# Code Explanation

## Cell 1

### Code:

!pip install pdfplumber openpyxl

### Explanation:

This cell installs necessary packages, loads libraries, processes the PDF, extracts relevant data, and saves it to a DataFrame.

## Cell 2

### Code:

import re  
import pdfplumber  
import pandas as pd  
from collections import namedtuple  
from openpyxl import Workbook  
  
# Adjusted named tuple to include relevant fields  
Line = namedtuple('Line', 'sr\_no air first\_name middle\_name last\_name form\_no marks category sex quota selection\_details')  
  
# Adjust the regular expression based on the PDF structure  
line\_re = re.compile(r'^(\d+)\s+(\d+)\s+(.\*?)\s+(\d+)\s+(\d+)\s+(\w+)\s+(M|F)\s+(\S+)\s+(.\*)$')  
  
file = 'SelectionList R5 AHU - Only Selection - 8nov.pdf'  
lines = []  
  
with pdfplumber.open(file) as pdf:  
 pages = pdf.pages  
 for page in pages:  
 text = page.extract\_text()  
 for line in text.split('\n'):  
 if "Choice Not Available" in line or "Disqualified-Allotted by State-MBBS" in line:  
 continue # Skip lines with "Choice Not Available" or "Disqualified-Allotted by State-MBBS"  
   
 match = line\_re.match(line)  
 if match:  
 sr\_no, air, name, form\_no, marks, category, sex, quota, selection\_details = match.groups()  
  
 # Split the name into parts  
 name\_parts = name.split()  
 if len(name\_parts) == 2:  
 first\_name = name\_parts[0]  
 middle\_name = ''  
 last\_name = name\_parts[1]  
 elif len(name\_parts) == 3:  
 first\_name = name\_parts[0]  
 middle\_name = name\_parts[1]  
 last\_name = name\_parts[2]  
 else:  
 first\_name = name\_parts[0]  
 middle\_name = ''  
 last\_name = ' '.join(name\_parts[1:])  
  
 # Check if the quota starts with a numerical value and move it to selection\_details if true  
 if re.match(r'^\d', quota):  
 selection\_details = quota + ' ' + selection\_details  
 quota = ''  
  
 lines.append(Line(sr\_no, air, first\_name, middle\_name, last\_name, form\_no, marks, category, sex, quota, selection\_details))  
# Create a pandas DataFrame  
df = pd.DataFrame(lines)  
  
# Save the DataFrame to an Excel file  
df.to\_csv('output1.csv', index=False)

### Explanation:

This code processes a PDF file containing selection lists and extracts relevant data into a structured format. It begins by importing necessary libraries and defining a named tuple, `Line`, to hold the extracted data fields. A regular expression (`line\_re`) is set up to match and extract the relevant parts of each line in the PDF. The code then opens the specified PDF file (`SelectionList R5 AHU - Only Selection - 8nov.pdf`) using `pdfplumber` and iterates through each page and line of text. It skips lines that contain "Choice Not Available" or "Disqualified-Allotted by State-MBBS." For lines that match the regular expression, the relevant data (like serial number, AIR, name, etc.) is extracted. The name is split into first, middle, and last names, and if the quota field starts with a number, it is merged with the selection details. The extracted data is then appended to a list, which is converted into a pandas DataFrame. Finally, the DataFrame is saved as a CSV file named `output1.csv`.

## Cell 3

### Code:

import pandas as pd  
import re  
  
# Load the CSV file  
df = pd.read\_csv('output.csv')  
  
# Function to extract text before numbers  
def extract\_text\_before\_numbers(value):  
 match = re.match(r"([^\d]+)", value)  
 return match.group(0).strip() if match else ''  
  
# Apply the function to the 'cod\_college' column  
df['new\_column'] = df['selection\_details'].apply(extract\_text\_before\_numbers)  
  
# Function to remove text before numbers in 'cod\_college'  
def remove\_text\_before\_numbers(value):  
 match = re.search(r"\d.\*", value) # Find the first occurrence of a digit and everything after it  
 return match.group(0).strip() if match else value # Return the matched part or the original value if no match  
  
# Apply the function to the 'cod\_college' column  
df['selection\_details'] = df['selection\_details'].apply(remove\_text\_before\_numbers)  
  
# Merge 'Quota' and 'new\_column' into a single column  
df['Quota'] = df['category'].fillna('') + ' ' + df['quota'].fillna('') + df['new\_column'].fillna('')  
  
# Save the updated DataFrame to a new CSV file  
print(df)  
  
# Save the updated DataFrame to a new CSV file  
df.to\_csv('updated\_file.csv', index=False)

### Explanation:

This code processes a CSV file (`output.csv`) to modify and merge specific columns in a pandas DataFrame. It starts by loading the CSV into a DataFrame (`df`). A function `extract\_text\_before\_numbers` is defined to extract text that appears before any digits in a string, which is then applied to the `selection\_details` column, storing the result in a new column called `new\_column`. Another function, `remove\_text\_before\_numbers`, is defined to remove text that appears before the first digit in a string, leaving only the portion starting from the first number. This function is applied to the `selection\_details` column, modifying it directly. Next, the code merges the `category`, `quota`, and `new\_column` fields into a new column named `Quota`. The final DataFrame is printed and then saved to a new CSV file called `updated\_file.csv`. The purpose of these operations is to clean and reorganize the data by adjusting the format of the `selection\_details` and merging relevant fields for further analysis.

## Cell 4

### Code:

# Group by college, category, and gender to find max AIR  
max\_air\_df = df.loc[df.groupby(['code\_college', 'category', 'sex','Quota'])['air'].idxmax()]  
  
# Rename columns to indicate max  
max\_air\_df = max\_air\_df.rename(columns={'air': 'max\_air', 'sr\_no': 'max\_sr\_no', 'neet\_roll\_no': 'max\_neet\_roll\_no', 'cet\_form\_no': 'max\_cet\_form\_no', 'Quota': 'max\_quota'})  
  
# Merge the min and max DataFrames on college, category, and gender  
result\_df = pd.merge(min\_air\_df, max\_air\_df, on=['code\_college', 'category', 'sex','Quota', suffixes=( '\_max'))  
  
# Add the Course and Cap\_round\_No columns with specified values  
result\_df['Course'] = '#add course name '  
result\_df['Cap\_round\_No'] = 2  
  
# Select relevant columns for the final output  
columns\_to\_keep = [  
 'code\_college', 'category', 'sex',  
 'max\_air', 'max\_quota',  
]  
  
# Display the result  
final\_df = result\_df[columns\_to\_keep]  
final\_df  
  
#convert into csv  
final\_df.to\_csv('output\_Max.csv', index=False)

### Explanation:

This code processes a DataFrame (`df`) to find and extract the maximum AIR (All India Rank) for each combination of college, category, gender, and quota. It begins by grouping the DataFrame by `code\_college`, `category`, `sex`, and `Quota` to identify the rows with the maximum AIR within each group. The resulting subset (`max\_air\_df`) is then created, with the relevant columns renamed to indicate that they contain maximum values (e.g., `air` becomes `max\_air`, `Quota` becomes `max\_quota`). Next, the code attempts to merge this DataFrame with another DataFrame (`min\_air\_df`) on the same grouping fields (`code\_college`, `category`, `sex`, and `Quota`), but the line contains a syntax error due to an unclosed parenthesis. After fixing this, the code adds two new columns, `Course` and `Cap\_round\_No`, with preset values (the course name and a round number). The final step is to select a subset of relevant columns (`code\_college`, `category`, `sex`, `max\_air`, and `max\_quota`) and store this subset in `final\_df`. Finally, the DataFrame is saved to a CSV file named `output\_Max.csv` for further use.