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.