What are the top 5 most popular genres?

The goal of the code is to find the top 5 most popular game genres from a dataset.

1. **Understanding Genres**: The `Genres` column in the dataset lists the types of genres each game belongs to. For example, a game might be labeled as `['Adventure']` or `['Shooter', 'Indie']`.
2. **Identifying Unique Genres**: The code checks all the unique combinations of genres that exist in the dataset. This means it looks at all different way genres are combined across all the games.
3. **Replacing Empty Genres**: If any game does not have a genre listed (an empty entry), the code replaces it with the word 'Unknown'. This ensures that every game has some genre information.
4. **Counting Genres**: The code then counts how many times each genre combination appears in the dataset.
5. **Top 5 Genres**: From these counts, the code identifies the top 5 most frequently occurring genres. In this case, the top 5 are:

    - No genre listed (represented as `[]`)
    - Adventure (listed as `['Adventure']`)
    - Shooter (listed as `['Shooter']`)
    - Adventure and Indie (listed as `['Adventure', 'Indie']`)
    - Indie (listed as `['Indie']`)

This helps to see which types of games are most common in the dataset.

5 most popular genres list:

```
Genres
[]                          6741
['Adventure']               2925
['Shooter']                 2415
['Adventure', 'Indie']      2248
['Indie']                   1972
Name: count, dtype: int64
```

This is the code for top 5 most popular genres:

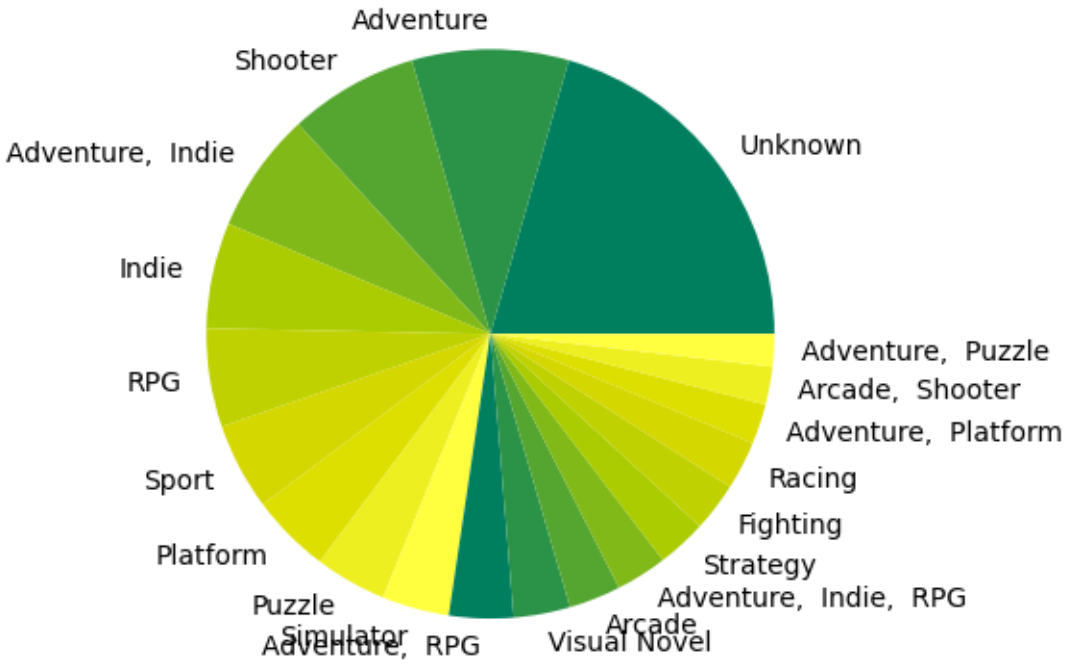
```
#Hassan work
game_df['Genres'].unique()
```

```
array(['Adventure,  RPG', 'Adventure,  Puzzle',
       'Adventure,  Brawler,  Indie,  RPG', ...,
       'Indie,  MOBA,  Shooter,  Strategy',
       'Card & Board Game,  Real Time Strategy,  RPG,  Turn Based Strategy',
       'Adventure,  Card & Board Game,  Indie,  RPG,  Strategy,  Tactical,  Turn Based Strategy,  Visual Novel'],
      dtype=object)
```

```
game_df['Genres']=game_df['Genres'].replace('','Unknown')
```

```
filter_data = game_df['Genres'].value_counts().sort_values(ascending=False).head(5)
```

Pie chart for genres:



Code for Pie chart:

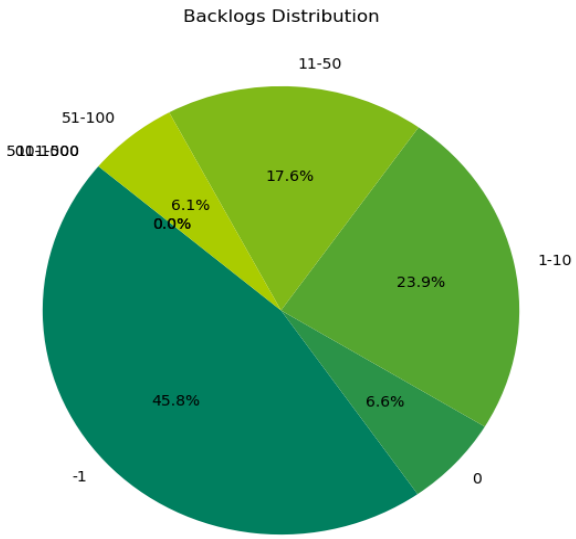
```
genres=filter_data.index
count=filter_data.values
```

```
plt.figure()
plt.pie(count, labels=genres)
```

```
([<matplotlib.patches.Wedge at 0x1a79196c110>,
  <matplotlib.patches.Wedge at 0x1a79196c050>,
  <matplotlib.patches.Wedge at 0x1a79196c860>,
  <matplotlib.patches.Wedge at 0x1a791962cc0>,
  <matplotlib.patches.Wedge at 0x1a791963160>,
  <matplotlib.patches.Wedge at 0x1a7919635c0>,
  <matplotlib.patches.Wedge at 0x1a791963a60>,
  <matplotlib.patches.Wedge at 0x1a791963cc0>,
  <matplotlib.patches.Wedge at 0x1a79195c340>,
  <matplotlib.patches.Wedge at 0x1a79195c7c0>,
  <matplotlib.patches.Wedge at 0x1a79195cc40>,
  <matplotlib.patches.Wedge at 0x1a79195d0c0>,
  <matplotlib.patches.Wedge at 0x1a79195d540>,
  <matplotlib.patches.Wedge at 0x1a79195d9c0>,
  <matplotlib.patches.Wedge at 0x1a79195de40>,
  <matplotlib.patches.Wedge at 0x1a79195e2c0>,
  <matplotlib.patches.Wedge at 0x1a79195e740>,
  <matplotlib.patches.Wedge at 0x1a79195ebc0>,
  <matplotlib.patches.Wedge at 0x1a79195f040>,
  <matplotlib.patches.Wedge at 0x1a791941870>],
 [Text(0.8786677381738015, 0.6617726240126106, 'Unknown'),
  Text(-0.0003686506677363942, 1.0999999382257644, 'Adventure'),
  Text(-0.5386191561447331, 0.9591086511099437, 'Shooter'),
  Text(-0.8999867916910407, 0.6324743273696313, 'Adventure,  Indie'),
  Text(-1.0761837783613795, 0.22765868134517872, 'Indie'),
  Text(-1.0866435601813373, -0.17089696637573307, 'RPG'),
  Text(-0.9723434682587166, -0.5143424731971981, 'Sport'),
  Text(-0.7790640606822256, -0.7765688568010706, 'Platform'),
  Text(-0.5432051789687777, -0.9565187575481718, 'Puzzle'),
  Text(-0.2910634447240831, -1.0607931330591045, 'Simulator'),
  Text(-0.036071052442165 64, -1.0994084223689187, 'Adventure,  RPG'),
  Text(0.2002268992487585 6, -1.0816234043405 44, 'Visual Novel'),
  Text(0.4049837115401454 3, -1.0227673168326288, 'Arcade'),
  Text(0.5854232140139075, -0.9312785085534976, 'Adventure,  Indie,  RPG'),
  Text(0.7424813058816878, -0.8116166030930021, 'Strategy'),
  Text(0.8727394385876128, -0.6695714094357512, 'Fighting'),
  Text(0.9754287159627768, -0.5084671278216605, 'Racing'),
  Text(1.0412505908187 65, -0.3487523544813 6793, 'Adventure,  Platform'),
  Text(1.0815370227154493, -0.2806929707184633 7, 'Arcade,  Shooter'),
  Text(1.0981605745144738, -0.0635873618106697 3, 'Adventure,  Puzzle')])
```

Number of Copies sold:

This code effectively simulates a large dataset of game backlogs based on provided statistical parameters, categorizes the data into meaningful bins, and visualizes the distribution through a pie chart. Visualization helps in understanding the frequency and proportion of different backlog categories.



This is code for number of copies sold:

```
[62]: # Creating a DataFrame based on the provided summary statistics
data = {
    'Backlogs': {
        'count': 60000, 'mean': 6.298450, 'std': 61.118711, 'min': -1.0, '25%': 0.0,
        '50%': 0.0, '75%': 0.0, 'max': 4600.0
    }
}

# Generating a simulated 'Backlogs' dataset
np.random.seed(0)
backlogs = np.random.normal(loc=6.298450, scale=61.118711, size=60000)
backlogs = np.clip(backlogs, -1, 4600)

# Creating a DataFrame
df = pd.DataFrame({'Backlogs': backlogs})

# Categorizing 'Backlogs' into different bins for the pie chart
bins = [-1, 0, 10, 50, 100, 500, 1000, 4600]
labels = ['-1', '0', '1-10', '11-50', '51-100', '101-500', '501-1000']
df['Backlog_Category'] = pd.cut(df['Backlogs'], bins=bins, labels=labels, include_lowest=True)

# Calculating the distribution for the pie chart
backlog_distribution = df['Backlog_Category'].value_counts().sort_index()

# Plotting the pie chart
plt.figure(figsize=(10, 7))
plt.pie(backlog_distribution, labels=backlog_distribution.index, autopct='%1.1f%%', startangle=140)
plt.title('Backlogs Distribution')
plt.show()
```