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 Titanic dataset from a CSV file on your desktop

- Contains passenger details like age, sex, fare, survival status, etc.

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

```python

df.isnull().sum() # Checks for missing values

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

df.dtypes # Shows data types of each column

df.shape # Returns (rows, columns) count

df.info() # Concise summary of the DataFrame

```

- These commands help understand:

- Missing values (Age, Cabin have many nulls)

- Data distributions (average age, fare ranges)

- Column types (numeric vs categorical)

3. \*\*One-Hot Encoding\*\*

```python

df\_encoded = pd.get\_dummies(df, columns=['Sex', 'Embarked'])

```

- Converts categorical text data (Sex, Embarked) into numeric format

- Creates new columns like:

- `Sex\_female` (1 if female, 0 otherwise)

- `Sex\_male` (1 if male, 0 otherwise)

- `Embarked\_C`, `Embarked\_Q`, `Embarked\_S` (for embarkation ports)

---

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

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

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

A1: Missing data can cause errors in machine learning models. We check to decide whether to:

- Drop rows/columns

- Fill with mean/median (for Age)

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

\*\*Q2: What does `describe(exclude="object")` show?\*\*

A2: Displays statistics only for numeric columns:

- Count

- Mean

- Std (standard deviation)

- Min/max

- 25%/50%/75% percentiles

\*\*Q3: Why one-hot encode categorical data?\*\*

A3: Because:

- ML algorithms require numeric input

- Prevents false ordinal relationships (e.g., treating "male" > "female")

- Better than label encoding for nominal categories

---

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

\*\*Q4: How would you handle missing 'Age' values?\*\*

A4: Common approaches:

1. Fill with median age (less affected by outliers)

```python

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

```

2. Predict age using other features (like title from Name)

\*\*Q5: What other preprocessing steps might this dataset need?\*\*

A5:

- Feature engineering (create FamilySize from SibSp+Parch)

- Normalize Fare (large range could skew models)

- Extract titles from Names (Mr, Mrs, etc.)

- Bin Age into groups (child/adult/senior)

\*\*Q6: When would you NOT use one-hot encoding?\*\*

A6: When:

- High cardinality (too many categories → too many columns)

- Categories have natural order (use label encoding instead)

- Using tree-based models (can handle categorical data directly)

---

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

\*\*Q7: How does `get\_dummies()` differ from `OneHotEncoder`?\*\*

A7:

| `get\_dummies()` | `OneHotEncoder` |

|-----------------|-----------------|

| Pandas function | Scikit-learn class |

| Immediate output | Needs `fit\_transform` |

| Handles strings directly | Requires numeric input first |

| Simpler for EDA | Better for ML pipelines |

\*\*Q8: What problem might arise with one-hot encoded 'Embarked'?\*\*

A8: Multicollinearity - knowing 2 of `[Embarked\_C, Embarked\_Q, Embarked\_S]` implies the third. Solution: Use `drop\_first=True` to remove one column.

\*\*Q9: How would you prepare this data for a survival prediction model?\*\*

A9: Typical steps:

1. Handle missing values

2. Encode all categorical features

3. Scale numeric features (StandardScaler)

4. Split into train/test sets

5. Optional: Feature selection

---

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

\*\*Task 1: Fill missing Ages with median\*\*

```python

df['Age'] = df['Age'].fillna(df['Age'].median())

```

\*\*Task 2: Create a new 'FamilySize' feature\*\*

```python

df['FamilySize'] = df['SibSp'] + df['Parch'] + 1 # +1 for self

```

\*\*Task 3: Normalize 'Fare'\*\*

```python

from sklearn.preprocessing import StandardScaler

df['Fare\_scaled'] = StandardScaler().fit\_transform(df[['Fare']])

```

---

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

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

2. Handle missing values before modeling

3. Encode categorical data appropriately

4. Feature engineering often improves model performance

Would you like me to demonstrate any specific preprocessing technique in more detail?