Here's a detailed explanation of your Titanic dataset preprocessing code, along with potential viva questions and answers:

### \*\*Code Explanation\*\*

1. \*\*Loading Data\*\*

```python

df = pd.read\_csv(r"C:\Users\sumit\OneDrive\Desktop\titanic.csv")

```

- Loads the Titanic dataset from a CSV file stored on your desktop

- The dataset contains passenger information like age, sex, fare, class, and survival status

2. \*\*Data Inspection\*\*

```python

df.isnull().sum() # Counts missing values in each column

df.describe(exclude="object") # Statistical summary of numeric columns

df.dtypes # Shows data types of all columns

df.shape # Shows (number of rows, number of columns)

df.info() # Provides concise DataFrame summary

```

These commands help you understand:

- Which columns have missing data (like Age or Cabin)

- Basic statistics of numerical features (mean age, fare ranges)

- The structure and size of your dataset

3. \*\*Data Type Conversion\*\*

```python

df["Survived"] = df['Survived'].astype('bool') # Converts to boolean

df["Pclass"] = df['Pclass'].astype('category') # Converts to category

```

- Changes `Survived` (0/1) to `True/False` (more semantically meaningful)

- Converts `Pclass` (1,2,3) to categorical data type (better for analysis)

4. \*\*Verification\*\*

```python

df.dtypes # Checks the updated data types

```

- Confirms the type conversions were successful

---

### \*\*Viva Questions & Answers\*\*

#### \*\*Basic Questions\*\*

\*\*Q1: Why check for missing values?\*\*

A1: Missing data can:

- Cause errors in machine learning models

- Skew statistical analyses

- Need to be handled by either:

- Removing rows/columns

- Filling with mean/median (for Age)

- Using special values (like "Unknown" for Cabin)

\*\*Q2: What's the difference between `describe()` and `info()`?\*\*

A2:

- `describe()`: Shows statistical summary (count, mean, min/max) for numeric columns

- `info()`: Shows column names, data types, and non-null counts

\*\*Q3: Why convert 'Survived' to boolean?\*\*

A3: Because:

- More intuitive than 0/1 (True=survived, False=didn't survive)

- Takes less memory than integer storage

- Makes filtering clearer: `df[df['Survived']]` vs `df[df['Survived']==1]`

#### \*\*Intermediate Questions\*\*

\*\*Q4: What are the advantages of categorical data type?\*\*

A4:

- Uses less memory than strings/numbers

- Enables efficient sorting in logical order (1st < 2nd < 3rd class)

- Works better with some visualization and ML algorithms

\*\*Q5: How would you handle the missing 'Age' values?\*\*

A5: Common approaches:

1. Fill with median age (robust to outliers):

```python

df['Age'].fillna(df['Age'].median(), inplace=True)

```

2. Use title from 'Name' to estimate age (Mr vs Master)

3. Use predictive modeling

\*\*Q6: What other preprocessing would this dataset need?\*\*

A6:

- One-hot encode 'Sex' and 'Embarked'

- Create new features (FamilySize = SibSp + Parch)

- Normalize 'Fare' (large range could skew models)

- Extract titles from 'Name' (Mr, Mrs, etc.)

#### \*\*Advanced Questions\*\*

\*\*Q7: When should you NOT use categorical data type?\*\*

A7: When:

- The column has high cardinality (too many unique values)

- You need to perform mathematical operations

- Working with libraries that don't support pandas categories

\*\*Q8: How does memory usage change with these conversions?\*\*

A8:

- `boolean` uses 1 byte per value vs 8 bytes for `int64`

- `category` uses even less memory for repetitive strings

- Example: Converting 'Sex' to category could reduce memory by 90%

\*\*Q9: How would these changes affect machine learning?\*\*

A9:

- Boolean/categorical types are handled natively by most ML algorithms

- Some models require one-hot encoding of categories first

- Proper typing helps avoid treating categories as continuous numbers

---

### \*\*Practical Tasks\*\*

\*\*Task 1: Convert 'Sex' to category\*\*

```python

df['Sex'] = df['Sex'].astype('category')

```

\*\*Task 2: Fill missing Embarked values with mode\*\*

```python

df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)

```

\*\*Task 3: Create age groups\*\*

```python

df['AgeGroup'] = pd.cut(df['Age'],

bins=[0, 12, 18, 60, 100],

labels=['Child', 'Teen', 'Adult', 'Senior'])

```

---

### \*\*Key Takeaways\*\*

1. Always inspect your data first (`info()`, `describe()`)

2. Choose appropriate data types for each column

3. Handle missing values before analysis/modeling

4. Categorical data often needs special treatment

Would you like me to demonstrate any specific aspect of this preprocessing pipeline in more detail?