**7/02/25**

**Task**

**Research on BFSI OCR and find the challenges and solutions**

**1.Introduction to BFSI OCR**

OCR (Optical Character Recognition) in BFSI (Banking, Financial Services, and Insurance) is used to extract, digitize, and process text from financial documents, reducing manual effort and improving accuracy.

🔹 Purpose: Automate document processing for banking, insurance, and financial workflows.  
🔹 Key Benefits: Speed, accuracy, cost reduction, fraud detection, and enhanced customer experience.

**2. BFSI OCR Use Cases**

**A.Banking**

* KYC (Know Your Customer) Automation – Extracts data from identity proofs (Aadhaar, Passport, PAN, etc.).
* Cheque Processing – Reads handwritten and printed cheques to verify transactions.
* Loan Document Processing – Digitizes application forms, agreements, and financial statements.
* Invoice & Receipt Processing – Automates bookkeeping and tax filings.
* Bank Statement Analysis – Extracts and categorizes transactions from bank PDFs**.**

**B. Financial Services**

* Automated Data Entry – Reads and structures financial statements.
* Credit Risk Assessment – Extracts financial metrics from balance sheets and tax returns.
* Stock Market & Trading – Reads printed reports, contracts, and invoices.

**C. Insurance**

* Claim Processing – Automates claim form verification.
* Policy Document Extraction – Reads and structures insurance policy details.
* Fraud Detection – Identifies discrepancies in scanned claims and documents.

**BFSI OCR Workflow**

**Step 1: Document Collection**

* Physical or digital documents (PDFs, images, scans).

**Step 2: Preprocessing (Image Enhancement)**

* **Grayscale conversion** – Improves clarity.
* **Binarization (Thresholding)** – Converts to black & white for better OCR accuracy.
* **Noise removal** – Removes distortions like dust, smudges.
* **Deskewing** – Corrects tilted images.

**Step 3: OCR Processing**

* **Text Extraction** – Reads printed and handwritten text.
* **Table & Layout Detection** – Identifies financial tables and forms.
* **Named Entity Recognition (NER)** – Identifies important financial terms like "Account Number," "Loan Amount," etc.

**Step 4: Data Validation & Extraction**

* Matches extracted data with financial databases.
* Validates against known formats (e.g., IBAN for bank accounts).

**Step 5: Integration & Automation**

* Connects with **banking systems, ERP, CRM, and RPA bots** for real-time decision-making.

**Technologies Used in BFSI OCR**

1. **OCR Engines**

* **Tesseract OCR** – Open-source, supports multiple languages.
* **Google Vision OCR** – Cloud-based, AI-powered.
* **Amazon Textract** – Best for financial document parsing.
* **Azure Form Recognizer** – Specialized in invoice, receipt, and form extraction.

**B. AI/ML Enhancements**

* **Natural Language Processing (NLP)** – Extracts context from financial text.
* **Deep Learning Models (LSTMs, CNNs)** – Improves handwriting recognition.
* **Computer Vision (CV)** – Enhances image and text detection.

**C. RPA (Robotic Process Automation) Integration**

* Automates **data entry into banking & insurance systems** using OCR results.

**5. Challenges & Limitations**

* **Handwritten Text Complexity** – Poor handwriting reduces accuracy.
* **Document Variability** – Different banks use different formats, making standardization hard.
* **Security & Compliance Risks** – BFSI documents contain sensitive financial data.
* **Language & Font Variability** – Local languages and non-standard fonts can impact accuracy.
* **Low-Quality Scans** – Blurry or damaged documents can cause misreading.
* **Solution:** Advanced OCR models + AI-based post-processing.

**6. BFSI OCR Compliance & Security**

**Data Protection Laws:**

* **GDPR (Europe)** – Protects financial data privacy.
* **CCPA (California)** – Regulates personal data extraction.
* **RBI & SEBI Compliance (India)** – Ensures secure banking data handling.

**Security Measures:**

* **End-to-End Encryption** – Protects document transfers.
* **Access Control** – Restricts sensitive data access.
* **Audit Trails & Logging** – Tracks document processing history.

**Solutions for BFSI OCR Challenges**

**Challenge: Handwritten Text Complexity**

Problem: Poor handwriting affects OCR accuracy.

Solution:

* Use AI-based handwriting recognition (HWR) models like Google Vision AI, AWS Textract, and Microsoft Azure OCR.
* Implement Recurrent Neural Networks (RNNs) & Long Short-Term Memory (LSTM) for better handwritten text recognition.
* Train custom OCR models on BFSI-specific handwritten data.

**Solutions for BFSI OCR Challenges 🚀**

**1Challenge: Handwritten Text Complexity**

* **Problem:** Poor handwriting affects OCR accuracy.  
  **Solution:**
* Use AI-based handwriting recognition (HWR) models like Google Vision AI, AWS Textract, and Microsoft Azure OCR.
* Implement Recurrent Neural Networks (RNNs) & Long Short-Term Memory (LSTM) for better handwritten text recognition.
* Train custom OCR models on BFSI-specific handwritten data.

**2️Challenge: Document Variability**

* **Problem:** Different banks and financial institutions use different formats, making extraction difficult.

**Solution:**

* Use **Template-Free OCR** like **Amazon Textract & Google Document AI**, which detects structure dynamically.
* Train **machine learning models** to recognize key-value pairs in documents.
* Use **Natural Language Processing (NLP)** to extract context from unstructured data.

**6️Challenge: Real-Time Processing Speed ⏳**

* **Problem:** BFSI institutions require **real-time document processing** for quick decision-making.  
  **Solution:**
* Use **cloud-based OCR APIs** (Google Vision, AWS Textract, Azure OCR) for faster results.
* Implement **edge computing & RPA (Robotic Process Automation)** for instant document validation.
* Use **GPU acceleration** for large-scale OCR processing.

**10/02/25**

**COMPLETE WORKFLOW OF OCR**

OCR is a multi-step process that converts scanned images, PDFs, or handwritten documents into machine-readable text. The workflow includes image acquisition, preprocessing, text detection, recognition, post-processing, and integration.

**Step 1: Image Acquisition**

The first step is collecting the document/image for OCR processing.

✅ **Sources of Input:**

* Scanned documents (bank statements, invoices, forms, etc.)
* Digital images (JPEG, PNG, TIFF)
* PDFs (searchable & non-searchable)
* Handwritten notes

✅ **Challenges:**

* Poor image quality
* Low resolution
* Blurred or skewed images

**Step 2: Preprocessing (Image Enhancement)**

* OCR accuracy depends on **clean and clear input images**. Preprocessing improves image quality by removing noise and distortions.
* ✅ **Key Preprocessing Techniques:**  
  ✔ **Grayscale Conversion** – Converts colored images to grayscale for better contrast.  
  ✔ **Binarization (Thresholding)** – Converts grayscale images to black & white.  
  ✔ **Noise Removal (Denoising)** – Removes unwanted marks, dust, and artifacts.  
  ✔ **Deskewing & Rotation Correction** – Aligns tilted documents for better text extraction.  
  ✔ **Contrast & Brightness Adjustment** – Enhances text visibility.  
  ✔ **Edge Detection & Cropping** – Identifies and removes unnecessary borders.

**Step 3: Text Detection**

* At this stage, the OCR engine identifies **text regions** in the image.
* ✅ **Techniques Used:**  
  ✔ **Connected Component Analysis (CCA)** – Detects separate letters and words.  
  ✔ **Edge Detection (Canny, Sobel Filters)** – Finds text boundaries.  
  ✔ **Deep Learning Models (YOLO, EAST, CRAFT)** – Detects text in complex layouts.  
  ✔ **Layout Analysis (PSM – Page Segmentation Mode)** – Divides text into blocks, lines, and words.

**Step 4: Text Recognition (OCR Engine)**

* Once the text areas are detected, the OCR engine **recognizes and converts** characters into digital text.
* ✅ **Types of OCR Engines:**  
  ✔ **Tesseract OCR** – Open-source, supports multiple languages.  
  ✔ **Google Vision API** – Cloud-based, AI-powered recognition.  
  ✔ **AWS Textract** – Best for financial and legal documents.  
  ✔ **Microsoft Azure OCR** – Recognizes structured text, tables, and forms.  
  ✔ **ABBYY FineReader** – High accuracy for printed & handwritten text.
* ✅ **Two Recognition Methods:**  
  1️**Traditional OCR (Pattern Matching & Feature Extraction)** – Uses templates to recognize characters.  
  2️**AI-Based OCR (Deep Learning – LSTMs, CNNs)** – Learns from large datasets to improve accuracy.

**Step 5: Post-Processing & Text Validation**

* After extracting text, post-processing improves accuracy by **correcting errors and formatting text**.
* **Techniques Used:**  
  ✔ **Spell Checking & Correction (Levenshtein Distance, Hunspell, Fuzzy Matching)**  
  ✔ **Named Entity Recognition (NER)** – Identifies names, dates, amounts, etc.  
  ✔ **Text Structuring (Regex, NLP)** – Formats extracted text into structured data.  
  ✔ **Table & Layout Reconstruction** – Detects tabular data and organizes it properly.

**Step 6: Data Extraction & Integration 🔗**

After post-processing, the structured text is integrated into a **database, application, or system** for further use.

✅ **Where the Extracted Data Goes?**  
✔ **Databases (MySQL, MongoDB, Firebase)** – Stores structured text.  
✔ **APIs & RPA Systems** – Automates document processing.  
✔ **Search Engines** – Enables text search in scanned documents.  
✔ **ERP & CRM Systems** – Uses extracted data in business applications.

**Step 7: Continuous Improvement (AI & Feedback Loop)**

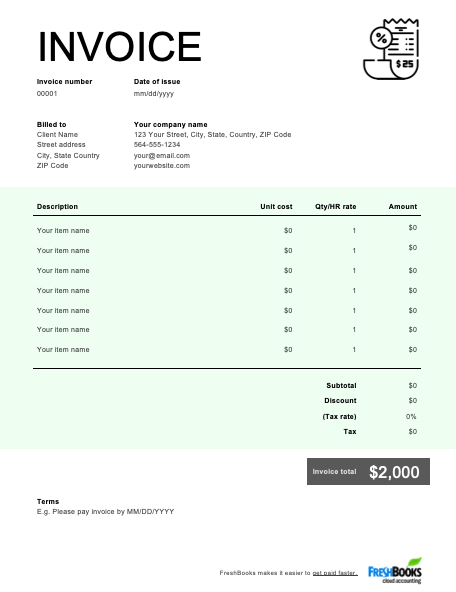
OCR systems improve over time using **AI & feedback mechanisms**.

✅ **Methods to Improve Accuracy:**  
✔ **Machine Learning Training** – Fine-tunes models for specific fonts & handwriting.  
✔ **Active Learning** – Users correct OCR errors to retrain the system.  
✔ **Image Quality Enhancements** – AI auto-adjusts image contrast & sharpness.  
✔ **Custom OCR Models** – BFSI, Legal, Medical sectors train OCR for domain-specific text.

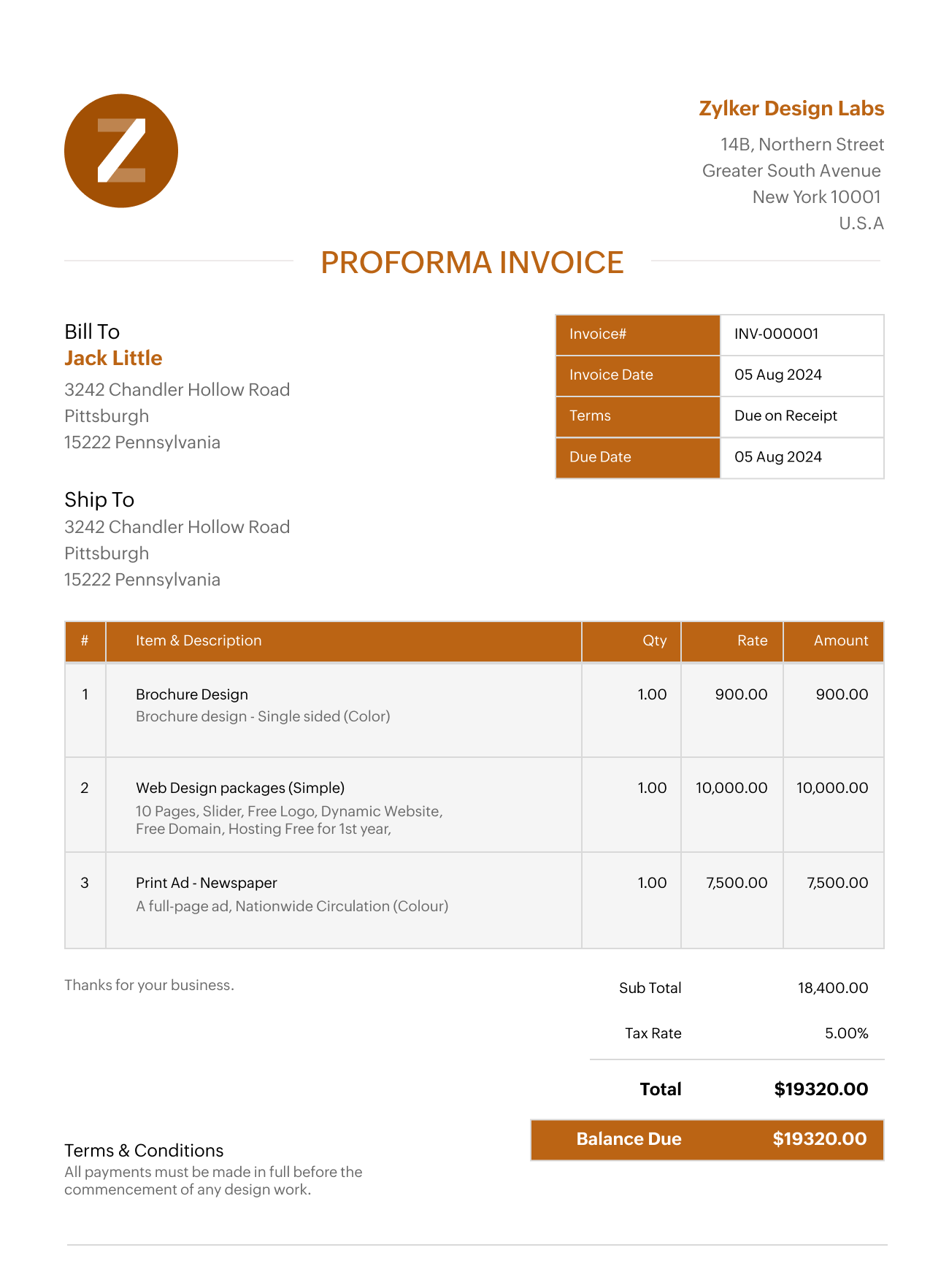
**11/02/2025 TASK**

**INVOICE**

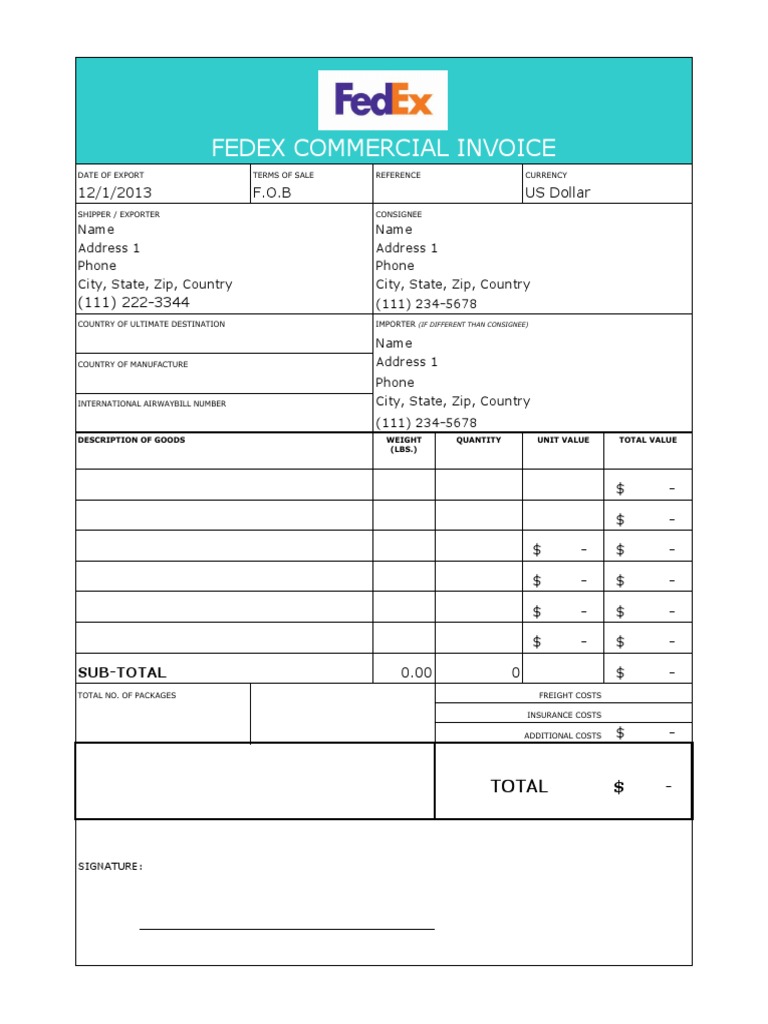
1 STANDARD INVOICE



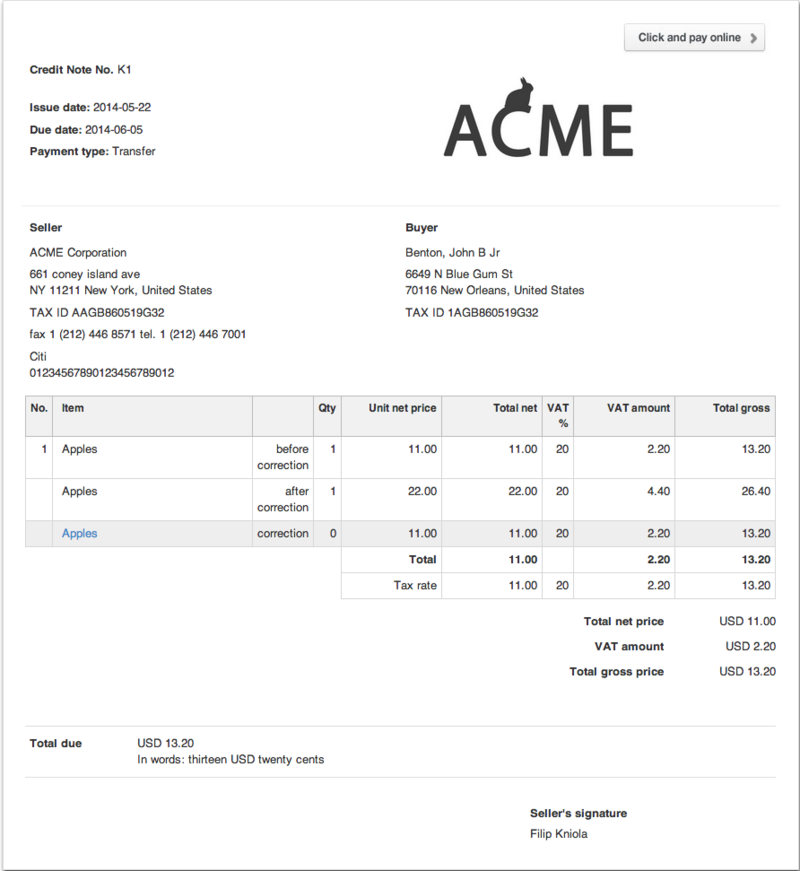
2 Proforma Invoice



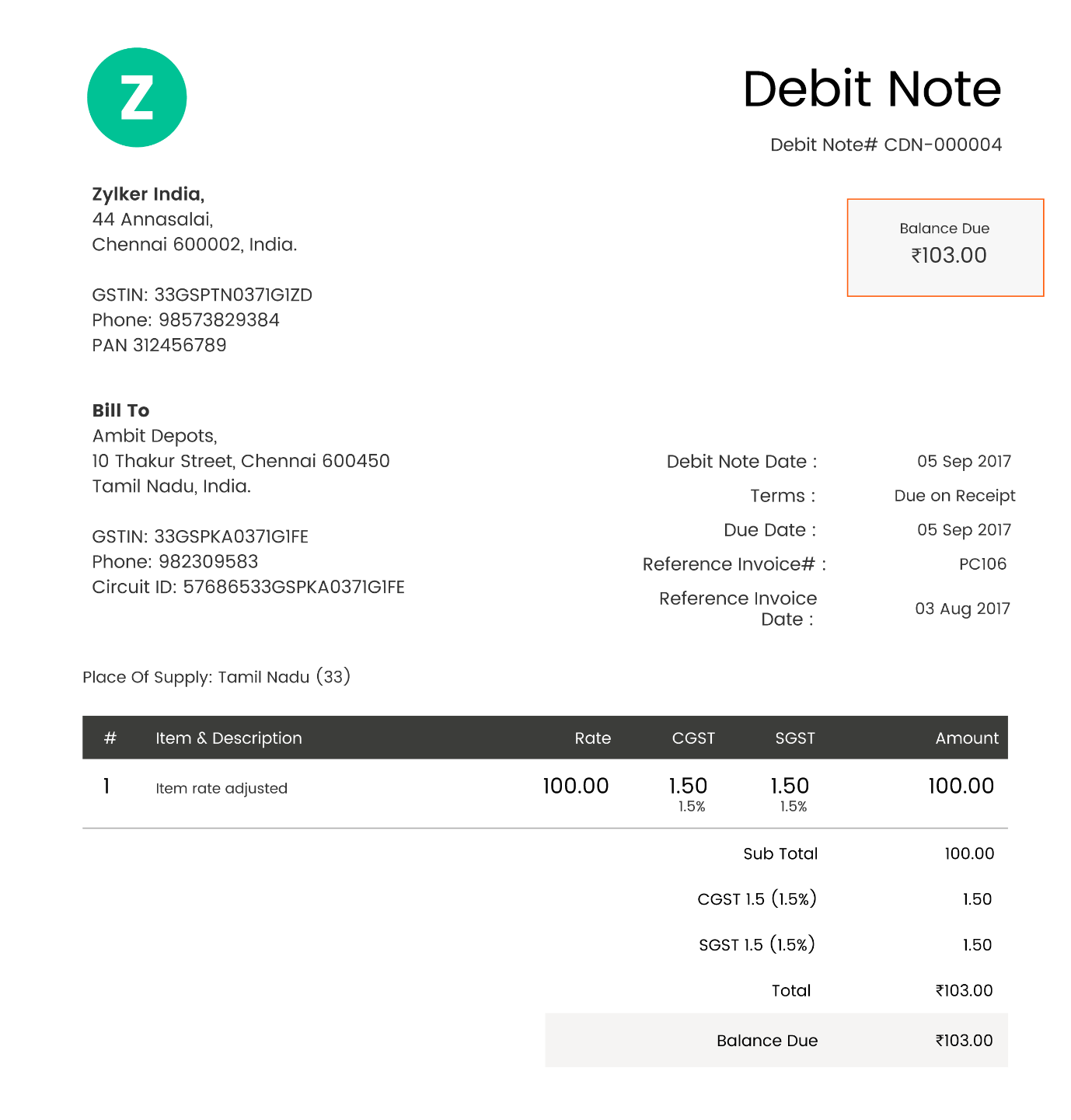
3 COMMERCIAL INVOICE



4 Credit Invoice

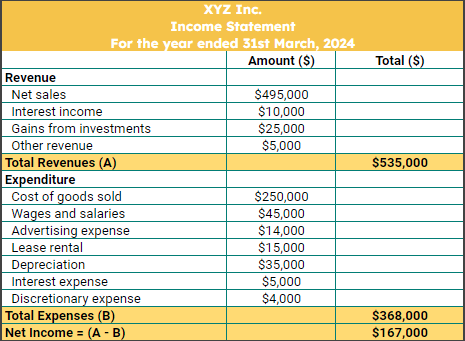


5 Debit Invoice

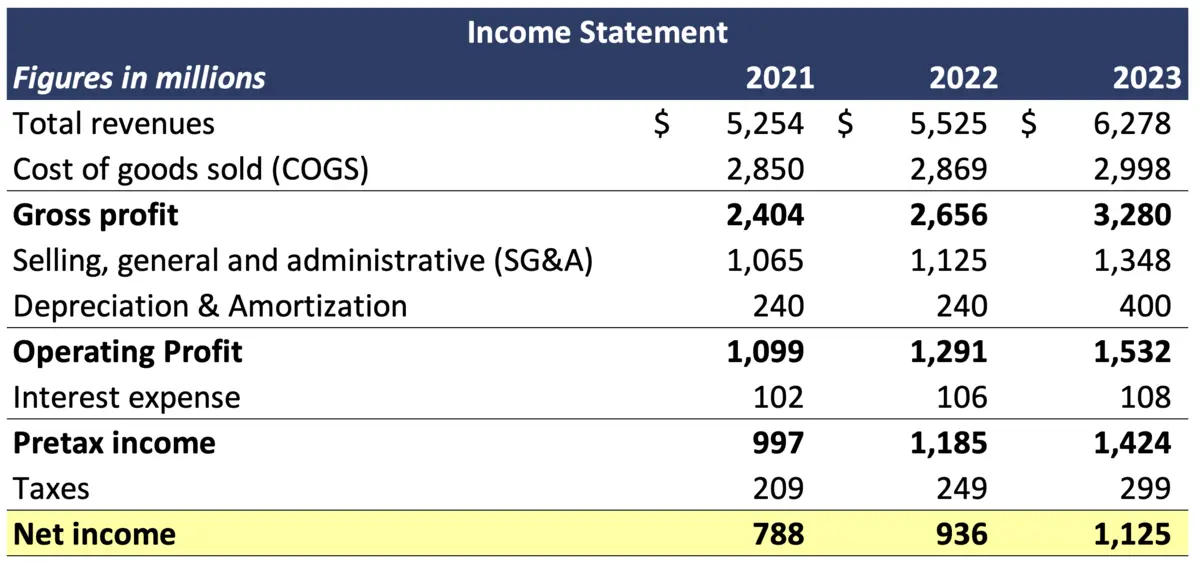


**PROFIT LOSS STATEMENT**

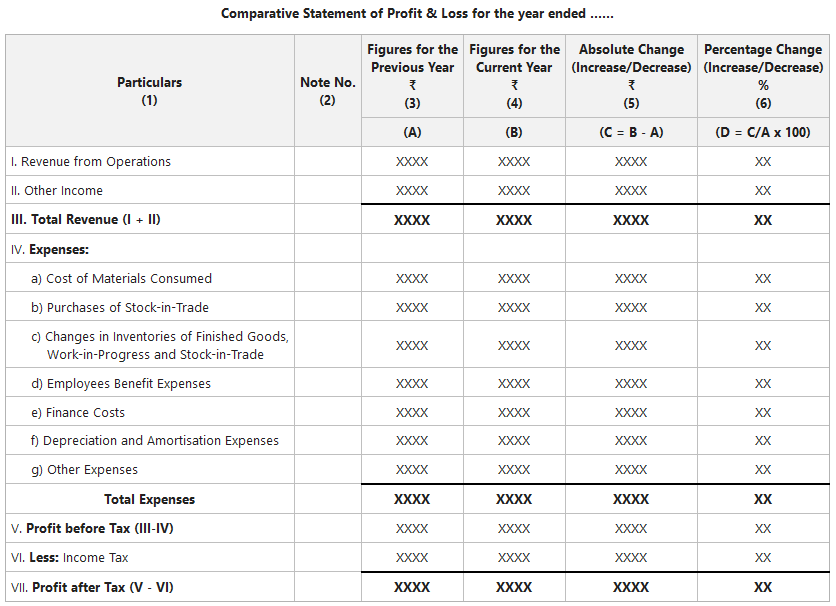
1 Single-Step Profit and Loss Statement



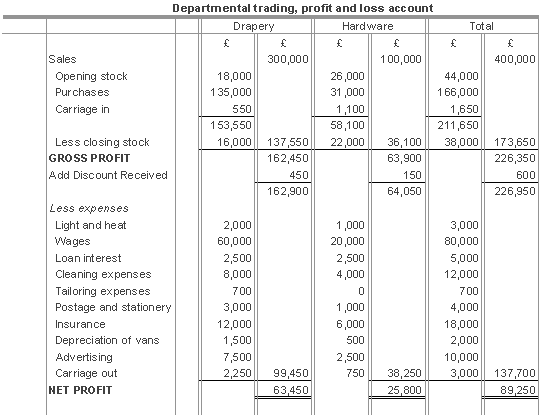
2 Multi Step Profit and Loss Statement



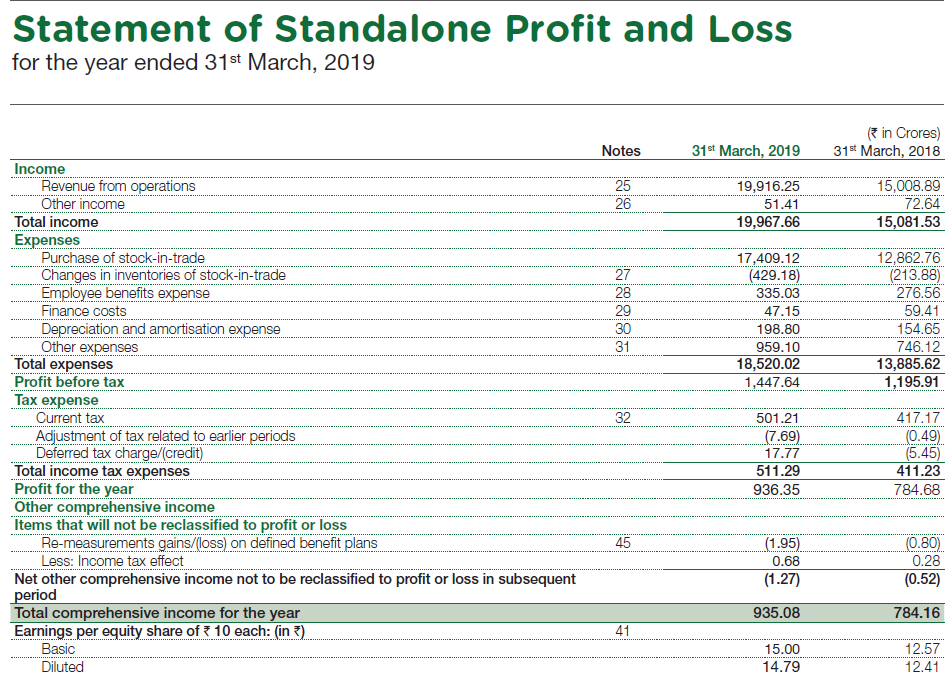
3 Comparative Profit and Loss Statement



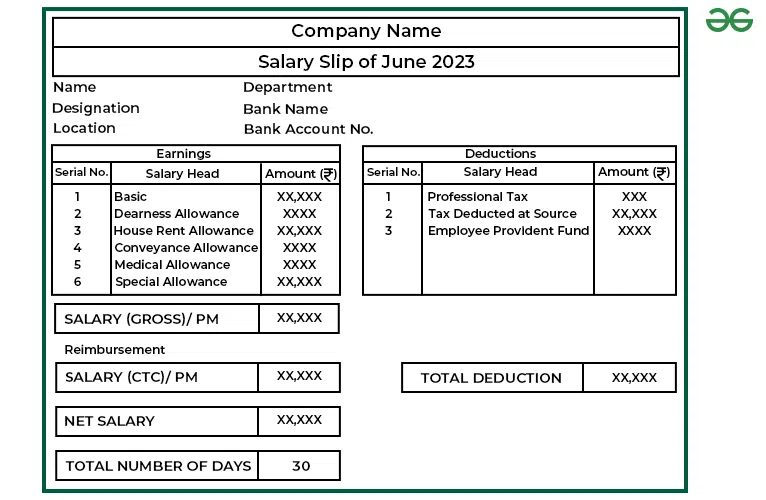
4 Departmental Profit and Loss Statement

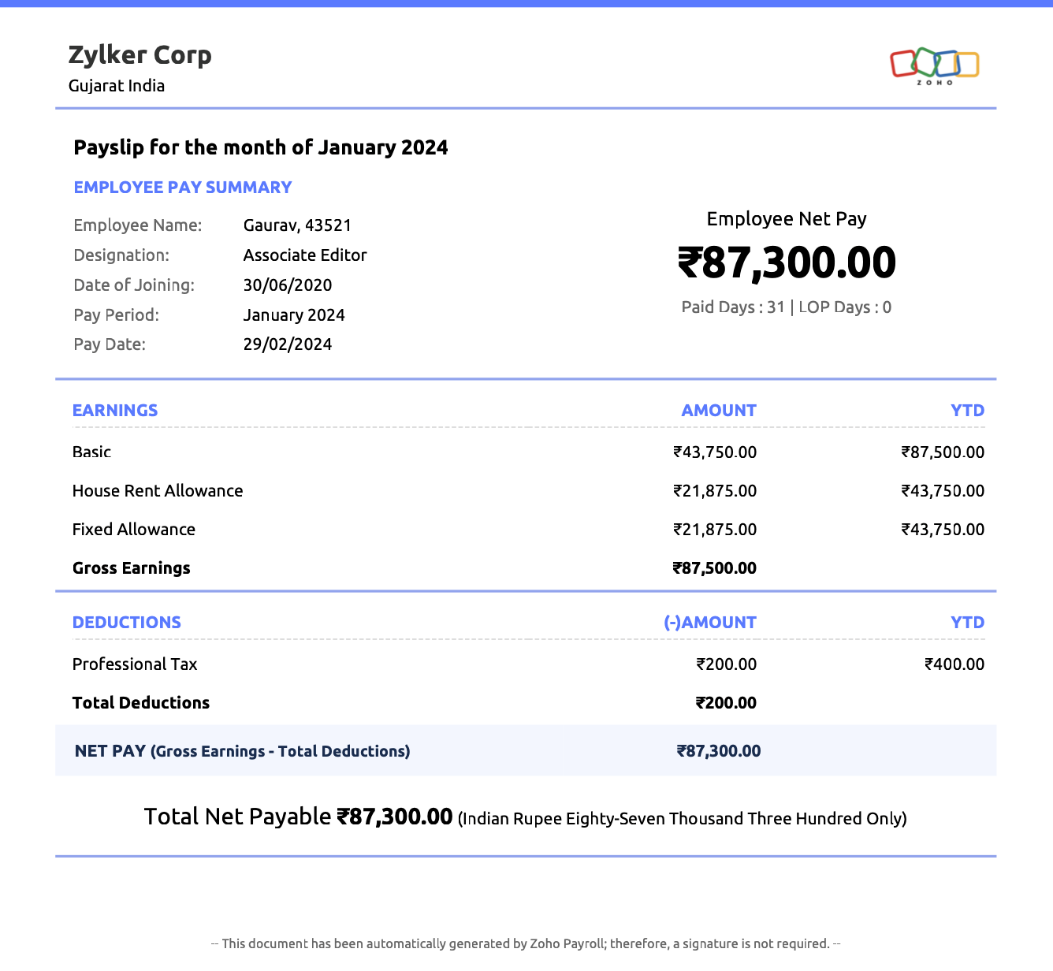


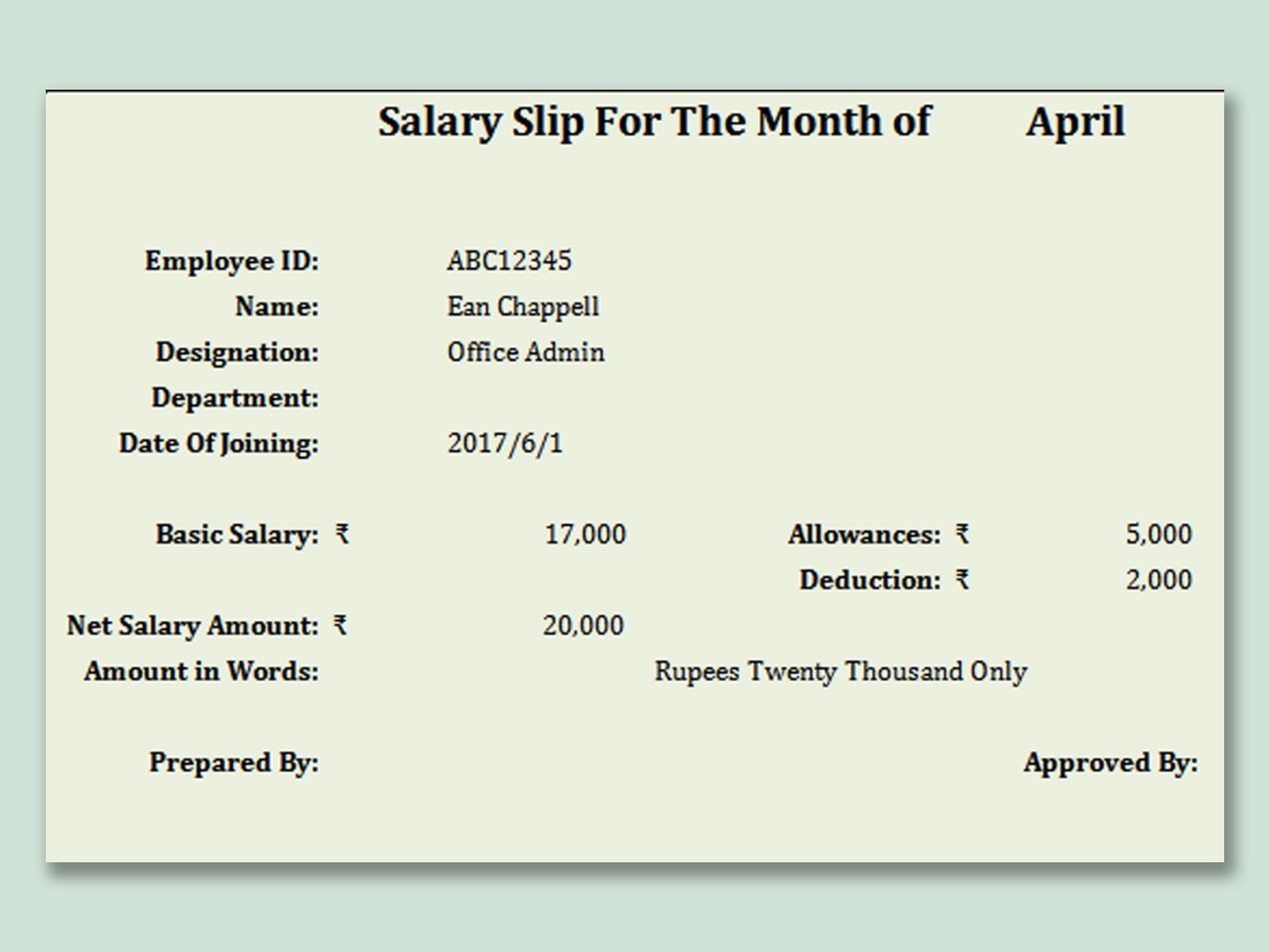
5 Consolidated Profit and Loss Statement

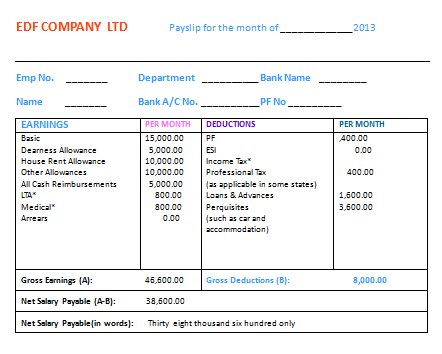


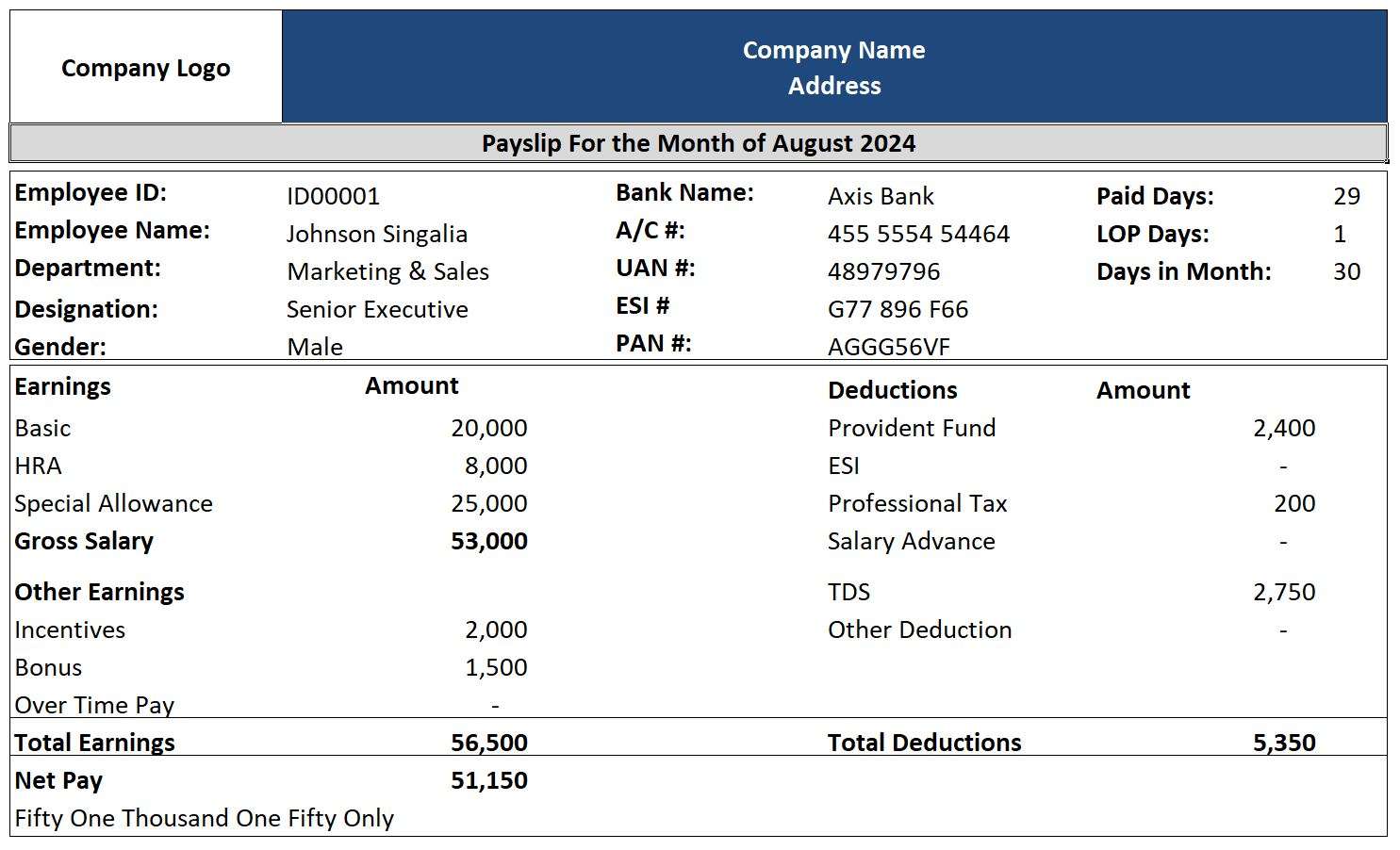
**INCOME**











**12/02/25**

**OTHER OCR ENGINES & ADVANTAGES, DISADVANTAGES**

OCR engines vary in accuracy, speed, and features. Here’s a breakdown of the most popular OCR engines, their **advantages**, and **disadvantages**.

**Tesseract OCR (Open-Source, Google)**

**Overview:**

* Free & open-source OCR developed by Google.
* Supports over 100 languages.
* Works well with printed & handwritten text.

**Advantages:**

* Free & open-source (cost-effective).
* Supports multiple languages & scripts.
* Can be trained for custom datasets.
* Works offline without cloud dependencies.

**Disadvantages:**

* Struggles with poor-quality images & complex layouts.
* Slower compared to cloud-based OCRs.
* Requires pre-processing for better accuracy.
* Lacks built-in handwriting recognition.

**Google Cloud Vision OCR (Cloud-Based AI OCR)**

**Overview:**

* AI-powered OCR by Google Cloud.
* Recognizes text, handwriting, and structured data (tables, forms).
* Supports multi-language & handwriting recognition.

**Advantages:**

* High accuracy for printed & handwritten text.
* Supports structured data extraction (tables, invoices, receipts).
* Cloud-based, scalable for large datasets.
* Integrated with AI & NLP for better text processing.

**Disadvantages:**

* Paid service (costs can be high for large-scale use).
* Requires internet connection (no offline support).
* API-dependent; customization is limited.

**AWS Textract (Amazon OCR for Documents & Forms)**

**Overview:**

* OCR engine by Amazon Web Services (AWS).
* Extracts text from scanned documents, PDFs, and images.
* Recognizes structured data like forms & tables.

**Advantages:**

* High accuracy for financial & legal documents.
* Supports form & table extraction.
* Scalable for enterprise applications.
* Integrates with AWS ecosystem (AI, ML, cloud storage).

**Disadvantages:**

* Expensive for high-volume processing.
* Cloud-based (requires internet).
* Limited language support compared to Tesseract & Google Vision.

**Microsoft Azure OCR (AI-Powered OCR by Microsoft)**

**Overview:**

* AI-driven OCR service by Microsoft Azure.
* Recognizes printed text, handwriting, and structured data.
* Works well for business automation.

**Advantages:**

* Supports handwritten & printed text.
* Extracts key-value pairs from invoices, IDs, and receipts.
* High-speed processing with cloud scalability.
* Easy integration with Microsoft 365, Dynamics, and Power BI.

**Disadvantages:**

* Paid service (cost varies based on usage).
* Cloud-based (requires internet).
* Some accuracy issues with handwritten text.

**ABBYY FineReader (High-Accuracy Enterprise OCR)**

**Overview:**

* Premium OCR software for business & enterprise applications.
* Best for scanned documents, PDFs, and books.
* Supports multi-language recognition & structured data extraction.

**Advantages:**

* High OCR accuracy (better than Tesseract & Azure).
* Supports complex document layouts (tables, columns).
* Works well for legal & financial documents.
* Can process bulk documents efficiently.

**Disadvantages:**

* Paid software (expensive for small businesses).
* Requires installation (not cloud-based).
* Not ideal for real-time processing.

**Adobe Acrobat OCR (PDF-Based OCR)**

**Overview:**

* OCR engine built into Adobe Acrobat for PDF text extraction.
* Converts scanned PDFs into editable & searchable documents.

**Advantages:**

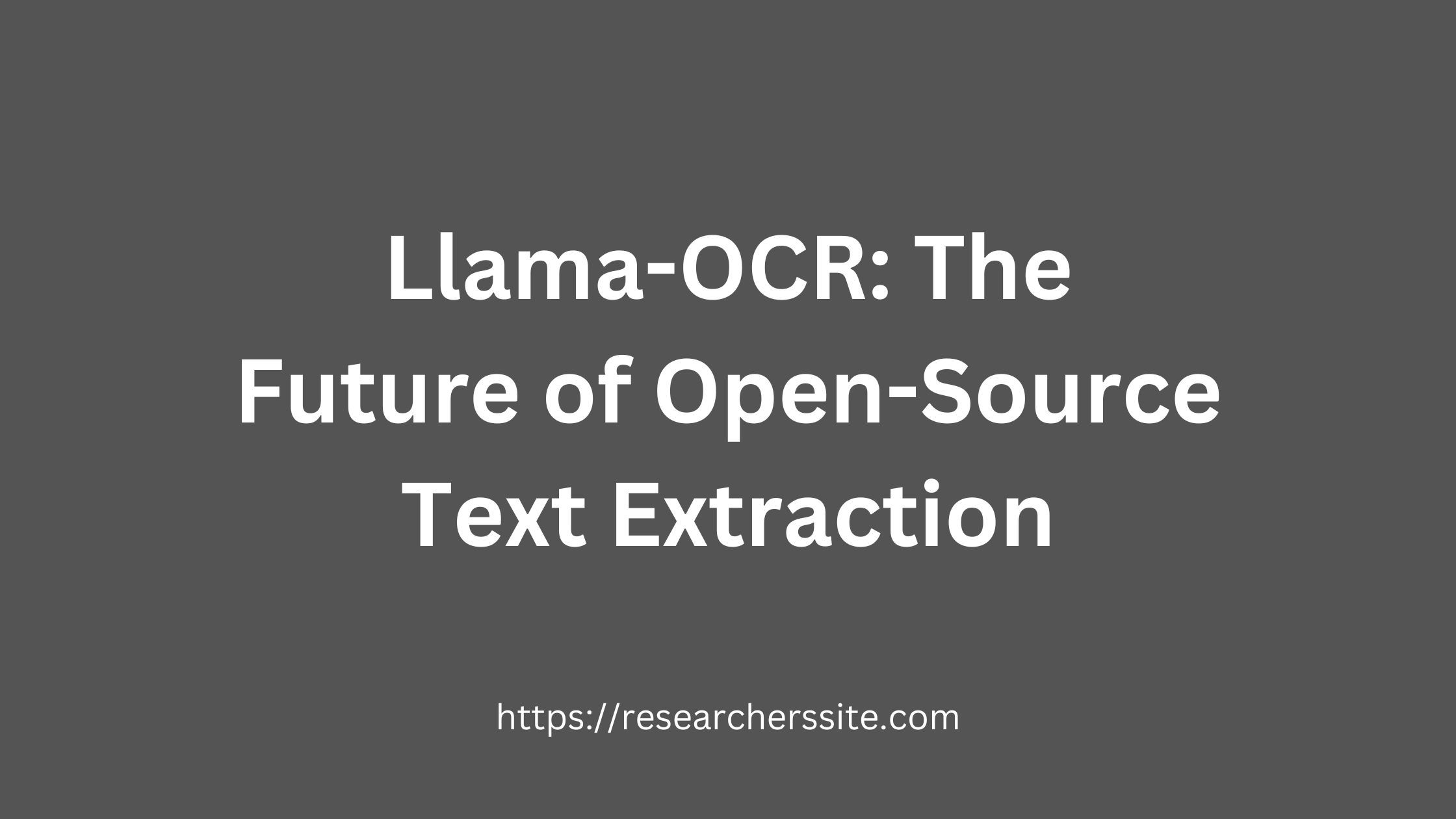
* Easy to use for PDF documents.
* Good accuracy for printed text.
* Maintains document formatting.
* Works offline.

**Disadvantages:**

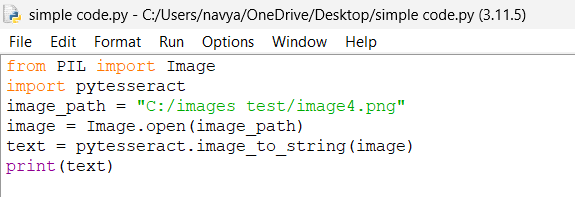
* Not ideal for handwriting recognition.
* Limited support for complex layouts.
* Requires Adobe Acrobat Pro (paid software).

**PYTHON PROGRAM TO EXTRACT TEXT IN THE IMAGES**

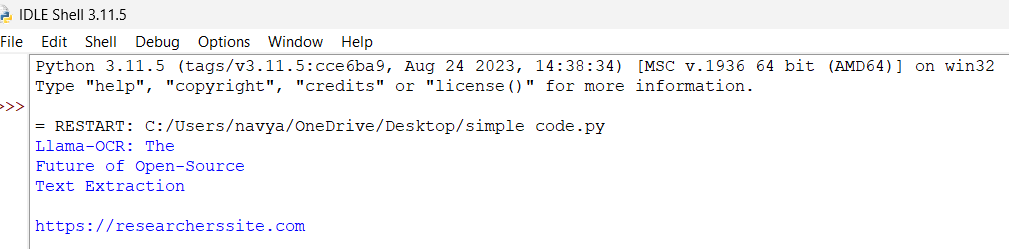
**Input**

****

**Code :**

****

**OUT PUT :**

****

**14/02/25**

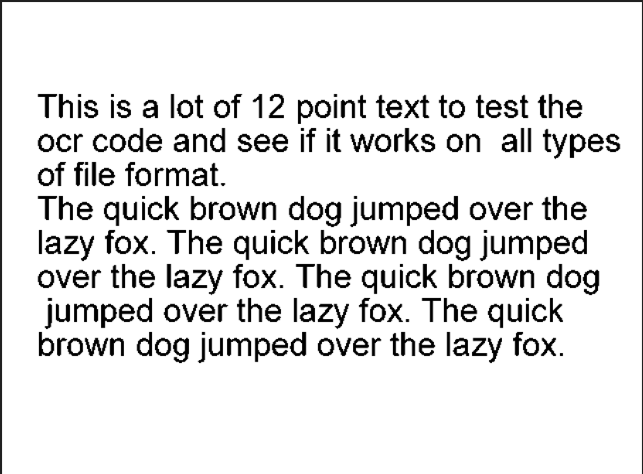
**PREPROCESSING AN IMAGE**

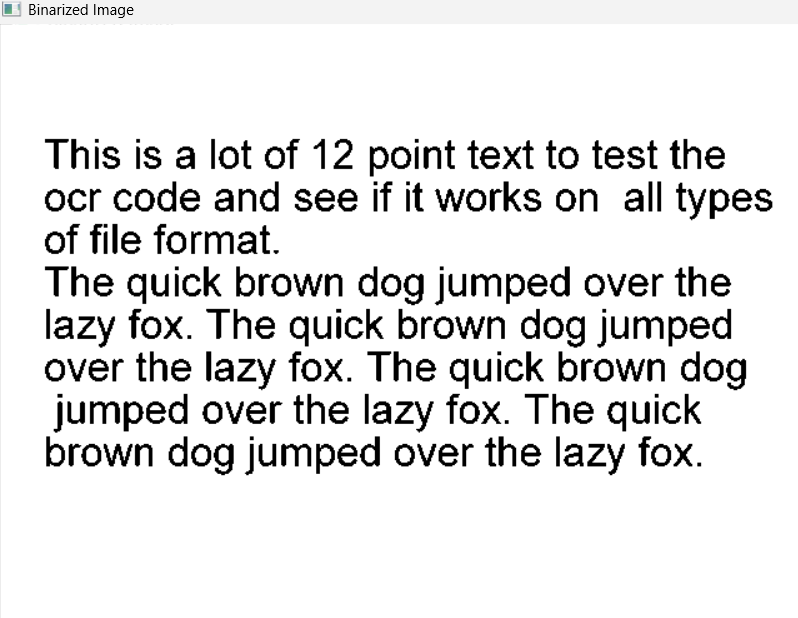
**GRAYSCALING:**

****

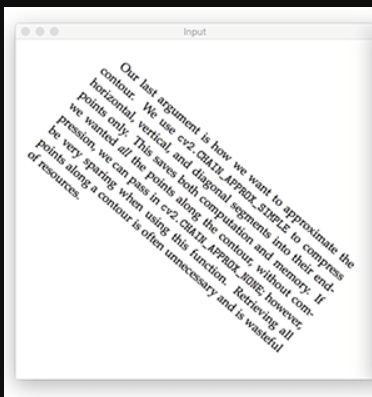


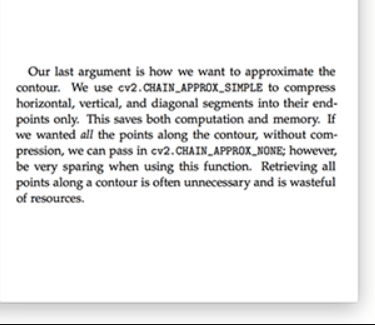
**BINARIZATION:**

****

****

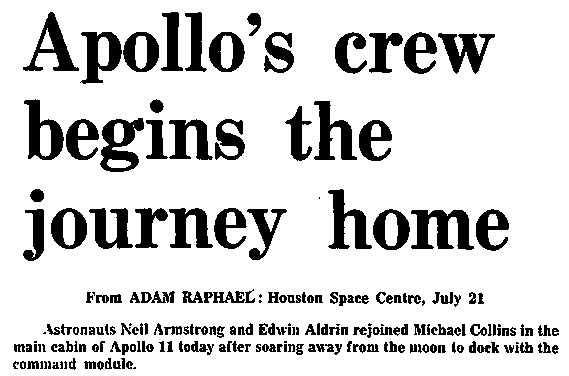
**DESKEWING:**

****

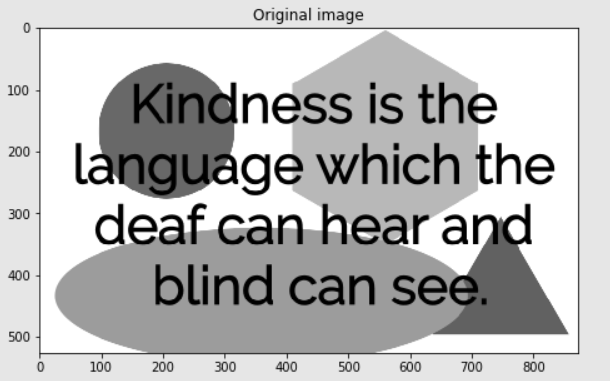
****

**NOISE REMOVAL:**





THRESHOLDING:





**18/02/25**

**TOOLS AND LIBRARIES USED TO EXTRACT STRUCTURED DOC**

**1 Preprocessing (Enhancing Image/Text for Extraction)**

* **OpenCV** → Used for binarization, noise removal, deskewing, and image enhancement.
* **Pillow (PIL)** → Handles image processing (resizing, format conversion).

2️ **Natural Language Processing (NLP) for Text Analysis**

* **SpaCy** → Named Entity Recognition (NER) to extract dates, names, invoice numbers, etc.
* **NLTK** → Text tokenization, stemming, and sentence segmentation.
* **Hugging Face Transformers (BERT, LayoutLM)** → Advanced AI models for document understanding.

3️ **Regular Expressions (Regex) for Pattern Matching**

* Extracts **dates, email IDs, invoice numbers, amounts, etc.**
* Example: re.findall(r"\d{4}-\d{2}-\d{2}", text) for date extraction.

4️ **Key-Value Pair & Form Extraction**

* **AWS Textract** → Extracts structured key-value pairs from documents.
* **Google Document AI** → Extracts key fields from structured/unstructured documents.

5️ **Table Extraction from PDFs**

* **Camelot** → Extracts tables from PDFs with proper structure.
* **pdfplumber** → Extracts tabular data from PDFs more flexibly.
* **Pandas** → Converts extracted tables into structured data (CSV, Excel, JSON).

6️ **Document Parsing & Understanding**

* **LayoutLM (by Hugging Face)** → AI model that understands document layouts (invoices, receipts).
* **Grobid** → Extracts structured information from research papers and official documents.

7️ **Metadata Extraction**

* **PyPDF2** → Extracts metadata (author, title, creation date) from PDFs.
* **pdfminer.six** → Extracts text and layout structure from PDFs.

8️ **Semantic Search & Information Retrieval**

* **FAISS (Facebook AI Search)** → Searches structured data in large documents.
* **Elasticsearch** → Efficient search for structured/unstructured document data.

**Better accuracy:**

**1** Google Document AI / AWS Textract → Best accuracy for structured data extraction.  
2️ LayoutLM (Hugging Face) → Best for AI-based document parsing.  
3️pdfplumber / Camelot → Best for extracting tables from PDFs.

19/02/25

**TASK**

**How OCR Works on Different Font Styles: A Comprehensive Research**

**1. Introduction to OCR and Font Styles**

Optical Character Recognition (OCR) is a technology that converts images of text into machine-readable text. The performance of OCR largely depends on the font style, size, spacing, and distortions present in the scanned text. Some fonts are easier to recognize, while others pose challenges due to their complexity or resemblance to handwritten text.

**2. OCR Processing Pipeline**

Regardless of the font, OCR generally follows these steps:

1. **Preprocessing** (grayscale conversion, binarization, noise removal, deskewing)
2. **Character Segmentation** (detecting and isolating characters)
3. **Feature Extraction** (analyzing font shape, stroke width, etc.)
4. **Recognition Model** (using machine learning algorithms or deep learning)
5. **Post-processing** (spell-checking, contextual correction)

**3. Impact of Font Styles on OCR Accuracy**

**3.1 Standard Fonts (Serif vs. Sans-serif)**

* **Serif fonts** (Times New Roman, Georgia): OCR performs well as these fonts have a consistent structure.
* **Sans-serif fonts** (Arial, Helvetica): These fonts are even easier for OCR because they lack decorative elements.

**3.2 Monospaced vs. Proportional Fonts**

* **Monospaced fonts** (Courier New, Consolas): OCR performs well as each character has equal spacing.
* **Proportional fonts** (Calibri, Garamond): Slightly harder to process due to varying character spacing.

**3.3 Decorative and Stylized Fonts**

* **Cursive/Script fonts** (Lucida Handwriting, Brush Script): OCR struggles due to letter connectivity.
* **Gothic/Blackletter fonts** (Old English, Fraktur): Recognition is difficult due to complex letterforms.

**3.4 Handwritten vs. Printed Fonts**

* **Handwritten text:** Traditional OCR struggles; modern deep learning-based OCR (like Tesseract with LSTM or Google Cloud Vision) is required.
* **Printed text:** Standard OCR models work effectively.

**3.5 Distorted or Decorative Fonts**

* OCR accuracy drops significantly when fonts have excessive curls, embellishments, or uneven spacing.

**4. OCR Models and Font Adaptability**

**4.1 Traditional OCR Engines (Rule-Based)**

* Tesseract OCR (with PSM and OEM settings) works well with standard printed fonts but struggles with cursive or decorative fonts.

**4.2 Machine Learning-Based OCR**

* Google Vision API, Microsoft Azure OCR, and Adobe OCR use AI to recognize a broader range of fonts with improved accuracy.

**4.3 Deep Learning-Based OCR**

* Models like CRNN (Convolutional Recurrent Neural Network) and Transformer-based OCR can handle a variety of fonts, including handwritten and distorted fonts.

**5. Font Adaptation Techniques for OCR**

* **Font Training:** Custom OCR training datasets improve recognition of unique fonts.
* **Adaptive Thresholding:** Enhances text visibility for difficult fonts.
* **Neural Networks:** Improve OCR performance for non-standard fonts.

**6. Conclusion**

OCR performs best on standard serif and sans-serif fonts but struggles with cursive, gothic, and distorted fonts. Advanced deep learning models improve recognition across various font styles. Custom training and preprocessing techniques can further enhance OCR accuracy for specific fonts.

**Why OCR Treats Arial and Times New Roman Similarly**

OCR engines, particularly Tesseract, Google Vision API, and other traditional rule-based models, tend to treat **Arial** and **Times New Roman** similarly because they share several key characteristics in terms of structure and readability. Here’s a detailed breakdown of why this happens:

**1. Structural Similarities in Arial and Times New Roman**

Both fonts, despite being different (sans-serif vs. serif), share **consistent character proportions**, which helps OCR engines recognize them efficiently.

**2. OCR Algorithms Focus on Character Geometry, Not Styling**

OCR engines recognize text primarily through **shape, spacing, and contrast** rather than stylistic elements like serifs.

* **Arial and Times New Roman have distinct visual differences, but their basic letterforms are well-defined and easy to segment.**
* The minor differences in serifs (e.g., small strokes at the ends of Times New Roman characters) do not impact OCR significantly because the core letter structure remains intact.

**3. Proportional Fonts vs. Monospaced Fonts**

Both Arial and Times New Roman are **proportional fonts**, meaning character widths vary depending on the letter.

* OCR engines are trained on proportional fonts, making it easier for them to process both Arial and Times New Roman with **high accuracy**.
* Unlike **monospaced fonts** (like Courier), where every letter takes the same space, proportional fonts mimic natural reading patterns, improving OCR detection.

**4. Impact of Font Differences on OCR Accuracy**

While OCR handles both fonts well, **Times New Roman may have a slight advantage in printed documents**:

* **Serifs in Times New Roman can help define letters better, especially in low-resolution scans.**
* In some cases, OCR engines may detect **Times New Roman more reliably in blurry or noisy images**, as the serifs help distinguish letters like **I (uppercase i) and l (lowercase L)**.

However, **for digital and high-resolution scans, the difference in OCR accuracy between Arial and Times New Roman is negligible**.

**5. When Does the Difference Matter?**

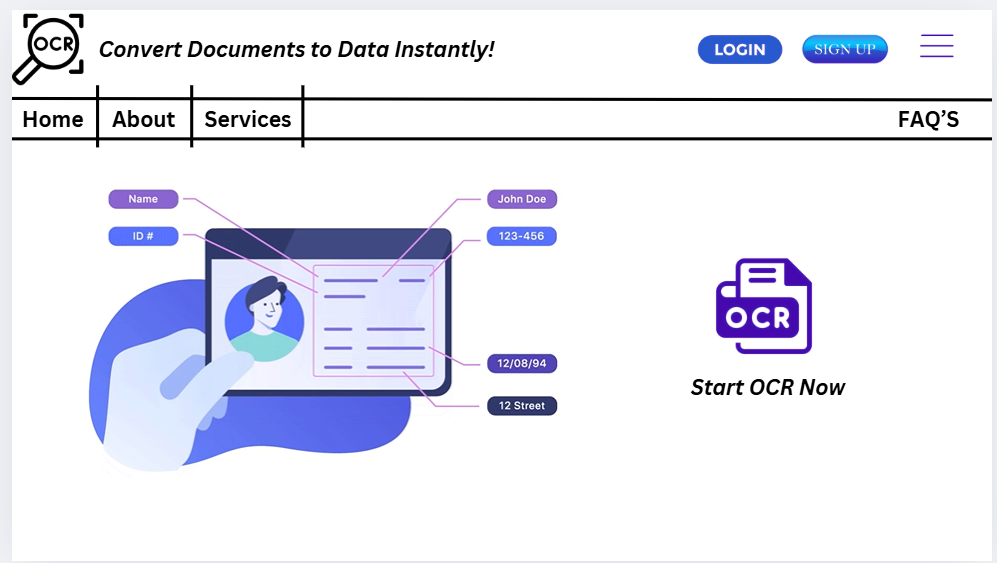
* **Low-resolution images:** Times New Roman may have an edge due to its serif structure, which helps separate characters.
* **Cursive or stylized variations:** Arial performs slightly better because it's more uniform and lacks additional styling.
* **Noisy backgrounds:** OCR might struggle equally with both unless additional preprocessing is applied.

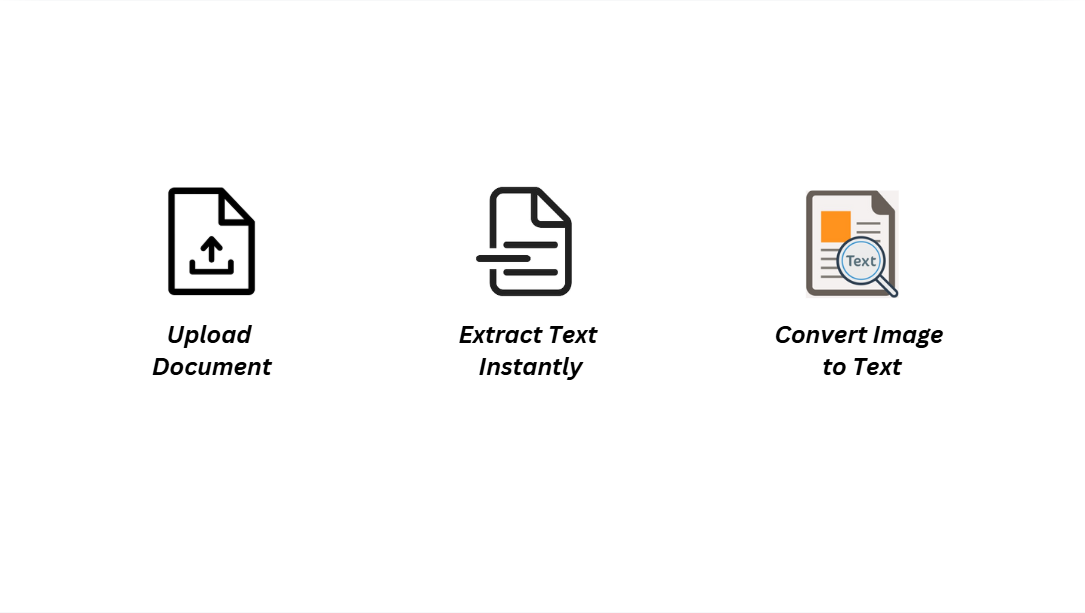
**Conclusion**

* OCR engines treat Arial and Times New Roman similarly because **they are both clear, well-structured, and commonly used fonts**.
* The presence or absence of **serifs does not significantly affect OCR accuracy** in most cases.
* **OCR errors are more likely due to image quality, noise, or text distortions rather than font selection between these two.**

**20/02/25**

**FRONT END PAGE**

****

****

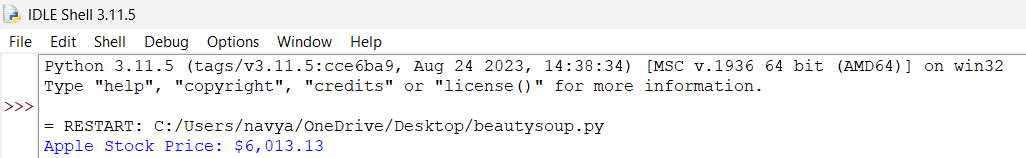
**21/02/25**

**SCRAP THE DATA FROM FINANCIAL DATA AND SAVE IN CSV FILE**

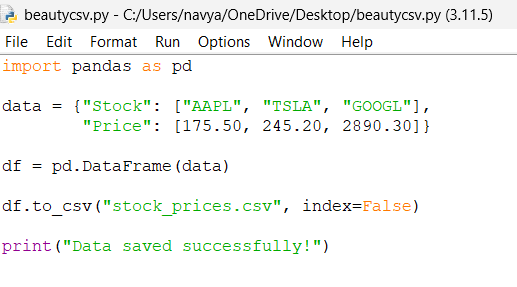
**Code**

****

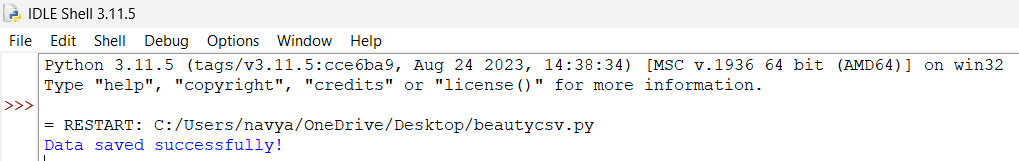
**Out put**

****

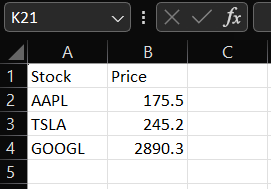
**Csv**

****

**Out put**

****

**Csv file**

****

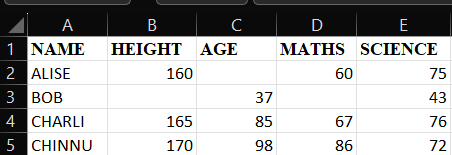
**24/02/25**

**TASK**

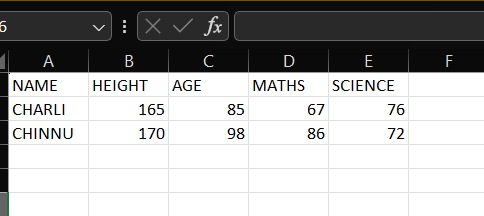
**Collecting missing data information helps in selecting the right method for handling it. By identifying missing values, analyzing their patterns, measuring their percentage, and tracking correlations, you can make informed decisions on whether to remove, impute, or predict missing values.**

**Removing Missing Values from a Database:**

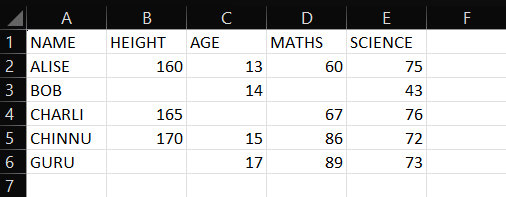
**1. Removing Rows with Missing Values**

****

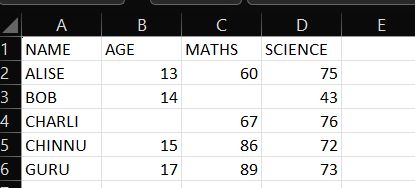
**Out put:**

****

**2. Removing Columns with Too Many Missing Values**

****

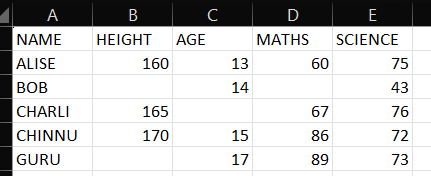
**Out put**

****

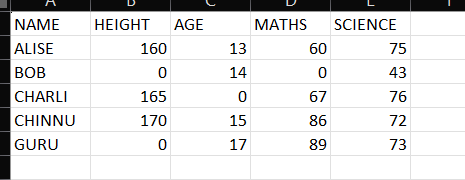
**2. Imputation**

**Fill with a Specific Value:**

**Input**

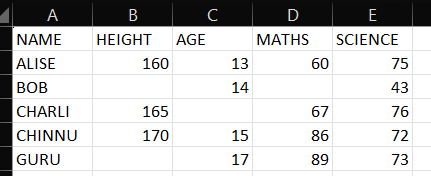
****

**Out put**

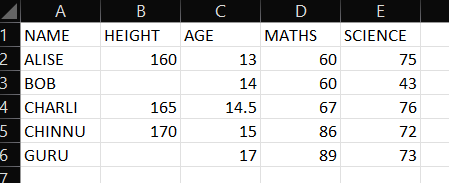
****

**Fill with Mean, Median, or Mode:**

**Input**

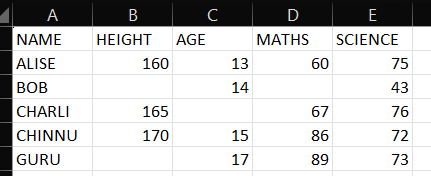
****

**Output**

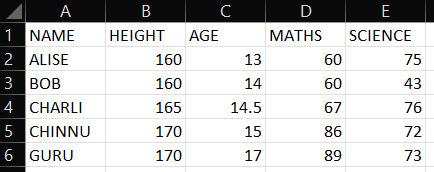
****

**Forward Fill & Backward Fill**

**Input**

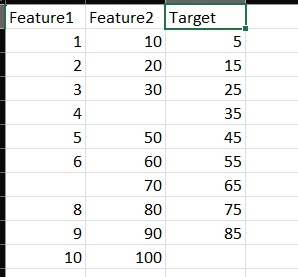
****

**Output**

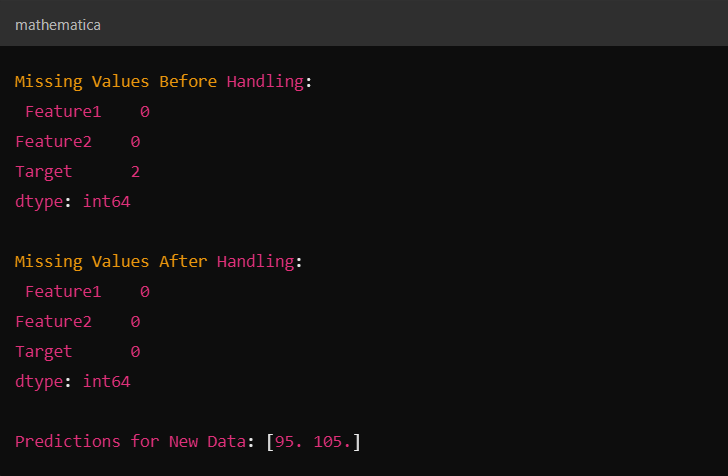
****

**PREDICTION MODEL**

**INPUT**

****

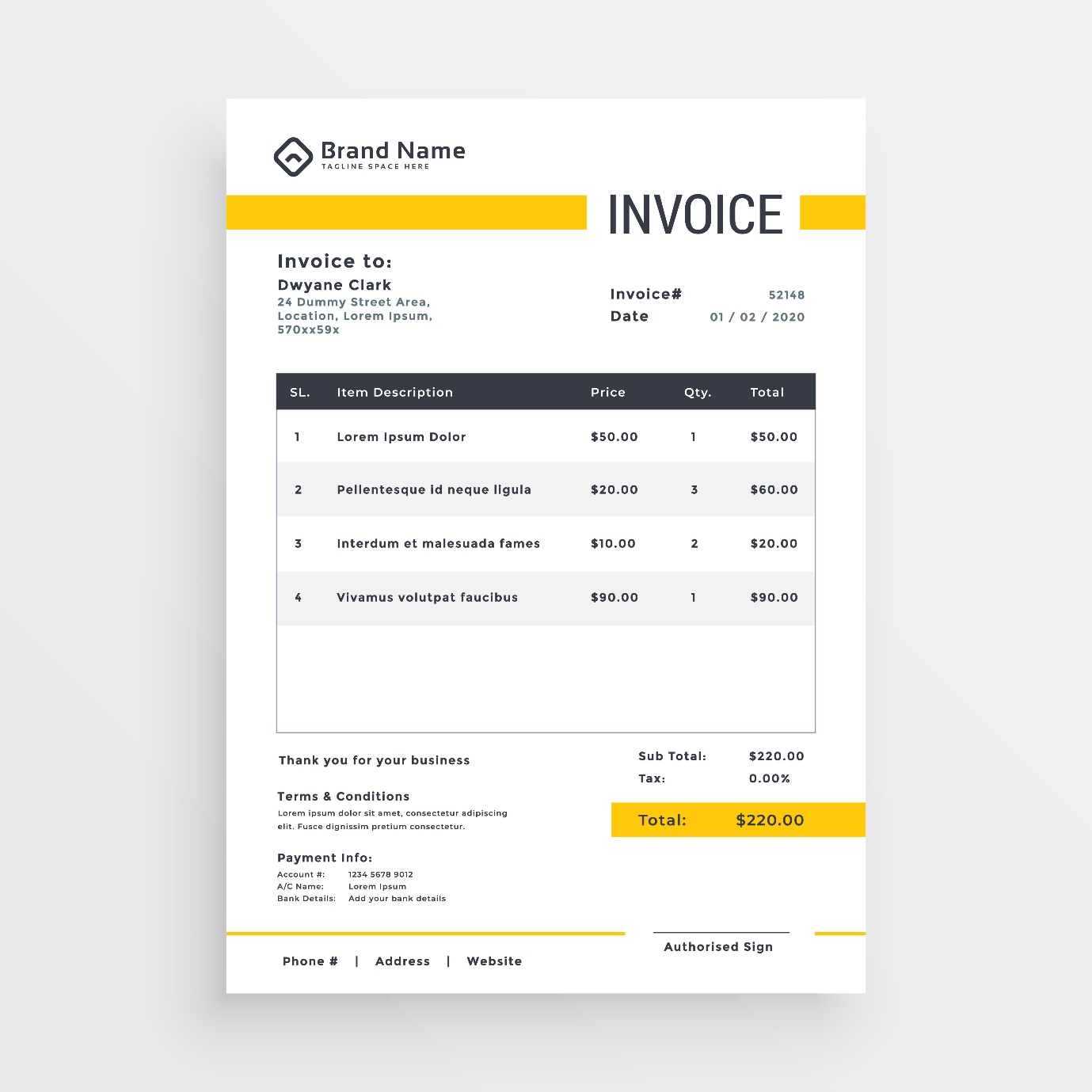
**Output**

****

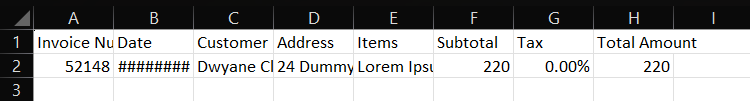
**25/2/25**

**VISUALIZATION**

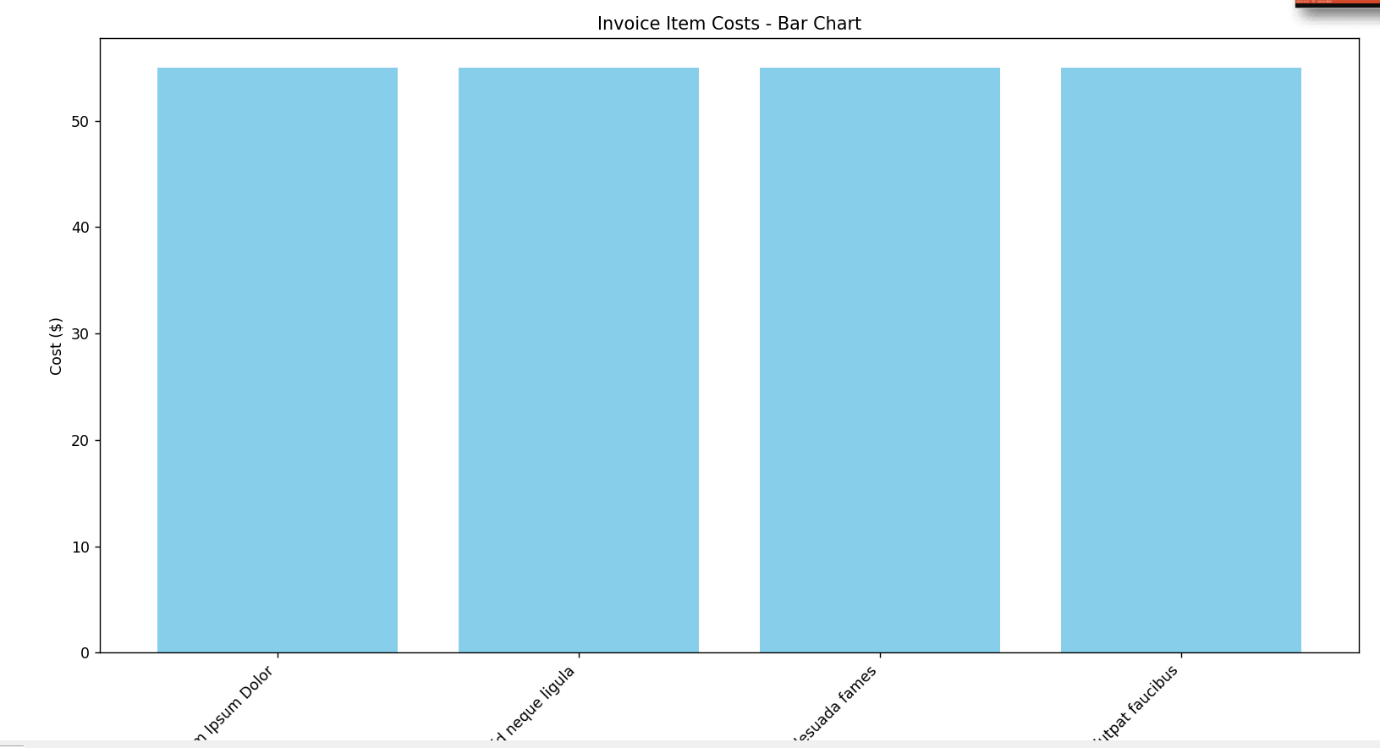
**Invoice 1**

****

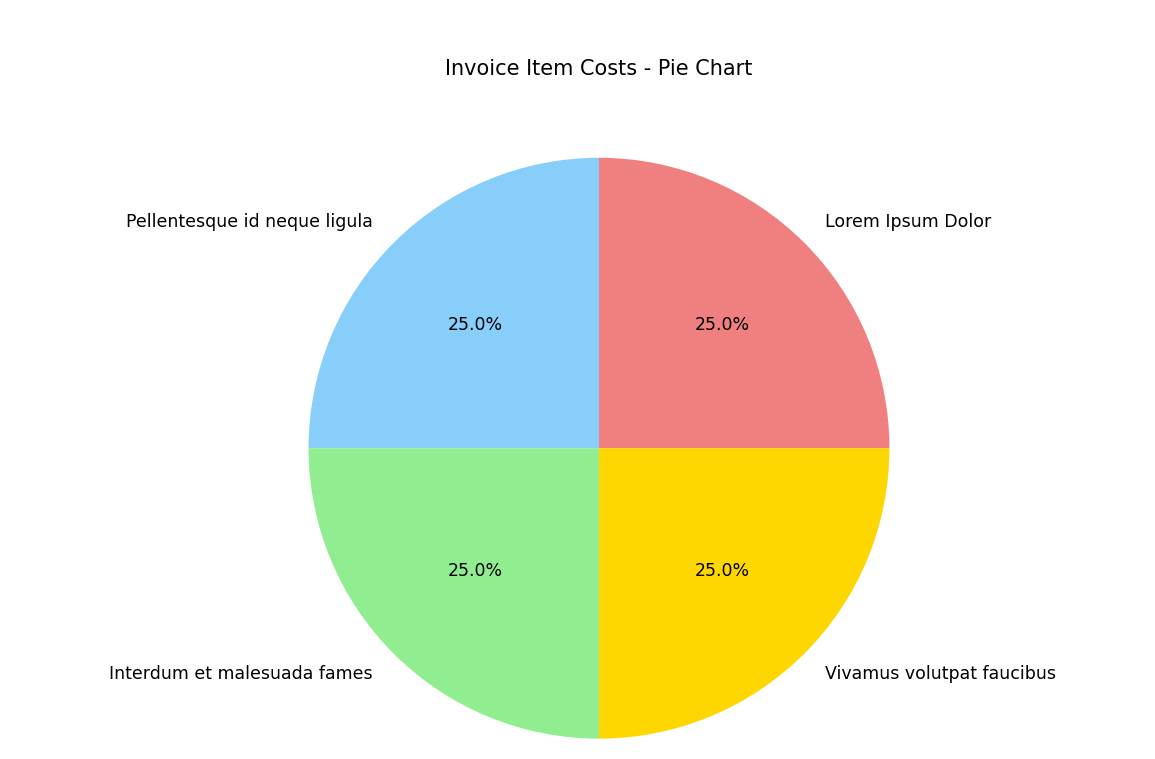
**Extracted data to csv**

****

**Bar Chart**

****

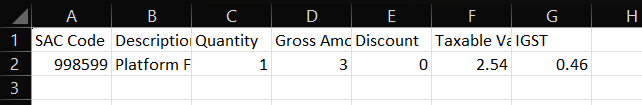
**Pie Chart**

****

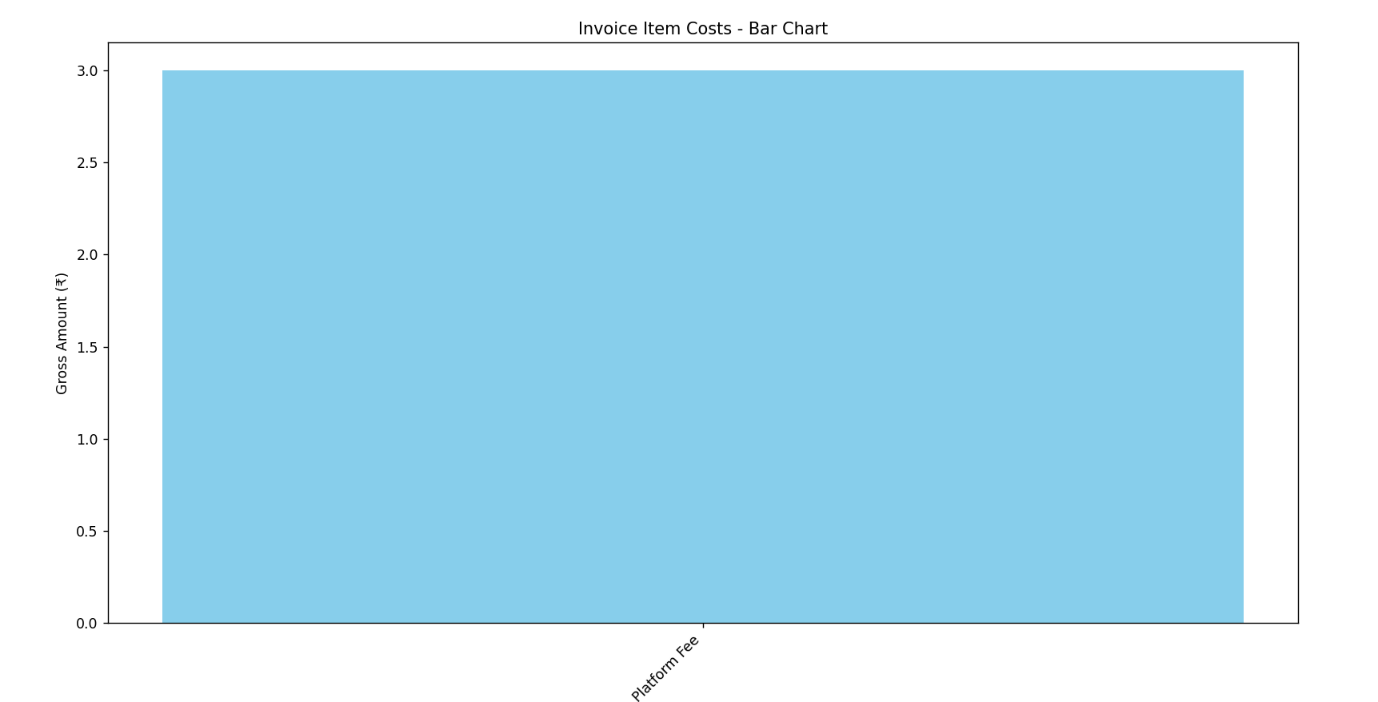
**Invoice 2**

****

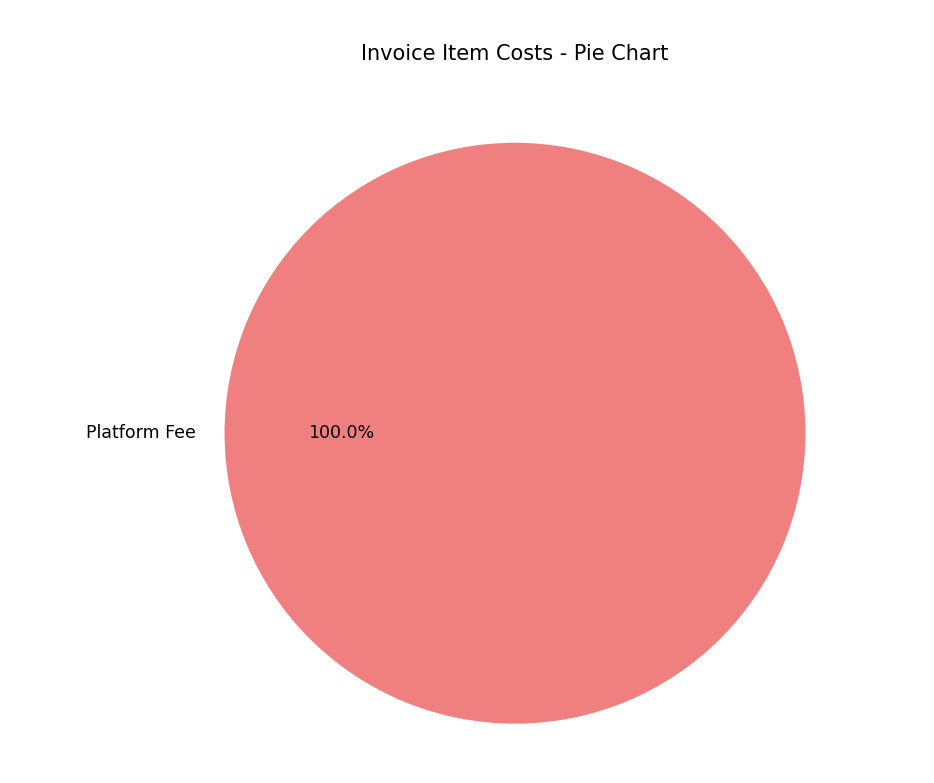
**Extracted data to csv**

****

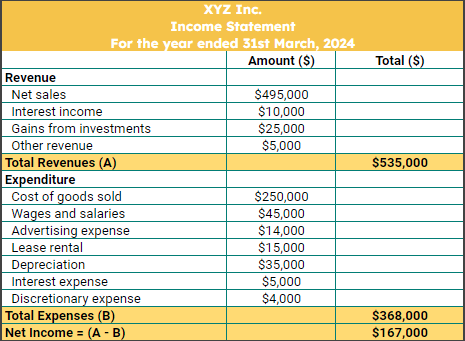
**Bar Chart**

****

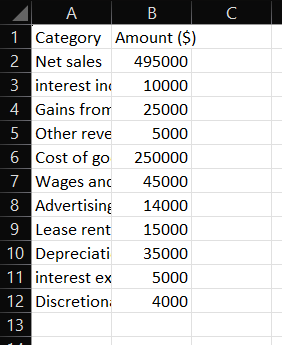
**Pie chart**

****

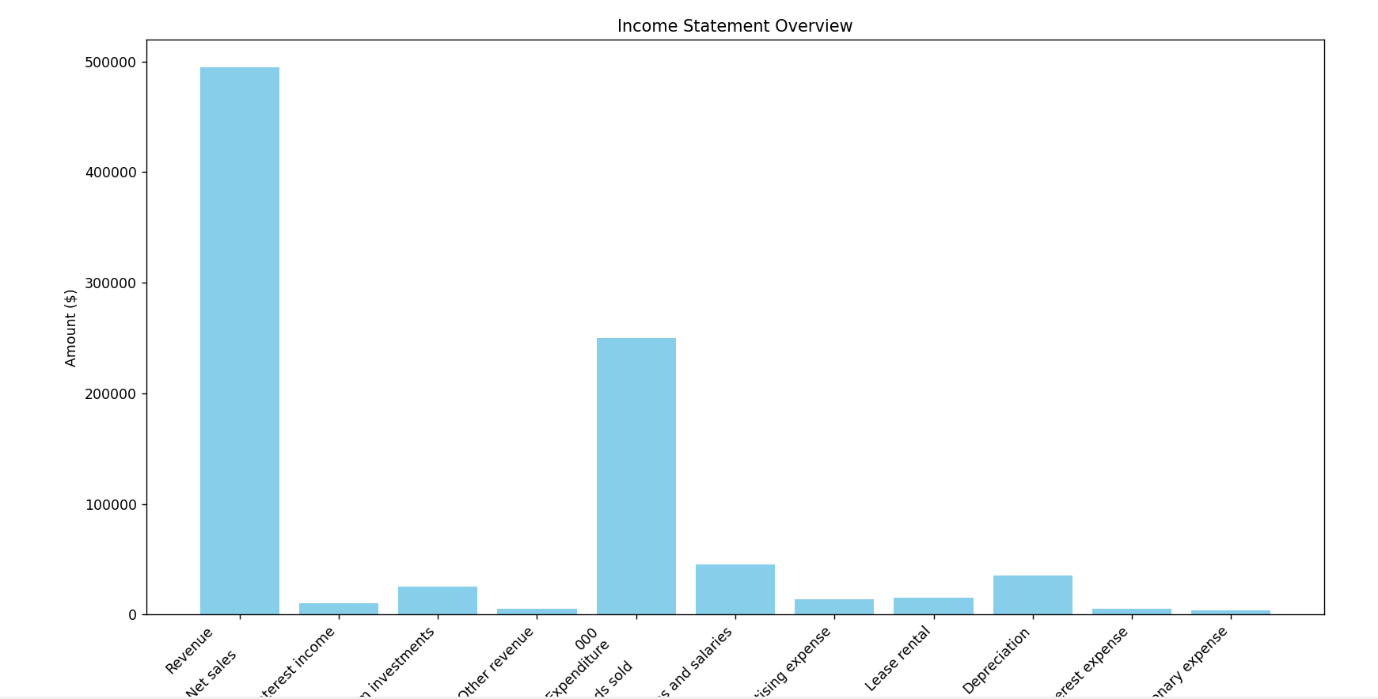
**Profit loss 1**

****

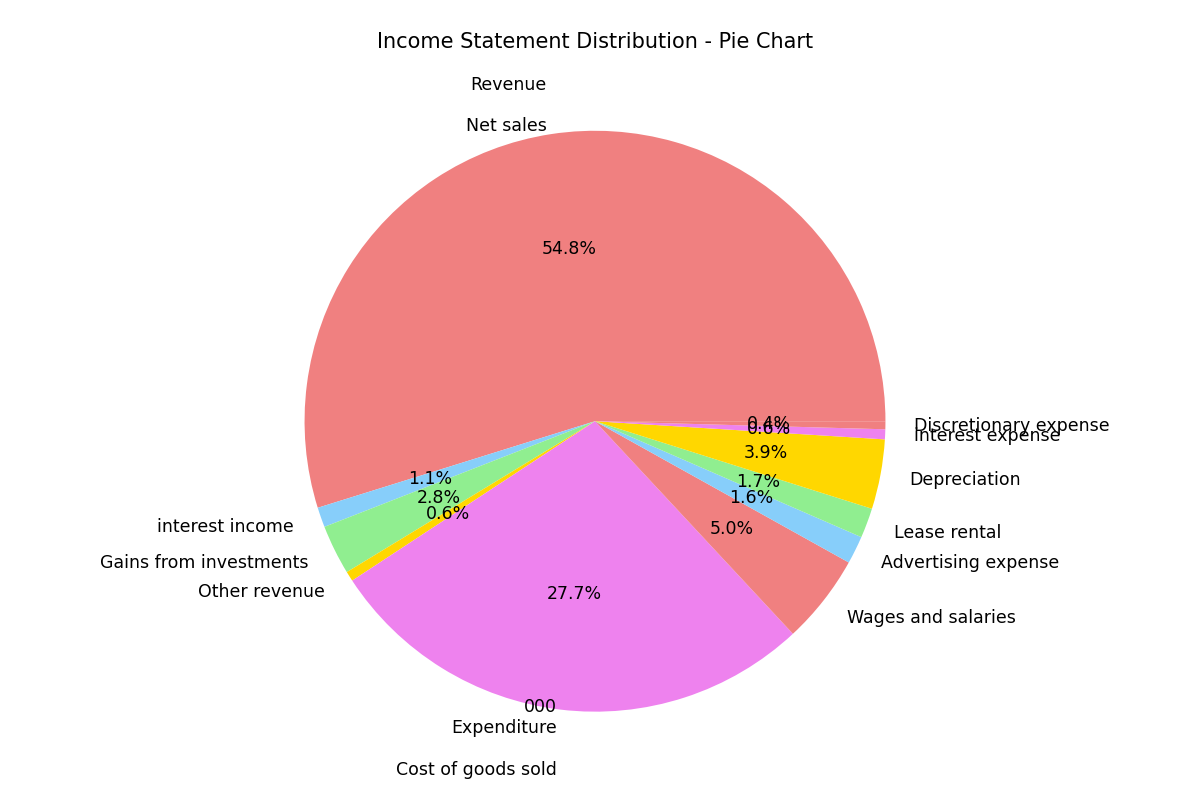
**Extract data to csv**

****

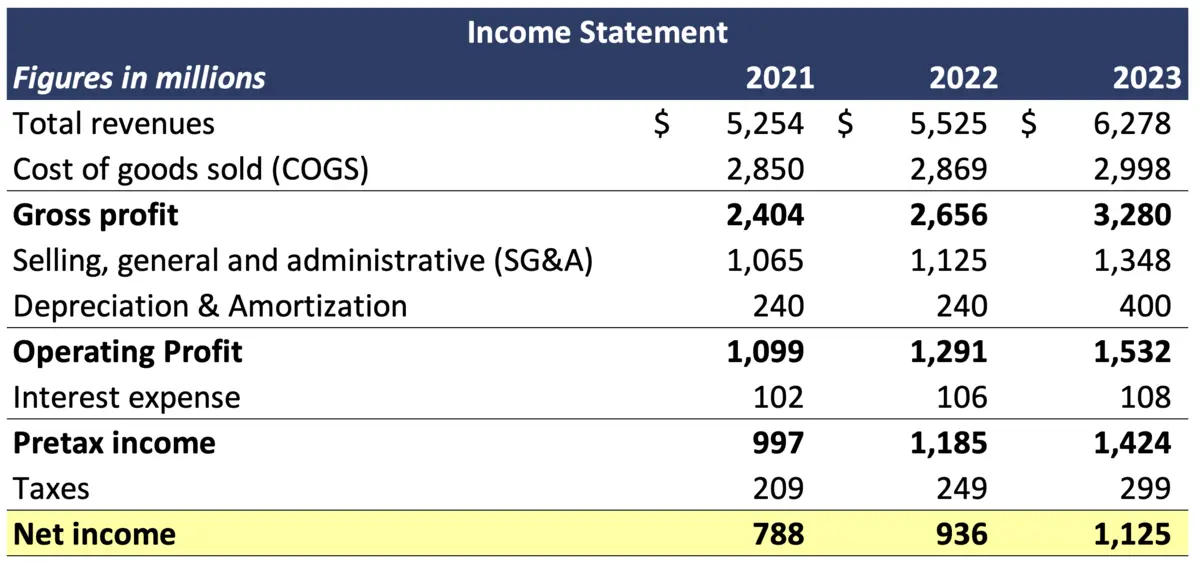
**Bar chart**

****

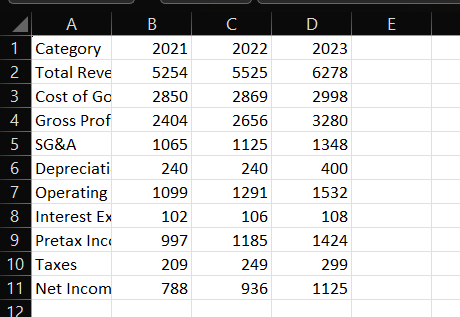
**Pie chart**

****

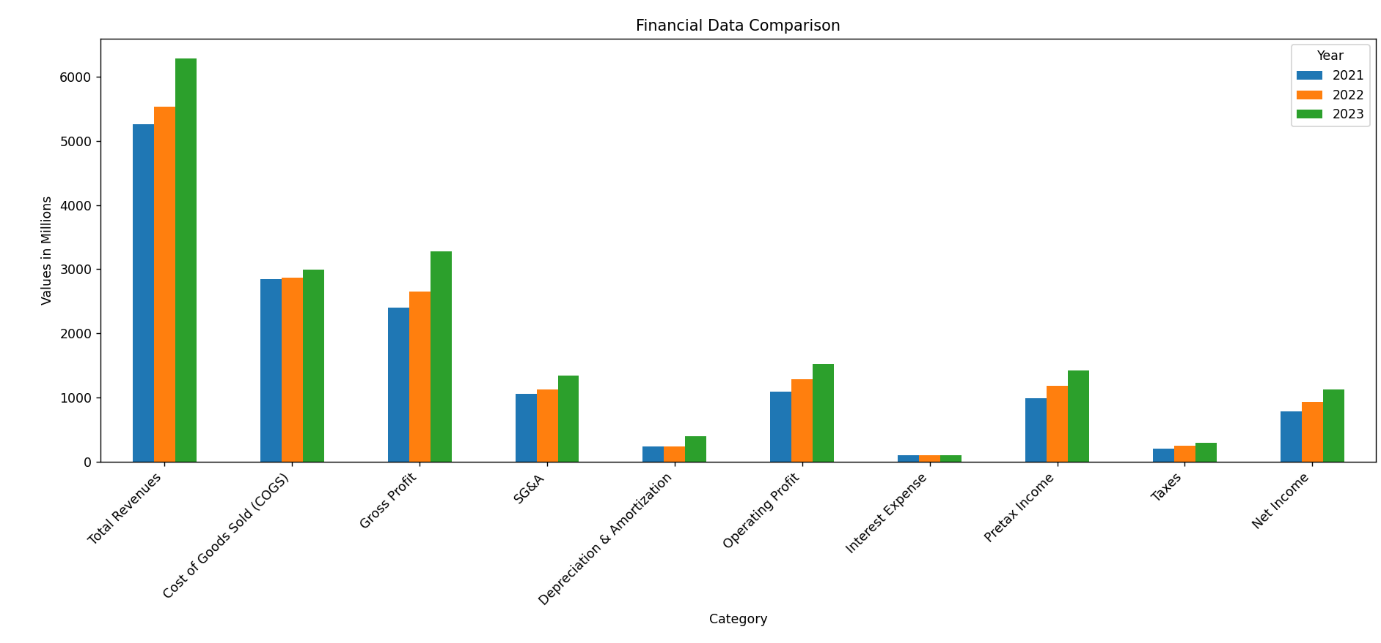
**Profit and loss 2**

****

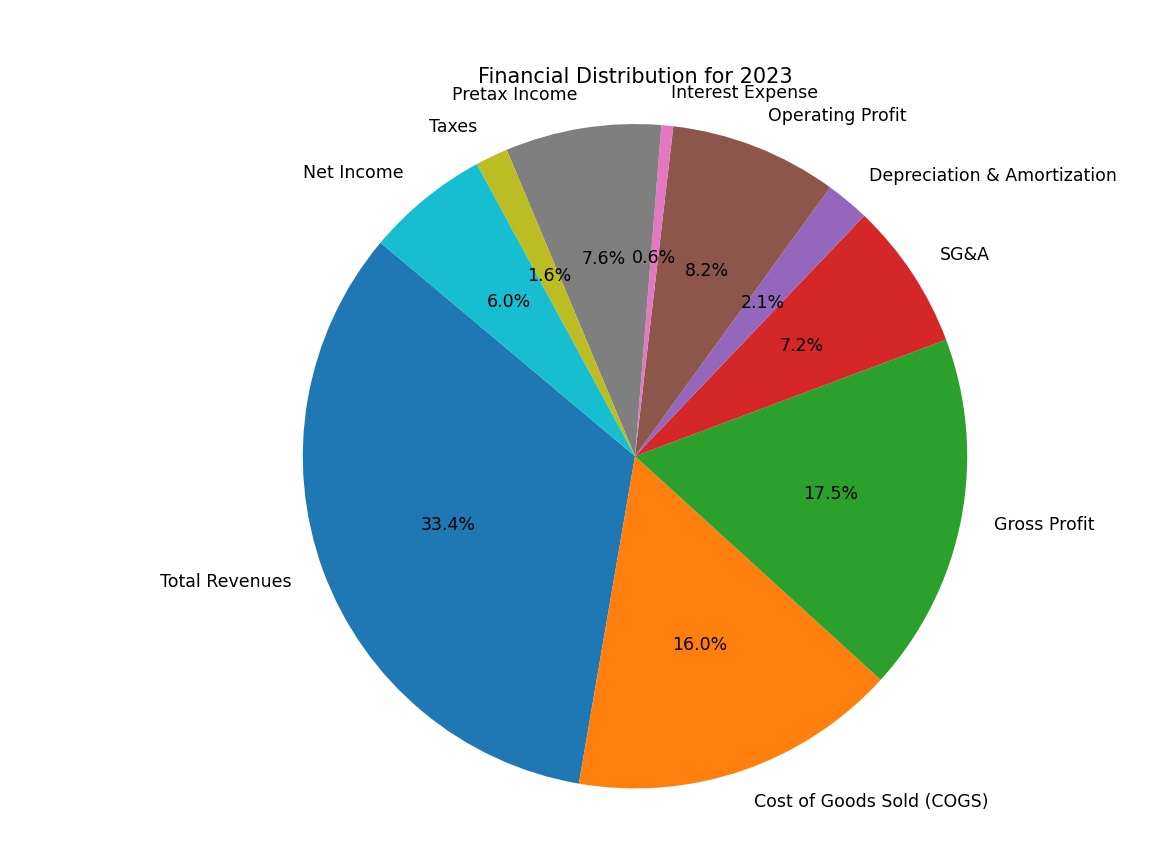
**Extract data to csv**

****

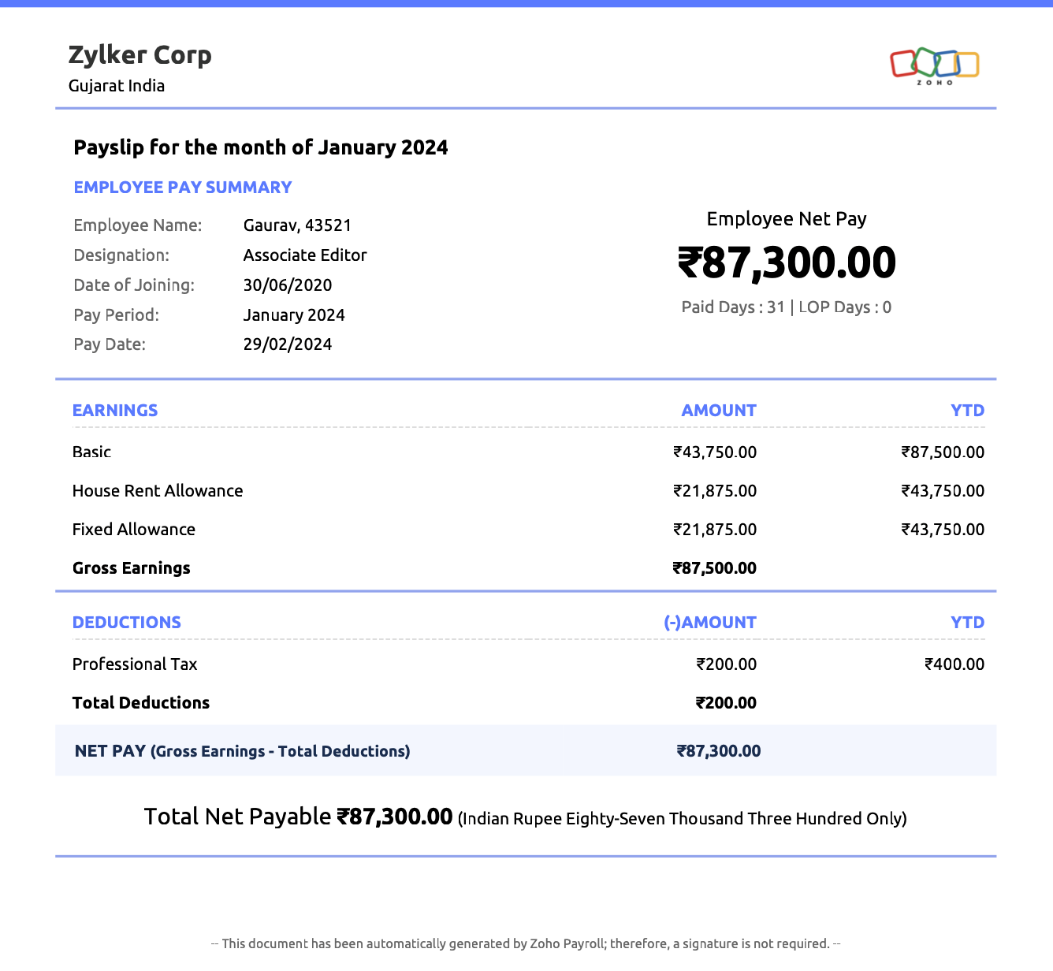
**Bar graph**

****

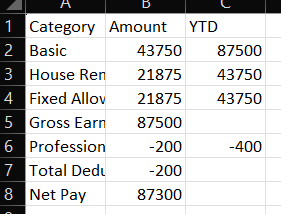
**Pie chart**

****

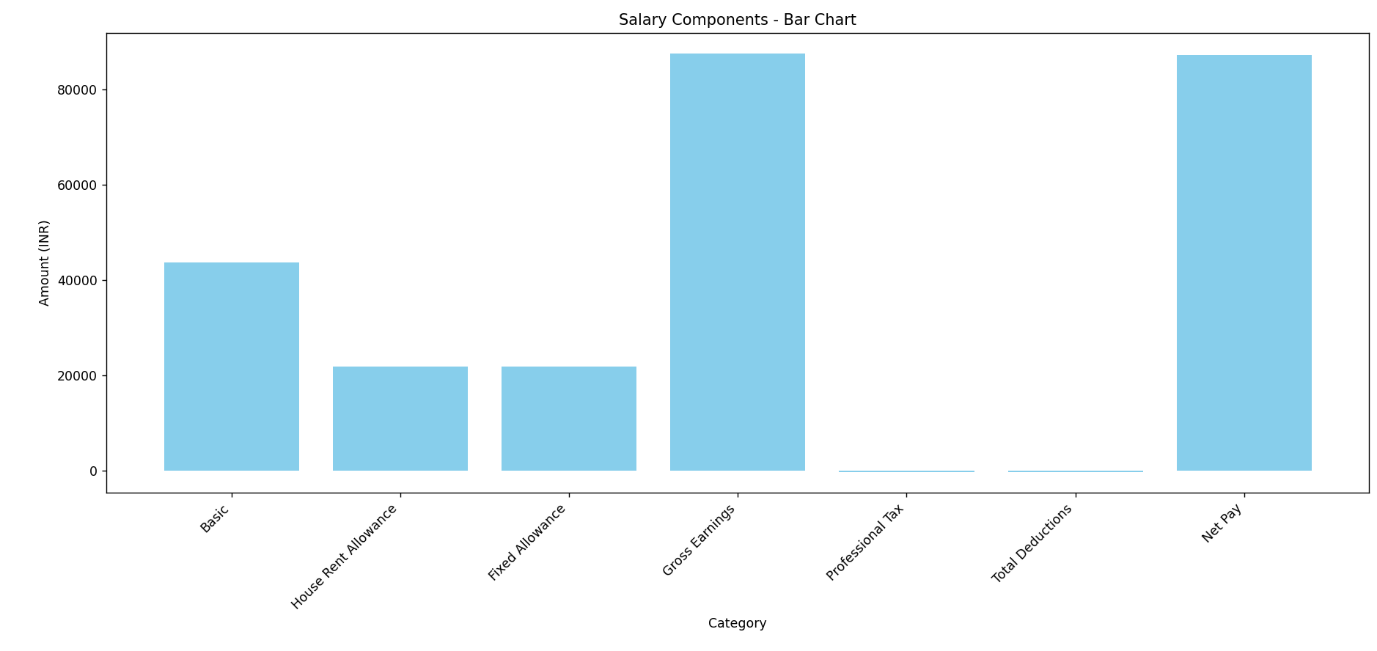
**Salary slip**

****

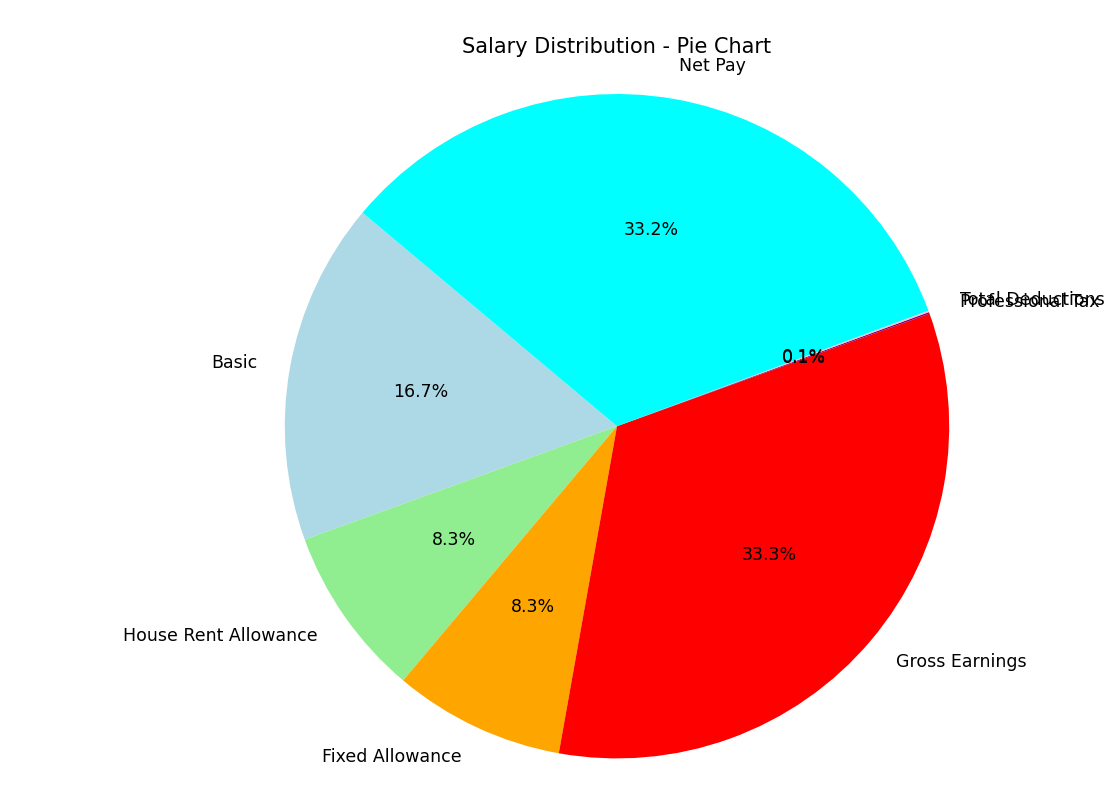
**Extracted data**

****

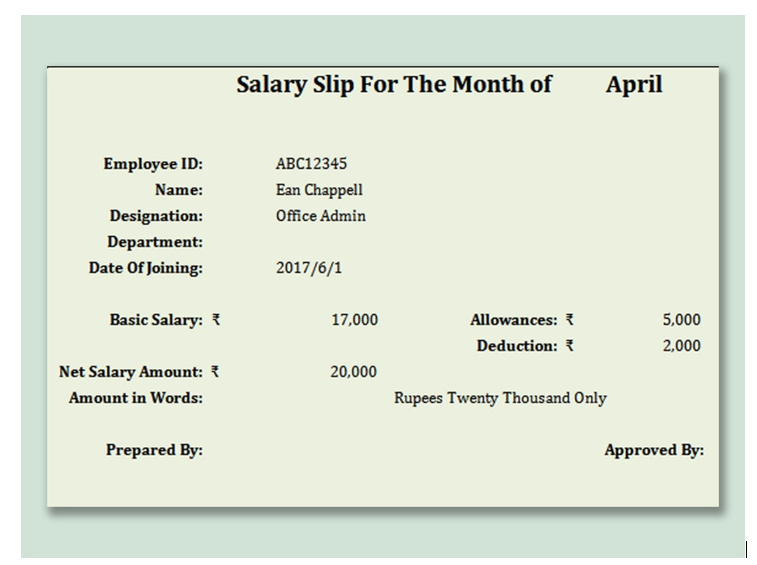
**Bar chart**

****

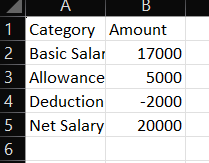
**Pie chart**

****

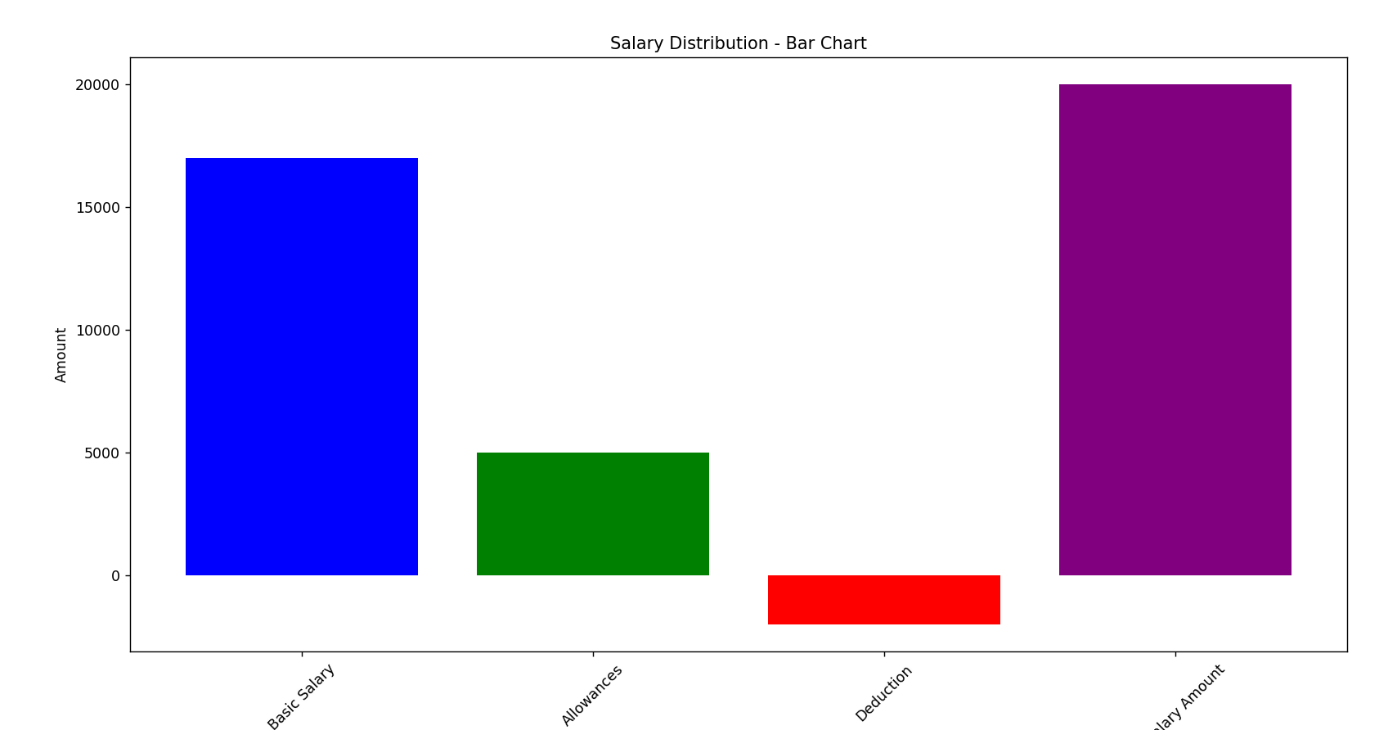
**Salary slip 2**

****

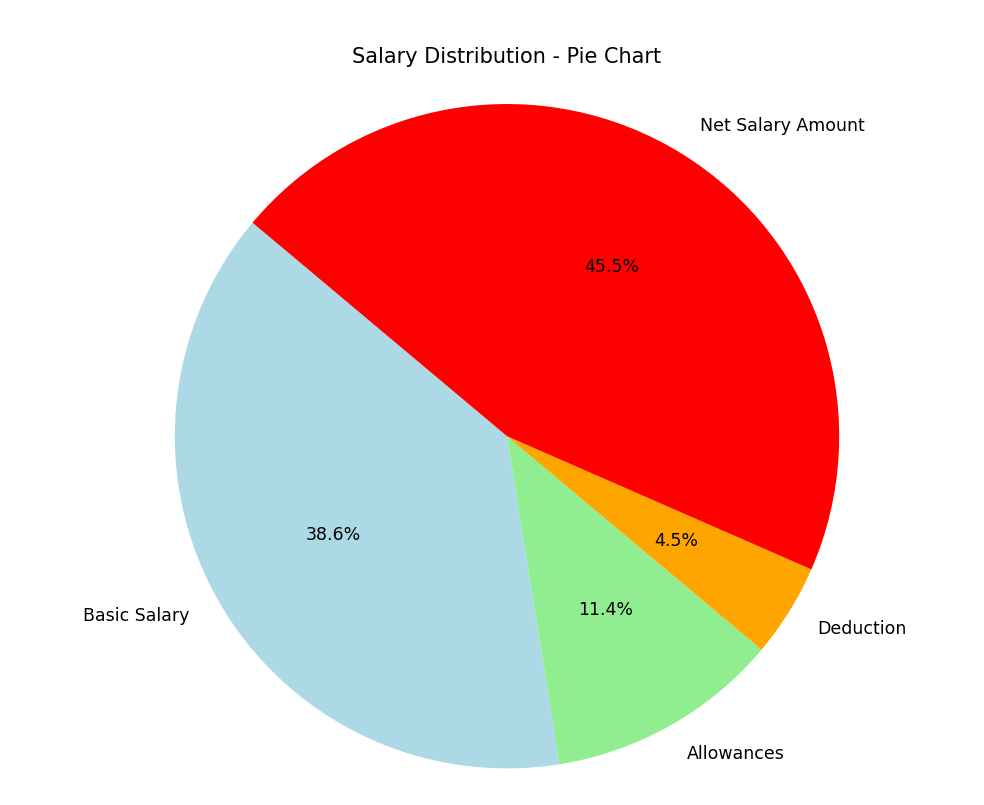
**Extracted data**

****

**Bar graph**

****

**Pie chart**

****

**26/02/25**

**Free installation of LLM model**

If you want to install a Large Language Model (LLM) for free, you have a few options depending on your hardware and use case. Below is a step-by-step guide for installing and running an LLM locally.

**1. Choose an LLM Model**

Popular free open-source LLMs:

* **Llama 2 (Meta)** – Available in 7B, 13B, and 70B parameters.
* **Mistral 7B** – A powerful 7B parameter model.
* **Falcon (TII)** – Falcon 7B and 40B.
* **GPT-NEO (EleutherAI)** – A free alternative to GPT-3.
* **StableLM (StabilityAI)** – A good lightweight model.

**2. System Requirements**

Your hardware affects which model you can run:

* **Low-end (4GB–8GB RAM, no GPU)** → Try **GPT4All, LLaMA 2 7B (quantized)**
* **Mid-range (16GB RAM, RTX 3060 or higher GPU)** → LLaMA 2 13B, Mistral 7B
* **High-end (32GB+ RAM, RTX 4090, or A100 GPU)** → LLaMA 2 70B, Falcon 40B
* **For Beginners:** Use **Ollama** (ollama run mistral) or **GPT4All**.
* **For Developers:** Use text-generation-webui.
* **For Cloud Users:** Try **Hugging Face** or **Google Colab**.

**Hugging Face**

Hugging Face is the leading platform for natural language processing (NLP) and machine learning (ML). It provides open-source models, datasets, and tools for building AI applications, including LLMs (Large Language Models), computer vision, audio processing, and more.

**What is Hugging Face?**

Hugging Face is a company and open-source platform that provides:

* **Pretrained models** (LLMs, Vision, Audio, Reinforcement Learning)
* **Datasets** for ML research
* **APIs** for easy AI deployment
* **Training & fine-tuning tools** for custom AI models
* **Hugging Face Spaces** – A free cloud environment for running ML models

**Key Features of Hugging Face**

**A. Hugging Face Transformers Library**

* Open-source library for **NLP, vision, and audio** tasks.
* Supports **PyTorch & TensorFlow**.
* Used for **text generation, chatbots, summarization, etc.**

**B. Hugging Face Models**

* **Thousands of pre-trained models** (LLMs, vision, audio, reinforcement learning).
* Open-source & easy to use.
* **Popular models available:**
  + **Llama 2** (Meta)
  + **Mistral 7B**
  + **Falcon 7B & 40B**
  + **BERT, GPT-2, T5, Bloom**

**C. Hugging Face Datasets**

* **Over 10,000 datasets** for NLP, vision, and more.
* Supports **large-scale dataset loading & preprocessing**.

**D. Hugging Face Spaces (Free Cloud Environment)**

* **Run models for free on Hugging Face’s cloud** (no installation required).
* Supports **Gradio, Streamlit, Flask, FastAPI**.
* Free **GPU & CPU resources**.

**E. Hugging Face Hub**

* A repository to **store & share models, datasets, and code**.
* Works like GitHub for AI models.
* Supports **model versioning & collaboration**.

**which model is best for categorizing transactions**

For categorizing transactions, the best models depend on the complexity of your data, computational resources, and whether you need a pre-trained or custom-trained model. Here are the best models based on different needs:

**1. Best Pre-trained Models for Transaction Categorization**

**a) FinBERT (Financial BERT)**

* **Best for:** Financial text classification, transaction categorization
* **Pretrained on:** Financial documents & reports
* **Pros:** High accuracy, understands financial terminology

**b) RoBERTa (General Text Classification)**

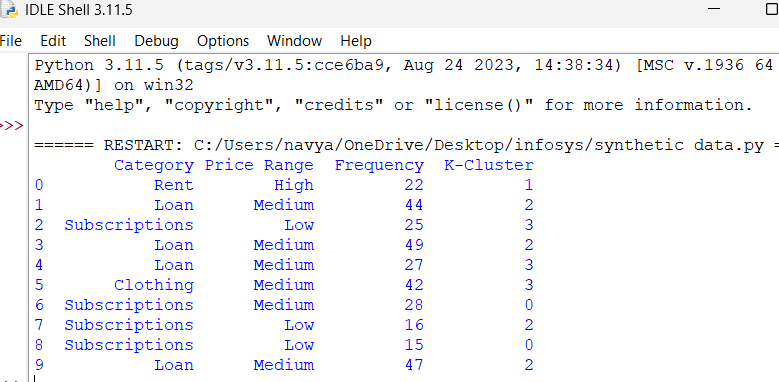
* **Best for:** Multi-category transaction labeling
* **Pros:** Works well if trained on a dataset of labeled transactions

**4. Which Model is Best for You?**

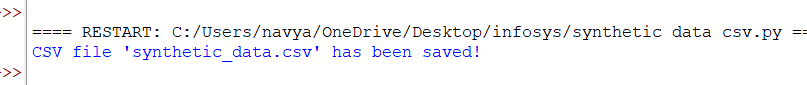
| **Use Case** | **Best Model** |
| --- | --- |
| **Pre-trained & No Training Needed** | FinBERT |
| **Custom Categorization (e.g., Business vs. Personal)** | Fine-tuned BERT |
| **Lightweight & Fast** | Logistic Regression |
| **Large-Scale Categorization (Millions of Transactions)** | RoBERTa |

**27/02/25**

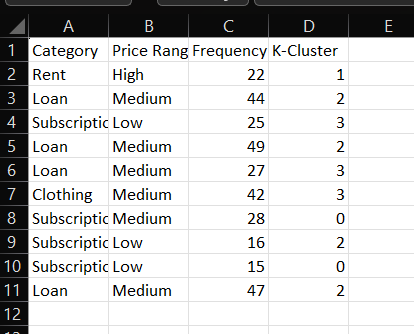
**SYNTHETIC DATA**

****

**Extract data to csv file format**

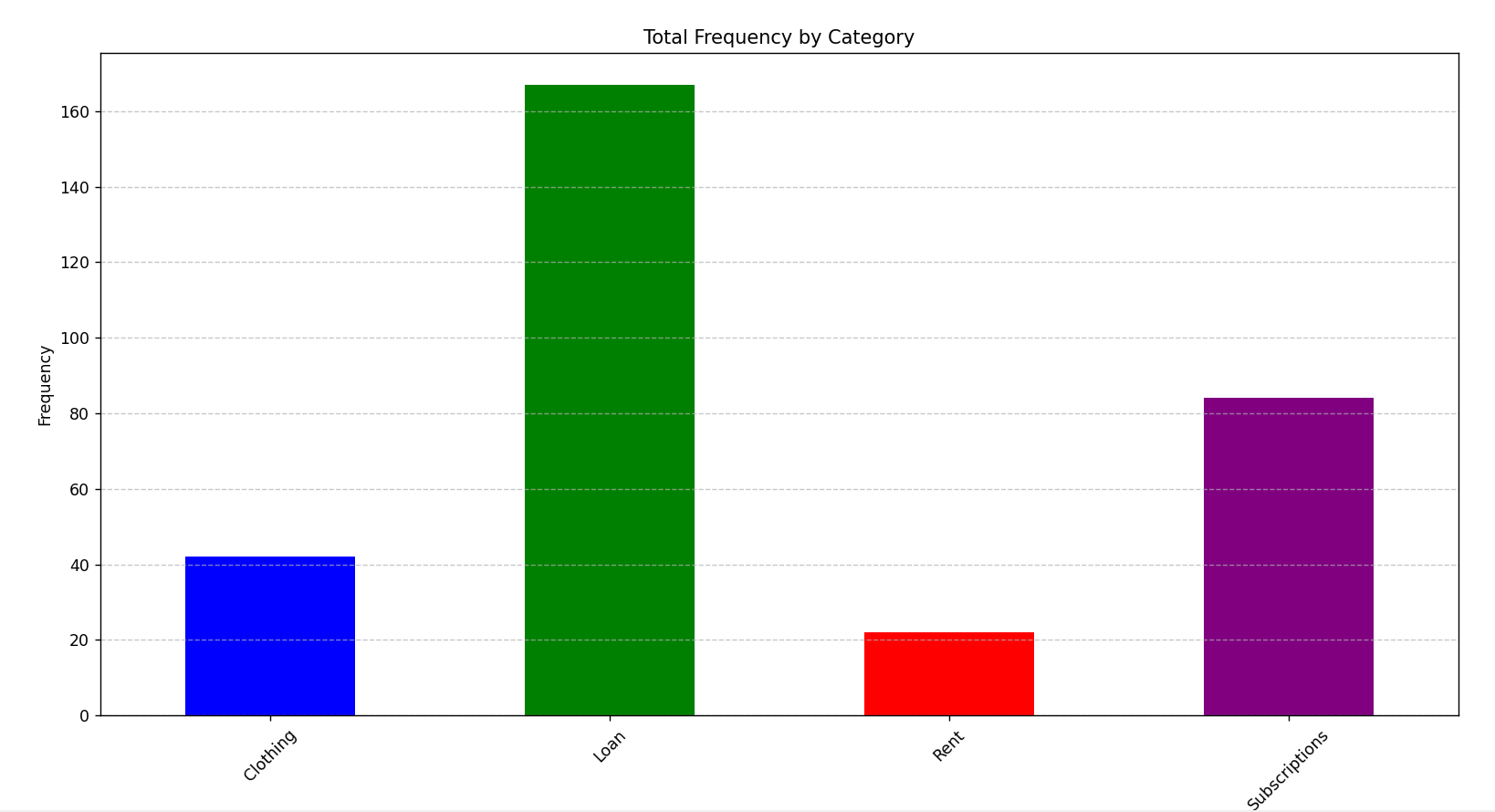
****

**Csv file**

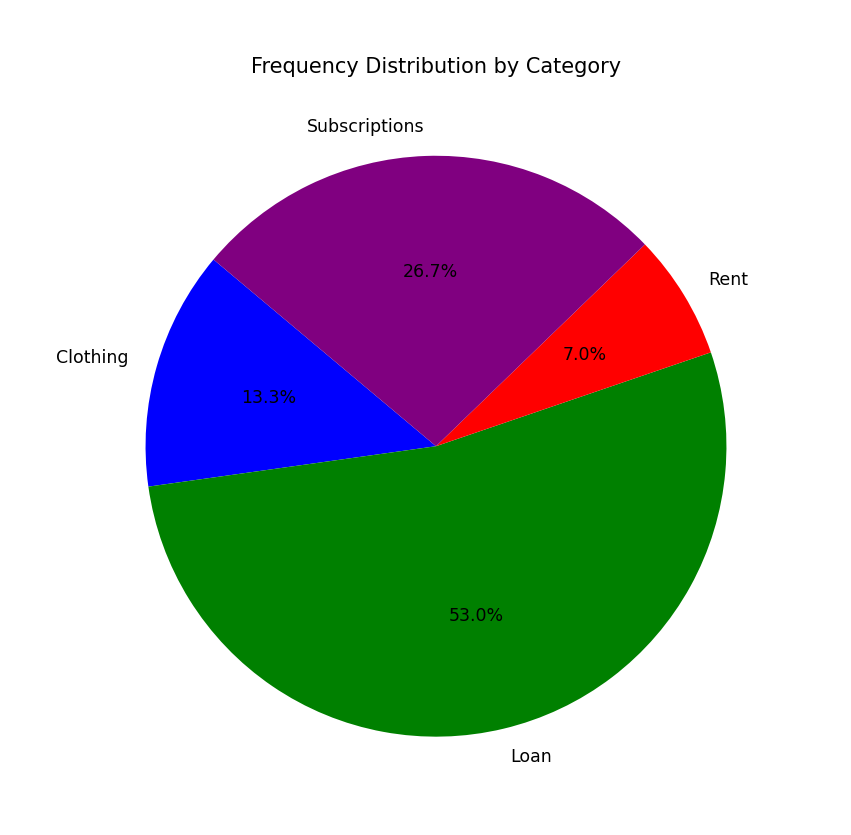
****

**Virtualization of Synthetic data**

**Bar chart**

****

**Pie chart**

****