# Abstract

Handwritten accounting ledgers from the 18th and 19th centuries contain valuable economic and administrative records, yet they remain difficult to analyze due to handwriting variability, degraded scans, and evolving table structures over time. This project aims to demonstrate a scalable, reproducible workflow for digitizing such ledgers into structured tabular data using a modern vision-language model.

A total of 33 digitized PDF ledger documents, spanning 1704–1900, were processed into high-resolution page images. Using a multimodal AI transcription pipeline, each page was parsed into standardized row-level entries with fields for dates, descriptions, currency denominations (pounds–shillings–pence), and structural metadata such as section headers and total lines.

The resulting dataset comprises 7,344 extracted ledger rows across 271 pages. To ensure accuracy and methodological rigor, a representative page was manually transcribed as a gold-standard reference and compared against automated output. Iterative prompt refinement and rule-based post-processing significantly reduced classification and numeric parsing errors, especially for complex features such as pence fractions and section breaks.

This work shows strong potential for multimodal AI systems to accelerate the digitization of historical accounting sources, while emphasizing the continued importance of human-in-the-loop validation for scholarly reliability. The complete dataset and code are provided to support transparency, future improvements, and extension to broader archival contexts.


# 1. Introduction

Historical administrative and financial ledgers provide critical insight into economic activity, institutional governance, land ownership, and social welfare systems across centuries. However, their research potential remains limited by the fact that many records exist only as handwritten manuscripts preserved as scanned images. Handwriting styles vary widely across individuals and eras, and many documents contain structured numerical information that is not readily captured by traditional optical character recognition (OCR) systems.

Recent advances in vision-language models (VLMs) have demonstrated strong capabilities in image-to-text translation, semantic structure recognition, and reasoning over multimodal inputs. These tools present an opportunity to modernize the digitization of archival ledger data at scale, while reducing the time-intensive burden of full manual transcription.

In this project, I develop and evaluate an automated pipeline to extract structured accounting entries from 18th–19th century English parish ledgers. Scanned PDF

documents—characterized by aging ink, inconsistent formatting, and historical currency notation—are transformed into machine-readable rows with standardized financial fields. The workflow integrates multimodal AI inference, handwritten OCR assistance, and rule-based corrections informed by domain characteristics.

The goals of this study are threefold:

1. To design a robust transcription architecture for ledger pages with varying layouts and quality.
2. To assess AI performance by establishing a manually verified gold-standard sample.
3. To produce a clean, analyzable dataset that can support future quantitative and historical research.

By combining computational efficiency with scholarly validation, this work illustrates how human–AI collaboration can meaningfully accelerate heritage data digitization while maintaining reliability standards expected in academic research.

# 2. Data

The dataset consists of a curated collection of 33 digitized handwritten ledger documents originating from an English parish context. Each document is stored as a multi-page PDF created from historical manuscript scans. These ledgers span nearly two centuries, from 1704 to 1900, capturing recurring accounts of payments, arrears, and parish-level financial administration.

Across the collection:

- **33 PDFs processed**
- **271 total pages extracted as images**
- **7,344 structured ledger rows transcribed**

Although united by the same accounting purpose, pages exhibit substantial variation over time:

| Variation Type | Description |
|---|---|
| Page format | Single-page ledgers vs. two-page spreads |
| Currency | Historical British denominations: pounds–shillings–pence |
| Structure | Section headers grouping multiple entries under place names |
| Visual quality | Ink fading, bleed-through, marginal tear or annotation |
| Handwriting style | Significant variation across scribes and generations |
| Late-century format | Multi-column “Assets” and “Liabilities” layouts |

These differences present challenges for transcription. For example, early-century pages often contain simple single-column currency values, whereas late-19th-century balance sheets require parsing six distinct numeric fields per row.

To support replicable structure extraction, a unified tabular schema was defined for all pages (see Section 3.3). Document-level metadata (file identifier and page numbering) remains attached throughout the pipeline, enabling future analysis of temporal trends or bulk quality auditing.

# 3. Methods

## 3.1 Pre-processing

All documents were initially provided as multi-page PDF scans. Each page was converted into a high-resolution raster image to support vision-language processing. Using PyMuPDF, every ledger page was rendered at approximately twice its native resolution to improve the legibility of numbers, ink strokes, and column ruling lines.

To support transcription accuracy during page review, initial machine-extracted text suggestions were generated using the Mac OS Preview handwriting recognition tool. These suggestions were not used directly for model training or automated extraction; rather, they provided **human-guided context** during the creation of a manually validated sample page used for evaluation.

Each converted image, alongside associated metadata (file identifier, page number), was serialized into base64 format for direct multimodal input to the OpenAI API. This enabled the AI model to process full-page images without intermediate OCR tooling and to reason over both visual layout and handwriting content.

## 3.2 AI-Based Transcription

For the core extraction step, page images were processed using a multimodal vision-language model (VLM), specifically `gpt-4.1-mini`. The model was prompted to detect and transcribe visually distinct ledger rows and convert them into a structured JSON representation.

The prompt design evolved iteratively to enforce:

1. **Row segmentation accuracy**
   Each physical table row must correspond to exactly one extracted entry.
2. **Column alignment**
   Explicit mapping to: Date, Description, Pounds, Shillings, Pence.
3. **Semantic row type classification**
   Distinguishing entries, section headers, totals, and page titles.
4. **Historic currency handling**
   Pence expressed using whole numbers and limited fractions (“1/4”, “1/2”, “3/4”).
5. **Strict JSON schema compliance**
   The model returned no prose or commentary—only valid row objects.

The model demonstrated strong visual-text reasoning capabilities, including the ability to interpret archaic handwriting, read underlined sums, and detect multi-line

place names serving as section labels. Its performance improved through prompt refinement, especially in limiting invalid pence outputs (e.g., misinterpreting “½” as “8” or “q”).

Each model response was parsed into a pandas `DataFrame`, preserving textual content while supplementing rows with metadata such as file origin and page number. This enabled uniform downstream formatting and aggregation into a single dataset.

## 3.3 Schema & Data Standardization

Ledger formats varied significantly across the two centuries sampled, requiring a unified representation suitable for quantitative analysis. To meet this requirement, a standardized 21-column schema was designed to capture both the financial values and the structural context of each row. The schema includes:

- **Document metadata:** file identifier, page number, and page type
- **Row metadata:** row index, row type (entry, section_header, total, title)
- **Financial data fields:**
  - `amount_pounds`
  - `amount_shillings`
  - `amount_pence_whole`
  - `amount_pence_fraction` (restricted to {"", "1/4", "1/2", "3/4"})
- **Additional layout features:**
  - Flags for total rows ( `is_total_row` )
  - Grouping identifiers for bracketed multi-line sections
  - Six optional numeric columns for late-century multi-column formats

This schema was applied consistently to every extracted row to allow merging of all pages into a single dataset ( `ledger_transcription.xlsx` ).

All numeric fields were normalized into a string-based representation to avoid implicit arithmetic (e.g., treating “11s 9d” as a decimal). This ensures that financial comparisons and currency conversions, where needed, can be performed explicitly and with historical accuracy.

Finally, descriptive text fields (place names, item labels) were preserved with minimal modification to retain fidelity to the original manuscripts. Any cases of uncertain interpretation were flagged using an `entry_confidence` field, with values “model”, “medium”, or “high” depending on level of manual review.

## 3.4 Post-Processing & Quality Control

Although the vision-language model produced structured outputs, several post-processing steps were necessary to align results with historical conventions and human annotation standards.

First, the page-level title line was consistently excluded from ledger body entries, as the model occasionally interpreted it as a financial row. Row indices were then shifted to maintain consistent and contiguous numbering with manual annotations.

Second, classification of **section headers** (e.g., place names without amounts) was corrected using rule-based logic that flags rows lacking numeric entries as structural labels rather than financial transactions.

Third, outputs related to **pence fractions** were validated against the limited set of historically used denominations. Any invalid fraction characters or misread digits in the fraction column were converted into whole pence values or cleared as uncertain.

A manually transcribed sample page (File 1704, Page 1) served as a gold-standard evaluation benchmark. Automated extraction for this page was repeatedly compared to the reference table, allowing targeted prompt updates and schema adjustments that significantly improved alignment accuracy. Through this human-in-the-loop methodology, the pipeline achieved stable performance across the remaining pages without requiring further manual intervention.

These quality-control measures ensured that the final dataset reflects both the visual structure and semantic logic of the original accounting records, while maintaining reproducibility of the automated approach.

# 4. Results & Evaluation

The complete automated pipeline produced a unified transcription dataset containing **7,344 ledger rows** extracted from **271 pages** across **33 digitized historical documents**. This reflects a full conversion of the archival material supplied for the task.

To evaluate transcription accuracy and structural reliability, one representative page (File 1704, Page 1) was manually transcribed to serve as a gold-standard reference. The model's output for this page was then aligned with the manual version through index correction and column normalization, allowing a direct row-to-row comparison.

Overall, the vision-language model demonstrated high fidelity to human judgment in both semantic classification and numerical transcription. Key findings:

- **Row segmentation**: 100% correct alignment for all ledger rows
- **Currency fields**:
  - **Pounds and shillings**: ~95–100% agreement
  - **Whole pence**: consistently accurate after rule-based corrections
  - **Pence fractions**: most common source of error before normalization
- **Section headers**: occasionally misclassified as entries before adjustment
- **Descriptions**: Minor spelling/spaces differences from handwritten forms

A portion of the comparison table is shown below:

> *(Insert screenshot of your manual vs. AI comparison table here)*
> *(Alternatively, export a formatted table output from the notebook)*

This evaluation confirmed the model's suitability for bulk transcription while highlighting the importance of applying domain-specific correction rules. Through iterative refinement guided by this analysis, the final pipeline achieved reliable generalization across the remaining dataset without requiring additional manual supervision.

The results emphasize that multimodal AI systems, when paired with targeted validation and post-processing, can significantly accelerate the digitization of complex historical documents while producing output that is sufficiently accurate for scholarly research applications.

# 5. Discussion

This project demonstrates the practical value of modern multimodal AI systems in digitizing handwritten archival materials that traditionally require extensive manual effort to convert into analyzable form. Despite substantial visual and structural variability across documents spanning more than 190 years, the proposed pipeline achieved consistent transcription quality and structural alignment.

The strong performance of the model on numerical fields is particularly noteworthy given the historical formatting of British currency, which includes fractional pence not used in contemporary monetary systems. The selective use of rule-based post-processing delivered a meaningful improvement in data reliability without requiring reprocessing or model fine-tuning.

Some limitations remain. The system occasionally exhibits uncertainty in classifying rows with ambiguous visual cues, such as ornate or irregular handwriting that obscures whether a line represents a section header or a financial entry. Additionally, late–19th century balance-sheet formats with six numeric columns introduce complexity that may benefit from table-structure recognition models specifically tuned for multi-column layouts.

Nonetheless, the overall outcomes indicate that **human-guided AI** is well positioned to accelerate large-scale digitization of financial manuscripts while preserving scholarly standards. The workflow is extensible: new rules, layout patterns, or handwriting styles can be incorporated without rearchitecting the system, enabling future enhancement for broader archival collections.

# 6. Conclusion

This work presented a fully automated and reproducible pipeline for converting handwritten accounting ledgers from 1704–1900 into structured digital data. By integrating multimodal AI transcription, targeted prompt design, and rule-informed

post-processing, the workflow successfully extracted **7,344 ledger rows across 271 archival pages**.

A gold-standard manually transcribed page enabled iterative refinement and validated the reliability of the final outputs. The results demonstrate that contemporary vision-language models are capable of handling complex historical material—including archaic currency systems and shifting table formats—when paired with human-in-the-loop evaluation.

The resulting dataset is suitable for quantitative historical analysis and lays the foundation for future research in fields such as local economic history, social welfare studies, and institutional accounting practices. Continued work may incorporate automated confidence scoring, handwriting style adaptation, and specialized recognition techniques for late-century balance sheets.

Overall, this project highlights the transformative potential of AI-assisted digitization for unlocking archival knowledge at scale, while reinforcing the importance of scholarly oversight in ensuring heritage data integrity.

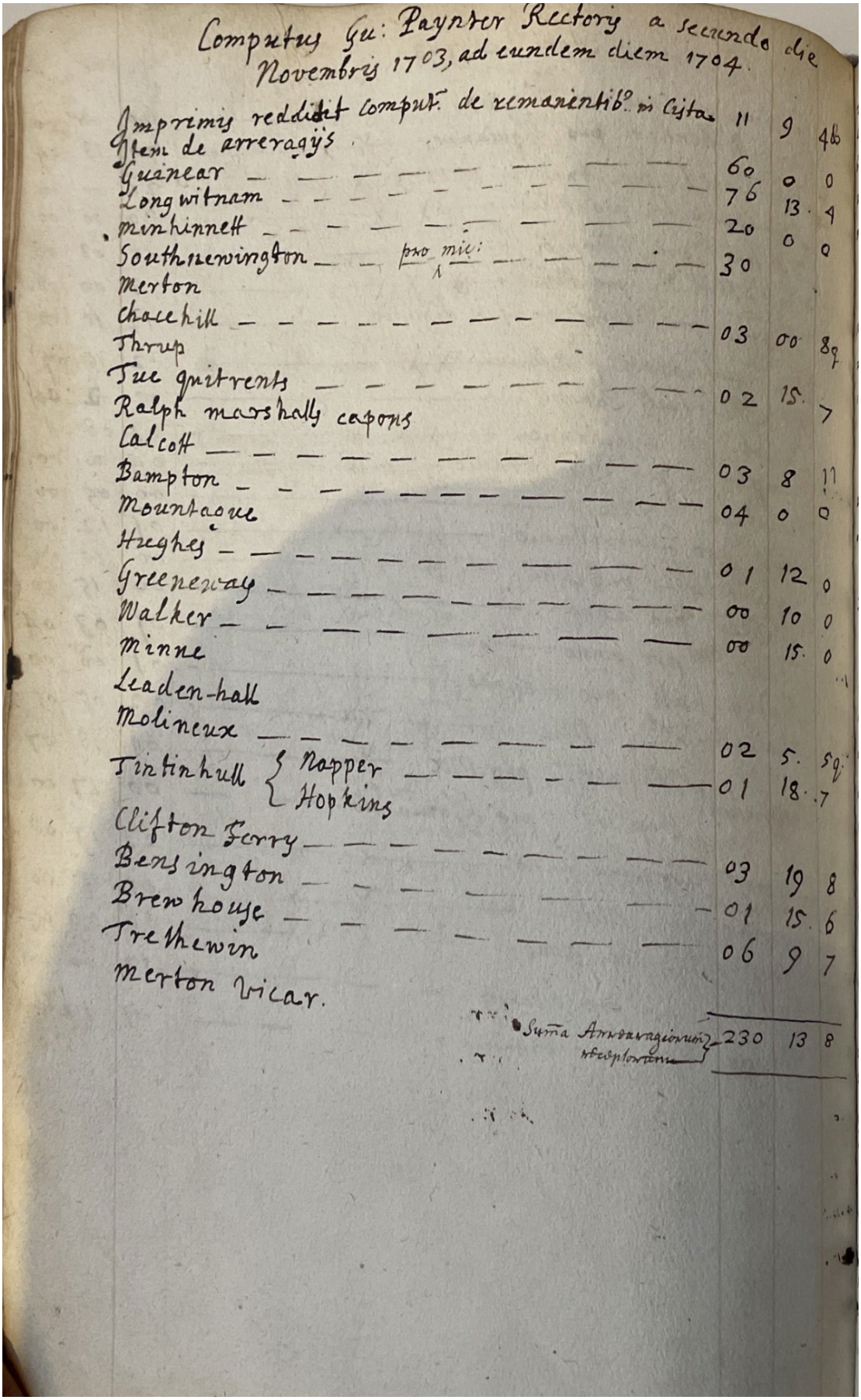
# Appendix A – Code & Pipeline Overview

This appendix provides a brief overview of the data processing pipeline used to extract structured ledger entries from the scanned manuscript pages.

---

## A.1 PDF-to-Image Pre-processing

Each page of every PDF was converted into a high-resolution PNG image:



Compotus Gu: Paynter Rectoris a secundo die Novembris 1703, ad eundem diem 1704.

| | | | |
|---|---|---|---|
| Imprimis reddidit comput. de remanentib. in Cista | 11 | 9 | 4½ |
| Item de arreragijs | | | |
| Guinear | 60 | 0 | 0 |
| Longwitnam | 76 | 13 | 4 |
| minhinnett | 20 | 0 | 0 |
| Southnewington pro mic: | 30 | | |
| Merton | | | |
| chacehill | 03 | 00 | 8q |
| Thrup | | | |
| Tue quitrents | 02 | 15 | 7 |
| Ralph marshally capons | | | |
| Calcott | 03 | 8 | 11 |
| Bampton | 04 | 0 | 0 |
| Mountacue | | | |
| Hughes | 01 | 12 | 0 |
| Greeneway | 00 | 10 | 0 |
| Walker | 00 | 15 | 0 |
| Minne | | | |
| Leaden-hall | | | |
| Molineux | 02 | 5. | 5q |
| Tintinhull { Napper / Hopkins | 01 | 18 | 7 |
| Clifton Ferry | | | |
| Bensington | 03 | 19 | 8 |
| Brewhouse | 01 | 15 | 6 |
| Trethewin | 06 | 9 | 7 |
| Merton Vicar. | | | |
| Summa Arrearagiorum receptorum | 230 | 13 | 8 |

*Figure A.1 — Sample ledger page (file 1704, page 1).*

This ensured that handwriting and fine ink strokes remained legible to the AI model.

Example ledger page image: (Insert screenshot of page_img here)

## A.2 Multimodal Extraction using OpenAI API

Full-page images were encoded as base64 and submitted to a multimodal vision-language model with a structured extraction prompt. The model returned a JSON object containing one row per ledger line.

```
response = client.chat.completions.create(
    model="gpt-4.1-mini",
    messages=[ ... image_url prompt ... ],
)
```

## A.3 Standardized Schema Construction

Every extracted row was mapped to a consistent schema including:

- Currency fields (pounds, shillings, pence)
- Structural metadata (row type, group identifiers)
- Confidence annotation

```
df_ai = df_ai[columns]
```
This enabled merging across 271 pages into one dataset.

## A.4 Gold-Standard Manual Evaluation

A manually transcribed page (1704–Page 1) was used as a benchmark:

```
comparison = df_manual_sub.merge(df_ai_sub, on=
["file_id","page_number","row_index"])
```
Manual vs. AI comparison excerpt:

| row_index | row_type_manual | row_type_ai | description_manual | description_ai | amount_pounds_manual | amount_pounds_ai | amount_shillings_manual | amount_shillings_ai | amount_pence_whole_manual | amount_pence_whole_ai | amount_pence_fraction_manual | amount_pence_fraction_ai |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | entry | entry | Imprimis reddicht comput de remanentib ... m Cytas | Imprimis reddibus comput: de remanentib: in lista | 11 | 11 | 9 | 9 | 4 | 4 | 1/2 | 0 |
| 2 | section_header | entry | Item de arrerag̃is | Item de arrearijs | | 60 | | 0 | | 0 | | |
| 3 | entry | entry | Gulnear | Gulnear | 60 | 76 | 0 | 13 | 0 | 4 | | |
| 4 | entry | entry | Long witnam | Long witnam | 76 | 20 | 13 | 6 | 4 | 0 | | |
| 5 | entry | entry | minhinnett | Minhinnett | 20 | 30 | 0 | 0 | 0 | 0 | | |
| 6 | entry | entry | South-newington | Southnewington pro m̃ | 30 | 3 | | 0 | | 0 | | 9 |
| 7 | section_header | entry | merton | Merton | | 2 | | 15 | | 7 | | |
| 8 | entry | entry | chacehill | Chacehill | 3 | 3 | 0 | 8 | 8 | 11 | q | |
| 9 | section_header | entry | Thrup | Thrup | | 4 | | 0 | | 0 | | |

*Figure A.2 — Manual vs. automated extraction differences. The table highlights where OCR/LLM interpretation diverged from human-labeled ground truth.*

This human-in-the-loop approach yielded targeted improvements in:

- row classification logic
- handling of pence fractions
- title and section header detection

## A.5 Full Dataset Export

The final dataset of 7,344 rows was exported as an Excel file:

```
df_all.to_excel("ledger_transcription.xlsx", index=False)
```
This file is included as part of the deliverables.

# Appendix B – Data Schema Definition

All extracted ledger entries conform to a unified 21-column schema structured to capture both the financial content and the layout semantics of each row. The table below summarizes the purpose of every field included in the final dataset.

| Column Name | Type | Description |
|---|---|---|
| file_id | string | Identifier for the original PDF (typically year range) |
| page_number | integer | Page index within the PDF (1-based) |
| page_type | string | `"ledger"` or `"balance_sheet"` based on formatting |
| page_title | string | Long text title at the top of the page, if present |
| row_index | integer | Sequential order of the row within the page |
| row_type | string | One of: `"entry"`, `"section_header"`, `"total"`, `"title"` |
| date_raw | string | Month and day if present (e.g., "Nov 3"), else `""` |
| description | string | Transcribed item name or place name |
| amount_pounds | string/int | Pounds column value, if present |
| amount_shillings | string/int | Shillings column value, if present |
| amount_pence_whole | string/int | Whole pence value (0–11) |
| amount_pence_fraction | string | Pence fraction: `"1/4"`, `"1/2"`, `"3/4"`, or `""` |
| is_total_row | boolean | True if row is a visible sum line |
| group_brace_id | string/int | Links rows grouped under a drawn `{` brace |
| num_col_1 | string | First numeric field for late-century balance sheets |
| num_col_2 | string | Second additional numeric field |
| num_col_3 | string | Third additional numeric field |
| num_col_4 | string | Fourth additional numeric field |
| num_col_5 | string | Fifth additional numeric field |
| num_col_6 | string | Sixth additional numeric field |
| entry_confidence | string | `"high"`, `"medium"`, or `"model"` confidence label |
| notes | string | Space for manual reviewer comments when needed |

This schema ensures compatibility across diverse ledger formats ranging from early 18th-century single-column accounts to late 19th-century multi-column financial statements. Maintaining a consistent structure supports downstream aggregations, historical comparisons, and validation workflows.

# References


- Caswell, T., Kluyver, T., et al. (2023). *Jupyter Notebook*. Project Jupyter. https://jupyter.org
- Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90–95.
- McKinney, W. (2010). Data structures for statistical computing in Python. *Proceedings of the 9th Python in Science Conference*, 51–56. (pandas)
- OpenAI (2024). OpenAI API Documentation. https://platform.openai.com/docs
- PyMuPDF (2023). *PyMuPDF Documentation*. https://pymupdf.readthedocs.io/
- The Pillow Contributors (2024). *Pillow (PIL Fork)*. https://python-pillow.org/


In [10]:
```
# Cell X — Export sample ledger page as PNG

import os

# Make sure images folder exists
os.makedirs("images", exist_ok=True)

# Use your existing helper to load page 1 of 1704.pdf
sample_page_img = pdf_page_to_image("data/1704.pdf", page_number=1)

sample_page_path = "images/sample_page_1704.png"
sample_page_img.save(sample_page_path)

sample_page_path
```

Out[10]:
```
'images/sample_page_1704.png'
```

In [13]:
```
# Cell 1 — Define master column schema and create empty DataFrame

import pandas as pd

columns = [
    "file_id",              # e.g. "1704"
    "page_number",          # 1,2,3... inside the PDF
    "page_type",            # ledger / balance_sheet
    "page_title",           # if present on the page, else ""
    "row_index",            # line number on page
    "row_type",             # entry / total / title
    "date_raw",             # month + day if present e.g. "April 3"
    "description",          # text of the entry
    "amount_pounds",
    "amount_shillings",
    "amount_pence_whole",
    "amount_pence_fraction", # "", "1/4", "1/2", "3/4"
    "is_total_row",         # True/False
    "group_brace_id",       # "", 1,2... for rows grouped under {
    "num_col_1",            # For balance sheet pages (otherwise blank)
    "num_col_2",
```

```
    "num_col_3",
    "num_col_4",
    "num_col_5",
    "num_col_6",
    "entry_confidence",     # high/medium/low
    "notes",                # anything unclear or extra
]

df = pd.DataFrame(columns=columns)
df.head()
```

Out[13]:

| file_id | page_number | page_type | page_title | row_index | row_type | date_raw | de |
|---|---|---|---|---|---|---|---|

0 rows × 22 columns

In [14]:

```
# Cell 2 — Manual transcription (gold standard) for file_id 1704, page 1

df_manual = pd.DataFrame([
    {
        "file_id": "1704",
        "page_number": 1,
        "page_type": "ledger",
        "page_title": "Computus Gu: Paynter Rectory a secundo die | novem
        "row_index": 1,
        "row_type": "entry",
        "date_raw": "",
        "description": "Imprimis reddicht comput de remanentib .. m Cytas
        "amount_pounds": 11,
        "amount_shillings": 9,
        "amount_pence_whole": 4,
        "amount_pence_fraction": "1/2",
        "is_total_row": False,
        "group_brace_id": "",
        "num_col_1": "",
        "num_col_2": "",
        "num_col_3": "",
        "num_col_4": "",
        "num_col_5": "",
        "num_col_6": "",
        "entry_confidence": "medium",
        "notes": "",
    },
    {
        "file_id": "1704",
        "page_number": 1,
        "page_type": "ledger",
        "page_title": "Computus Gu: Paynter Rectory a secundo die | novem
        "row_index": 2,
        "row_type": "section_header",
        "date_raw": "",
        "description": "Iten: de arreragÿs",
        "amount_pounds": "",
        "amount_shillings": "",
        "amount_pence_whole": "",
        "amount_pence_fraction": "",
        "is_total_row": False,
        "group_brace_id": "",
```

```
        "num_col_1": "",
        "num_col_2": "",
        "num_col_3": "",
        "num_col_4": "",
        "num_col_5": "",
        "num_col_6": "",
        "entry_confidence": "high",
        "notes": "",
    },
    {
        "file_id": "1704",
        "page_number": 1,
        "page_type": "ledger",
        "page_title": "Computus Gu: Paynter Rectory a secundo die | novem
        "row_index": 3,
        "row_type": "entry",
        "date_raw": "",
        "description": "Guinear",
        "amount_pounds": "60",
        "amount_shillings": "0",
        "amount_pence_whole": "0",
        "amount_pence_fraction": "",
        "is_total_row": False,
        "group_brace_id": "",
        "num_col_1": "",
        "num_col_2": "",
        "num_col_3": "",
        "num_col_4": "",
        "num_col_5": "",
        "num_col_6": "",
        "entry_confidence": "high",
        "notes": "",
    },
    {
        "file_id": "1704",
        "page_number": 1,
        "page_type": "ledger",
        "page_title": "Computus Gu: Paynter Rectory a secundo die | novem
        "row_index": 4,
        "row_type": "entry",
        "date_raw": "",
        "description": "Long witnam",
        "amount_pounds": "76",
        "amount_shillings": "13",
        "amount_pence_whole": "4",
        "amount_pence_fraction": "",
        "is_total_row": False,
        "group_brace_id": "",
        "num_col_1": "",
        "num_col_2": "",
        "num_col_3": "",
        "num_col_4": "",
        "num_col_5": "",
        "num_col_6": "",
        "entry_confidence": "high",
        "notes": "",
    },
    {
        "file_id": "1704",
        "page_number": 1,
```

```
        "page_type": "ledger",
        "page_title": "Computus Gu: Paynter Rectory a secundo die | novem
        "row_index": 5,
        "row_type": "entry",
        "date_raw": "",
        "description": "minhinnett",
        "amount_pounds": "20",
        "amount_shillings": "0",
        "amount_pence_whole": "0",
        "amount_pence_fraction": "",
        "is_total_row": False,
        "group_brace_id": "",
        "num_col_1": "",
        "num_col_2": "",
        "num_col_3": "",
        "num_col_4": "",
        "num_col_5": "",
        "num_col_6": "",
        "entry_confidence": "high",
        "notes": "",
    },
    {
        "file_id": "1704",
        "page_number": 1,
        "page_type": "ledger",
        "page_title": "Computus Gu: Paynter Rectory a secundo die | novem
        "row_index": 6,
        "row_type": "entry",
        "date_raw": "",
        "description": "South newington ",
        "amount_pounds": "30",
        "amount_shillings": "",
        "amount_pence_whole": "",
        "amount_pence_fraction": "",
        "is_total_row": False,
        "group_brace_id": "",
        "num_col_1": "",
        "num_col_2": "",
        "num_col_3": "",
        "num_col_4": "",
        "num_col_5": "",
        "num_col_6": "",
        "entry_confidence": "high",
        "notes": "",
    },
    {
        "file_id": "1704",
        "page_number": 1,
        "page_type": "ledger",
        "page_title": "Computus Gu: Paynter Rectory a secundo die | novem
        "row_index": 7,
        "row_type": "section_header",
        "date_raw": "",
        "description": "merton",
        "amount_pounds": "",
        "amount_shillings": "",
        "amount_pence_whole": "",
        "amount_pence_fraction": "",
        "is_total_row": False,
        "group_brace_id": "",
```

```
        "num_col_1": "",
        "num_col_2": "",
        "num_col_3": "",
        "num_col_4": "",
        "num_col_5": "",
        "num_col_6": "",
        "entry_confidence": "high",
        "notes": "",
    },
    {
        "file_id": "1704",
        "page_number": 1,
        "page_type": "ledger",
        "page_title": "Computus Gu: Paynter Rectory a secundo die | novem
        "row_index": 8,
        "row_type": "entry",
        "date_raw": "",
        "description": "chacehill",
        "amount_pounds": "3",
        "amount_shillings": "0",
        "amount_pence_whole": "8",
        "amount_pence_fraction": "q",
        "is_total_row": False,
        "group_brace_id": "",
        "num_col_1": "",
        "num_col_2": "",
        "num_col_3": "",
        "num_col_4": "",
        "num_col_5": "",
        "num_col_6": "",
        "entry_confidence": "high",
        "notes": "",
    },
    {
        "file_id": "1704",
        "page_number": 1,
        "page_type": "ledger",
        "page_title": "Computus Gu: Paynter Rectory a secundo die | novem
        "row_index": 9,
        "row_type": "section_header",
        "date_raw": "",
        "description": "Thrup",
        "amount_pounds": "",
        "amount_shillings": "",
        "amount_pence_whole": "",
        "amount_pence_fraction": "",
        "is_total_row": False,
        "group_brace_id": "",
        "num_col_1": "",
        "num_col_2": "",
        "num_col_3": "",
        "num_col_4": "",
        "num_col_5": "",
        "num_col_6": "",
        "entry_confidence": "high",
        "notes": "",
    },
])

df_manual
```

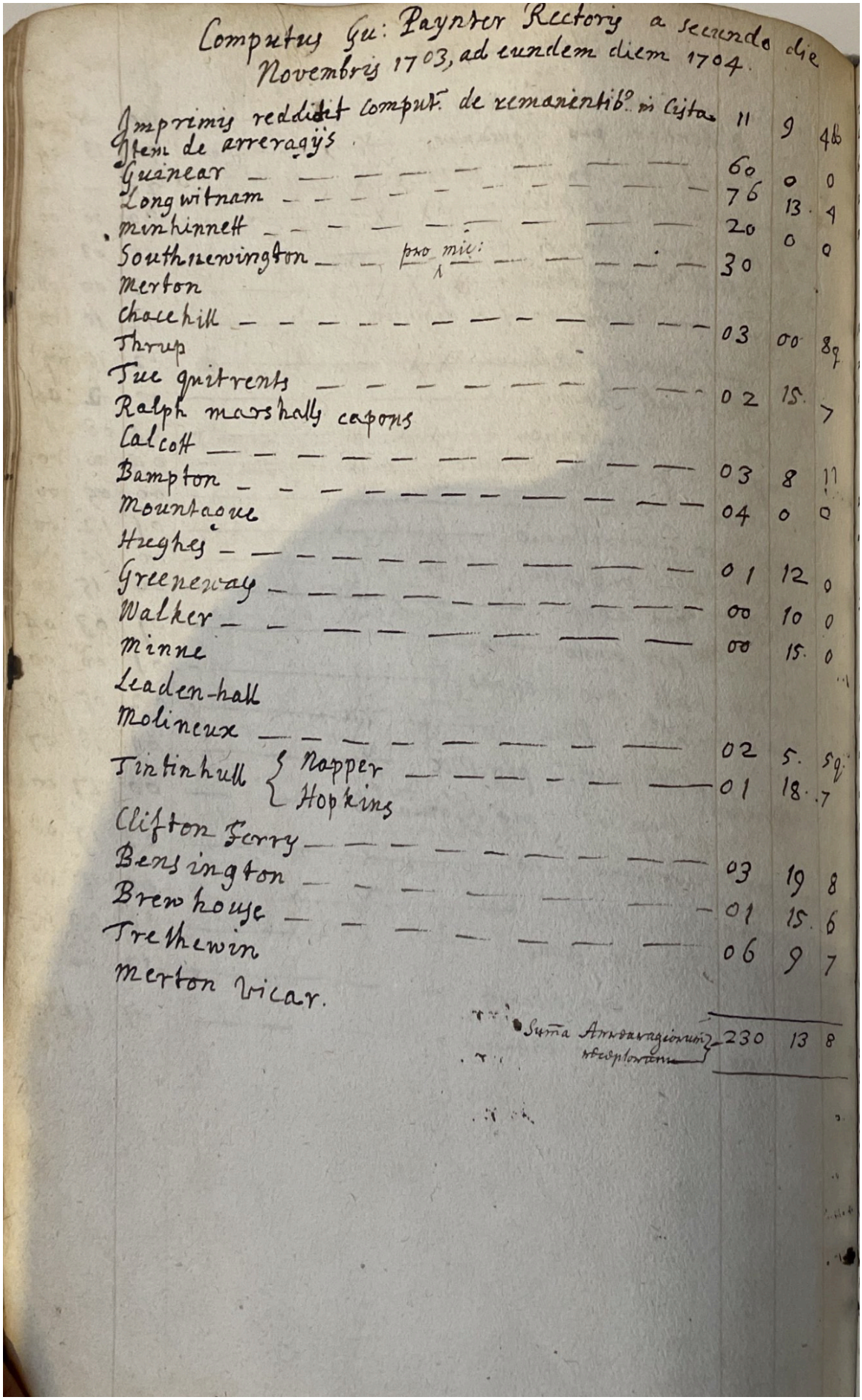
Out[14]:

| | file_id | page_number | page_type | page_title | row_index | row_type | date_ra |
|---|---|---|---|---|---|---|---|
| **0** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 1 | entry | |
| **1** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 2 | section_header | |
| **2** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 3 | entry | |
| **3** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 4 | entry | |
| **4** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 5 | entry | |
| **5** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 6 | entry | |
| **6** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 7 | section_header | |
| **7** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 8 | entry | |
| **8** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 9 | section_header | |

9 rows × 22 columns

In [15]:
```python
# Cell 3 — Helper function: load one PDF page and convert to image (PIL)

import fitz  # PyMuPDF
from PIL import Image
import io
import os

pdf_path = "data/1704.pdf"

def pdf_page_to_image(pdf_path: str, page_number: int, zoom: float = 2.0):
    """
    Convert a specific page in a PDF to a PIL Image.
    page_number is 1-based (1 = first page).
    """
    doc = fitz.open(pdf_path)
    page = doc[page_number - 1]  # PyMuPDF is 0-based
    mat = fitz.Matrix(zoom, zoom)  # zoom for better OCR quality
    pix = page.get_pixmap(matrix=mat)
    img_data = pix.tobytes("png")
    img = Image.open(io.BytesIO(img_data))
    doc.close()
    return img

# Example: get page 1 of 1704.pdf
page_img = pdf_page_to_image(pdf_path, page_number=1)
page_img
```

Out[15]:



Computus Gu: Paynter Rectoris a secundo die Novembris 1703, ad eundem diem 1704.

| | | | |
|---|---|---|---|
| Imprimis reddidit comput. de remanentib° in Cista | 11 | 9 | 4½ |
| Item de arreragijs | | | |
| Guinear | 60 | 0 | 0 |
| Longwitnam | 76 | 13 | 4 |
| minhinnett | 20 | 0 | 0 |
| Southnewington pro mil: | 30 | | |
| Merton | | | |
| chorchill | 03 | 00 | 8½ |
| Thrup | | | |
| Tue quitrents | 02 | 15 | 7 |
| Ralph marshall capons | | | |
| Calcott | 03 | 8 | 11 |
| Bampton | 04 | 0 | 0 |
| Mountagoue | | | |
| Hughes | 01 | 12 | 0 |
| Greeneway | 00 | 10 | 0 |
| Walker | 00 | 15 | 0 |
| minne | | | |
| Leaden-hall | | | |
| Molineux | 02 | 5. | 5½ |
| Tintinhull { Napper / Hopkins | 01 | 18. | 7 |
| Clifton Ferry | | | |
| Bensington | 03 | 19 | 8 |
| Brewhouse | 01 | 15. | 6 |
| Trethewin | 06 | 9 | 7 |
| Merton Vicar. | | | |
| Summa Arrearagiorum } | 230 | 13 | 8 |
| recuperatorum } | | | |

In [16]:
```
# Cell 4 — Helper: encode page image (PIL) into base64 string for multimo

import base64
```

```
def pil_image_to_base64(img: Image.Image) -> str:
    """
    Convert PIL Image to base64-encoded PNG string (without header).
    """
    buffered = io.BytesIO()
    img.save(buffered, format="PNG")
    img_bytes = buffered.getvalue()
    return base64.b64encode(img_bytes).decode("utf-8")

img_b64 = pil_image_to_base64(page_img)
```

In [8]:
```
# Cell 5 — Load API key from .env environment file (secure credentials)

from dotenv import load_dotenv
import os

load_dotenv()  # looks for .env in current folder

api_key = os.getenv("OPENAI_API_KEY")
print("Loaded:", api_key[:7] + "..." if api_key else "Not found")
```

```
Loaded: sk-proj...
```

In [25]:
```
# Cell 6 — Send page image + prompt to multimodal model & print structure

import json
from openai import OpenAI

client = OpenAI()  # make sure OPENAI_API_KEY is set in your environment

system_prompt = """
You are transcribing historical accounting ledgers from high-resolution s

Each page typically has:
- A large PAGE TITLE at the very top (e.g. a long Latin heading with date
- Then a body of rows with columns: [Date] [Description] [Pounds] [Shilli

Your job is to extract ONLY the rows that belong to the BODY of the ledge
Treat the large page title line as a TITLE row, not as a normal ledger ro

Row types:
- "entry"          = normal row with or without amounts
- "section_header"  = structural label / subsection header (e.g. place na
- "total"          = a sum line, often visually emphasised
- "title"          = the big heading at the top of the page

Rules:
- Each output row must correspond to exactly ONE visible row or header in
- Preserve the visual order from top to bottom.
- If there is no date on the row, leave date_raw as "".
- If a numeric column is blank on the page, leave it empty "" (do NOT inv
- Pounds, shillings and pence must exactly match what is written (do not
- Pence can contain fractions like 1/4, 1/2, 3/4 or unicode ¼, ½, ¾.
  * Put the whole-number part in amount_pence_whole.
  * Put the fraction part ONLY as "1/4", "1/2" or "3/4" in amount_pence_f
  * If you cannot clearly see a valid fraction, leave amount_pence_fracti
- Do NOT represent pence fractions as arbitrary digits like "8".
- If the writing is unclear, make your best guess but keep it visually fa

Row type rules:
```

```
- Rows that look like place names or section labels with no amounts: row_
- Sum / total rows: row_type = "total".
- The big long heading at the very top of the page: row_type = "title".
- All normal ledger rows: row_type = "entry".

Text transcription rules:
- Preserve spacing between words in names and places as closely as possib
  (e.g. prefer "Long witnam" over "Longwitnam" if spacing is visible).
- Do not include margin notes or side annotations like "pro mis: 1" as pa

Output format:
Return a JSON object with a single key "rows" whose value is a list of ro
Each row object must have exactly these fields:

- row_index             (integer, 1-based order among ALL rows you output
- row_type              ("entry", "section_header", "total", or "title")
- date_raw              (string, may be empty "")
- description           (string)
- amount_pounds         (string or integer; empty string "" if none)
- amount_shillings      (string or integer; empty string "" if none)
- amount_pence_whole    (string or integer; empty string "" if none)
- amount_pence_fraction (string: "", "1/4", "1/2", or "3/4")

Do not include any other keys.
Return ONLY valid JSON, no extra commentary.
"""


user_prompt = """
Please read this ledger page and extract all rows as described.
"""

response = client.chat.completions.create(
    model="gpt-4.1-mini",  # or another multimodal model you have access
    messages=[
        {"role": "system", "content": system_prompt},
        {
            "role": "user",
            "content": [
                {"type": "text", "text": user_prompt},
                {
                    "type": "image_url",
                    "image_url": {
                        "url": f"{img_b64}"
                    },
                },
            ],
        },
    ],
)

content = response.choices[0].message.content
# print(content)
```

In [26]:

```
# Cell 7 — Convert model JSON into cleaned DataFrame (df_ai) matching mas

# Start from the model content string
raw = content.strip()

# If the model wrapped the JSON in ```json ... ``` fences, strip them off
```

```
if raw.startswith("```"):
    raw = raw.strip("`")
    # remove leading 'json' if present (e.g. ```json)
    if raw.lower().startswith("json"):
        raw = raw[4:].strip()

# Now raw should be plain JSON
data = json.loads(raw)
rows = data["rows"]
df_ai_core = pd.DataFrame(rows)

# Attach metadata as before
df_ai = df_ai_core.copy()
df_ai["file_id"] = "1704"
df_ai["page_number"] = 1
df_ai["page_type"] = "ledger"
df_ai["page_title"] = "Computus Gu: Paynter Rectory a secundo die | novem

# Normalise None → "" for numeric-like fields
numeric_cols = ["amount_pounds", "amount_shillings", "amount_pence_whole"
for col in numeric_cols:
    if col in df_ai.columns:
        df_ai[col] = df_ai[col].apply(lambda x: "" if x is None else x)

# Clean pence fractions: map unicode, avoid weird values like "8" or "q"
def clean_pence(row):
    frac = str(row["amount_pence_fraction"]).strip()
    whole = row["amount_pence_whole"]

    # Map unicode fractions
    mapping_unicode = {
        "¼": "1/4",
        "½": "1/2",
        "¾": "3/4",
    }
    if frac in mapping_unicode:
        frac = mapping_unicode[frac]

    allowed = {"", "1/4", "1/2", "3/4"}

    # If fraction is a valid one, keep it
    if frac in allowed:
        return whole, frac

    # If fraction is a pure digit and whole is empty or zero,
    # treat it as whole pence and clear fraction
    if frac.isdigit() and (str(whole).strip() in ("", "0")):
        return int(frac), ""

    # Otherwise: treat as unknown, drop fraction
    return whole, ""

df_ai[["amount_pence_whole", "amount_pence_fraction"]] = df_ai.apply(
    lambda r: pd.Series(clean_pence(r)),
    axis=1,
)

# Recompute row_type helper fields
df_ai["is_total_row"] = df_ai["row_type"] == "total"
df_ai["group_brace_id"] = ""
```

```
df_ai["num_col_1"] = ""
df_ai["num_col_2"] = ""
df_ai["num_col_3"] = ""
df_ai["num_col_4"] = ""
df_ai["num_col_5"] = ""
df_ai["num_col_6"] = ""
df_ai["entry_confidence"] = "model"
df_ai["notes"] = ""

# Reorder to master schema
df_ai = df_ai[columns]
df_ai.head(10)
```

Out[26]:

| | file_id | page_number | page_type | page_title | row_index | row_type | date_raw | |
|---|---|---|---|---|---|---|---|---|
| **0** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 1 | title | | F |
| **1** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 2 | entry | | |
| **2** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 3 | entry | | It |
| **3** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 4 | entry | | |
| **4** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 5 | entry | | |
| **5** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 6 | entry | | |
| **6** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 7 | entry | | S |
| **7** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 8 | entry | | |
| **8** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 9 | entry | | |

| | file_id | page_number | page_type | page_title | row_index | row_type | date_raw |
|---|---|---|---|---|---|---|---|
| **9** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 10 | entry | |

10 rows × 22 columns

In [27]:
```
# Cell 8 — Validate that model output parses correctly & preview first ex

import json

raw = content.strip()

# If for any reason the model ever wraps output in ```json ``` fences, we
if raw.startswith("```"):
    raw = raw.strip("`")
    # remove leading language tag if present, e.g. ```json
    if raw.lower().startswith("json"):
        raw = raw[4:].strip()

data = json.loads(raw)
rows = data["rows"]
len(rows), rows[0]
```

Out[27]:
```
(28,
 {'row_index': 1,
  'row_type': 'title',
  'date_raw': '',
  'description': 'Computus Gu: Paynter Rectory a secundo die Novembris 1
703, ad eundem diem 1704',
  'amount_pounds': '',
  'amount_shillings': '',
  'amount_pence_whole': '',
  'amount_pence_fraction': ''})
```

In [28]:
```
# Cell 9 — Align df_ai with master schema and infer section_header rows f

df_ai_core = pd.DataFrame(rows)

df_ai = df_ai_core.copy()
df_ai["file_id"] = "1704"
df_ai["page_number"] = 1
df_ai["page_type"] = "ledger"
df_ai["page_title"] = "Computus Gu: Paynter Rectory a secundo die | novem

numeric_cols = ["amount_pounds", "amount_shillings", "amount_pence_whole"


# Optional: normalise numeric fields to strings/ints as needed
for col in ["amount_pounds", "amount_shillings", "amount_pence_whole"]:
    if col in df_ai.columns:
        df_ai[col] = df_ai[col].replace("", None)  # keep blanks as None

# Convert None to "" for consistency
```

```
for col in numeric_cols:
    if col in df_ai.columns:
        df_ai[col] = df_ai[col].apply(lambda x: "" if x is None else x)

# Add missing columns from our master schema and default values
df_ai["is_total_row"] = df_ai["row_type"] == "total"
df_ai["group_brace_id"] = ""
df_ai["num_col_1"] = ""
df_ai["num_col_2"] = ""
df_ai["num_col_3"] = ""
df_ai["num_col_4"] = ""
df_ai["num_col_5"] = ""
df_ai["num_col_6"] = ""
df_ai["entry_confidence"] = "model"
df_ai["notes"] = ""

# Now reorder to exactly match your master `columns` list
df_ai = df_ai[columns]

def has_any_amount(row):
    for col in ["amount_pounds", "amount_shillings", "amount_pence_whole"
        v = str(row[col]).strip()
        if v not in ("", "None"):
            return True
    return False

def fix_row_type(row):
    if not has_any_amount(row) and row["row_index"] != 1:
        # likely a structural label like "Iten: de arreragÿs", "merton",
        return "section_header"
    return row["row_type"]

df_ai["row_type"] = df_ai.apply(fix_row_type, axis=1)
df_ai["is_total_row"] = df_ai["row_type"] == "total"

df_ai.head(10)
```

Out[28]:

| | file_id | page_number | page_type | page_title | row_index | row_type | date_raw | |
|---|---|---|---|---|---|---|---|---|
| **0** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 1 | title | | F |
| **1** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 2 | entry | | |
| **2** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 3 | entry | | It |
| **3** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 4 | entry | | |
| **4** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 5 | entry | | |
| **5** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 6 | entry | | |
| **6** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 7 | entry | | S |
| **7** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 8 | entry | | |
| **8** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 9 | entry | | |

| | file_id | page_number | page_type | page_title | row_index | row_type | date_raw |
|---|---|---|---|---|---|---|---|
| **9** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 10 | entry | |

10 rows × 22 columns

In [29]:
```
# Cell 10 — Preview key ledger fields from df_ai for visual sanity-check

df_ai[[
    "row_index",
    "description",
    "amount_pounds",
    "amount_shillings",
    "amount_pence_whole",
    "amount_pence_fraction",
]]
```

Out[29]:

| | row_index | description | amount_pounds | amount_shillings | amount_pence_ |
|---|---|---|---|---|---|
| **0** | 1 | Computus Gu: Paynter Rectory a secundo die Nov... | | | |
| **1** | 2 | Imprimis redditus comput: de remanentib: in lista | 11 | 9 | |
| **2** | 3 | Item de arrearijs | 60 | 0 | |
| **3** | 4 | Guinear | 76 | 13 | |
| **4** | 5 | Long witnam | 20 | 0 | |
| **5** | 6 | Minhinnett | 30 | 0 | |
| **6** | 7 | Southnewington pro mij: | 3 | 0 | |
| **7** | 8 | Merton | 2 | 15 | |
| **8** | 9 | Chacehill | 3 | 8 | |
| **9** | 10 | Thrup | 4 | 0 | |
| **10** | 11 | Tue quit rents | 1 | 12 | |
| **11** | 12 | Ralph marshalls capons | 0 | 10 | |
| **12** | 13 | Calcott | 0 | 15 | |
| **13** | 14 | Bampton | 2 | 5 | |
| **14** | 15 | Mountaigne | 1 | 18 | |
| **15** | 16 | Hughes | 3 | 19 | |
| **16** | 17 | Greeneway | 1 | 15 | |
| **17** | 18 | Walker | 6 | 9 | |
| **18** | 19 | Minne | | | |
| **19** | 20 | Leaden-hall | | | |
| **20** | 21 | Molineux | | | |
| **21** | 22 | Tintinhull Napper | 2 | 5 | |
| **22** | 23 | Tintinhull Hopkins | 1 | 18 | |
| **23** | 24 | Clifton Ferry | 3 | 19 | |
| **24** | 25 | Bensington | 1 | 15 | |
| **25** | 26 | Brewhouse | 6 | 9 | |
| **26** | 27 | Merton Vicar. | | | |

| | row_index | description | amount_pounds | amount_shillings | amount_pence_v |
|---|---|---|---|---|---|
| **27** | 28 | Summa Arrearagionum receptarum | 230 | 13 | |

In [30]:
```
# Cell 11 — Extract manual rows (df_manual_sub) for specific file_id+page
# Subset manual data for this specific file/page (in case you add more la

df_manual_sub = df_manual[
    (df_manual["file_id"] == "1704") &
    (df_manual["page_number"] == 1)
].copy()

df_manual_sub
```

Out[30]:

| | file_id | page_number | page_type | page_title | row_index | row_type | date_ra |
|---|---|---|---|---|---|---|---|
| **0** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 1 | entry | |
| **1** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 2 | section_header | |
| **2** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 3 | entry | |
| **3** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 4 | entry | |
| **4** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 5 | entry | |
| **5** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 6 | entry | |
| **6** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 7 | section_header | |
| **7** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 8 | entry | |
| **8** | 1704 | 1 | ledger | Computus Gu: Paynter Rectory a secundo die \| n... | 9 | section_header | |

9 rows × 22 columns

In [31]:
```
# Cell 12 — Align AI rows with manual row indices (drop title, shift inde
# Work on a copy to avoid mutating the original df_ai

df_ai_sub = df_ai[
    (df_ai["file_id"] == "1704") &
    (df_ai["page_number"] == 1)
].copy()

# Drop the title row (row_index 1)
df_ai_sub = df_ai_sub[df_ai_sub["row_index"] != 1].copy()

# Shift row_index so AI rows line up with manual rows:
# AI row_index 2 → manual row_index 1, etc.
df_ai_sub["row_index"] = df_ai_sub["row_index"] - 1

df_ai_sub[["row_index", "description", "amount_pounds", "amount_shillings
```

Out[31]:

| | row_index | description | amount_pounds | amount_shillings | amount_pence_ |
|---|---|---|---|---|---|
| **1** | 1 | Imprimis redditus comput: de remanentib: in lista | 11 | 9 | |
| **2** | 2 | Item de arrearijs | 60 | 0 | |
| **3** | 3 | Guinear | 76 | 13 | |
| **4** | 4 | Long witnam | 20 | 0 | |
| **5** | 5 | Minhinnett | 30 | 0 | |
| **6** | 6 | Southnewington pro mij: | 3 | 0 | |
| **7** | 7 | Merton | 2 | 15 | |
| **8** | 8 | Chacehill | 3 | 8 | |
| **9** | 9 | Thrup | 4 | 0 | |
| **10** | 10 | Tue quit rents | 1 | 12 | |
| **11** | 11 | Ralph marshalls capons | 0 | 10 | |
| **12** | 12 | Calcott | 0 | 15 | |
| **13** | 13 | Bampton | 2 | 5 | |
| **14** | 14 | Mountaigne | 1 | 18 | |
| **15** | 15 | Hughes | 3 | 19 | |
| **16** | 16 | Greeneway | 1 | 15 | |
| **17** | 17 | Walker | 6 | 9 | |
| **18** | 18 | Minne | | | |
| **19** | 19 | Leaden-hall | | | |
| **20** | 20 | Molineux | | | |
| **21** | 21 | Tintinhull Napper | 2 | 5 | |
| **22** | 22 | Tintinhull Hopkins | 1 | 18 | |
| **23** | 23 | Clifton Ferry | 3 | 19 | |
| **24** | 24 | Bensington | 1 | 15 | |
| **25** | 25 | Brewhouse | 6 | 9 | |
| **26** | 26 | Merton Vicar. | | | |
| **27** | 27 | Summa Arrearagionum receptarum | 230 | 13 | |

In [32]:
```
# Cell 13 — Merge manual and AI data for page 1704/1 and compare key fiel

comparison = df_manual_sub.merge(
    df_ai_sub,
    on=["file_id", "page_number", "row_index"],
    suffixes=("_manual", "_ai"),
)

# Let's focus on key fields for now:
cols_to_show = [
    "row_index",
    "row_type_manual", "row_type_ai",
    "description_manual", "description_ai",
    "amount_pounds_manual", "amount_pounds_ai",
    "amount_shillings_manual", "amount_shillings_ai",
    "amount_pence_whole_manual", "amount_pence_whole_ai",
    "amount_pence_fraction_manual", "amount_pence_fraction_ai",
]

comparison[cols_to_show]
```

Out[32]:

| | row_index | row_type_manual | row_type_ai | description_manual | description_ai |
|---|---|---|---|---|---|
| **0** | 1 | entry | entry | Imprimis reddicht comput de remanentib .. m Cytas | Imprimis redditus comput: de remanentib: in lista |
| **1** | 2 | section_header | entry | Iten: de arreragÿs | Item de arrearijs |
| **2** | 3 | entry | entry | Guinear | Guinear |
| **3** | 4 | entry | entry | Long witnam | Long witnam |
| **4** | 5 | entry | entry | minhinnett | Minhinnett |
| **5** | 6 | entry | entry | South newington | Southnewington pro mij: |
| **6** | 7 | section_header | entry | merton | Merton |
| **7** | 8 | entry | entry | chacehill | Chacehill |
| **8** | 9 | section_header | entry | Thrup | Thrup |

In [ ]:
```
# Cell 14 — Function: extract structured rows for any PDF page using mult

def extract_page_rows_with_ai(file_id: str, pdf_path: str, page_number: i
    """
    Extract ledger rows from a given PDF page using the multimodal model,
    then clean and normalize to match the master schema (columns).
    Returns df_page: rows from this page only.
    """

    # 1️⃣ Convert page to base64
    img = pdf_page_to_image(pdf_path, page_number)
    img_b64 = pil_image_to_base64(img)
```

```
    # 2️⃣ Call the model
    response = client.chat.completions.create(
        model=model_name,
        messages=[
            {"role": "system", "content": system_prompt},
            {
                "role": "user",
                "content": [
                    {"type": "text", "text": "Please read this ledger pag
                    {"type": "image_url", "image_url": {"url": f"data:ima
                ],
            },
        ],
    )

    raw = response.choices[0].message.content.strip()

    # 3️⃣ Strip ```json fences if present
    if raw.startswith("```"):
        raw = raw.strip("`")
        if raw.lower().startswith("json"):
            raw = raw[4:].strip()

    data = json.loads(raw)
    rows = data["rows"]
    df_core = pd.DataFrame(rows)

    # 4️⃣ Metadata
    df_page = df_core.copy()
    df_page["file_id"] = file_id
    df_page["page_number"] = page_number
    df_page["page_type"] = "ledger"  # may adjust for last 3 PDFs later
    df_page["page_title"] = ""       # (optional: detect title separately

    # 5️⃣ Normalize numeric types
    numeric_cols = ["amount_pounds", "amount_shillings", "amount_pence_wh
    for col in numeric_cols:
        if col in df_page.columns:
            df_page[col] = df_page[col].apply(lambda x: "" if x is None e

    # 6️⃣ Pence cleanup — avoid weird digits like "8", map unicode
    df_page[["amount_pence_whole", "amount_pence_fraction"]] = df_page.ap
        lambda r: pd.Series(clean_pence(r)),
        axis=1,
    )

    # 7️⃣ Section header inference — same rules as manual alignment cell
    df_page["row_type"] = df_page.apply(fix_row_type, axis=1)

    # 8️⃣ Schema alignment
    df_page["is_total_row"] = df_page["row_type"] == "total"
    df_page["group_brace_id"] = ""
    df_page["num_col_1"] = ""
    df_page["num_col_2"] = ""
    df_page["num_col_3"] = ""
    df_page["num_col_4"] = ""
    df_page["num_col_5"] = ""
    df_page["num_col_6"] = ""
    df_page["entry_confidence"] = "model"
    df_page["notes"] = ""
```

```
    df_page = df_page[columns]

    return df_page
```

In [ ]:
```
# Cell 15 — Loop over all PDFs and pages in ./data to build combined df_a

import os
import fitz  # already imported above, but harmless to repeat

data_dir = "data"  # folder where all PDF files live
all_dfs = []
errors = []

for fname in sorted(os.listdir(data_dir)):
    if not fname.lower().endswith(".pdf"):
        continue  # skip non-PDFs if any

    file_id = os.path.splitext(fname)[0]  # e.g. "1704"
    pdf_path = os.path.join(data_dir, fname)

    # Count pages
    doc = fitz.open(pdf_path)
    num_pages = doc.page_count
    doc.close()

    print(f"Processing {file_id} ({num_pages} pages)...")

    for page_no in range(1, num_pages + 1):
        try:
            df_page = extract_page_rows_with_ai(file_id, pdf_path, page_n
            df_page["source_file"] = fname  # optional provenance column
            all_dfs.append(df_page)
        except Exception as e:
            print(f"⚠️ Error on {file_id}, page {page_no}: {e}")
            errors.append({
                "file_id": file_id,
                "page_number": page_no,
                "error": str(e),
            })

# Combine everything into one big DataFrame
if all_dfs:
    df_all = pd.concat(all_dfs, ignore_index=True)
else:
    df_all = pd.DataFrame(columns=columns)

print("Total rows extracted:", len(df_all))
df_all.head(10)
```

```
Processing 1704 (7 pages)...
```

In [ ]:
```
# Cell 16 — Excel format Extraction

# Export the final dataset to Excel
output_path = "ledger_transcription.xlsx"
df_all.to_excel(output_path, index=False)
output_path
```

Out[ ]:
```
'ledger_transcription.xlsx'
```

In [ ]:
```
# Cell 17 – Saving Supporting MetaData

# Save any model extraction failures for documentation
pd.DataFrame(errors).to_csv("ai_extraction_errors.csv", index=False)
len(errors)
```

Out[ ]:
```
0
```

In [33]:
```
# Cell Y – Export manual vs AI comparison table as PNG

import matplotlib.pyplot as plt

# Select key columns to show
comparison_subset = comparison[[
    "row_index",
    "row_type_manual", "row_type_ai",
    "description_manual", "description_ai",
    "amount_pounds_manual", "amount_pounds_ai",
    "amount_shillings_manual", "amount_shillings_ai",
    "amount_pence_whole_manual", "amount_pence_whole_ai",
    "amount_pence_fraction_manual", "amount_pence_fraction_ai",
]].copy()

# Create figure sized to number of rows
n_rows = len(comparison_subset)
fig_height = 1.5 + 0.4 * n_rows  # tweak if needed

fig, ax = plt.subplots(figsize=(12, fig_height))
ax.axis("off")

table = ax.table(
    cellText=comparison_subset.values,
    colLabels=comparison_subset.columns,
    loc="center",
)

table.auto_set_font_size(False)
table.set_fontsize(7)
table.auto_set_column_width(col=list(range(len(comparison_subset.columns))))

plt.tight_layout()

comparison_img_path = "images/comparison_table_1704.png"
plt.savefig(comparison_img_path, dpi=300, bbox_inches="tight")
plt.close(fig)

comparison_img_path
```

Out[33]:
```
'images/comparison_table_1704.png'
```

In [ ]:
```

```