In [154]:

```
import pandas as pd
import numpy as np


# In[2]:


df = pd.read_csv("Patient's Data - Sheet1.csv")
```

In [155]:

```
df.head()
```

Out[155]:

| | Patient's Name | Date Visited | Age | Gender | Followup/New | Contact No | Reference | Dr Name (Consultant) | Total Amount |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Sachin Pawar | 01/11/2024 | - | M | NaN | NaN | - | Dr. Nikesh | 200 |
| **1** | Sushma Sharaf | 01/11/2024 | - | F | NaN | NaN | Dressing | NaN | 300 |
| **2** | Kanchan Mahadik | 01/11/2024 | 63 | - | New | 8657974155 | Dr. R.R Singh | Dr. Sidra Khot | 800 |
| **3** | Vineeta Chaturvedi | 01/11/2024 | 52 | F | New | 7715995596 | Mr.Praveen Dhende | Dr. Sidra Khot | 1000 |
| **4** | Mahesh | 01/11/2024 | NaN | M | NaN | NaN | Injection | - | 100 |

In [156]:

```
df.describe()
```

Out[156]:

| | Patient's Name | Date Visited | Age | Gender | Followup/New | Contact No | Reference | Dr Name (Consultant) | Total Amount |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 1407 | 1406 | 1003 | 1399 | 1050 | 1095 | 773 | 1387 | 1349 |
| **unique** | 1087 | 166 | 82 | 6 | 7 | 726 | 216 | 199 | 71 |
| **top** | Rupali Ambede | 22/02/2025 | - | F | - | - | - | Dr Sidra Khot | 500 |
| **freq** | 12 | 30 | 269 | 556 | 370 | 215 | 369 | 88 | 315 |

In [157]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1407 entries, 0 to 1406
Data columns (total 15 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   Patient's Name        1407 non-null   object
 1   Date Visited          1406 non-null   object
 2   Age                   1003 non-null   object
 3   Gender                1399 non-null   object
 4   Followup/New          1050 non-null   object
```

```
 5   Contact No            1095 non-null   object
 6   Reference             773 non-null    object
 7   Dr Name (Consultant)  1387 non-null   object
 8   Total Amount          1349 non-null   object
 9   Consultant Amount     1164 non-null   object
 10  Hospital Amount       622 non-null    object
 11  Ref G.B               768 non-null    object
 12  Payment Mode          1229 non-null   object
 13  Unnamed: 13           481 non-null    object
 14  Unnamed: 14           481 non-null    object
dtypes: object(15)
memory usage: 165.0+ KB
```

In [158]:

```
df.columns = (
    df.columns
    .str.strip()
    .str.lower()
    .str.replace(' ', '_')
    .str.replace('(', '', regex=False)
    .str.replace(')', '', regex=False)
)
```

In [159]:

```
df['date_visited'] = df['date_visited'].str.replace('-', '/', regex=False)

# Step 2: Use pd.to_datetime with dayfirst=True and infer_datetime_format=True
df['date_visited'] = pd.to_datetime(df['date_visited'], dayfirst=True, infer_datetime_fo

# Optional: Check if any dates failed to convert (NaT)
invalid_dates = df[df['date_visited'].isna()]
if not invalid_dates.empty:
    print("These dates could not be converted:")
    print(invalid_dates)

print("\nConverted datetime column:")
df['date_visited'].tail()
```

```
These dates could not be converted:
       patient's_name date_visited  age gender followup/new  contact_no  \
16      Rizwana Patel          NaT   30      F    Follow up  9503093418
17             Maheki          NaT   33      F          NaN  9930348235
18       Sushma Saraf          NaT    -      F          NaN  9920289301
19              Mohit          NaT   26      M          New  7039297434
20      Kashaf Shaikh          NaT    -      M    Follow up         NaN
21    Shandaar Pujari          NaT   55      M          New  7715983423
22       Antara Pawar          NaT    -      -    Follow up         NaN
23       Antara Pawar          NaT   55      M    Follow up         NaN
24        Pooja Singh          NaT    -      F    Follow up         NaN
25         Rambha Rai          NaT    -      M    Follow up         NaN
26    Salman Sikander          NaT   28      M    Follow up  9821586280
27       Yogesh Palvi          NaT   32      F    Follow up  9082792092
28           Mrs Geri          NaT  NaN      F          NaN         NaN
29  Meera Vishwakarma          NaT    -      F    Follow up  7977886098
30     Metali Raghani          NaT    -      F          NaN  9967905104
31           Meera V.          NaT    -      F    Follow up  9977886098
32              Safal          NaT    -    NaN          NaN  9867849097
33     Savita Bharati          NaT   21      F          NaN  7498440935
34      Aarti Pardesi          NaT   44      F          NaN         NaN
```

| | | | | | | |
|---|---|---|---|---|---|---|
| 35 | Sardha Ramvarma | NaT | 36 | M | NaN | 9889881212 |
| 36 | Radhika More | NaT | - | F | Follow up | - |
| 37 | Chandrabhaga | NaT | - | M | Follow up | - |
| 38 | Sushma Sharaf | NaT | - | F | Follow up | - |
| 39 | Chaitanya | NaT | - | M | Follow up | - |
| 40 | Kishore Naikwade | NaT | - | M | Follow up | - |
| 41 | Akash Yadav | NaT | 15 | M | New | 8928798601 |
| 42 | Fayeen Khan | NaT | 21 | F | New | 9833476861 |
| 43 | Aditi Pawar | NaT | 23 | F | NaN | 9004226848 |
| 44 | Alia | NaT | NaN | F | Follow up | NaN |
| 45 | Bandu Jagtap | NaT | 71 | M | Follow up | 7738762726 |
| 46 | Netradevi Rathod | NaT | 4 | F | Follow up | 3636980431 |
| 47 | Shashi Raujan | NaT | NaN | M | NaN | NaN |
| 48 | Shiv Patel | NaT | NaN | M | NaN | 9702941999 |
| 49 | Samrudhi Khadare | NaT | NaN | F | NaN | 9769554286 |
| 68 | Nanda Bhorge | NaT | NaN | F | NaN | 7777074901 |

| | reference | dr_name_consultant |
|---|---|---|
| 16 | Follow-up | Dr. Ravi Sangle(Dressing+Stiches) |
| 17 | Injection | - |
| 18 | Dressing | NaN |
| 19 | Dressing + Consult | NaN |
| 20 | - | Dr. Sidra Khot |
| 21 | - | Dr Sagar P. |
| 22 | - | Dr. Nikhil Dagdu |
| 23 | Lab | Dr. Nikhil Dagdu |
| 24 | - | Dr. Nikhil Dagdu |
| 25 | NaN | Dr. Nikhil Dagdu |
| 26 | Baitulmal Foundation (Kalwa) | Dr. Deepak Pawar |
| 27 | NaN | Dr. Nikhil Dagdu |
| 28 | Dr jigar Gori | Dr Bhagyashree Daulatabadkar |
| 29 | Follow-up | Dr. Sidra Khot |
| 30 | Ref O.P Shukla | Dr. Sidra Khot |
| 31 | Follow up | Dr Pooja Sharma |
| 32 | Day Care | Dr Arthi Gadwaikar |
| 33 | Dr Amruta Jog | Dr Sidra Khot |
| 34 | NaN | Dr RMO |
| 35 | NaN | Dr. Mahesh Puri |
| 36 | Follow up | Dr. Nikhil Dagdu |
| 37 | FUP | Dr. Nikhil Dagdu |
| 38 | FUP | Dr Deepak pawar |
| 39 | FUP | Dr Smeeyenei + Dressing |
| 40 | FUP | Dr Duyanesh Sury |
| 41 | Azim | Dr.Duyanesh + IV |
| 42 | Dr Maria Vohra | Dr Sidra Khot |
| 43 | Dr Neha Kolhe | Dr. Sidra Khot |
| 44 | NaN | Dr. Sidra Khot |
| 45 | Followup | Dr Saurabh Chalke |
| 46 | Followup | Dr Abhijeet Sawant |
| 47 | NaN | Dr Dnyanesh Suryavanshi |
| 48 | NaN | Dr Dnyanesh Suryavanshi |
| 49 | NaN | Dr Deven Kuruwa |
| 68 | NaN | Dr Nikhil Dagdu |

| | total_amount | consultant_amount | hospital_amount | ref_g.b | payment_mode |
|---|---|---|---|---|---|
| 16 | 500 | NaN | - | 500 | Cash |
| 17 | 100 | NaN | - | 100 | Cash |
| 18 | 300 | NaN | NaN | 300 | UPI |
| 19 | 500 | NaN | NaN | 500 | UPI |

| | | | | | |
|---|---|---|---|---|---|
| 20 | 500 | 400 | - | 100 | UPI |
| 21 | 300 | 200 | NaN | 100 | Cash |
| 22 | 500 | NaN | NaN | - | UPI |
| 23 | 500 | NaN | NaN | NaN | UPI |
| 24 | 500 | 400 | NaN | 100 | Cash |
| 25 | 500 | 400 | NaN | 100 | UPI |
| 26 | 600 | 400 | - | - | UPI |
| 27 | 500 | 400 | - | 100 | UPI |
| 28 | 500 | 500 | NaN | NaN | NaN |
| 29 | 500 | 400 | - | 100 | UPI |
| 30 | 1000 | 400 | 200 | 200 | Cash |
| 31 | 600 | 500 | - | 100 | UPI |
| 32 | 1500 | 1000 | NaN | 500 | Cash |
| 33 | NaN | NaN | NaN | - | NaN |
| 34 | 200 | - | NaN | 200 | Cash |
| 35 | No charges | - | NaN | NaN | NaN |
| 36 | 500 | 400 | NaN | 100 | Cash |
| 37 | 500 | 400 | NaN | 100 | Cash |
| 38 | 500 | 400 | NaN | 100 | UPI |
| 39 | 1000 | 700 | NaN | 300 | UPI |
| 40 | 600 | 500 | NaN | 100 | UPI |
| 41 | 1500 | 700 | NaN | NaN | UPI |
| 42 | 6000 | 6000 | NaN | 100 | Cash |
| 43 | 15000 | - | NaN | - | Cash |
| 44 | 500 | 400 | NaN | 100 | Cash |
| 45 | 1200 | 1000 | NaN | 200 | Cash |
| 46 | 1200 | 1000 | NaN | NaN | UPI |
| 47 | 2100 | 500 | NaN | NaN | Card |
| 48 | 2500 | 1500 | NaN | 1000 | UPI |
| 49 | 500 | 400 | NaN | 100 | Cash |
| 68 | 500 | 400 | NaN | NaN | Cash |

| | unnamed:_13 | unnamed:_14 |
|---|---|---|
| 16 | NaN | NaN |
| 17 | NaN | NaN |
| 18 | NaN | NaN |
| 19 | NaN | NaN |
| 20 | NaN | NaN |
| 21 | NaN | NaN |
| 22 | NaN | NaN |
| 23 | NaN | NaN |
| 24 | NaN | NaN |
| 25 | NaN | NaN |
| 26 | NaN | NaN |
| 27 | NaN | NaN |
| 28 | NaN | NaN |
| 29 | NaN | NaN |
| 30 | NaN | NaN |
| 31 | NaN | NaN |
| 32 | NaN | NaN |
| 33 | NaN | NaN |
| 34 | NaN | NaN |
| 35 | NaN | NaN |
| 36 | NaN | NaN |
| 37 | NaN | NaN |
| 38 | NaN | NaN |
| 39 | NaN | NaN |
| 40 | NaN | NaN |
| 41 | NaN | NaN |

```
42         NaN         NaN
43         NaN         NaN
44         NaN         NaN
45         NaN         NaN
46         NaN         NaN
47         NaN         NaN
48         NaN         NaN
49         NaN         NaN
68         NaN         NaN

Converted datetime column:
/var/folders/1w/677cvfp14kd9g4hm22pmcm9c0000gn/T/ipykernel_6086/1122392345.py:4: UserWar
ning: The argument 'infer_datetime_format' is deprecated and will be removed in a future
version. A strict version of it is now the default, see https://pandas.pydata.org/pdeps/
0004-consistent-to-datetime-parsing.html. You can safely remove this argument.
  df['date_visited'] = pd.to_datetime(df['date_visited'], dayfirst=True, infer_datetime_
format=True, errors='coerce')
Out[159]:
1402   2025-04-30
1403   2025-04-30
1404   2025-04-30
1405   2025-04-30
1406   2025-04-30
Name: date_visited, dtype: datetime64[ns]
```

In [160]:

```
df.drop(columns=['unnamed:_13', 'unnamed:_14'], inplace=True, errors='ignore')
```

In [161]:

```
df.replace('-', np.nan, inplace=True)
```

In [162]:

```
# D. Convert data types
for col in ['age', 'total_amount', 'consultant_amount', 'hospital_amount', 'ref_g.b']:
    df[col] = pd.to_numeric(df[col], errors='coerce')
```

In [163]:

```
# Preview cleaned data
print(df.info())
print(df.tail(30))
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1407 entries, 0 to 1406
Data columns (total 13 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   patient's_name      1407 non-null   object
 1   date_visited        1372 non-null   datetime64[ns]
 2   age                 734 non-null    float64
 3   gender              1376 non-null   object
 4   followup/new        680 non-null    object
 5   contact_no          880 non-null    object
 6   reference           404 non-null    object
 7   dr_name_consultant  1381 non-null   object
 8   total_amount        1249 non-null   float64
 9   consultant_amount   1047 non-null   float64
 10  hospital_amount     512 non-null    float64
 11  ref_g.b             332 non-null    float64
 12  payment_mode        1175 non-null   object
```

```
dtypes: datetime64[ns](1), float64(5), object(7)
memory usage: 143.0+ KB
None
       patient's_name date_visited   age gender followup/new  contact_no  \
1377             Neha   2025-04-28  37.0      f          New  9769903701
1378    Manbalai Yadav   2025-04-28   NaN      f          NaN         NaN
1379      Sheela Gupta   2025-04-28   NaN      f          NaN         NaN
1380             Kumar   2025-04-28  47.0      n          New  8660567645
1381  Chaitanya Chunge   2025-04-28   NaN      m          NaN  9322149329
1382  Vatsala Peshmane   2025-04-28  63.0      f          New  9320506619
1383    Vrinda Jhalani   2025-04-28  35.0      f          New  9920773768
1384   Roshni Shigvain   2025-04-28  30.0      f          New   836941566
1385       Jhalak Jain   2025-04-29   NaN      f          NaN  9920566657
1386        Kajal Gade   2025-04-29  23.0      f          New         NaN
1387       Usha Shinde   2025-04-29  59.0      f          New  9870289379
1388       Deepak Naik   2025-04-29   NaN      m          NaN         NaN
1389   Suchita Ghadshi   2025-04-29   NaN      f          NaN         NaN
1390     Shaguna yadav   2025-04-29   NaN      f          NaN         NaN
1391            Mangal   2025-04-29   NaN      m          NaN         NaN
1392       Nisha Halde   2025-04-29  44.0      f          New         NaN
1393       Kunal Kadam   2025-04-29   NaN      m          NaN         NaN
1394            Govind   2025-04-29   NaN      m          NaN         NaN
1395     Ruksana Patha   2025-04-29   NaN      f          NaN         NaN
1396       Aslam Memon   2025-04-29  57.0      m          New  9152159165
1397   Vidhisha Bhogal   2025-04-29   NaN      f          NaN         NaN
1398             Sujog   2025-04-29  40.0      m          New  9930961690
1399     Yogita Sanant   2025-04-29  24.0      f          New  8433799072
1400     Yogita Sanant   2025-04-30   NaN      f          NaN         NaN
1401    Poonam Jaiswal   2025-04-30   NaN      f          NaN         NaN
1402       Sayed Firoz   2025-04-30  31.0      m          New         NaN
1403     Dinkar Dongre   2025-04-30  33.0      m          New         NaN
1404            Najeeb   2025-04-30   NaN      m          New         NaN
1405      Aahang Ambre   2025-04-30   NaN      f          NaN  9930270422
1406       Kishor Kini   2025-04-30  60.0      m          NaN  9869241712

                  reference               dr_name_consultant  total_amount  \
1377                    NaN                   Dr.Sachin Wani        1200.0
1378                    NaN                  Dr.Nikhil Dagdu           NaN
1379                    NaN                   Dr.Sachin Wani           NaN
1380                    NaN                   Dr.Sachin Wani        1200.0
1381                    NaN             Dr.Chetashree Gawale         500.0
1382         Dr.Sunil Gabale                   Dr.Sachin Wani        1500.0
1383                    NaN                   Dr.Sachin Wani        1200.0
1384           Dr.Neha kolne                    Dr.Sidra Khot         800.0
1385                    NaN              Dr.Arohi Tasgaonkar         600.0
1386                    NaN                         Day Care        3000.0
1387          Dr.ravi Shinde          Dr.Dyanesh Suryavanshi        1000.0
1388                    NaN                             NaN           NaN
1389                    NaN                  Dr.Nikhil Dagdu         500.0
1390                    NaN  Dr.Dyanesh Suryavanshi+dressing        1000.0
1391                    NaN           Dr.Dyanesh Suryavanshi         300.0
1392                  Trust           Dr.Dyanesh Suryavanshi         250.0
1393                    NaN           Dr.Dyanesh Suryavanshi         300.0
1394                    NaN                  Dr.Nikhil Dagdu         500.0
1395                  Trust           Dr.Dyanesh Suryavanshi         300.0
1396        Dr.Asgar Mukadam                Dr.Saurabh Chalke        1500.0
1397                    NaN                    Dr.Sidra Khot         500.0
1398                    NaN                   Dr.Sachin Wani        1200.0
1399        Dr.Ankita Tiwari                    Dr.Sidra Khot         500.0
```

```
1400                    NaN                   Dr.Sidra Khot           NaN
1401                    NaN                   Dr.Sidra Khot           NaN
1402                    NaN                  Dr.Sachin Wani        1000.0
1403                    NaN                        Day Care        1000.0
1404              Azim Sir                  Dr.Deepak Pawar         500.0
1405              Azim Sir                 Dr.Yogesh Tiwari        1100.0
1406  Dr.Dyanesh Suryavanshi                 Dr.Dinesh Pimple        2000.0

      consultant_amount  hospital_amount  ref_g.b payment_mode
1377             1000.0            200.0      NaN         Cash
1378                NaN              NaN      NaN          NaN
1379                NaN              NaN      NaN          NaN
1380             1000.0            200.0      NaN         Cash
1381              400.0            100.0      NaN         Gpay
1382             1000.0            300.0    200.0         Gpay
1383             1000.0            200.0      NaN         Cash
1384              400.0            200.0    200.0         Gpay
1385              500.0            100.0      NaN         Gpay
1386                NaN           3000.0      NaN         Gpay
1387              600.0            200.0    200.0         Gpay
1388                NaN              NaN      NaN         Gpay
1389              400.0            100.0      NaN         Cash
1390              700.0            300.0      NaN         Gpay
1391              300.0              NaN      NaN         Gpay
1392              250.0              NaN      NaN         Gpay
1393              300.0              NaN      NaN         Cash
1394              400.0            100.0      NaN         Cash
1395              300.0              NaN      NaN         cash
1396             1000.0            300.0    200.0         Gpay
1397              400.0            100.0      NaN         Gpay
1398             1000.0            200.0      NaN         Cash
1399              400.0            100.0      NaN         Gpay
1400                NaN              NaN      NaN          NaN
1401                NaN              NaN      NaN          NaN
1402              800.0            200.0      NaN         Cash
1403                NaN           1000.0      NaN         Gpay
1404              400.0            100.0      NaN         Cash
1405             1000.0            100.0      NaN         Gpay
1406             1500.0            500.0      NaN         Cash
```

In [164]:

```
# Standardize column name if not already done
df.rename(columns={'followup/new': 'followup_new'}, inplace=True)
```

In [165]:

```
df['followup_new'] = df['followup_new'].fillna('New')
```

In [166]:

```
df['followup_new'] = df['followup_new'].str.strip().str.capitalize()
```

In [167]:

```
df.rename(columns={'gender': 'gender'}, inplace=True)  # just to be consistent
```

In [168]:

```
df['gender'] = df['gender'].fillna('Unknown')
```

In [169]:

```
df['gender'] = df['gender'].str.strip().str.lower()
```

In [170]:

```
gender_map = {
    'm': 'Male',
    'male': 'Male',
    'f': 'Female',
    'female': 'Female',
    'other': 'Other',
    'unknown': 'Unknown',
    'not specified': 'Unknown',
    '': 'Unknown',
    np.nan: 'Unknown'
}

df['gender'] = df['gender'].map(gender_map).fillna('Unknown')
```

In [171]:

```
df['age'] = df['age'].replace(['-', '', ' '], np.nan)
```

In [172]:

```
# Convert to numeric (integer)
df['age'] = pd.to_numeric(df['age'], errors='coerce')

# Optional: Check basic stats
print(df['age'].describe())


median_age = df['age'].median()
df['age'] = df['age'].fillna(median_age)
```

```
count    734.000000
mean      40.856948
std       15.851028
min        1.000000
25%       30.000000
50%       38.000000
75%       52.000000
max       92.000000
Name: age, dtype: float64
```

In [173]:

```
# Fill missing payment modes with 'Unknown'
df['payment_mode'] = df['payment_mode'].fillna('Unknown')

# Standardize values (strip spaces and capitalize)
df['payment_mode'] = df['payment_mode'].str.strip().str.capitalize()

# Optional: Check unique payment modes to spot inconsistencies
print(df['payment_mode'].value_counts())
```

```
payment_mode
Cash          585
Upi           357
Unknown       232
Gpay          215
Card           16
Cash+ upi       1
No charges      1
Name: count, dtype: int64
```

In [174]:

```
amount_cols = ['total_amount', 'consultant_amount', 'hospital_amount']

for col in amount_cols:
    # Replace '-' or empty with NaN
    df[col] = df[col].replace(['-', '', ' '], np.nan)

    # Convert to numeric (float)
    df[col] = pd.to_numeric(df[col], errors='coerce')

# Optional: Check summary stats
print(df[amount_cols].describe())
```

```
       total_amount  consultant_amount  hospital_amount
count   1249.000000        1047.000000       512.000000
mean    1009.753403         630.888252       315.617188
std     1482.227976         468.000582       468.847404
min        0.000000           0.000000        50.000000
25%      500.000000         400.000000       100.000000
50%      700.000000         500.000000       200.000000
75%     1000.000000         800.000000       262.500000
max    30000.000000        6000.000000      3400.000000
```

In [175]:

```
print(df[['total_amount', 'consultant_amount', 'hospital_amount']].describe())
```

```
       total_amount  consultant_amount  hospital_amount
count   1249.000000        1047.000000       512.000000
mean    1009.753403         630.888252       315.617188
std     1482.227976         468.000582       468.847404
min        0.000000           0.000000        50.000000
25%      500.000000         400.000000       100.000000
50%      700.000000         500.000000       200.000000
75%     1000.000000         800.000000       262.500000
max    30000.000000        6000.000000      3400.000000
```

In [176]:

```
print(df['gender'].value_counts())
```

```
gender
Female     817
Male       558
Unknown     32
Name: count, dtype: int64
```

In [177]:

```
print(df['followup_new'].value_counts())
```

```
followup_new
New           1086
Follow up      311
Followup         5
Folllow up       4
Neew             1
Name: count, dtype: int64
```

In [178]:

```
print(df['dr_name_consultant'].value_counts().head(15))  # top 10 doctors by patient cou
```

```
dr_name_consultant
Dr Sidra Khot               88
```

```
Dr.Sidra Khot               76
DR. SIDRA KHOT              72
Dr.Dyanesh Suryavanshi      69
DR. DNYANESH SURYAVANSHI    66
DR. SACHIN WANI             55
Dressing                    50
DR. NIKHIL DAGDU            50
Dr.Deepak Pawar             46
Dr Deepak Pawar             45
Dr Nikhil Dagdu             41
Dr.Sachin Wani              39
Dr Dnyanesh Suryavanshi     36
Dr Sachin Wani              27
Dr.Nikhil D                 25
Name: count, dtype: int64
```

In [179]:

```
revenue_by_payment = df.groupby('payment_mode')['total_amount'].sum()
print(revenue_by_payment)
```

```
payment_mode
Card           38876.0
Cash          512626.0
Cash+ upi       1100.0
Gpay          216550.0
No charges         0.0
Unknown        41950.0
Upi           450080.0
Name: total_amount, dtype: float64
```

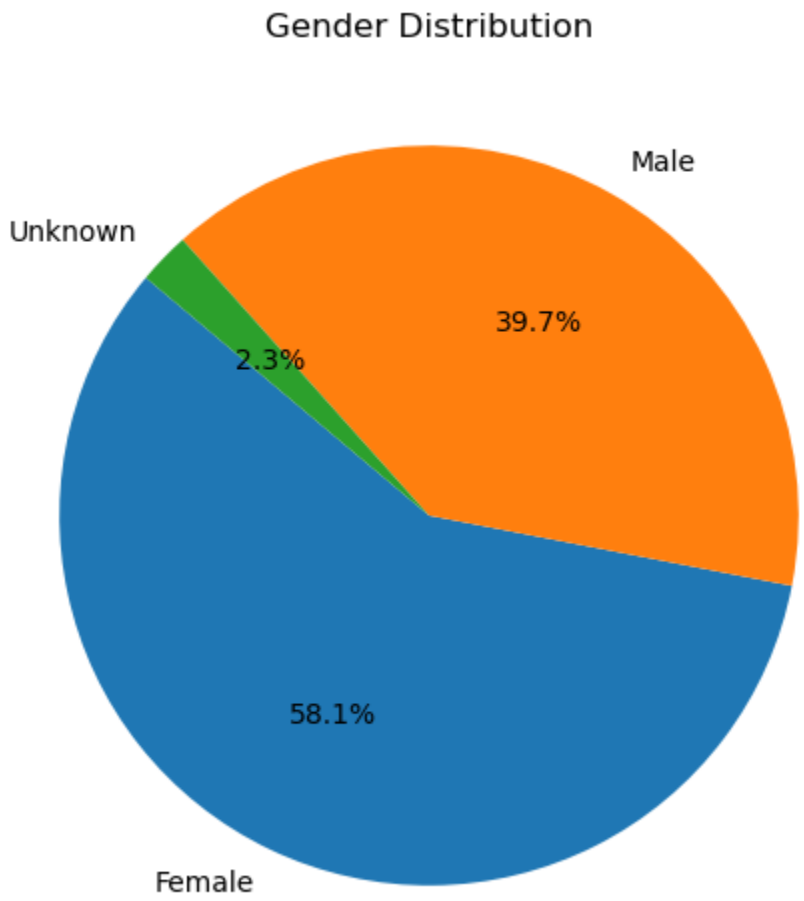
In [180]:

```
avg_consultant_per_doc = df.groupby('dr_name_consultant')['consultant_amount'].mean().so
print(avg_consultant_per_doc.head(10))
```

```
dr_name_consultant
Dr Shruti Chedha                  2000.0
Dr.Dyanesh Suryavanshi+daycare    2000.0
DR. AJIT GHADGE                   1600.0
Dr Abhijeet Sawat                 1500.0
Dr  Pradny Jain                   1500.0
Dr.Aniket Wankhede                1500.0
DR. DIPESH PIMPLE                 1500.0
Dr. Abhijeet Sawant               1500.0
Dr.Dinesh Pimple                  1500.0
Dr.Dipesh Pimple                  1500.0
Name: consultant_amount, dtype: float64
```

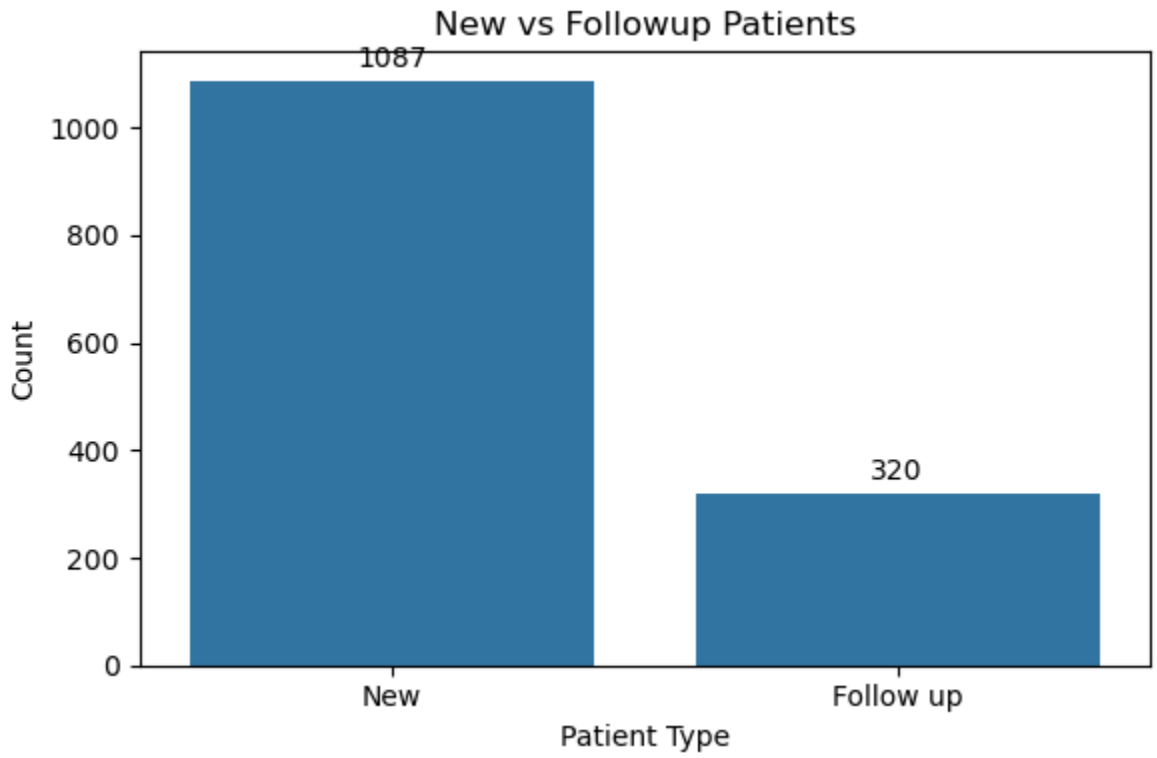
In [181]:

```
import matplotlib.pyplot as plt

gender_counts = df['gender'].value_counts()
plt.figure(figsize=(6,6))
plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%', startangle=140)
plt.title('Gender Distribution')
plt.show()
```




In [182]:

```
import seaborn as sns

df['followup_new'] = df['followup_new'].apply(lambda x: 'New' if str(x).startswith('N')
df['followup_new'] = df['followup_new'].apply(lambda x: 'Follow up' if str(x).startswith

# Plot
plt.figure(figsize=(6, 4))
ax = sns.countplot(data=df, x='followup_new', order=df['followup_new'].value_counts().in

# Add bar labels
for container in ax.containers:
    ax.bar_label(container, padding=3)

# Labels and title
plt.title('New vs Followup Patients')
plt.xlabel('Patient Type')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```




In [183]:

```
standard_names = ['Dr. Sidra Khot', 'Dr. Sagar Patil', 'Dr. Nikhil Dagdu','Dr. Deepak Pa
                  'Dr. Dnyanesh Suryavanshi','Dr. Amruta Pawar','Dr. Pooja Sharma','Dr. D
                  'Dr. Akshay Pednekar','Dr Vinay Pawar', 'Dr. Tejas Ghude','Dr. Sachin G
                  'Dr. Rakesh Patel','Dr. Bhagyashree Dualatabadkar','Dr. Ajit Ghadge','D
                  'Dr. Kaustubh Mehta','Dr. Chetashree Gavate','Dr. Ravi Sangle','Dr. Sid
```

In [184]:

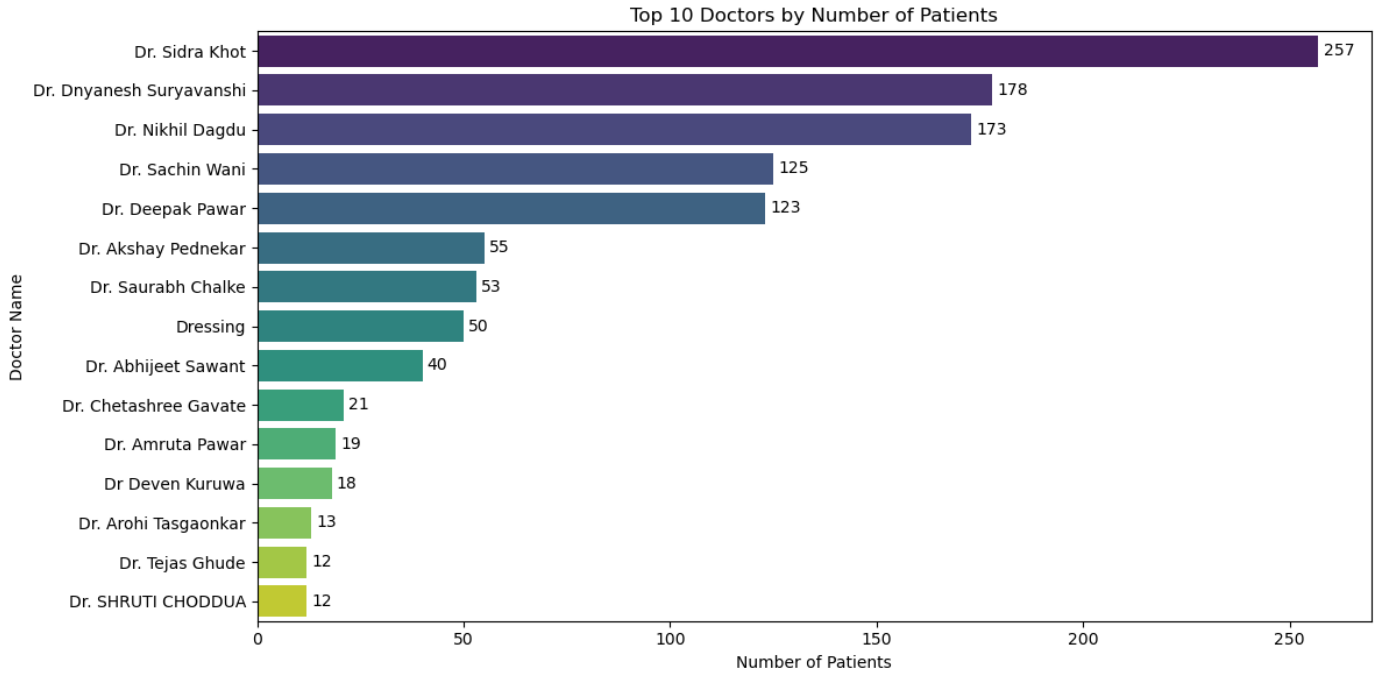
```
from difflib import SequenceMatcher
def normalize(name):
    if pd.isnull(name):  # Handle NaN or missing entries
        return set()
    name = str(name).lower().replace('.', ' ').strip()
    tokens = name.split()
    return set(tokens)

# Matching function (word match ≥ threshold)
def get_best_match(name, standard_list, threshold=0.6):
    name_tokens = normalize(name)
    for std in standard_list:
        std_tokens = normalize(std)
        match_score = len(name_tokens & std_tokens) / max(len(std_tokens), len(name_toke
        if match_score >= threshold:
            return std
    return name  # return original if no good match

# Apply matching and standardization
df['dr_clean'] = df['dr_name_consultant'].apply(lambda x: get_best_match(x, standard_nam

# Result
print(df['dr_clean'].value_counts().head(20))
```

```
dr_clean
Dr. Sidra Khot              257
Dr. Dnyanesh Suryavanshi    178
```

```
Dr. Nikhil Dagdu            173
Dr. Sachin Wani             125
Dr. Deepak Pawar            123
Dr. Akshay Pednekar          55
Dr. Saurabh Chalke           53
Dressing                     50
Dr. Abhijeet Sawant          40
Dr. Chetashree Gavate        21
Dr. Amruta Pawar             19
Dr Deven Kuruwa              18
Dr. Arohi Tasgaonkar         13
Dr. Tejas Ghude              12
Dr. SHRUTI CHODDUA           12
Day Care                     10
Dr. Pooja Sharma              9
Dr Aarthi                     9
Dr.Gaurav Dhanawat            8
DR. RMO                       7
Name: count, dtype: int64
```

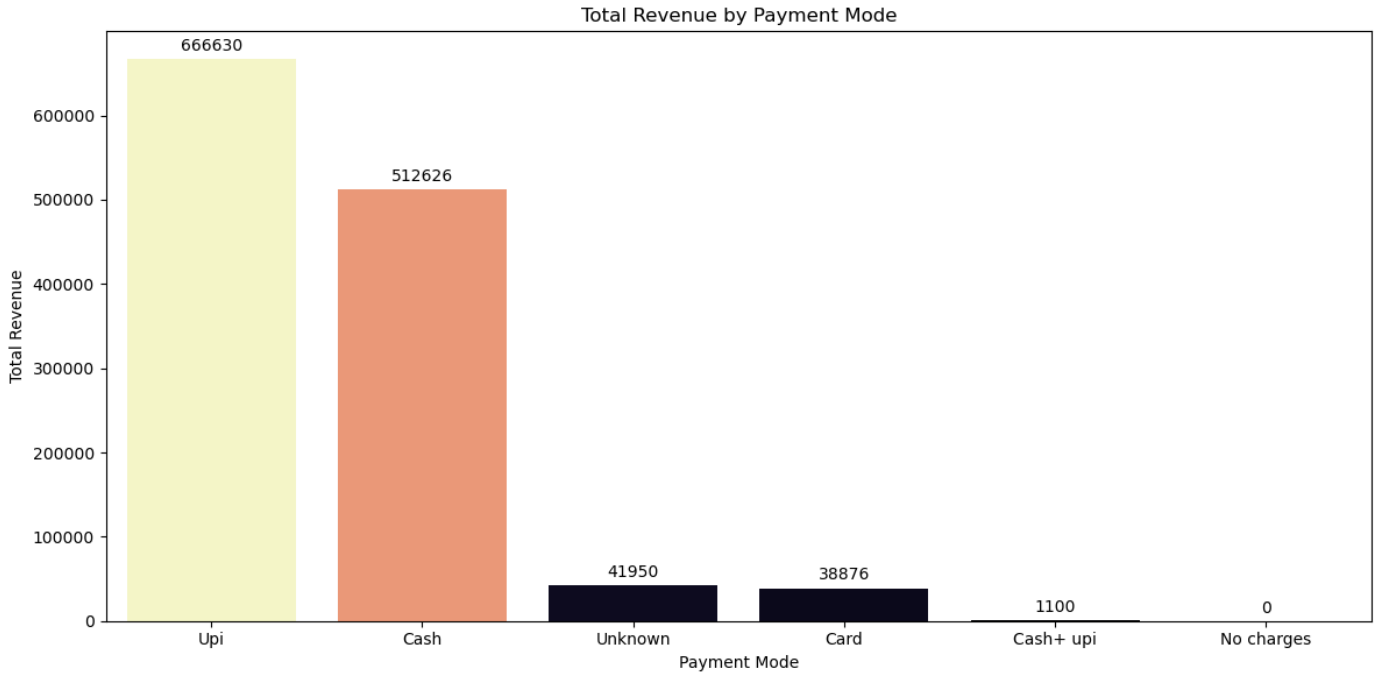
In [185]:

```python
top_doctors = df['dr_clean'].value_counts().head(15)
plt.figure(figsize=(12, 6))
ax = sns.barplot(
    x=top_doctors.values,
    y=top_doctors.index,
    hue=top_doctors.index,
    palette='viridis',
    legend=False
)

# Ensure labels are added to every bar
for container in ax.containers:
    ax.bar_label(container, label_type='edge', padding=3)

# Titles and labels
plt.title('Top 10 Doctors by Number of Patients')
plt.xlabel('Number of Patients')
plt.ylabel('Doctor Name')

# Improve spacing
plt.tight_layout()
plt.show()
```



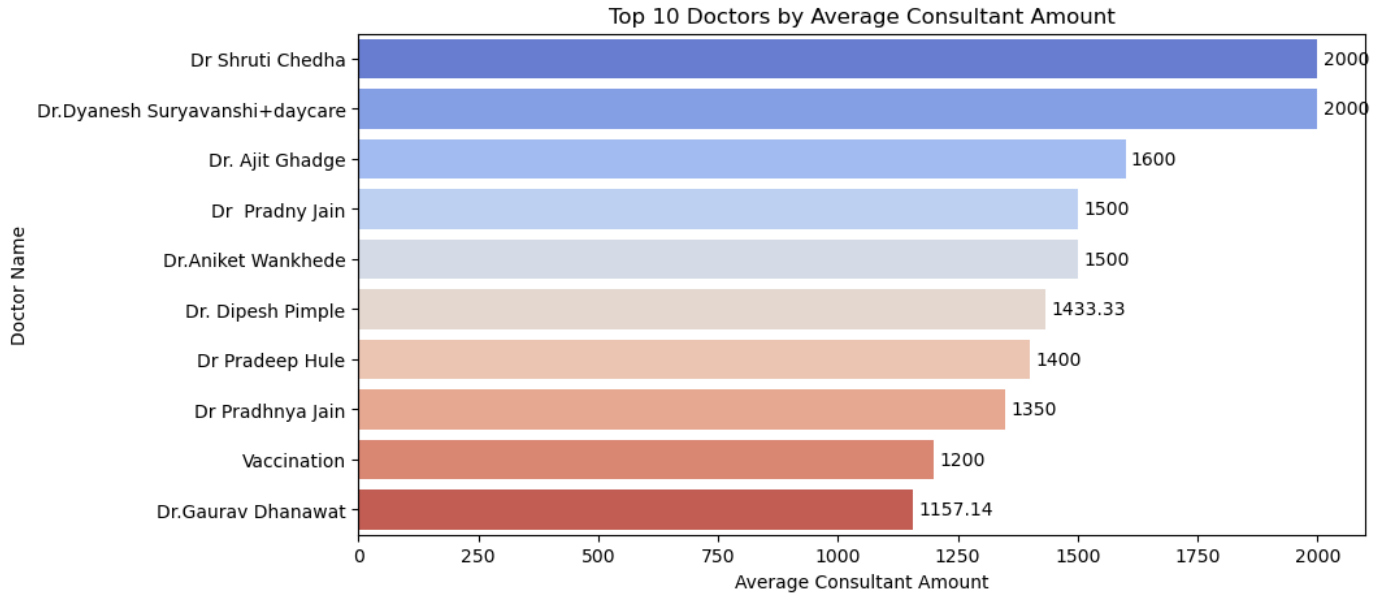

In [186]:

```
df['payment_mode'] = df['payment_mode'].replace({'Gpay':'Upi'})
revenue_by_payment = df.groupby('payment_mode')['total_amount'].sum().sort_values(ascend
plt.figure(figsize=(12, 6))
# sns.barplot(x=revenue_by_payment.index, y=revenue_by_payment.values, palette='magma')
ax = sns.barplot(
    x=revenue_by_payment.index,
    y=revenue_by_payment.values,
    hue=revenue_by_payment.values,
    palette='magma',
    legend=False
)

# Ensure labels are added to every bar
for container in ax.containers:
    ax.bar_label(container, label_type='edge', padding=3)
plt.title('Total Revenue by Payment Mode')
plt.xlabel('Payment Mode')
plt.ylabel('Total Revenue')
plt.tight_layout()

plt.show()
```



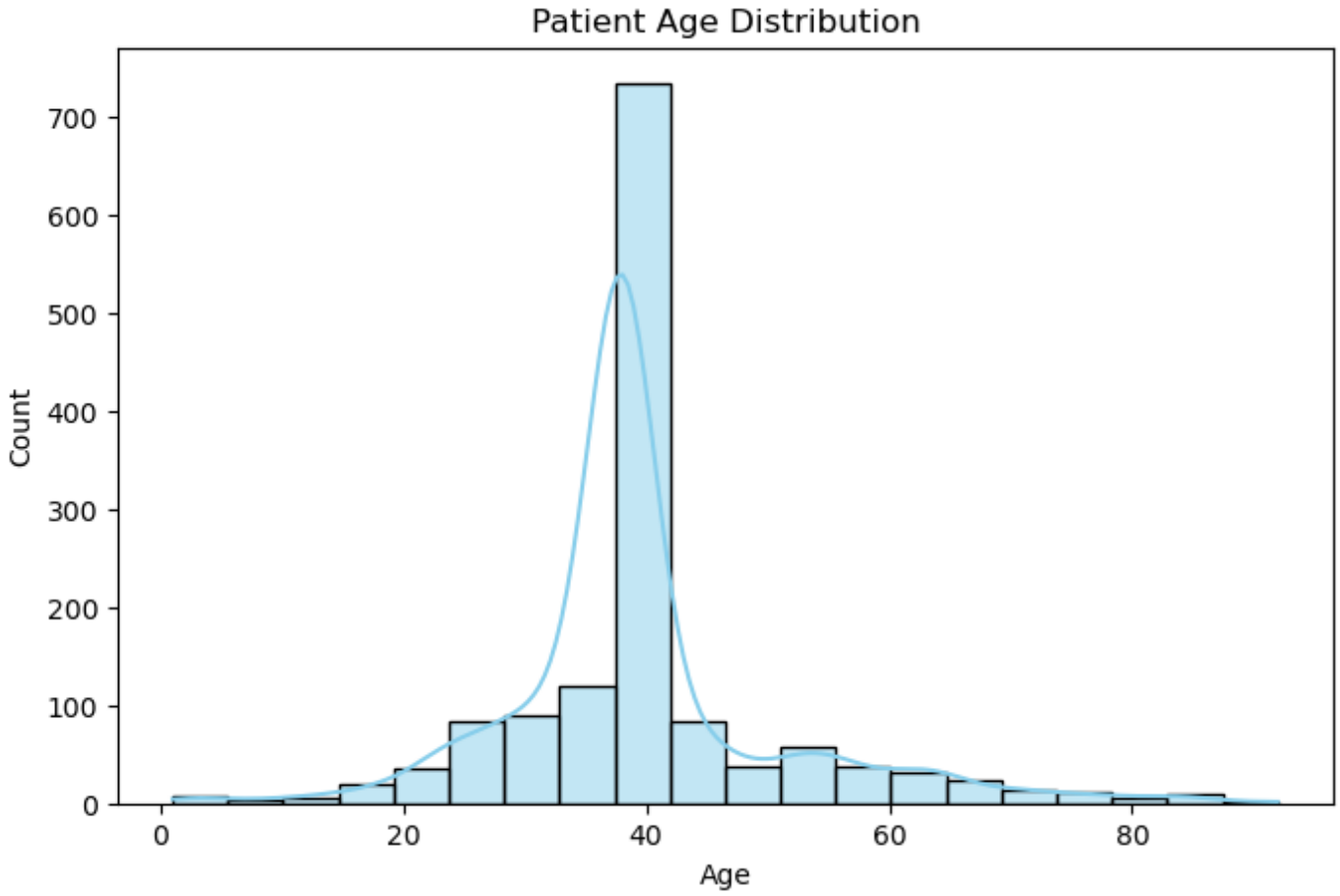

In [187]:

```
avg_consultant_per_doc = df.groupby('dr_clean')['consultant_amount'].mean().sort_values(

plt.figure(figsize=(10,5))
ax = sns.barplot(
    x=avg_consultant_per_doc.values,
    y=avg_consultant_per_doc.index,
    hue=avg_consultant_per_doc.index,
    palette='coolwarm',
    legend=False
)

# Ensure labels are added to every bar
for container in ax.containers:
    ax.bar_label(container, label_type='edge', padding=3)
plt.title('Top 10 Doctors by Average Consultant Amount')
plt.xlabel('Average Consultant Amount')
plt.ylabel('Doctor Name')
plt.show()
```



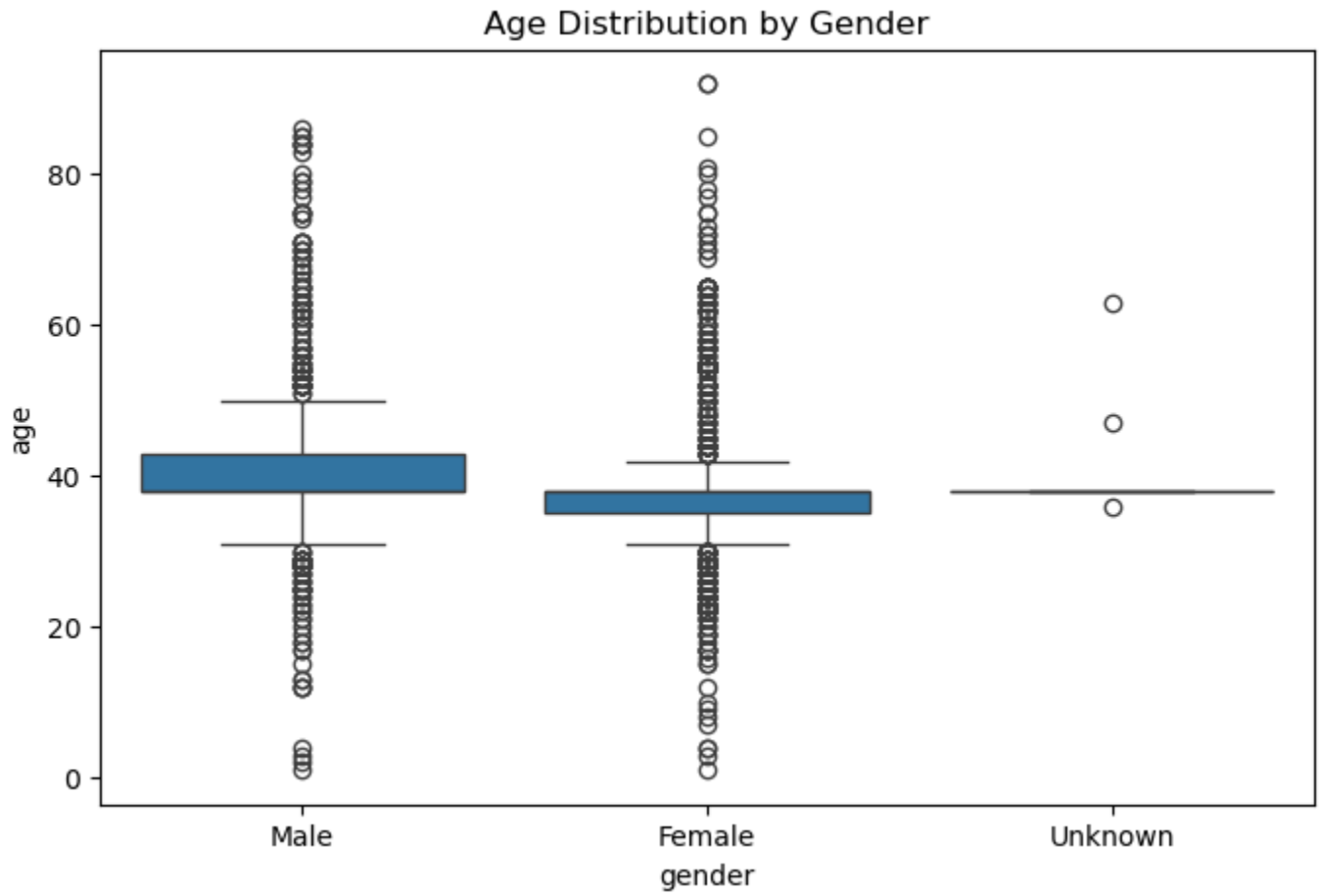
In [188]:

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8,5))
sns.histplot(df['age'].dropna(), bins=20, kde=True, color='skyblue')
plt.title('Patient Age Distribution')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```




In [189]:

```
plt.figure(figsize=(8,5))
sns.boxplot(data=df, x='gender', y='age')
plt.title('Age Distribution by Gender')
plt.show()
```




In [190]:

```
revenue_per_doctor = df.groupby('dr_name_consultant').agg({
    'total_amount': 'sum',
    'consultant_amount': 'sum',
    'hospital_amount': 'sum'
}).sort_values('total_amount', ascending=False).head(10)
```

In [191]:

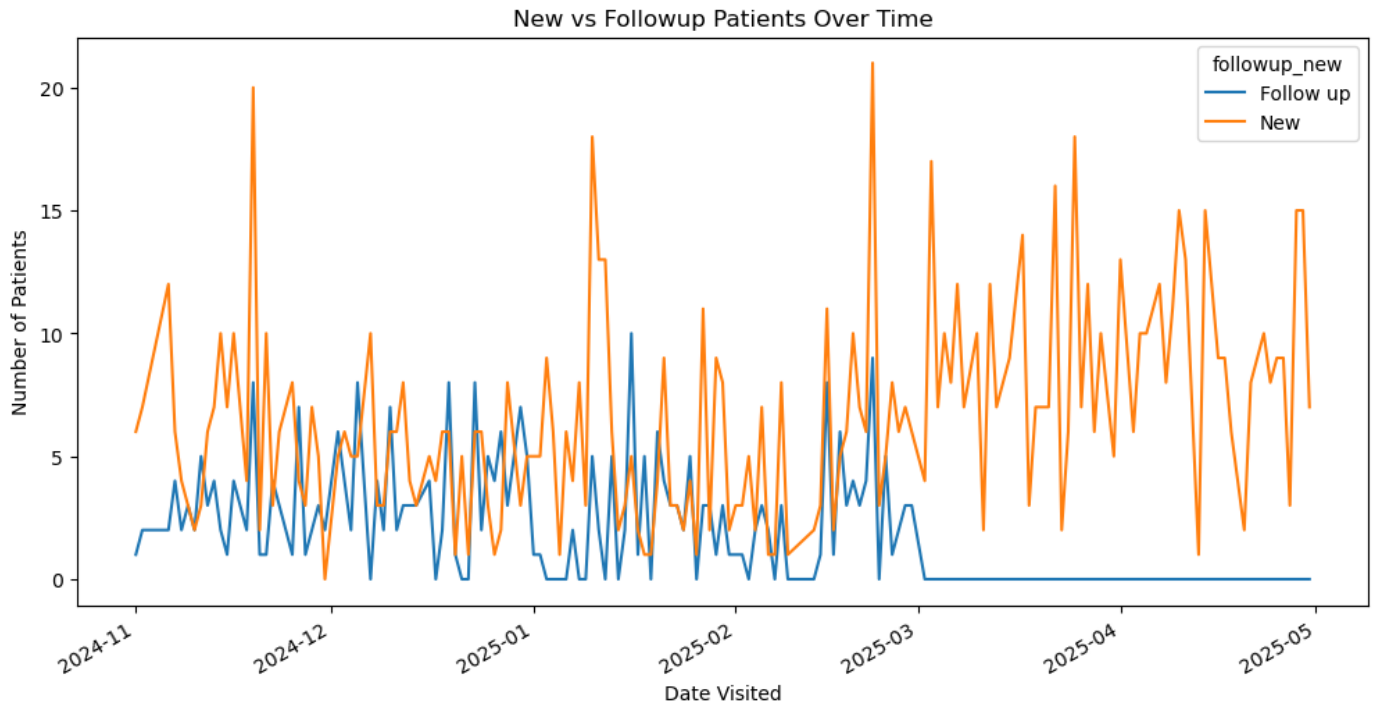
```
print(revenue_per_doctor)
```

```
                         total_amount  consultant_amount  hospital_amount
dr_name_consultant
Dr Sidra Khot                101450.0            51450.0           3400.0
DR. SIDRA KHOT                84500.0            40300.0           5000.0
Dr.Sidra Khot                 65100.0            41250.0          17650.0
Dr.Dyanesh Suryavanshi        64750.0            38200.0          20250.0
Dr Deepak Pawar               56660.0            20700.0           1600.0
DR. SACHIN WANI               55000.0            42500.0           2300.0
Dr.Sachin Wani                41300.0            31500.0           7900.0
DR. DNYANESH SURYAVANSHI      31900.0            28050.0           2200.0
DR. S.CHEDDVA                 30000.0                0.0              0.0
Dr.Deepak Pawar               29100.0            25500.0           7200.0
```

In [192]:

```
df['date_visited'] = pd.to_datetime(df['date_visited'], errors='coerce')

# Count of New vs Followup patients per day
followup_trends = df.groupby(['date_visited', 'followup_new']).size().unstack(fill_value

followup_trends.plot(kind='line', figsize=(12,6))
plt.title('New vs Followup Patients Over Time')
```

```
plt.xlabel('Date Visited')
plt.ylabel('Number of Patients')
plt.show()
```




In [193]:

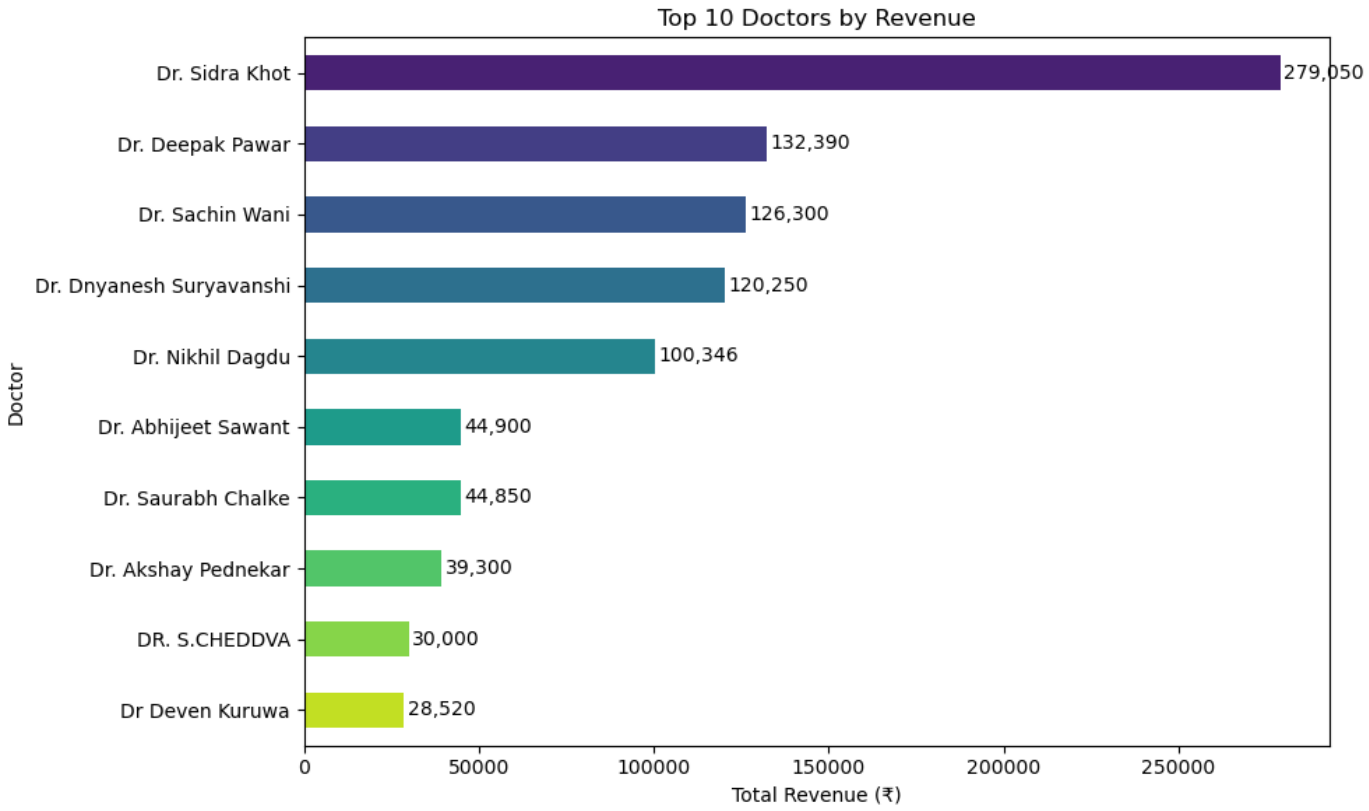
```
revenue_by_doctor = df.groupby('dr_clean')['total_amount'].sum().sort_values(ascending=F
print(revenue_by_doctor.head(10))
```

```
dr_clean
Dr. Sidra Khot              279050.0
Dr. Deepak Pawar            132390.0
Dr. Sachin Wani             126300.0
Dr. Dnyanesh Suryavanshi    120250.0
Dr. Nikhil Dagdu            100346.0
Dr. Abhijeet Sawant          44900.0
Dr. Saurabh Chalke           44850.0
Dr. Akshay Pednekar          39300.0
DR. S.CHEDDVA                30000.0
Dr Deven Kuruwa              28520.0
Name: total_amount, dtype: float64
```

In [194]:

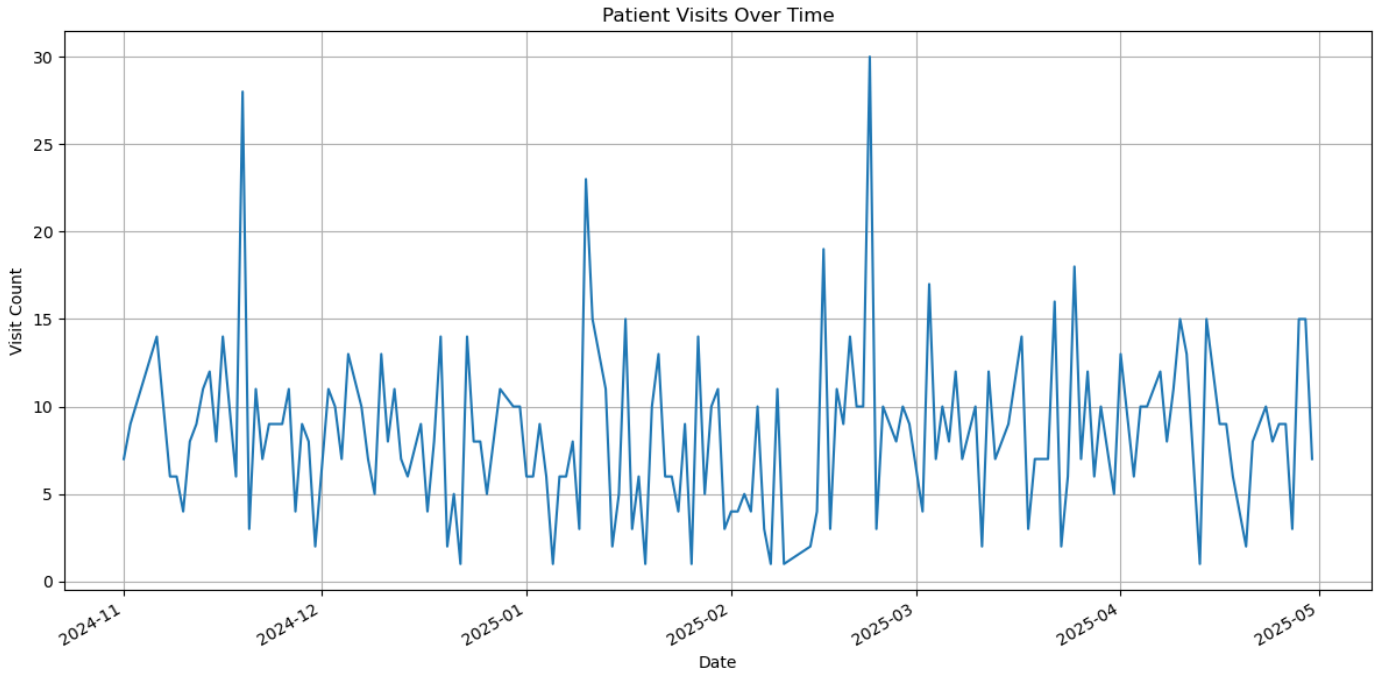
```
# Select top 10 revenue-generating doctors
top_revenue_doctors = revenue_by_doctor.head(10)

# Set a color palette (sorted to match values)
colors = sns.color_palette("viridis", len(top_revenue_doctors))

# Create plot
plt.figure(figsize=(10, 6))
ax = top_revenue_doctors.plot(kind='barh', color=colors)

# Add labels to each bar
for i, v in enumerate(top_revenue_doctors.values):
    ax.text(v + 1000, i, f"{int(v):,}", va='center', fontsize=10)

# Axis labels and title
plt.xlabel('Total Revenue (₹)')
```

```
plt.ylabel('Doctor')
plt.title('Top 10 Doctors by Revenue')

# Invert y-axis so highest revenue is at the top
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()
```



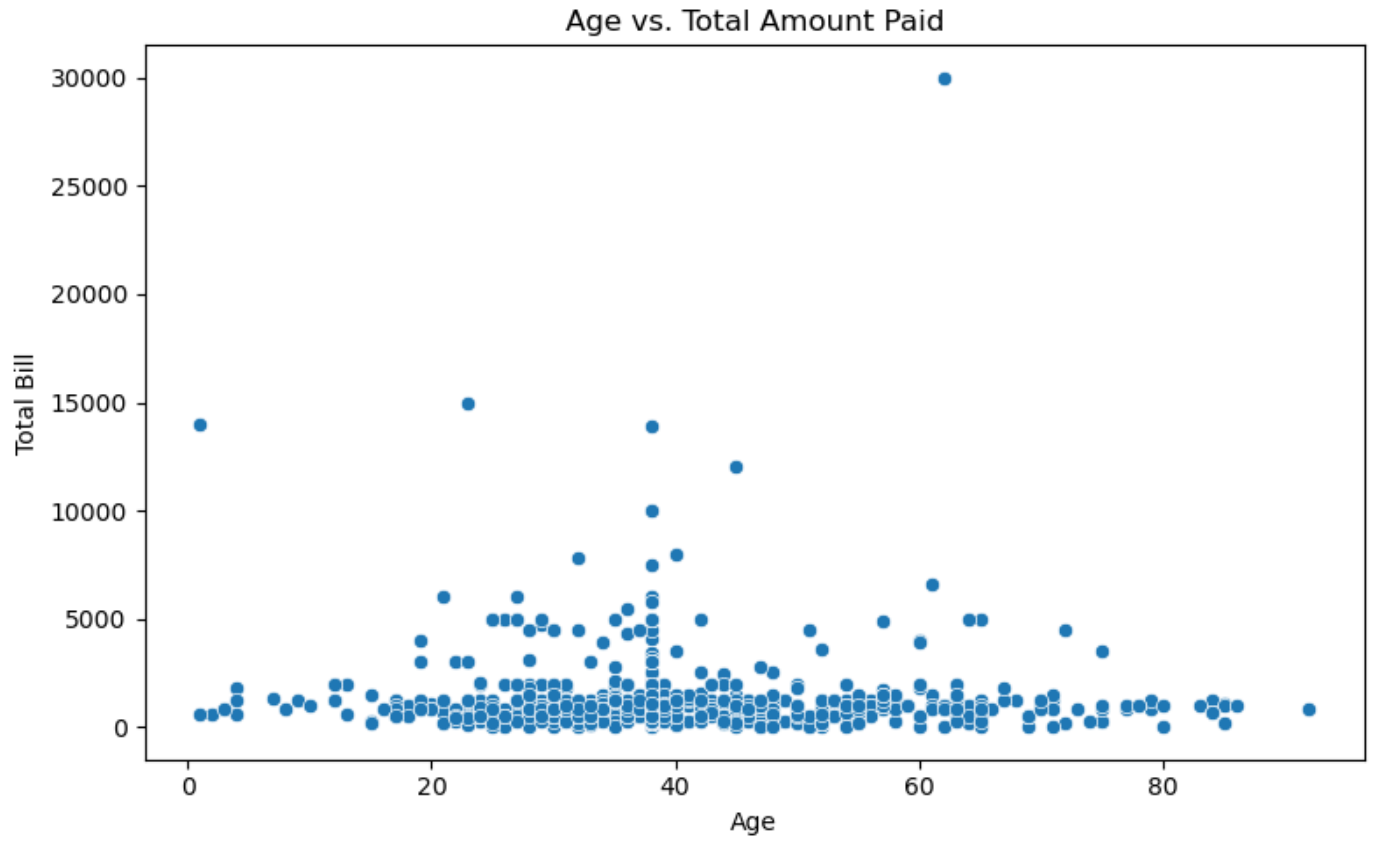

In [197]:

```
df['date_visited'] = pd.to_datetime(df['date_visited'], errors='coerce')
daily_visits = df['date_visited'].value_counts().sort_index()

plt.figure(figsize=(12,6))
daily_visits.plot(kind='line')
plt.title('Patient Visits Over Time')
plt.xlabel('Date')
plt.ylabel('Visit Count')
plt.grid(True)
plt.tight_layout()
plt.show()
```




In [195]:

```
import seaborn as sns

plt.figure(figsize=(8,5))
sns.scatterplot(data=df, x='age', y='total_amount')
plt.title('Age vs. Total Amount Paid')
plt.xlabel('Age')
plt.ylabel('Total Bill')
plt.tight_layout()
plt.show()
```



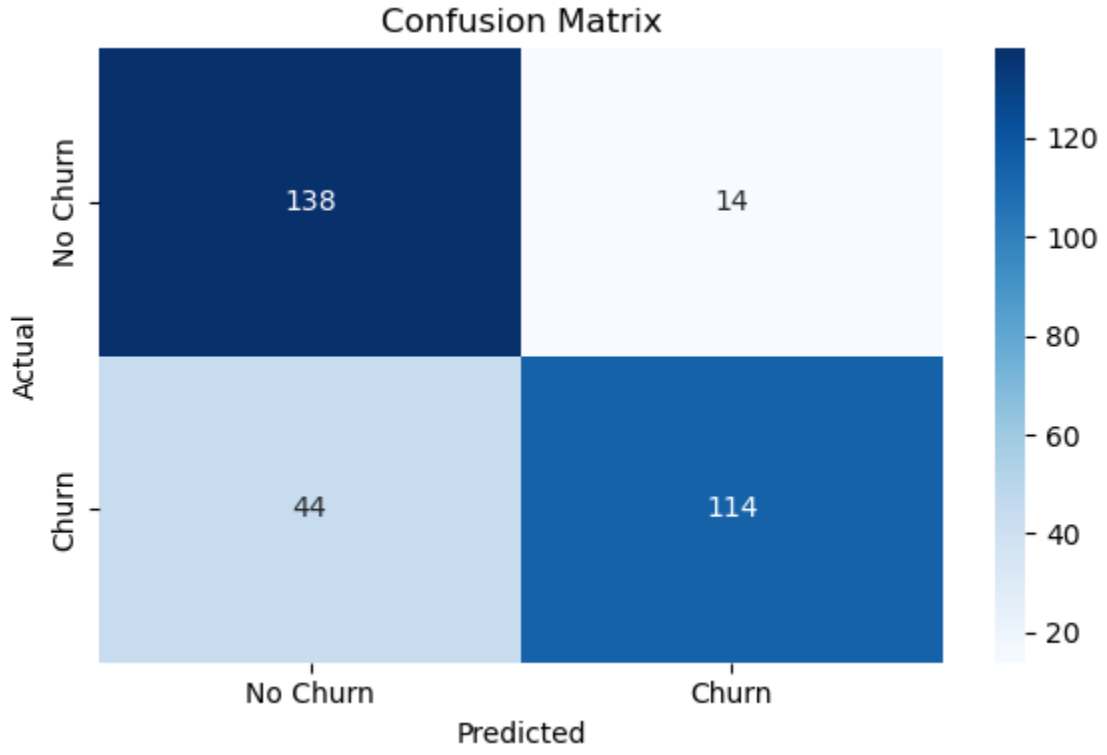

In [196]:

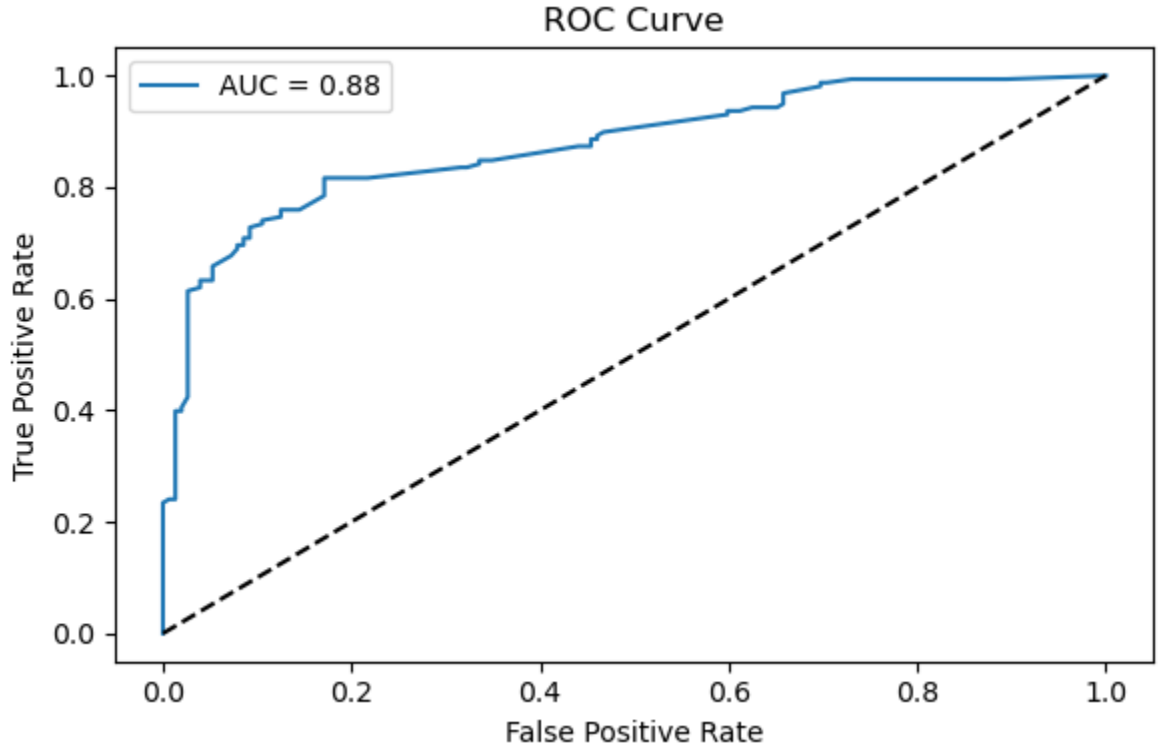
```python
# 📦 Import libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report, roc_auc_score, roc_
from sklearn.preprocessing import LabelEncoder
from sklearn.utils import resample

# -------------------------------------
# STEP 1: Data Preparation
# -------------------------------------

# 🧼 Clean and prepare your data
df['followup_new'] = df['followup_new'].str.strip().str.lower()
df['churn'] = df['followup_new'].apply(lambda x: 1 if x == 'new' else 0)

# Select features for prediction
features = ['age', 'gender', 'payment_mode', 'total_amount', 'consultant_amount']
df_model = df[features + ['churn']].dropna()

# 🎯 Label encoding for categorical columns
label_encoders = {}
for col in ['gender', 'payment_mode']:
    le = LabelEncoder()
    df_model[col] = le.fit_transform(df_model[col])
    label_encoders[col] = le

# -------------------------------------
# STEP 2: Balance the Dataset
# -------------------------------------
churn_yes = df_model[df_model['churn'] == 1]
churn_no = df_model[df_model['churn'] == 0]

# Upsample the minority class (if needed)
churn_no_upsampled = resample(churn_no,
                              replace=True,
                              n_samples=len(churn_yes),
                              random_state=42)

df_balanced = pd.concat([churn_yes, churn_no_upsampled])

X = df_balanced[features]
y = df_balanced['churn']

# -------------------------------------
# STEP 3: Train-Test Split
# -------------------------------------
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42

# -------------------------------------
# STEP 4: Train Random Forest Model
# -------------------------------------
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

```python
y_prob = model.predict_proba(X_test)[:, 1]  # for ROC curve

# -----------------------------------
# STEP 5: Evaluation
# -----------------------------------
print("🔍 Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\n📄 Classification Report:\n", classification_report(y_test, y_pred))

# -----------------------------------
# STEP 6: Visualize Confusion Matrix
# -----------------------------------
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['No Churn', 'Churn'],
            yticklabels=['No Churn', 'Churn'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.tight_layout()
plt.show()

# -----------------------------------
# STEP 7: ROC Curve
# -----------------------------------
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
plt.figure(figsize=(6,4))
plt.plot(fpr, tpr, label=f"AUC = {roc_auc_score(y_test, y_prob):.2f}")
plt.plot([0,1], [0,1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.tight_layout()
plt.show()

# -----------------------------------
# STEP 8: Feature Importance
# -----------------------------------
importances = model.feature_importances_
feature_imp = pd.Series(importances, index=features).sort_values(ascending=False)

plt.figure(figsize=(8,5))
sns.barplot(x=feature_imp.values, y=feature_imp.index, palette='viridis')
plt.title("Feature Importance")
plt.xlabel("Importance Score")
plt.ylabel("Feature")
plt.tight_layout()
plt.show()
```

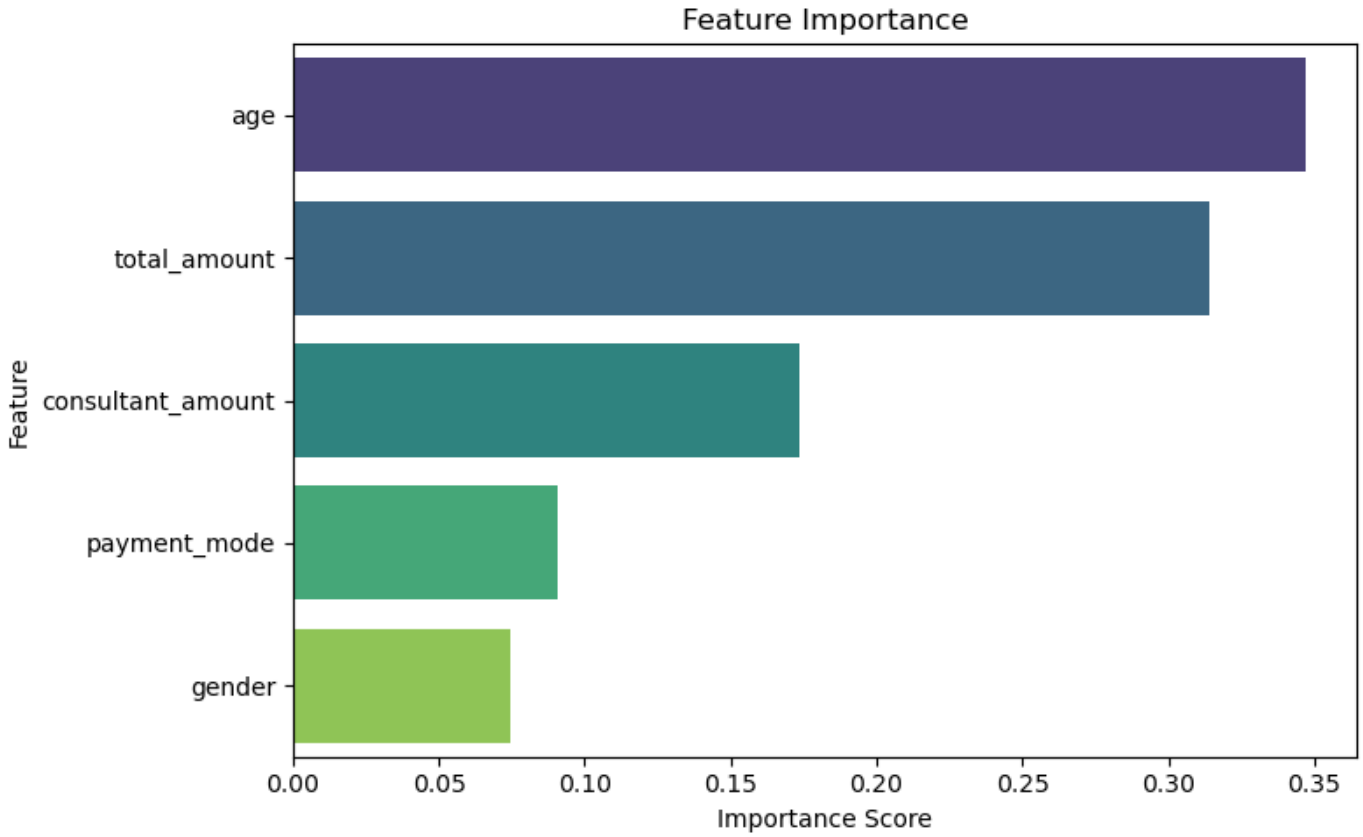
```
🔍 Confusion Matrix:
 [[138  14]
 [ 44 114]]

📄 Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.91      0.83       152
           1       0.89      0.72      0.80       158
```

```
              precision    recall  f1-score   support

    accuracy                           0.81       310
   macro avg       0.82      0.81      0.81       310
weighted avg       0.83      0.81      0.81       310
```




```
/var/folders/1w/677cvfp14kd9g4hm22pmcm9c0000gn/T/ipykernel_6086/961561886.py:103: Future
Warning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.
Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=feature_imp.values, y=feature_imp.index, palette='viridis')
```




In [ ]: