Extraction_Italy_SII_demo

September 23, 2025

1 Italian Solvency reports Table S.02.01.02; Part 1 Extraction

The scope of this script is to transcribe SFCR table S.02.01.02 for the 18 life insurance companies on the Italian market. The notebook is organized into the following sections:

1) Companies and tables in scope
2) Packages and tools
3) Generation of DataFrames

1.1 Companies in scope

For the year 2024, the companies in scope are the following:

- Credemvita S.p.A.
- AXA MPS Assicurazioni Vita
- CRÈDIT AGRICOLE VITA
- Società Reale Mutua di Assicurazioni
- Cardif Vita S.p.A.
- MEDIOLANUM VITA S.p.A.
- Generali Italia S.p.A.
- Banco BPM Vita S.p.A.
- HDI ASSICURAZIONI S.p.A.
- Gruppo Assicurativo Poste Vita
- FIDEURAM VITA S.P.A.
- CNP Vita Assicura S.p.A.
- ITAS VITA
- Helvetia Vita S.p.A.
- Vittoria Assicurazioni S.p.A.
- GROUPAMA ASSICURAZIONI S.P.A.
- UniCredit Allianz Vita S.p.A.
- Zurich Investments Life S.p.A.

1.1.1 Solvency and Financial condition reports

According to Article 51 of the Solvency II Directive 2009/138/EC, companies under the regulatory umbrella of EIOPA, companies must publish annually a Solvency and Financial Condition Reports (SFCR) for all legal entities.

Part of the report is mandatory tables that show some financial and actuarial indicators. One such table is S.02.01.02 which shows a simplified balance sheet of the legal entity. This table is inside

the scope of this demo.

1.2 Description of the process

The process of extraction is performed in 5 phases:

1.2.1 Phase 1: Find the reports and identify the relevant tables.

1) Identify the new SFCR report and save it into the folder Input.
2) Identify the pages where the tables of interest are.
3) Compile the map of the company run in the master_list.csv.

1.2.2 Phase 2: Run the Extraction script (this script).

The script performs the following steps (with slight modifications depending on the table format): 1) Save the page with the table into a separate folder Single_pdf. 2) Use either a Python package or specialized LLM to create a digital equivalent of the table. 3) Fix the systemic errors that prevent the table from being saved as DataFrame. 4) Save the DataFrame into the Output folder.

1.2.3 Phase 3: Run the Processing script.

The script applies fixes to the DataFrame to make the numbers closer to the reported numbers. It joins all the tables into a single dataset.

1.2.4 Phase 4: Run the Cross-Validation script.

Applies a series of tests that check for the internal consistency between the numbers. Flags the potential errors.

1.2.5 Phase 5: Final modifications to the table and a manual inspection.

1.3 Necessary Python packages

[1]:
```
pip install PyPDF2 pdfplumber
```

```
Requirement already satisfied: PyPDF2 in c:\users\grego\anaconda3\lib\site-
packages (3.0.1)
Requirement already satisfied: pdfplumber in c:\users\grego\anaconda3\lib\site-
packages (0.11.7)
Requirement already satisfied: pdfminer.six==20250506 in
c:\users\grego\anaconda3\lib\site-packages (from pdfplumber) (20250506)
Requirement already satisfied: Pillow>=9.1 in c:\users\grego\anaconda3\lib\site-
packages (from pdfplumber) (10.3.0)
Requirement already satisfied: pypdfium2>=4.18.0 in
c:\users\grego\anaconda3\lib\site-packages (from pdfplumber) (4.30.0)
Requirement already satisfied: charset-normalizer>=2.0.0 in
c:\users\grego\anaconda3\lib\site-packages (from
pdfminer.six==20250506->pdfplumber) (2.0.4)
Requirement already satisfied: cryptography>=36.0.0 in
c:\users\grego\anaconda3\lib\site-packages (from
```

```
pdfminer.six==20250506->pdfplumber) (42.0.5)
Requirement already satisfied: cffi>=1.12 in c:\users\grego\anaconda3\lib\site-
packages (from cryptography>=36.0.0->pdfminer.six==20250506->pdfplumber)
(1.16.0)
Requirement already satisfied: pycparser in c:\users\grego\anaconda3\lib\site-
packages (from
cffi>=1.12->cryptography>=36.0.0->pdfminer.six==20250506->pdfplumber) (2.21)
Note: you may need to restart the kernel to use updated packages.
```

[2]:
```
!pip install mistralai
```

```
Requirement already satisfied: mistralai in c:\users\grego\anaconda3\lib\site-
packages (1.9.3)
Requirement already satisfied: eval-type-backport>=0.2.0 in
c:\users\grego\anaconda3\lib\site-packages (from mistralai) (0.2.2)
Requirement already satisfied: httpx>=0.28.1 in
c:\users\grego\anaconda3\lib\site-packages (from mistralai) (0.28.1)
Requirement already satisfied: pydantic>=2.10.3 in
c:\users\grego\anaconda3\lib\site-packages (from mistralai) (2.11.7)
Requirement already satisfied: python-dateutil>=2.8.2 in
c:\users\grego\anaconda3\lib\site-packages (from mistralai) (2.9.0.post0)
Requirement already satisfied: typing-inspection>=0.4.0 in
c:\users\grego\anaconda3\lib\site-packages (from mistralai) (0.4.1)
Requirement already satisfied: anyio in c:\users\grego\anaconda3\lib\site-
packages (from httpx>=0.28.1->mistralai) (4.2.0)
Requirement already satisfied: certifi in c:\users\grego\anaconda3\lib\site-
packages (from httpx>=0.28.1->mistralai) (2025.8.3)
Requirement already satisfied: httpcore==1.* in
c:\users\grego\anaconda3\lib\site-packages (from httpx>=0.28.1->mistralai)
(1.0.9)
Requirement already satisfied: idna in c:\users\grego\anaconda3\lib\site-
packages (from httpx>=0.28.1->mistralai) (3.7)
Requirement already satisfied: h11>=0.16 in c:\users\grego\anaconda3\lib\site-
packages (from httpcore==1.*->httpx>=0.28.1->mistralai) (0.16.0)
Requirement already satisfied: annotated-types>=0.6.0 in
c:\users\grego\anaconda3\lib\site-packages (from pydantic>=2.10.3->mistralai)
(0.6.0)
Requirement already satisfied: pydantic-core==2.33.2 in
c:\users\grego\anaconda3\lib\site-packages (from pydantic>=2.10.3->mistralai)
(2.33.2)
Requirement already satisfied: typing-extensions>=4.12.2 in
c:\users\grego\anaconda3\lib\site-packages (from pydantic>=2.10.3->mistralai)
(4.14.1)
Requirement already satisfied: six>=1.5 in c:\users\grego\anaconda3\lib\site-
packages (from python-dateutil>=2.8.2->mistralai) (1.16.0)
Requirement already satisfied: sniffio>=1.1 in
c:\users\grego\anaconda3\lib\site-packages (from
anyio->httpx>=0.28.1->mistralai) (1.3.0)
```

[3]:
```
!pip install pycryptodome
```

```
Requirement already satisfied: pycryptodome in
c:\users\grego\anaconda3\lib\site-packages (3.23.0)
```

1.4 Python packages

[4]:
```
import re
import os
import json
import base64
import pdfplumber
import pandas as pd
from pathlib import Path
from PyPDF2 import PdfReader, PdfWriter
from mistralai.models import OCRResponse
from IPython.display import Markdown, display
from mistralai import Mistral, DocumentURLChunk, ImageURLChunk, TextChunk
```

1.5 API key

The model of choice for this project is Mistral. This was identified as the most economical choice. Additionally, Mistral released a custom model for OCR tasks.

Useful links:

- https://docs.mistral.ai/getting-started/quickstart/
- https://mistral.ai/news/mistral-ocr

[5]:
```
api_key = "[YOUR MISTRAL API KEY]"
```

1.6 Functions

[6]:
```
def set_code_index_and_save(table: pd.DataFrame, path: str) -> None:
    """
    Set the 'CODE' column as the index of a DataFrame and save it as a CSV file.
    """
    table = table.set_index("CODE")
    table.to_csv(path)
```

[7]:
```
def extract_tables_from_pdf(pdf_path: str, page_number:int=1) -> pd.DataFrame:
    """
    Extract the first table from a specific page of a PDF and return it as a␣
 ↪DataFrame.

    Parameters
    ----------
    pdf_path : str or pathlib.Path
```

```
        Path to the PDF file.
    page_number : int, optional (default=1)
        The page number to extract the table from (1-indexed, i.e., first page␣
↪is 1).

    Returns
    -------
    pd.DataFrame
        A DataFrame containing the extracted table. The first row of the table␣
↪is
        treated as the header, and the remaining rows as data.

    """

    with pdfplumber.open(pdf_path) as pdf:
        page = pdf.pages[page_number - 1]  # Pages are zero-indexed
        tables = page.extract_tables()
        df_tables = pd.DataFrame(tables[0][1:], columns=tables[0][0])
        return df_tables
```

[8]:
```
def extract_page(input_pdf_path: str, output_pdf_path: str, page_number: int,␣
↪password: str = "")-> None:
    """
    Extract a single page from a PDF file and save it as a new PDF.
    """

    pdf_reader = PdfReader(input_pdf_path)
    pdf_writer = PdfWriter()

    # Decrypt if necessary
    if pdf_reader.is_encrypted:
        if password:
            pdf_reader.decrypt(password)
        else:
            pdf_reader.decrypt("")  # try empty password

    # Add the specified page to the PdfWriter object
    pdf_writer.add_page(pdf_reader.pages[page_number - 1])

    # Write the selected page to a new PDF file
    with open(output_pdf_path, 'wb') as output_pdf_file:
        pdf_writer.write(output_pdf_file)
```

[9]:
```
def encode_pdf(pdf_path: str)-> None:
    """Encode the pdf to base64."""
    try:
        with open(pdf_path, "rb") as pdf_file:
```

```
            return base64.b64encode(pdf_file.read()).decode('utf-8')
    except FileNotFoundError:
        print(f"Error: The file {pdf_path} was not found.")
        return None
    except Exception as e:  # Added general exception handling
        print(f"Error: {e}")
        return None
```

[10]:
```
def markdown_table_to_dataframe(markdown_text, shift_row = 0):
    """
    Convert a Markdown-formatted table into a pandas DataFrame.
    """

    # Split the markdown text into lines
    lines = markdown_text.strip().split('\n')

    # Filter out lines that are part of the table
    table_lines = [line for line in lines if line.startswith('|')]

    # Remove the markdown table syntax
    cleaned_lines = [line.strip('|').strip() for line in table_lines]

    # Split each line into columns
    data = [line.split('|') for line in cleaned_lines]

    # Extract headers and rows
    headers = [header.strip() for header in data[1+shift_row] if header.strip()]
    rows = [[cell.strip() for cell in row if cell.strip()] for row in data[3:]]

    # Ensure each row has the same number of columns as the headers
    for row in rows:
        if len(row) < len(headers):
            row += [''] * (len(headers) - len(row))

    # Create a DataFrame
    df = pd.DataFrame(rows, columns=headers)
    return df
```

[11]:
```
def split_strings_to_df(strings):
    """
    Convert a list of strings in the format `"text ..... number"` into a␣
 ↪DataFrame.

    Each string is expected to contain a text label followed by dots and a␣
 ↪numeric value.
    The numeric value is cleaned (spaces and dots removed) and converted to␣
 ↪either
```

```
    `int` or `float`. If no numeric value is found, `None` is used instead.

    Parameters
    ----------
    strings : list of str
        A list of strings to be parsed.
        Example: ["Technical provisions ..... 2.337.991", "Best Estimate .....␣
↪0"]

    Returns
    -------
    pd.DataFrame
        A DataFrame with two columns:
        - "NAME": str, the extracted text (trimmed of dots and whitespace).
        - "C0010": int, float, or None, the extracted numeric value.

    Notes
    -----
    - Numbers with thousand separators like `"2.337.991"` are cleaned into␣
↪integers (`2337991`).
    - If the number cannot be parsed, it remains as a string.
    - Strings without a numeric value will have `None` in the `"C0010"` column.
    """

    data = []
    for s in strings:
        # Try to match "text ..... number"
        match = re.match(r'^(.*?)\.{2,}\s*([\d\s\.]+)$', s.strip())
        if match:
            left = match.group(1).strip()
            # Clean up number (remove spaces in middle, convert to float/int)
            right = match.group(2).replace(" ", "").replace(".", "")
            if right.isdigit():
                right = int(right)
            else:
                try:
                    right = float(right)
                except:
                    pass
            data.append((left, right))
        else:
            # No number found, just keep text
            data.append((s.strip(), None))

    df = pd.DataFrame(data, columns=["NAME", "C0010"])
    return df
```

[12]:
```python
def run_mistral_ocr(output_pdf_path: str, api_key: str)-> str:
    """
    Run OCR on a PDF file using the Mistral OCR API and return the extracted␣
 ↪text in Markdown format.

    The function:
    1. Encodes the PDF as a base64 string.
    2. Sends it to the Mistral OCR API.
    3. Extracts the recognized text in Markdown format from the response.
    """

    # Getting the base64 string
    base64_pdf = encode_pdf(Path(output_pdf_path))

    client = Mistral(api_key=api_key)

    ocr_response = client.ocr.process(
        model="mistral-ocr-latest",
        document={
            "type": "document_url",
            "document_url": f"data:application/pdf;base64,{base64_pdf}"
        },
        include_image_base64=True
    )

    # Assuming ocr_response is your instance of OCRResponse
    markdown_texts = [page.markdown for page in ocr_response.pages]

    # If you want to concatenate all markdown texts from all pages
    full_markdown_text = "\n".join(markdown_texts)

    # Assuming ocr_response is your OCRResponse object
    markdown_text = ocr_response.pages[0].markdown
    return markdown_text
```

[13]:
```python
def parse_table_to_df(text: str, value_column_name:str) -> pd.DataFrame:
    """
    Parse a long text table into a DataFrame with columns:
    ['code', 'name', 'value'].
    """
    # Split text into lines
    lines = text.splitlines()

    rows = []
    buffer = []
    for line in lines:
```

```
        # Match lines with pattern: code (Rxxxx) followed by number
        match = re.match(r"^(R\d{4})\s*(.*)$", line)
        if match:
            code = match.group(1)
            rest = match.group(2).strip()

            # Check if 'rest' is a number (the value)
            if re.match(r"^[\d\.,]+$", rest):
                name = " ".join(buffer).strip()
                value = rest
                rows.append((code, name, value))
                buffer = []  # reset for next name
            else:
                # It's not just a number, so treat it as name continuation
                buffer.append(line)
        else:
            # If line is just a number after a code line
            if re.match(r"^[\d\.,]+$", line.strip()):
                name = " ".join(buffer).strip()
                value = line.strip()
                # Last buffer entry should contain the code
                code_match = re.search(r"(R\d{4})", buffer[-1]) if buffer else␣
↪None
                code = code_match.group(1) if code_match else None
                # Remove code from name
                buffer[-1] = buffer[-1].replace(code, "").strip() if code else␣
↪buffer[-1]
                rows.append((code, " ".join(buffer).strip(), value))
                buffer = []  # reset
            else:
                # Accumulate description
                buffer.append(line)

    # Build DataFrame
    df = pd.DataFrame(rows, columns=["CODE", "NAME", value_column_name])

    # Convert numeric values to float
    df[value_column_name] = df[value_column_name].str.replace(".", "",␣
↪regex=False).str.replace(",", ".", regex=False)
    df[value_column_name] = pd.to_numeric(df[value_column_name],␣
↪errors="coerce")

    return df
```

[14]:
```
def extract_paths(master_list: pd.DataFrame, unique_id: str):
    """
    Extract file paths and page number from master_list for a given unique_id.
```

```
    Parameters:
        master_list (pd.DataFrame): DataFrame containing metadata.
        unique_id: Index or identifier in the DataFrame.

    Returns:
        dict: Dictionary with keys 'pdf_path', 'page_number',
              'output_pdf_path', 'output_final_path', 'codes_path'.
    """
    return master_list.loc[unique_id, "document_name"], int(master_list.
↪loc[unique_id, "page_number"]), master_list.loc[unique_id,␣
↪"output_pdf_path"], master_list.loc[unique_id, "output_final_path"],␣
↪master_list.loc[unique_id, "codes_path"]
```

[15]:
```
def convert_to_dataframe(text: str) -> pd.DataFrame:
    """
    Convert raw AnnexI/Solvency II balance sheet text into a structured␣
↪DataFrame.

    Parameters
    ----------
    text : str
        Raw text containing balance sheet items.

    Returns
    -------
    pd.DataFrame
        DataFrame with columns: CODE, NAME, VALUE
    """
    rows = []

    # Split text into lines
    for line in text.splitlines():
        # Match rows with a code like R001 and a value at the end
        match = re.match(r"^(.*)\s+(R\d+)\s+([-]?\d[\d\.\,]*)$", line.strip())
        if match:
            name = match.group(1).strip()
            code = match.group(2).strip()
            value_str = match.group(3).replace(".", "")  # remove thousand␣
↪separators
            value = float(value_str.replace(",", "."))   # convert to float
            rows.append((code, name, value))

    df = pd.DataFrame(rows, columns=["CODE", "NAME", "VALUE"])
    return df
```

[16]:
```python
def append_zero_if_len4(df: pd.DataFrame, col: str = "CODE") -> pd.DataFrame:
    """
    Append '0' to the code if its length is 4 characters.

    Parameters
    ----------
    df : pd.DataFrame
        Input dataframe containing the code column.
    col : str
        Column name for codes (default = 'CODE').

    Returns
    -------
    pd.DataFrame
        DataFrame with modified codes.
    """
    df = df.copy()
    df[col] = df[col].apply(lambda x: x + "0" if len(x) == 4 else x)
    return df
```

[17]:
```python
def run_ocr_and_convert_to_df(path: str, api_key: str)-> pd.DataFrame:
    """
    Run OCR on a PDF file and convert the extracted Markdown table into a␣
 ↪DataFrame.
    """

    markdown_text = run_mistral_ocr(path, api_key)
    table = markdown_table_to_dataframe(markdown_text)
    return table
```

[18]:
```python
def ocr_to_dataframe(ocr_text: str) -> pd.DataFrame:
    """
    Converts OCR extracted financial text into a structured DataFrame.

    Extracts description, code (Rxxxx, Cxxxx), and numeric value if present.
    """
    rows = []

    # Split text into lines
    for line in ocr_text.splitlines():
        # Look for patterns like "..... R0030 ..... 123.456"
        match = re.match(r"^(.*?)\.{2,}\s*(R\d{4}|C\d{4})(?:\s*\.{2,}\s*([\d\.
 ↪\s-]+))?$", line.strip())
        if match:
            desc = match.group(1).strip()
            code = match.group(2).strip()
            raw_value = match.group(3)
```

```
            if raw_value:
                # Remove spaces inside numbers (e.g. "4.275 .590" -> "4.275.
↪590")
                cleaned = raw_value.replace(" ", "")
                # Convert to float if numeric, else keep as string
                try:
                    value = float(cleaned.replace(".", "").replace(",", "."))
                except ValueError:
                    value = cleaned
            else:
                value = None

            rows.append((code, desc, value))

    df = pd.DataFrame(rows, columns=["CODE", "DESCRIPTION", "VALUE"])
    return df
```

1.7 The list of companies

[19]:
```
master_list = pd.read_csv("master_list.csv", header=0, index_col=0)
```

[20]:
```
display(master_list)
```

```
                                company  \
id
CREDEM_VITA_02_1                 CREDEM
CREDEM_VITA_02_2                 CREDEM
AXA_VITA_02_01                      AXA
CREDAG_VITA_02_01       CREDIT_AGRICOLE
CREDAG_VITA_02_02       CREDIT_AGRICOLE
REALE_02_01                 REALE_MUTUA
REALE_02_02                 REALE_MUTUA
CARDIF_02_01                     CARDIF
MEDIO_02_01                   MEDIOLANUM
MEDIO_02_02                   MEDIOLANUM
GEN_ITA_02_01           GENERALI ITALIA
GEN_ITA_02_02           GENERALI ITALIA
BMP_VITA_02_01                 BMP VITA
BMP_VITA_02_02                 BMP VITA
HDI_01                              HDI
HDI_02                              HDI
POSTE_VITA_01                POSTE_VITA
POSTE_VITA_02                POSTE_VITA
INTESA_VITA_01              INTESA_VITA
INTESA_VITA_02              INTESA_VITA
CNP_VITA_01                    CNP_VITA
CNP_VITA_02                    CNP_VITA
```

```
ITAS_VITA_01                  ITAS_VITA
ITAS_VITA_02                  ITAS_VITA
HELVETIA_VITA_01          HELVETIA_VITA
HELVETIA_VITA_02          HELVETIA_VITA
VITTORIA_01                    VITTORIA
VITTORIA_02                    VITTORIA
GROUPAMA_01                    GROUPAMA
ALLIANZ_UNICREDIT_01  ALLIANZ_UNICREDIT
ALLIANZ_UNICREDIT_02  ALLIANZ_UNICREDIT
ZURICH_LIFE_01              ZURICH_LIFE
ZURICH_LIFE_02              ZURICH_LIFE

                                                          document_name  \
id
CREDEM_VITA_02_1                        Input\SFCR 2024 CREDEMVITA.pdf
CREDEM_VITA_02_2                        Input\SFCR 2024 CREDEMVITA.pdf
AXA_VITA_02_01        Input\2024.12 QRT SFCR AXA MPS Assicurazioni V…
CREDAG_VITA_02_01                          Input\ca_vita_sfcr_2024.pdf
CREDAG_VITA_02_02                          Input\ca_vita_sfcr_2024.pdf
REALE_02_01                              Input\SFCR_A123S_20241231.pdf
REALE_02_02                              Input\SFCR_A123S_20241231.pdf
CARDIF_02_01                             Input\SFCR_A421S_20241231.pdf
MEDIO_02_01             Input\Mediolanum_Vita_Relazione_Unica_2024.pdf
MEDIO_02_02             Input\Mediolanum_Vita_Relazione_Unica_2024.pdf
GEN_ITA_02_01         Input\2024 Relazione sulla Solvibilita e Condi…
GEN_ITA_02_02         Input\2024 Relazione sulla Solvibilita e Condi…
BMP_VITA_02_01        Input\Fascicolo-SFCR_2024_BBPMV-signed_con-opi…
BMP_VITA_02_02        Input\Fascicolo-SFCR_2024_BBPMV-signed_con-opi…
HDI_01                Input\HDI Assicurazioni S.p.A. - Avviso pubbli…
HDI_02                Input\HDI Assicurazioni S.p.A. - Avviso pubbli…
POSTE_VITA_01         Input\relazioneunicasolvibilita_condizionefina…
POSTE_VITA_02         Input\relazioneunicasolvibilita_condizionefina…
INTESA_VITA_01                  Input\SFCR Gruppo ISPA 2024_ENG.pdf
INTESA_VITA_02                  Input\SFCR Gruppo ISPA 2024_ENG.pdf
CNP_VITA_01           Input\SFCR_2024_CNP_Vita_Assicura_con_Relazion…
CNP_VITA_02           Input\SFCR_2024_CNP_Vita_Assicura_con_Relazion…
ITAS_VITA_01                           Input\SFCR_G0010_20241231.pdf
ITAS_VITA_02                           Input\SFCR_G0010_20241231.pdf
HELVETIA_VITA_01             Input\SFCR_Annex_Helvetia Vita_2024.pdf
HELVETIA_VITA_02             Input\SFCR_Annex_Helvetia Vita_2024.pdf
VITTORIA_01                             Input\SFCR-2024_Vittoria.pdf
VITTORIA_02                             Input\SFCR-2024_Vittoria.pdf
GROUPAMA_01           Input\SFCR-2024-Relazione-solvibilita-condizio…
ALLIANZ_UNICREDIT_01  Input\UAV_Fascicolo_SFCR_2024_con_relazioni_Pw…
ALLIANZ_UNICREDIT_02  Input\UAV_Fascicolo_SFCR_2024_con_relazioni_Pw…
ZURICH_LIFE_01                           Input\zil_scfr_2024_web.pdf
ZURICH_LIFE_02                           Input\zil_scfr_2024_web.pdf
```

id	table_name	page_number	output_pdf_path
CREDEM_VITA_02_1	S.02.01.02_A	197	Single_pdf/CREDEM_S02_01_02_1_2024.pdf
CREDEM_VITA_02_2	S.02.01.02_L	198	Single_pdf/CREDEM_S02_01_02_2_2024.pdf
AXA_VITA_02_01	S.02.01.02_A	1	Single_pdf/AXA_S02_01_02_1_2024.pdf
CREDAG_VITA_02_01	S.02.01.02_A	61	Single_pdf/CA_S02_01_02_1_2024.pdf
CREDAG_VITA_02_02	S.02.01.02_L	62	Single_pdf/CA_S02_01_02_2_2024.pdf
REALE_02_01	S.02.01.02_A	158	Single_pdf/RM_S02_01_02_1_2024.pdf
REALE_02_02	S.02.01.02_L	159	Single_pdf/RM_S02_01_02_2_2024.pdf
CARDIF_02_01	S.02.01.02_A	106	Single_pdf/CARDIF_S02_01_02_1_2024.pdf
MEDIO_02_01	S.02.01.02_A	164	Single_pdf/MEDIO_S02_01_02_1_2024.pdf
MEDIO_02_02	S.02.01.02_L	165	Single_pdf/MEDIO_S02_01_02_2_2024.pdf
GEN_ITA_02_01	S.02.01.02_A	144	
GEN_ITA_02_02	S.02.01.02_L	145	
BMP_VITA_02_01	S.02.01.02_A	80	
BMP_VITA_02_02	S.02.01.02_L	81	
HDI_01	S.02.01.02_A	104	
HDI_02	S.02.01.02_L	105	
POSTE_VITA_01	S.02.01.02_A	136	
POSTE_VITA_02	S.02.01.02_L	137	
INTESA_VITA_01	S.02.01.02_A	250	
INTESA_VITA_02	S.02.01.02_L	251	
CNP_VITA_01	S.02.01.02_A	116	
CNP_VITA_02	S.02.01.02_L	117	
ITAS_VITA_01	S.02.01.02_A	233	
ITAS_VITA_02	S.02.01.02_L	234	
HELVETIA_VITA_01	S.02.01.02_A	2	
HELVETIA_VITA_02	S.02.01.02_L	3	
VITTORIA_01	S.02.01.02_A	93	
VITTORIA_02	S.02.01.02_L	94	
GROUPAMA_01	S.02.01.02_A	71	
ALLIANZ_UNICREDIT_01	S.02.01.02_A	110	
ALLIANZ_UNICREDIT_02	S.02.01.02_L	111	
ZURICH_LIFE_01	S.02.01.02_A	58	
ZURICH_LIFE_02	S.02.01.02_L	59	

```
GEN_ITA_02_01                   Single_pdf/GEN_ITA_S02_01_02_1_2024.pdf
GEN_ITA_02_02                   Single_pdf/GEN_ITA_S02_01_02_2_2024.pdf
BMP_VITA_02_01                 Single_pdf/BMP_VITA_S02_01_02_1_2024.pdf
BMP_VITA_02_02                 Single_pdf/BMP_VITA_S02_01_02_2_2024.pdf
HDI_01                              Single_pdf/HDI_S02_01_02_1_2024.pdf
HDI_02                              Single_pdf/HDI_S02_01_02_2_2024.pdf
POSTE_VITA_01                Single_pdf/POSTE_VITA_S02_01_02_1_2024.pdf
POSTE_VITA_02                Single_pdf/POSTE_VITA_S02_01_02_2_2024.pdf
INTESA_VITA_01              Single_pdf/INTESA_VITA_S02_01_02_1_2024.pdf
INTESA_VITA_02              Single_pdf/INTESA_VITA_S02_01_02_2_2024.pdf
CNP_VITA_01                    Single_pdf/CNP_VITA_S02_01_02_1_2024.pdf
CNP_VITA_02                    Single_pdf/CNP_VITA_S02_01_02_2_2024.pdf
ITAS_VITA_01                  Single_pdf/ITAS_VITA_S02_01_02_1_2024.pdf
ITAS_VITA_02                  Single_pdf/ITAS_VITA_S02_01_02_2_2024.pdf
HELVETIA_VITA_01          Single_pdf/HELVETIA_VITA_S02_01_02_1_2024.pdf
HELVETIA_VITA_02          Single_pdf/HELVETIA_VITA_S02_01_02_2_2024.pdf
VITTORIA_01                     Single_pdf/VITTORIA_S02_01_02_1_2024.pdf
VITTORIA_02                     Single_pdf/VITTORIA_S02_01_02_2_2024.pdf
GROUPAMA_01                    Single_pdf/GROUPAMA_S02_01_02_1_2024.pdf
ALLIANZ_UNICREDIT_01  Single_pdf/ALLIANZ_UNICREDIT_S02_01_02_1_2024.pdf
ALLIANZ_UNICREDIT_02  Single_pdf/ALLIANZ_UNICREDIT_S02_01_02_2_2024.pdf
ZURICH_LIFE_01              Single_pdf/ZURICH_LIFE_S02_01_02_1_2024.pdf
ZURICH_LIFE_02              Single_pdf/ZURICH_LIFE_S02_01_02_2_2024.pdf

                                                    output_final_path  \
id
CREDEM_VITA_02_1                Output/CREDEM_S02_01_02_1_2024.csv
CREDEM_VITA_02_2                Output/CREDEM_S02_01_02_2_2024.csv
AXA_VITA_02_01                     Output/AXA_S02_01_02_1_2024.csv
CREDAG_VITA_02_01                   Output/CA_S02_01_02_1_2024.csv
CREDAG_VITA_02_02                   Output/CA_S02_01_02_2_2024.csv
REALE_02_01                         Output/RM_S02_01_02_1_2024.csv
REALE_02_02                         Output/RM_S02_01_02_2_2024.csv
CARDIF_02_01                   Output/CARDIF_S02_01_02_1_2024.csv
MEDIO_02_01                     Output/MEDIO_S02_01_02_1_2024.csv
MEDIO_02_02                     Output/MEDIO_S02_01_02_2_2024.csv
GEN_ITA_02_01                  Output/GEN_ITA_S02_01_02_1_2024.csv
GEN_ITA_02_02                  Output/GEN_ITA_S02_01_02_2_2024.csv
BMP_VITA_02_01                Output/BMP_VITA_S02_01_02_1_2024.csv
BMP_VITA_02_02                Output/BMP_VITA_S02_01_02_2_2024.csv
HDI_01                              Output/HDI_S02_01_02_1_2024.csv
HDI_02                              Output/HDI_S02_01_02_2_2024.csv
POSTE_VITA_01                Output/POSTE_VITA_S02_01_02_1_2024.csv
POSTE_VITA_02                Output/POSTE_VITA_S02_01_02_2_2024.csv
INTESA_VITA_01              Output/INTESA_VITA_S02_01_02_1_2024.csv
INTESA_VITA_02              Output/INTESA_VITA_S02_01_02_2_2024.csv
CNP_VITA_01                    Output/CNP_VITA_S02_01_02_1_2024.csv
CNP_VITA_02                    Output/CNP_VITA_S02_01_02_2_2024.csv
```

```
ITAS_VITA_01                 Output/ITAS_VITA_S02_01_02_1_2024.csv
ITAS_VITA_02                 Output/ITAS_VITA_S02_01_02_2_2024.csv
HELVETIA_VITA_01          Output/HELVETIA_VITA_S02_01_02_1_2024.csv
HELVETIA_VITA_02          Output/HELVETIA_VITA_S02_01_02_2_2024.csv
VITTORIA_01                   Output/VITTORIA_S02_01_02_1_2024.csv
VITTORIA_02                   Output/VITTORIA_S02_01_02_2_2024.csv
GROUPAMA_01                   Output/GROUPAMA_S02_01_02_1_2024.csv
ALLIANZ_UNICREDIT_01  Output/ALLIANZ_UNICREDIT_S02_01_02_1_2024.csv
ALLIANZ_UNICREDIT_02  Output/ALLIANZ_UNICREDIT_S02_01_02_2_2024.csv
ZURICH_LIFE_01              Output/ZURICH_LIFE_S02_01_02_1_2024.csv
ZURICH_LIFE_02              Output/ZURICH_LIFE_S02_01_02_2_2024.csv

                     codes_path  leto
id
CREDEM_VITA_02_1            NaN  2024
CREDEM_VITA_02_2            NaN  2024
AXA_VITA_02_01              NaN  2024
CREDAG_VITA_02_01           NaN  2024
CREDAG_VITA_02_02           NaN  2024
REALE_02_01                 NaN  2024
REALE_02_02                 NaN  2024
CARDIF_02_01                NaN  2024
MEDIO_02_01                 NaN  2024
MEDIO_02_02                 NaN  2024
GEN_ITA_02_01               NaN  2024
GEN_ITA_02_02               NaN  2024
BMP_VITA_02_01              NaN  2024
BMP_VITA_02_02              NaN  2024
HDI_01                      NaN  2024
HDI_02                      NaN  2024
POSTE_VITA_01               NaN  2024
POSTE_VITA_02               NaN  2024
INTESA_VITA_01              NaN  2024
INTESA_VITA_02              NaN  2024
CNP_VITA_01                 NaN  2024
CNP_VITA_02                 NaN  2024
ITAS_VITA_01                NaN  2024
ITAS_VITA_02                NaN  2024
HELVETIA_VITA_01            NaN  2024
HELVETIA_VITA_02            NaN  2024
VITTORIA_01                 NaN  2024
VITTORIA_02                 NaN  2024
GROUPAMA_01                 NaN  2024
ALLIANZ_UNICREDIT_01        NaN  2024
ALLIANZ_UNICREDIT_02        NaN  2024
ZURICH_LIFE_01              NaN  2024
ZURICH_LIFE_02              NaN  2024
```

2 Code

2.1 Credemvita S.p.A.

S.02.01.02 1

[21]:
```
unique_id = "CREDEM_VITA_02_1"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[22]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[23]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
table.drop(index=[0],inplace=True)
```

[24]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[25]:
```
del table
```

S.02.01.02 2

[26]:
```
unique_id = "CREDEM_VITA_02_2"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[27]:
```
table = extract_tables_from_pdf(pdf_path=output_pdf_path, page_number=1)
```

[28]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
table.drop(index=[39],inplace=True)
```

[29]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[30]:
```
del table
```

2.2 AXA MPS Assicurazioni Vita

S.02.01.02 1

[31]:
```
unique_id = "AXA_VITA_02_01"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[32]:
```
table = extract_tables_from_pdf(pdf_path=output_pdf_path, page_number=1)
```

```
Cannot set gray non-stroke color because /'R768' is an invalid float value
Cannot set gray non-stroke color because /'R770' is an invalid float value
```

[33]:
```
table = table.iloc[:,[2,3]]
table.columns = ["CODE", "C0010"]
table.drop(index=[0, 1, 43, 44, 45, 86],inplace=True)
```

[34]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[35]:
```
del table
```

2.3 CRÈDIT AGRICOLE VITA

S.02.01.02 1

[36]:
```
unique_id = "CREDAG_VITA_02_01"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[37]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[38]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
```

[39]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[40]:
```
del table
```

S.02.01.02 2

[41]:
```
unique_id = "CREDAG_VITA_02_02"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[42]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[43]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
```

[44]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[45]:
```
del table
```

2.4 Società Reale Mutua di Assicurazioni

S.02.01.02 1

[46]:
```
unique_id = "REALE_02_01"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[47]:
```
table = extract_tables_from_pdf(pdf_path=output_pdf_path, page_number=1)
```

[48]:
```
table = convert_to_dataframe(table.columns.values[0])
```

[49]:
```
table = append_zero_if_len4(table,"CODE")
```

[50]:
```
table.columns = ["CODE","DESCRIPTION", "C0010"]
```

[51]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[52]:
```
del table
```

S.02.01.02 2

[53]:
```
unique_id = "REALE_02_02"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[54]:
```
table_index = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[55]:
```
with pdfplumber.open(pdf_path) as pdf:
    page = pdf.pages[page_number - 1]  # Pages are zero-indexed
    tables = page.extract_tables()
    table_numbers = pd.DataFrame(tables[0])
```

[56]:
```
table = pd.concat([table_index, table_numbers], axis=1)
```

[57]:
```
table.iloc[-1,1] = table.iloc[-1,0]
```

[58]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
```

[59]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[60]:
```
del table
del table_index
del table_numbers
```

2.5 Cardif Vita S.p.A.

S.02.01.02

[61]:
```
unique_id = "CARDIF_02_01"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[62]:
```
table = extract_tables_from_pdf(pdf_path=output_pdf_path, page_number=1)
```

[63]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
table.drop(index=[0,1,2],inplace=True)
```

[64]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[65]:
```
del table
```

2.6 MEDIOLANUM VITA S.p.A.

S.02.01.02 1

[66]:
```
unique_id = "MEDIO_02_01"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[67]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[68]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
```

[69]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[70]:
```
del table
```

S.02.01.02 2

[71]:
```
unique_id = "MEDIO_02_02"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[72]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[73]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
```

[74]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[75]:
```
del table
```

2.7 Generali Italia S.p.A.

S.02.01.02 1

[76]:
```
unique_id = "GEN_ITA_02_01"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[77]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[78]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
```

[79]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[80]:
```
del table
```

S.02.01.02 2

[81]:
```
unique_id = "GEN_ITA_02_02"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[82]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[83]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
```

[84]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[85]:
```
del table
```

2.8 Banco BPM Vita S.p.A.

S.02.01.02 1

[86]:
```
unique_id = "BMP_VITA_02_01"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[87]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[88]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
```

[89]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[90]:
```
del table
```

S.02.01.02 2

[91]:
```
unique_id = "BMP_VITA_02_02"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[92]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[93]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
```

[94]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[95]:
```
del table
```

2.9 HDI ASSICURAZIONI S.p.A.

S.02.01.02 1

[96]:
```
unique_id = "HDI_01"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[97]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[98]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
```

[99]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[100]:
```
del table
```

S.02.01.02 2

[101]:
```
unique_id = "HDI_02"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[102]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[103]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
```

[104]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[105]:
```
del table
```

2.10 Gruppo Assicurativo Poste Vita

S.02.01.02 1

[106]:
```
unique_id = "POSTE_VITA_01"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[107]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[108]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
```

[109]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[110]:
```
del table
```

S.02.01.02 2

[111]:
```
unique_id = "POSTE_VITA_02"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[112]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[113]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
```

[114]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[115]:
```
del table
```

2.11 FIDEURAM VITA S.P.A.

S.02.01.02 1

[116]:
```
unique_id = "INTESA_VITA_01"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[117]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[118]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
```

[119]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[120]:
```
del table
```

S.02.01.02 2

[121]:
```
unique_id = "INTESA_VITA_02"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[122]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[123]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
```

[124]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[125]:
```
del table
```

2.12 CNP Vita Assicura S.p.A.

S.02.01.02 1

[126]:
```
unique_id = "CNP_VITA_01"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[127]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[128]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
```

[129]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[130]:
```
del table
```

S.02.01.02 2

[131]:
```
unique_id = "CNP_VITA_02"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[132]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[133]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
```

[134]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[135]:
```
del table
```

2.13 ITAS VITA

S.02.01.02 1

[136]:
```
unique_id = "ITAS_VITA_01"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[137]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[138]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
```

[139]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[140]:
```
del table
```

S.02.01.02 2

[141]:
```
unique_id = "ITAS_VITA_02"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[142]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[143]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
```

[144]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[145]:
```
del table
```

2.14 Helvetia Vita S.p.A.

S.02.01.02 1

[146]:
```
unique_id = "HELVETIA_VITA_01"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[147]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[148]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
```

[149]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[150]:
```
del table
```

S.02.01.02 2

[151]:
```
unique_id = "HELVETIA_VITA_02"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[152]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[153]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
```

[154]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[155]:
```
del table
```

2.15 Vittoria Assicurazioni S.p.A.

S.02.01.02 1

[156]:
```
unique_id = "VITTORIA_01"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[157]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[158]:
```
table.columns = ["CODE", "C0010"] # ONE COLUMN LESS
```

[159]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[160]:
```
del table
```

S.02.01.02 2

[161]:
```
unique_id = "VITTORIA_02"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[162]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[163]:
```
table.columns = ["CODE", "C0010"] # ONE COLUMN LESS
```

[164]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[165]:
```
del table
```

2.16 GROUPAMA ASSICURAZIONI S.P.A.

S.02.01.02

[166]:
```
unique_id = "GROUPAMA_01"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[167]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[168]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
```

[169]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[170]:
```
del table
```

2.17 UniCredit Allianz Vita S.p.A.

S.02.01.02 1

[171]:
```
unique_id = "ALLIANZ_UNICREDIT_01"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[172]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[173]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
```

[174]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[175]:
```
del table
```

S.02.01.02 2

[176]:
```
unique_id = "ALLIANZ_UNICREDIT_02"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[177]:
```
table = run_ocr_and_convert_to_df(path=output_pdf_path, api_key=api_key)
```

[178]:
```
table.columns = ["DESCRIPTION","CODE", "C0010"]
```

[179]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[180]:
```
del table
```

2.18 Zurich Investments Life S.p.A.

S.02.01.02 1

[181]:
```
unique_id = "ZURICH_LIFE_01"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[182]:
```
markdown_text = run_mistral_ocr(output_pdf_path, api_key)
table = ocr_to_dataframe(markdown_text)
```

[183]:
```
table.columns = ["CODE", "DESCRIPTION", "C0010"]
```

[184]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[185]:
```
del table
```

S.02.01.02 2

[186]:
```
unique_id = "ZURICH_LIFE_02"
pdf_path, page_number, output_pdf_path, output_final_path, codes_path = ␣
 ↪extract_paths(master_list=master_list, unique_id=unique_id)
extract_page(input_pdf_path=pdf_path, output_pdf_path=output_pdf_path,␣
 ↪page_number=page_number, password = "")
```

[187]:
```
markdown_text = run_mistral_ocr(output_pdf_path, api_key)
table = ocr_to_dataframe(markdown_text)
```

[188]:
```
table.columns = ["CODE", "DESCRIPTION", "C0010"]
```

[189]:
```
set_code_index_and_save(table=table, path=output_final_path)
```

[190]:
```
del table
```