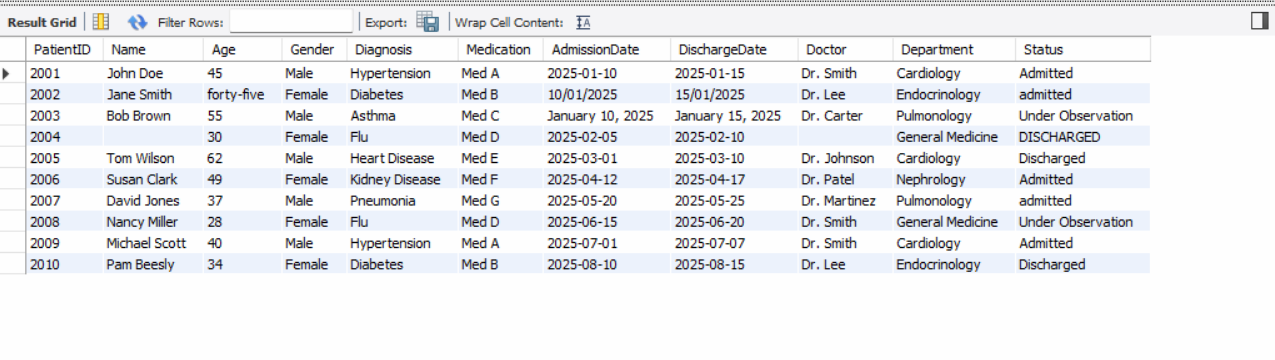
**GROUP 3: Hospital patient records Project**

**Part 1: Data Cleaning (SQL and Python)**

1. **Load the Dataset** 
2. **Data Cleaning**

**SQL Tasks:**

1. **Remove duplicate records:**

A screenshot of a computer

Description automatically generated

1. **Standardize the AdmissionDate column to a consistent format:**

A screenshot of a computer

Description automatically generated

A screen shot of a computer program

Description automatically generated

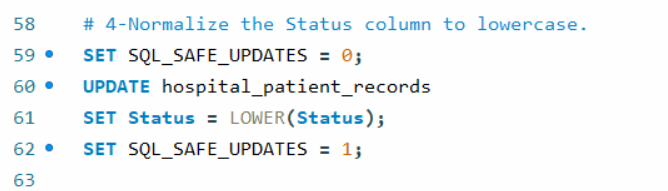
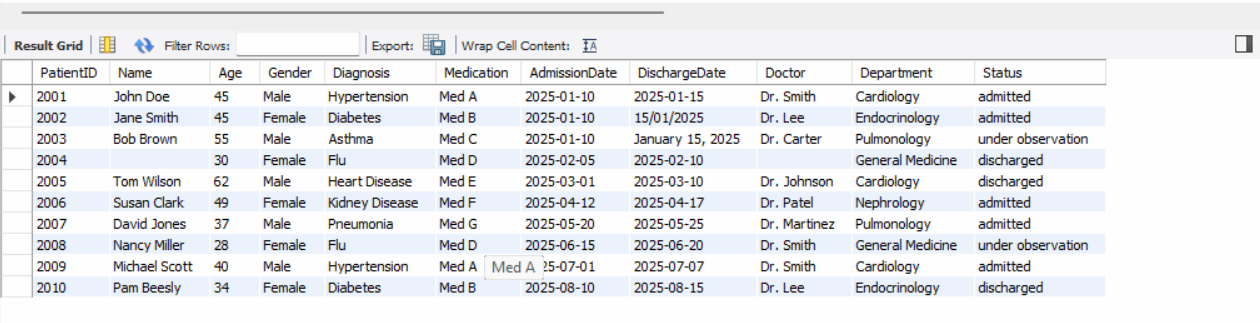
1. **Convert the Age column to a numeric type:**

A screenshot of a computer

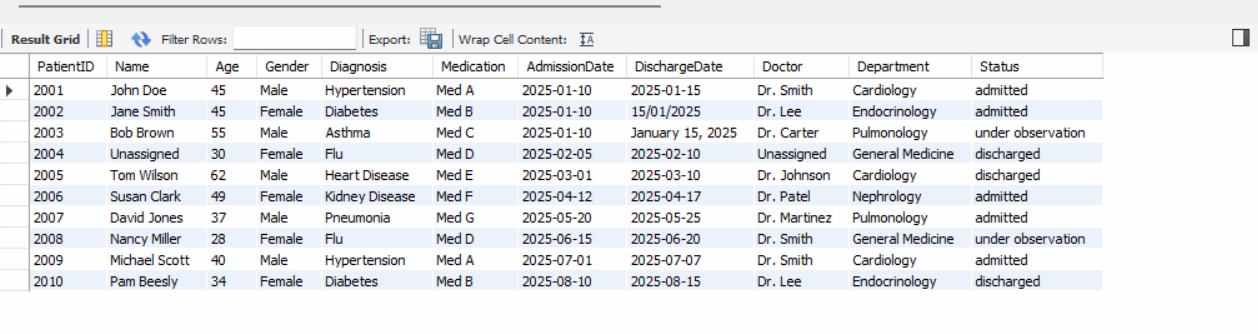
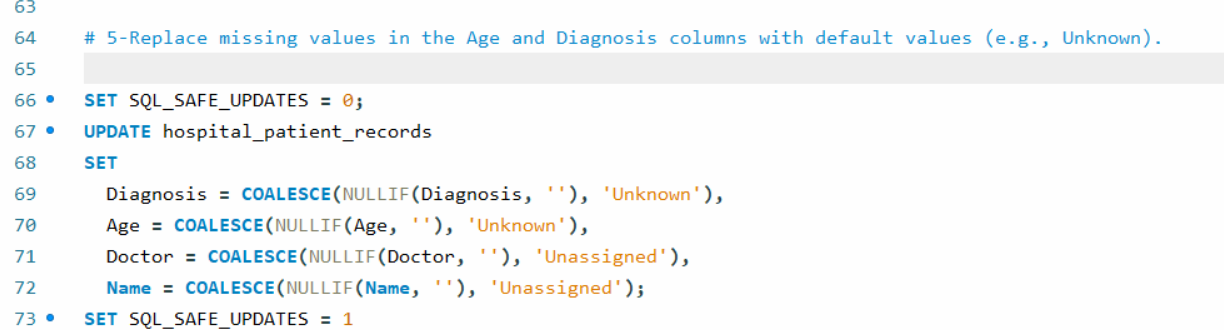
Description automatically generatedA screenshot of a computer code

Description automatically generated

1. **Normalize the Status column to lowercase:**



1. **Replace missing values in the Age and Diagnosis columns with default values:**

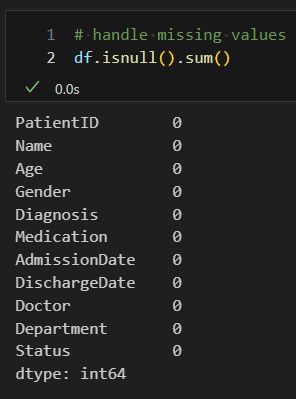


**Python Tasks** Use Pandas to clean the dataset further

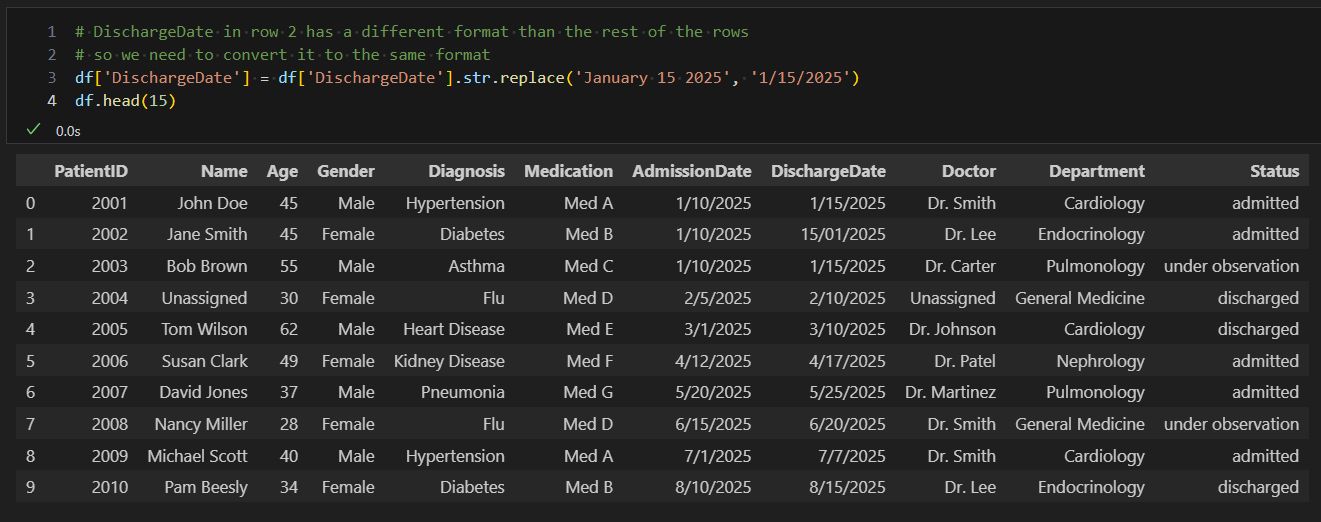
**Remove leading/trailing spaces**



**Handle missing values**

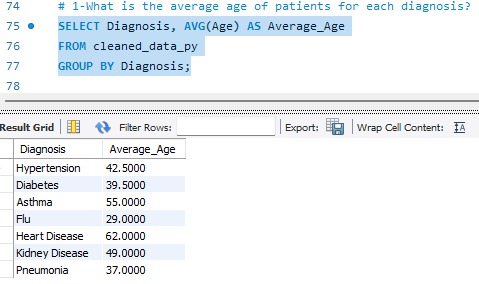


**DischargeDate Format in Row 2**

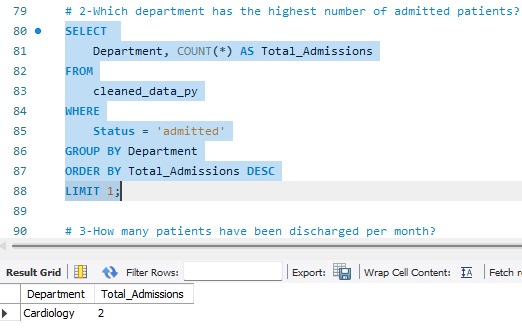


**Part 2: Data Exploration and Analysis**

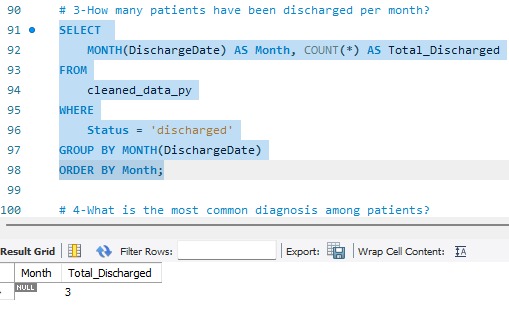
1. **What is the average age of patients for each diagnosis?**



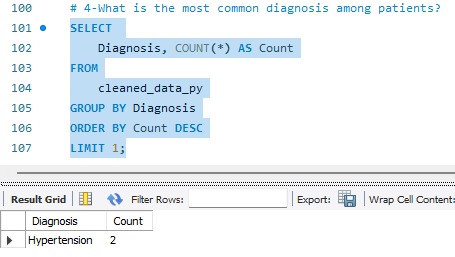
1. **Which department has the highest number of admitted patients?**



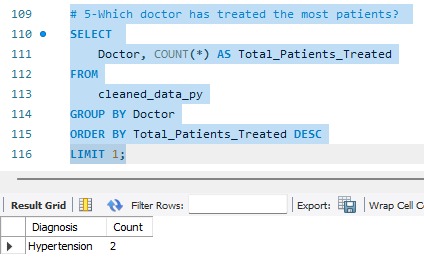
1. **How many patients have been discharged per month?**



1. **What is the most common diagnosis among patients?**

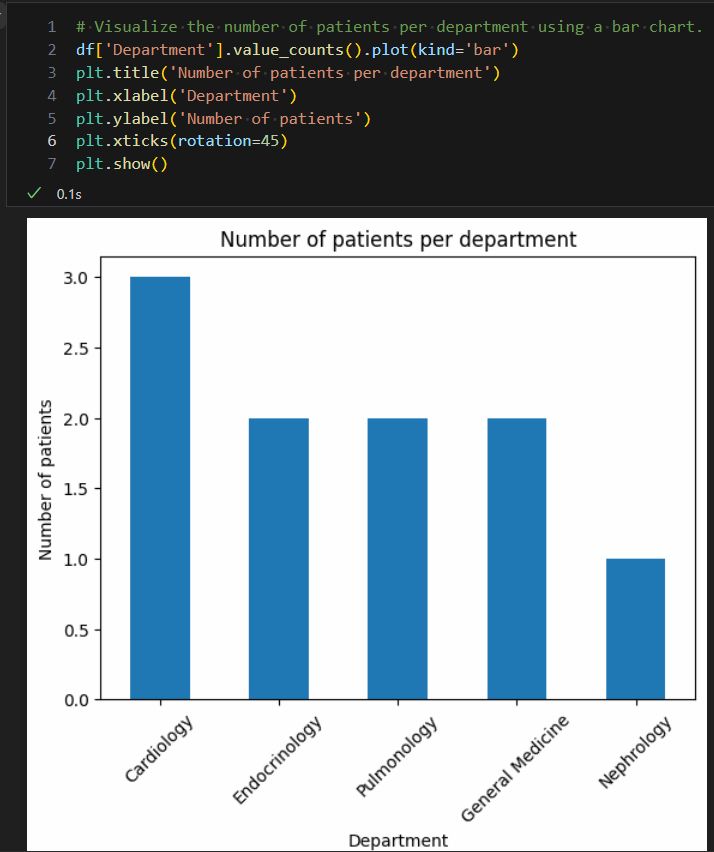


1. **Which doctor has treated the most patients?**

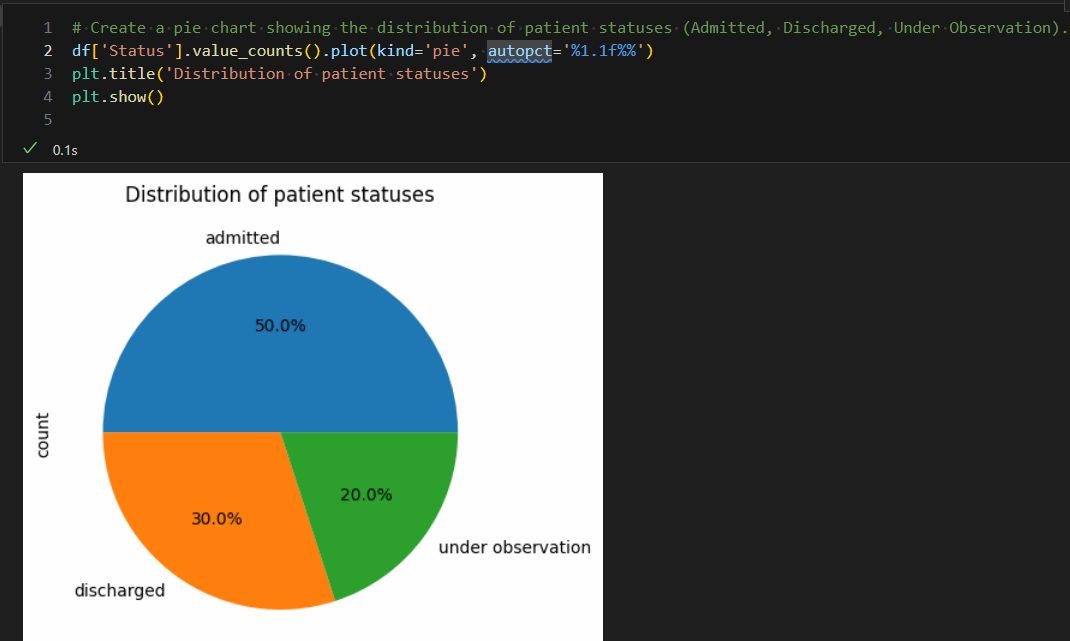


**Python Analysis**

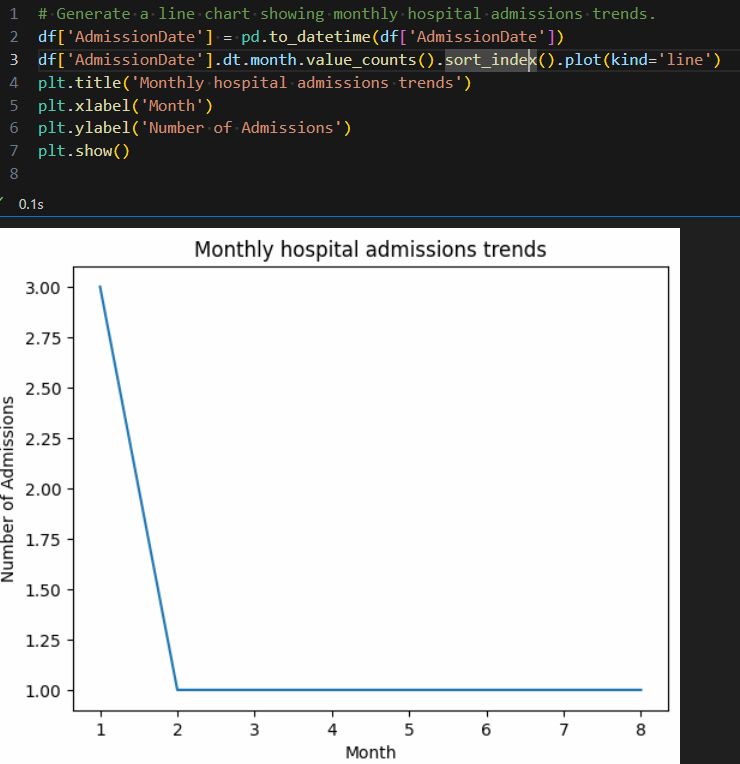
1. **Visualize the number of patients per department using a bar chart.**



1. **Create a pie chart showing the distribution of patient statuses (Admitted, Discharged, Under Observation).**



1. **Generate a line chart showing monthly hospital admissions trends.**



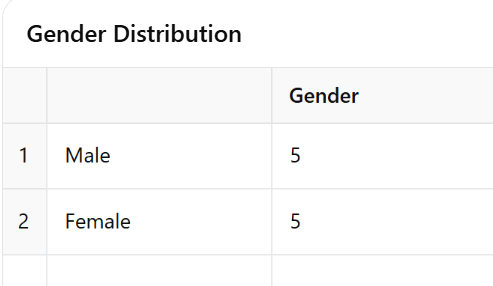
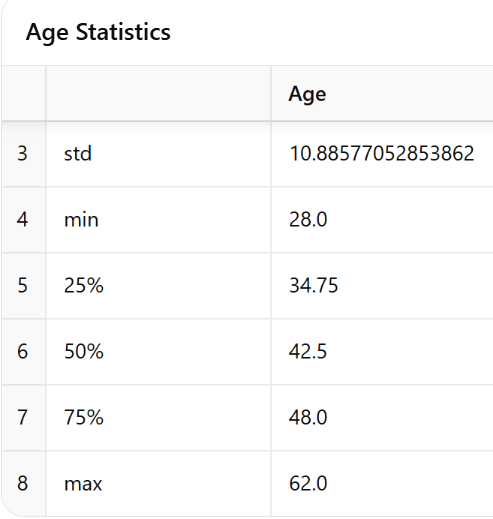
**Part 3: Reporting**

**Recommendations for hospital management:**

**which departments need more resources:**

* Cardiology
* Endocrinology
* Nephrology
* Pulmonology

**Common patient demographics**

****

**A screenshot of a white table

Description automatically generated**

**Theoretical Questions:**

**1. Data Cleaning**

* **Common Issues in a Messy Dataset:**
  + **Duplicate Data:** Records that appear more than once.
  + **Missing Data:** Absences of values in one or more fields.
  + **Errors in Data Entry:** Typographical mistakes or incorrect values.
  + **Outliers:** Data points that deviate significantly from the norm.
* **Handling Missing Values:**
  + A common strategy is to replace missing values with the mean (or median/mode) of the column.
  + Other techniques include interpolation, using model-based imputations, or even deletion if appropriate.
* **Importance of Data Type Consistency:**
  + Ensures that calculations and comparisons can be performed accurately.
  + Simplifies the data cleaning process.
  + Reduces the risk of errors during analysis by ensuring that each column holds the expected type of data.

**2. SQL Queries**

* **Difference Between INNER JOIN and LEFT JOIN:**
  + **INNER JOIN:** Returns only the rows where there is a match on both tables.
  + **LEFT JOIN:** Returns all rows from the left (first) table and the matching rows from the right table; if there is no match, the result will include NULLs for columns from the right table.
* **Using the GROUP BY Clause to Aggregate Data:**
  + The GROUP BY clause is used to group rows that have the same values in one or more columns.
  + It is typically paired with aggregate functions (such as SUM (), AVG (), COUNT (), etc.) to perform calculations on each group.
* **Purpose of the HAVING Clause:**
  + After grouping data using GROUP BY, the HAVING clause is applied to filter groups based on a condition.
  + It serves a similar purpose to the WHERE clause but is used for aggregated data rather than individual rows.

**3. Python Analysis**

* **Cleaning a Dataset with Mixed Data Types Using Pandas:**
  + **df.convert\_dtypes():** Automatically converts columns to the most suitable data types.
  + **df.infer\_objects():** Attempts to infer better data types for columns that are currently stored as objects.
  + **df.apply(pd.to\_numeric, errors='coerce'):** Converts columns to numeric values, coercing invalid parsing to NaN if necessary.
* **Benefits of Using Visualizations in Data Analysis:**
  + **Enhanced Understanding:** Quickly identifies trends, patterns, and outliers.
  + **Improved Decision-Making:** Empowers stakeholders to make data-driven decisions.
  + **Simplified Communication:** Makes complex data more accessible to non-technical audiences.
  + **Faster Insights:** Visual summaries can speed up the analysis process.
  + **Effective Storytelling:** Helps to convey key messages and insights in a compelling manner.