database (`military_slots.db`) is generated, the deployment leverages containerization through Docker. The Dockerfile packages the application using a Python 3.11 slim base image, installs required dependencies including Datasette and its plugins (datasette-cluster-map, datasette-vega, datasette-dashboards), and configures the container to serve the database on port 8001 with CORS enabled for cross-origin access. The containerization ensures consistent runtime environments across local development and production deployments.

The final deployment stage utilizes Render's infrastructure-as-code approach through `render.yaml`, which automates the entire build and deployment lifecycle. When code is pushed to the GitHub repository, Render automatically detects the configuration file, builds the Docker image, and deploys it as a web service. The platform handles environment variables, health checks, and automatic restarts, while the free tier provides sufficient resources for public data exploration. The resulting application serves an interactive Datasette interface that allows users to query revenue data, visualize geographic distributions on cluster maps, generate charts through the Vega plugin, and explore pre-built dashboards with filters for fiscal year, branch, and district—all accessible through a public URL without requiring any client-side installation or configuration.

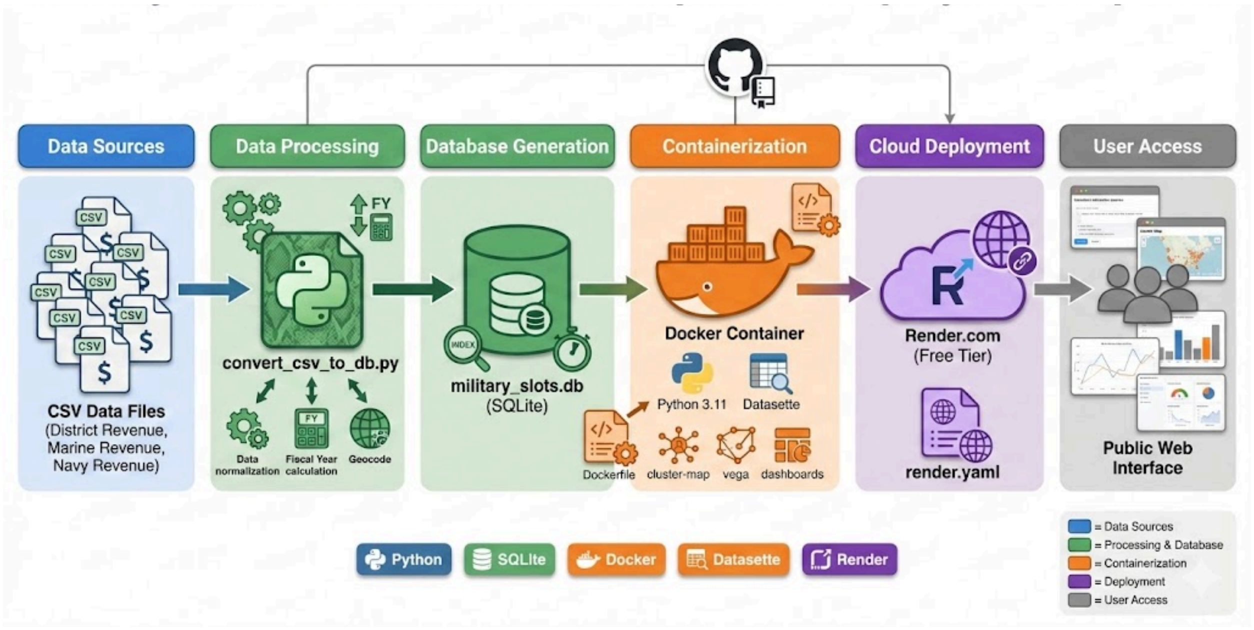


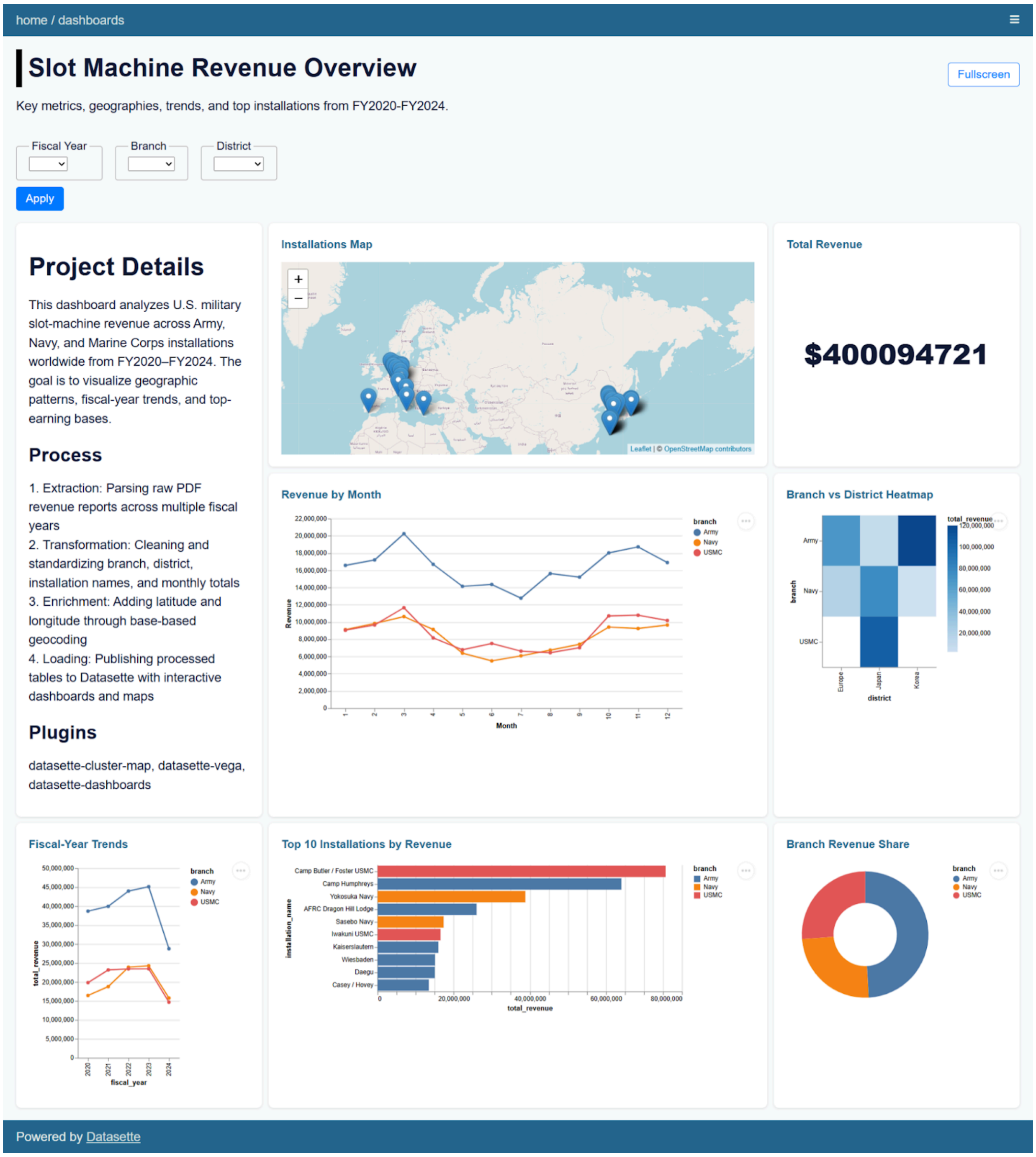
Figure 8: Diagram of the Datasette UI Deployment Process

2) Slot Machine Revenue Dashboard

a) Overview

After publishing all processed slot-machine revenue tables to Datasette, we built this dashboard to make the data more accessible and user-friendly. This dashboard provides users with an interactive way to explore and understand the detailed information contained in our uploaded Datasette dataset. It visualizes U.S. military slot-machine revenue across Army, Navy, and Marine Corps installations from FY2020–FY2024, integrating multi-year district revenue reports obtained via MuckRock into a single interactive analytical tool built with Datasette and the datasette-dashboards plugin.

A global filtering system is implemented across the entire dashboard, allowing users to adjust year, service branch, and region selections once and immediately see all visualizations updated consistently. This unified filter design ensures that users can seamlessly navigate the dataset from multiple analytical perspectives without having to reapply filters in each section.


home / dashboards

Fullscreen

Slot Machine Revenue Overview

Key metrics, geographies, trends, and top installations from FY2020-FY2024.

Fiscal Year | Branch | District

Apply

Project Details

This dashboard analyzes U.S. military slot-machine revenue across Army, Navy, and Marine Corps installations worldwide from FY2020–FY2024. The goal is to visualize geographic patterns, fiscal-year trends, and top-earning bases.

Process

1. Extraction: Parsing raw PDF revenue reports across multiple fiscal years
2. Transformation: Cleaning and standardizing branch, district, installation names, and monthly totals
3. Enrichment: Adding latitude and longitude through base-based geocoding
4. Loading: Publishing processed tables to Datasette with interactive dashboards and maps

Plugins

datasette-cluster-map, datasette-vega, datasette-dashboards

Installations Map

Total Revenue

$400094721

Revenue by Month

Branch vs District Heatmap

Fiscal-Year Trends

Top 10 Installations by Revenue

Branch Revenue Share

Powered by Datasette

Figure 9: Slot Machine Revenue Dashboard runs in real time

b) Objective

The purpose of this dashboard is to transform fragmented PDF revenue reports into a unified dataset that enables transparent analysis of global slot-machine operations. We primarily use the District_Revenue_filtered_FY20-FY24_final.csv dataset in this dashboard, which was derived from MuckRock-provided documents and enhanced with additional latitude and longitude fields to support geospatial visualization.

c) **Dashboard Components**
   - **Installations Map:** Interactive world map showing Army, Navy, and Marine Corps slot-machine locations, filterable by fiscal year, branch, and district.
   - **Total Revenue Summary:** Aggregated revenue displayed dynamically based on selected filters.
   - **Revenue by Month:** Line chart illustrating monthly revenue patterns and general seasonal trends across fiscal years.
   - **Branch vs. District Heatmap:** Heatmap showing revenue concentration patterns across service branches and districts.
   - **Fiscal-Year Trends:** Multi-year comparison of annual slot-machine revenue across Army, Navy, and USMC.
   - **Top 10 Installations:** Ranking of highest-earning installations worldwide.
   - **Branch Revenue Share:** Breakdown of total revenue contribution by Army, Navy, and Marine Corps.

d) **Analytical Value**
   - Converts multiple pages of financial documents into a transparent, public-facing analytical product.
   - Supports investigative journalism, academic research, and data-driven oversight.
   - Reveals multi-year revenue and geographic patterns.
   - Highlights where slot-machine operations are most concentrated globally.
   - Makes Department of Defense MWR gaming operations more accessible to non-technical audiences.

Conclusion and Recommendation

This project

Individual Contribution

Task Description	Cameron Moore	Ching Hsuan Lin	Jyun-Ru Huang	Nithya Priya Jayakumar	Pin-Hao Pan	Quoc Dat Nguyen
Extract Asset Reports		V	V	V	V	V
Extract Marine Revenue			V			
Extract Navy Revenue		V		V		