Programação para Dados
Fase 1

Nome do estudante	Isadora Ferraz e Figueiredo
Curso	Banco de Dados ênfase em Data Analytics

O código-fonte deste projeto está disponível no GitHub: **Programacao-para-Dados**

Pergunta 1: Qual o percentual de jogos gratuitos e pagos na plataforma?

Para calcular a porcentagem de jogos gratuitos e pagos na Steam, foram criados dois métodos dentro da classe `GameStats`.

O método `game_counts()` contabiliza o total de jogos únicos com base na coluna `AppID`, garantindo que não haja duplicatas. Além disso, ele também conta os jogos gratuitos, onde `Price = 0`.

Em seguida, o método `percentage_games()` utiliza os valores retornados por `game_counts()` para calcular a porcentagem de jogos gratuitos e pagos, arredondando o resultado para duas casas decimais.

O resultado obtido foi:

Jogos gratuitos: **17.39%**
Jogos pagos: **82.61%**

A implementação dos métodos pode ser vista a seguir:

Listing 1: Métodos utilizados

```
        def game_counts(self):
            """

            Retorna o total de jogos unicos e gratuitos unicos.

            """
            unique_games = set()
            unique_free_games = set()

            for game in self.games:
                app_id = game['AppID']
                unique_games.add(app_id)

                try:
                    if float(game.get('Price', 0)) == 0.0:
                        unique_free_games.add(app_id)
                except ValueError:
                    continue

            return len(unique_games), len(unique_free_games)

        def percentage_games(self):
            """
            Retorna as porcentagens de jogos gratuitos e pagos.
            """
            total_games, total_free = self.game_counts()
            total_paid = total_games - total_free

            percent_free = (total_free / total_games) * 100
            percent_paid = (total_paid / total_games) * 100

            return {
                'free_percentage': round(percent_free, 2),
                'paid_percentage': round(percent_paid, 2)
            }
```

Pergunta 2: Qual o ano com o maior número de novos jogos? Em caso de empate, retorne uma lista com os anos empatados.

Primeiro, foi criado o método `yearly_releases()` para contar a quantidade de jogos lançados por ano. No entanto, a data de lançamento não seguia um formato padronizado, o que poderia comprometer a análise. Para corrigir esse problema, utilizou-se a função `re.search()`, que extrai automaticamente um conjunto de quatro dígitos representando o ano. Dessa forma, garantiu-se um agrupamento correto das informações.

Em seguida, foi implementado o método `peak_years()` para identificar o ano com o maior número de lançamentos. Esse método primeiro obtém o dicionário de lançamentos por ano a partir de `yearly_releases()`. Depois, encontra o valor máximo de lançamentos e filtra os anos que possuem essa quantidade. Se houver apenas um ano com mais lançamentos, ele é retornado diretamente; caso contrário, retorna-se uma lista com todos os anos empatados.

O resultado obtido foi:

Ano com maior número de lançamento: 2022

A implementação dos métodos pode ser vista a seguir:

Listing 2: Métodos utilizados

```
    def yearly_releases(self):
        """

        Retorna um dicionario com o total de jogos unicos lancados por
             ano.

        """
        games_year = defaultdict(int)
        seen_games = set()

        for game in self.games:
            app_id = game['AppID']
            if app_id in seen_games:
                continue

            seen_games.add(app_id)

            release_date = game.get('Release date', '').strip()
            year_match = re.search(r'\b\d{4}\b', release_date)
            year = year_match.group() if year_match else 'Unknown'

            games_year[year] += 1

            return dict(games_year)

    def peak_years(self):
        """

        Retorna o ano ou anos com o maior numero de lancamentos de
            jogos.

        """
        games_year = self.yearly_releases()
        max_releases = max(games_year.values())
        most_years = [year for year, count in games_year.items() if
            count == max_releases]

        return most_years if len(most_years) > 1 else most_years[0]
```

Pergunta 3: Existe uma relação entre o preço e o tempo mediano de um jogo?

Para responder essa questão, foi implementado o método `price_playtime()`, que agrupa os jogos em diferentes faixas de preço: grátis, até 5 dólares, entre 5 e 20 dólares, entre 20 e 50 dólares, e acima de 50 dólares.

Para cada jogo, o preço é convertido para um valor numérico e comparado às faixas definidas. Em seguida, o tempo mediano de jogo correspondente é armazenado na categoria apropriada. Após a classificação de todos os jogos, calcula-se a média do tempo mediano para cada faixa de preço, arredondando o resultado para duas casas decimais. Dessa forma, é possível verificar se jogos mais caros tendem a ser jogados por mais tempo.

O resultado obtido foi:

Faixa de Preço	**Tempo Mediano (min)**
Grátis	102.93
Até 5	48.74
5 a 20	122.14
20 a 50	411.27
Acima de 50	854.9

Notou-se que, de maneira geral, jogos mais caros apresentam um tempo mediano de jogo maior, sugerindo que jogadores tendem a dedicar mais tempo a títulos com maior investimento.

A implementação do método pode ser vista a seguir:

Listing 3: Métodos utilizados

```
def price_playtime(self):
    """

    Analisa a relacao entre o preco dos jogos e o tempo mediano de jogo.

    Retorna um dicionario com faixas de preco e a media do tempo mediano de
        jogo para cada faixa.

    """
    price_categories = defaultdict(list)

    for game in self.games:
        try:
            price = float(game.get('Price', 0))
            playtime = float(game.get('Median playtime forever', 0))

            if price == 0:
                price_categories["Gratis"].append(playtime)

            elif price <= 5:
                price_categories["Ate $5"].append(playtime)

            elif price <= 20:
                price_categories["De $5 a $20"].append(playtime)

            elif price <= 50:
                price_categories["De $20 a $50"].append(playtime)

            else:
                price_categories["Acima de $50"].append(playtime)

        except ValueError:
            continue

    # Calcula a media do tempo mediano de jogo por categoria
    result = {cat: round(sum(times) / len(times), 2) if times else 0 for cat,
         times in price_categories.items()}
    return result
```

PROGRAMA TESTE

Para validar a precisão do programa, foi desenvolvido um teste utilizando uma amostra de 20 jogos selecionados aleatoriamente do arquivo `steam.csv`. Os valores dessa amostra foram analisados manualmente e comparados com os resultados gerados pelo código, garantindo a consistência dos cálculos e a confiabilidade do programa.

Primeiramente, os valores abaixo foram extraídos aleatoriamente da base de dados:

AppID	**Release date**	**Price ($)**	**Median Playtime Forever (min)**
1967950	May 18, 2022	9.99	0
812580	Sep 17, 2018	9.99	0
220700	Dec 10, 2012	10.49	219
675010	Oct 30, 2017	19.99	369
696460	Aug 30, 2017	3.99	0
572010	Oct 4, 2018	0.99	292
1473100	Dec 8, 2020	0.99	0
1139490	Mar 28, 2023	17.99	0
1202860	Dec 13, 2019	9.99	0
1366930	Sep 27, 2021	9.99	0
1841580	Jan 15, 2022	4.99	0
576130	Mar 31, 2017	2.99	314
814770	Mar 13, 2018	19.99	0
981790	Nov 26, 2018	0.99	0
925700	Sep 13, 2018	2.99	0
2147260	Nov 13, 2022	4.99	0
1815170	Jan 11, 2022	0	0
1477660	Dec 1, 2020	6.99	0
2357130	Mar 31, 2023	2.39	0
1045010	Jun 17, 2020	0.99	0

Tabela 1: Colunas e linhas extraídas da amostra

Em seguida, fora realizada a contagem dos jogos dentro de cada faixa de preço:

Categoria de Preço	**Quantidade de Jogos**
Grátis	1
Até 5 dólares	10
Entre 5 e 20 dólares	9
Entre 20 e 50 dólares	0
Acima de 50 dólares	0

Tabela 2: Quantidade de jogos por categoria de preço

Com a quantidade de jogos gratuitos e pagos contabilizados, fora possível calcular a porcentagem de cada categoria.

Valor	Porcentagem
Gratuitos	$\frac{1}{20} \cdot 100\% = 5\%$
Pagos	$\frac{19}{20} \cdot 100\% = 95\%$

Tabela 3: Porcentagem de jogos gratuitos e pagos

A seguir, fora calculada a média do tempo mediano de jogo por categoria de preço:

Categoria de Preço	Média de Tempo Mediano (min)
Grátis	$\frac{0}{1} = 0$
Até 5 dólares	$\frac{292 + 314}{10} = 60.6$
Entre 5 e 20 dólares	$\frac{219 + 369}{9} = 65.33$
Entre 20 e 50 dólares	–
Acima de 50 dólares	–

Tabela 4: Média de tempo mediano por categoria de preço

Na sequência fora agrupada a quantidade de jogo de acordo com o ano de lançamento.

Ano	Quantidade de Jogos
2012	1
2017	3
2018	5
2019	1
2020	3
2021	1
2022	4
2023	2

Tabela 5: Quantidade de jogos lançados por ano

Por fim, os valores calculados pelo programa foram comparados com os valores obtidos manualmente:

Métrica	**Valores Python**	**Valores Manual**
Porcentagem de jogos gratuitos	5%	5%
Porcentagem de jogos pagos	95%	95%
Ano com maior lançamento	2018	2018
Média de tempo - Jogos grátis	0.0 min	0 min
Média de tempo - Até 5 dólares	60.6 min	60.6 min
Média de tempo - De 5 a 20 dólares	65.33 min	65.33 min
Média de tempo - De 20 a 50 dólares	-	-
Média de tempo - Acima de 50 dólares	-	-

Tabela 6: Comparação entre valores obtidos no Python e manualmente em relação a amostra

Como os resultados do programa coincidem com os cálculos manuais, sua precisão foi confirmada. Isso demonstra que ele pode ser utilizado de forma confiável em análises futuras.