

MVP Sprint 3: Engenharia de Dados

Eduardo Vasconcelos Sampaio

PUC-RIO

Objetivo

A partir de uma base de dados com estatísticas dos jogadores da NBA(Liga Americana de Basquete) definir as seguintes perguntas :

- Quais são as equipes com a melhor e a pior média de percentual de arremesso de campo?
 - Identificar as equipes mais eficazes ofensivamente.
- Quais são as equipes com a melhor média de pontos na temporada?
 - Avaliar a precisão e eficiência das equipes nos arremessos de campo.
- Como se comparam as equipes em termos de classificação média dos jogadores?
 - Identificar equipes com jogadores de alto desempenho individual.

Com isso iremos pesquisar dados da ultima temporada da liga, afim de obter as respostas e entendermos perspectivas para a proxima temporada.

Busca pelos Dados

No seguinte link : https://www.basketball-reference.com/leagues/NBA_2024_per_game.html#per_game_stats, iremos utilizar os dados por jogo dos jogadores

https://www.basketball-reference.com/leagues/NBA_2024_per_game.html#per_game_stats

2023-24 NBA Season

Standings

Schedule and Results

Leaders

Coaches

Player Stats

Other

2024 Playoffs Summary

Back to top

Player Per Game

Share & Export

☒ When table is sorted, hide non-qualifiers for rate stats

Glossary

Hide Partial Rows

Rk	Player	Pos	Age	Tm	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%	eFG%	FT	FTA	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS
1	Precious Achiuwa	PF-C	24	TOT	74	18	21.9	3.2	6.3	.501	0.4	1.3	.268	2.8	5.0	.562	.529	0.9	1.5	.616	2.6	4.0	6.6	1.3	0.6	0.9	1.1	1.9	7.6
1	Precious Achiuwa	C	24	TOR	25	0	17.5	3.1	6.8	.459	0.5	1.9	.277	2.6	4.9	.528	.497	1.0	1.7	.571	2.0	3.4	5.4	1.8	0.6	0.5	1.2	1.6	7.7
1	Precious Achiuwa	PF	24	NYK	49	18	24.2	3.2	6.1	.525	0.3	1.0	.260	2.9	5.1	.578	.547	0.9	1.4	.643	2.9	4.3	7.2	1.1	0.6	1.1	1.1	2.1	7.6
2	Bam Adebayo	C	26	MIA	71	71	34.0	7.5	14.3	.521	0.2	0.6	.357	7.3	13.7	.528	.529	4.1	5.5	.755	2.2	8.1	10.4	3.9	1.1	0.9	2.3	2.2	19.3
3	Ochai Agbaji	SG	23	TOT	78	28	21.0	2.3	5.6	.411	0.8	2.7	.294	1.5	2.8	.523	.483	0.5	0.7	.661	0.9	1.8	2.8	1.1	0.6	0.6	0.8	1.5	5.8
3	Ochai Agbaji	SG	23	UTA	51	10	19.7	2.1	4.9	.426	0.9	2.8	.331	1.2	2.1	.551	.520	0.3	0.4	.750	0.7	1.8	2.5	0.9	0.5	0.6	0.7	1.3	5.4
3	Ochai Agbaji	SG	23	TOR	27	18	23.6	2.7	6.8	.391	0.6	2.6	.217	2.1	4.3	.496	.432	0.8	1.3	.611	1.4	1.9	3.3	1.3	0.7	0.6	1.1	1.9	6.7
4	Santi Aldama	PF	23	MEM	61	35	26.5	4.0	9.3	.435	1.7	5.0	.349	2.3	4.3	.534	.528	0.9	1.4	.621	1.2	4.6	5.8	2.3	0.7	0.9	1.1	1.5	10.7
5	Nickeil Alexander-Walker	SG	25	MIN	82	20	23.4	2.9	6.6	.439	1.6	4.1	.391	1.3	2.5	.517	.560	0.6	0.8	.800	0.4	1.6	2.0	2.5	0.8	0.5	0.9	1.7	8.0
6	Grayson Allen	SG	28	PHO	75	74	33.5	4.5	9.1	.499	2.7	5.9	.461	1.8	3.2	.570	.649	1.7	2.0	.878	0.6	3.3	3.9	3.0	0.9	0.6	1.3	2.1	13.5
7	Jarrett Allen	C	25	CLE	77	77	31.7	6.7	10.6	.634	0.0	0.1	.000	6.7	10.6	.638	.634	3.0	4.1	.742	3.2	7.4	10.5	2.7	0.7	1.1	1.6	1.9	16.5
8	Timmy Allen	SF	24	MEM	5	0	25.0	1.2	4.6	.261	0.0	1.4	.000	1.2	3.2	.375	.261	0.2	0.4	.500	0.8	2.6	3.4	1.0	0.8	0.0	0.4	3.6	2.6
9	Jose Alvarado	PG	25	NOP	56	0	18.4	2.5	6.2	.412	1.4	3.7	.377	1.1	2.5	.464	.525	0.6	0.9	.673	0.4	1.8	2.3	2.1	1.1	0.3	0.7	1.6	7.1
10	Kyle Anderson	PF	30	MIN	79	10	22.6	2.5	5.5	.460	0.1	0.6	.229	2.4	4.9	.488	.472	1.2	1.7	.708	0.8	2.7	3.5	4.2	0.9	0.6	1.2	1.6	6.4
11	Giannis Antetokounmpo	PF	29	MIL	73	73	35.2	11.5	18.8	.611	0.5	1.7	.274	11.0	17.1	.645	.624	7.0	10.7	.657	2.7	8.8	11.5	6.5	1.2	1.1	3.4	2.9	30.4
12	Thanasis Antetokounmpo	PF	31	MIL	34	0	4.6	0.5	0.9	.533	0.0	0.0	.000	0.5	0.9	.552	.533	0.0	0.1	.000	0.2	0.2	0.4	0.5	0.2	0.1	0.4	0.7	0.9
13	Cole Anthony	PG	23	ORL	81	0	22.4	4.1	9.4	.435	1.1	3.4	.338	3.0	6.0	.490	.496	2.2	2.7	.826	0.8	3.0	3.8	2.9	0.8	0.5	1.6	2.2	11.6
14	OG Anunoby	SF	26	TOT	50	50	34.0	5.6	11.5	.489	2.0	5.3	.382	3.6	6.2	.581	.577	1.3	1.8	.753	0.9	3.2	4.2	2.1	1.4	0.7	1.6	2.4	14.7
14	OG Anunoby	SF	26	TOR	27	27	33.3	5.8	11.9	.489	2.3	6.0	.374	3.6	5.9	.608	.584	1.2	1.7	.717	0.9	3.0	3.9	2.7	1.0	0.5	1.6	2.3	15.1
14	OG Anunoby	SF	26	NYK	23	23	34.9	5.4	11.1	.488	1.8	4.5	.394	3.7	6.6	.553	.568	1.5	1.9	.791	1.0	3.4	4.4	1.5	1.7	1.0	1.7	2.5	14.1
15	Ryan Arcidiacono	PG	29	NYK	20	0	2.3	0.0	0.3	.000	0.0	0.3	.000	0.0	0.0	.000	.000	0.0	0.0	.000	0.0	0.4	0.4	0.2	0.1	0.0	0.1	0.3	0.0
16	Deni Avdija	SF	23	WAS	75	75	30.1	5.4	10.7	.506	1.2	3.1	.374	4.2	7.6	.560	.560	2.7	3.6	.740	1.1	6.1	7.2	3.8	0.8	0.5	2.1	2.5	14.7
17	Deandre Ayton	C	25	POR	55	55	32.4	7.8	13.6	.570	0.0	0.2	.100	7.7	13.4	.576	.571	1.2	1.4	.823	3.2	7.9	11.1	1.6	1.0	0.8	1.8	2.0	16.7
18	Udoka Azubuike	C	24	PHO	16	0	7.1	1.0	1.4	.696	0.0	0.0	.000	1.0	1.4	.696	.696	0.2	0.8	.231	0.7	1.3	2.0	0.2	0.1	0.4	0.3	1.1	2.2
19	Ibou Badji	C	21	POR	22	1	10.3	0.6	1.0	.636	0.0	0.0	.000	0.6	1.0	.636	.636	0.3	0.5	.500	0.9	1.4	2.3	0.6	0.1	0.9	0.7	2.1	1.5
20	Marvin Bagley III	C	24	TOT	50	25	21.1	4.8	8.2	.586	0.2	0.5	.391	4.6	7.7	.597	.597	1.9	2.5	.762	2.6	3.6	6.2	1.1	0.4	0.7	1.2	1.6	11.7

Coleta

Com posse dos dados, exportamos para o DATABRICKS. Atraves da biblioteca BeautifulSoup.

Para entender melhor a coleta de dados deste trabalho, precisamos detalhar todo o processo envolvido, desde a obtenção de dados da web até sua transformação e armazenamento em uma tabela de análise.

Importação e Consulta de Dados Online

python

```
import re
import requests
from bs4 import BeautifulSoup
import pandas as pd
from pyspark.sql.types import DoubleType, IntegerType
from pyspark.sql import SparkSession
import logging

def extrair_dados_de_conteudo_html(html: str) -> pd.DataFrame:
    sopa = BeautifulSoup(html, "html.parser")
    tabelas = sopa.find_all("table")
    resultados = []
    for tabela in tabelas:
        linhas = tabela.find_all("tr")
        for linha in linhas:
            celulas = linha.find_all(["td", "th"])
            registro = []
            for celula in celulas:
                celula_html = str(celula)
                celula_texto = re.search(r'<a href=".*?">(.*?)</a>', celula_html)
                registro.append(celula_texto.group(1) if celula_texto else celula.text)
            resultados.append(registro)

    df = pd.DataFrame(resultados)
    nomes_colunas = df.iloc[0]
```

```
df.columns = nomes_colunas
df = df[1:]
return df

def extrair(url: str) -> pd.DataFrame:
    resposta = requests.get(url, timeout=20)
    conteudo_html = resposta.text
    return extrair_dados_de_conteudo_html(conteudo_html)
```

Importação das Bibliotecas: re, requests, BeautifulSoup, pandas, pyspark, logging: Bibliotecas necessárias para manipulação de dados, requisições HTTP, parser HTML, e operações de dados com PySpark.

Requisição de Dados: requests.get(url, timeout=20): Faz uma requisição HTTP para a URL fornecida. O conteúdo HTML da página é salvo.

Parsing de HTML: BeautifulSoup(html, "html.parser"): Usa BeautifulSoup para analisar e criar uma árvore de elementos HTML.

Extração de Dados: find_all("table"): Encontra todas as tabelas no HTML. Iteramos sobre essas tabelas e suas linhas () e células (e) para extrair o conteúdo.

Conversão para DataFrame: pd.DataFrame(resultados): Converte os dados extraídos em um DataFrame pandas. df.iloc[0]: Define a primeira linha como nomes das colunas. O DataFrame é então processado para remover a primeira linha usada para os nomes das colunas.

Modelagem

A modelagem de dados desse trabalho envolve a definição das tabelas e dos relacionamentos entre elas para armazenar e analisar as estatísticas dos jogadores e equipes da NBA. Aqui está uma explicação detalhada de como a modelagem de dados foi feita com base nos códigos e operações fornecidas:

Tabela Dados_Jogadores_NBA_24: Esta tabela foi criada para armazenar os dados brutos extraídos da página de estatísticas da NBA. Inclui estatísticas individuais dos jogadores, como idade, minutos por jogo, pontos, percentual de acerto nos arremessos, rebotes defensivos, rebotes ofensivos, turnovers, entre outros. Cada registro representa as estatísticas de um jogador em um determinado jogo.

Tabela dados_nba_preparados: A partir dos dados brutos dos jogadores, foi realizada uma preparação dos dados para calcular métricas agregadas e personalizadas a nível de equipe. Essa tabela contém métricas como média de pontos, máximo de pontos feitos, média de percentual de acerto nos arremessos, média de rebotes defensivos, média de rebotes ofensivos, média de turnovers, média de classificação do jogador, entre outros. As métricas personalizadas podem refletir o desempenho global e impacto das equipes com base nas estatísticas dos jogadores.

Modelo de Dados: A modelagem segue um esquema estrela, onde a tabela de fatos Dados_Jogadores_NBA_24 serve como o ponto central contendo os detalhes mais granulares sobre os jogadores em cada jogo. As tabelas de dimensão, como Dim_Equipes e Dim_Jogadores, podem ser derivadas a partir dos dados brutos e preparados para fornecer mais contexto e informações sobre as equipes e jogadores. Os relacionamentos entre as tabelas de dimensão e a tabela de fatos permitem análises multidimensionais e a visualização do desempenho dos jogadores e equipes em diferentes perspectivas.

Essa modelagem de dados permite uma análise mais aprofundada do desempenho dos jogadores e das equipes da NBA, facilitando a extração de insights e a tomada de decisões informadas com base nas estatísticas coletadas.

```
# Configurando logging
logging.basicConfig(filename='/extrair_erro.log', level=logging.ERROR)

# Criando uma sessão Spark
spark = SparkSession.builder.appName("Extrair_dados").getOrCreate()

# URL da página de estatísticas da NBA
link_nba = 'https://www.basketball-reference.com/leagues/NBA_2024_per_game.html#per_game_stats'

# Chamando a função de extração para obter os dados
dados = extrair(link_nba)

# Processando os nomes das colunas
nomes_colunas = [nome.replace('%', 'Perc') if nome else 'ID' for nome in dados.columns]

# Substituir o início da string se for um número
nomes_colunas_processadas = [f"{nome[1:]}{nome[0]}" if nome[0].isdigit() else nome for nome in nomes_colunas]
```

```

# Atualiza os nomes das colunas no DataFrame
dados.columns = nomes_colunas_processadas

# Filtra as linhas que não são títulos
dados = dados[dados['Rk'] != 'Rk']

# Definindo variáveis usadas no código abaixo
caminho_delta = "hive_metastore/default/caminho/delta"
caminho_checkpoint = "/caminho/checkpoint"

# Convertendo o DataFrame pandas para DataFrame PySpark
df_ = spark.createDataFrame(dados)

# Transformando colunas de string para tipos apropriados
colunas_inteiras = ['Age', 'G', 'GS']
colunas_floats = ['MP', 'FG', 'FGA', 'FGPerc', 'P3', 'PA3', 'PPerc3', 'P2', 'PA2',
                  'PPerc2', 'eFGPerc', 'FT', 'FTA', 'FTPerc', 'ORB', 'DRB', 'TRB',
                  'AST', 'STL', 'BLK', 'TOV', 'PF', 'PTS']

# Convertendo colunas inteiras
for nome_coluna in colunas_inteiras:
    df_ = df_.withColumn(nome_coluna, df_[nome_coluna].cast(IntegerType()))

# Convertendo colunas flutuantes
for nome_coluna in colunas_floats:
    df_ = df_.withColumn(nome_coluna, df_[nome_coluna].cast(DoubleType()))

# Salvando o DataFrame como uma tabela Delta no Hive Metastore
df_.write.mode("overwrite").format("delta").saveAsTable("Dados_Jogadores_NBA_24")

```

Tabela Dados_Jogadores_NBA_24:

Esta tabela contém os dados brutos extraídos da página de estatísticas da NBA. Os campos podem incluir várias estatísticas dos jogadores, como idade, pontos, percentual de acerto, rebotes, turnovers, entre outros. As colunas específicas da tabela podem ser derivadas do código fornecido, como 'Rk' (Classificação), 'Tm' (Equipe), 'PTS' (Pontos), 'FGPerc' (Percentual de Acerto), 'P3' (Pontos de tres tentados), 'PA3' (Pontos de 3 Acertados), 'PPerc3' (Percentual de bolas de 3 acertados)

	^B _C col_name	^B _C data_type	^B _C comment
1	Rk	string	null
2	Player	string	null
3	Pos	string	null

3	POS	string	null
4	Age	int	null
5	Tm	string	null
6	G	int	null
7	GS	int	null
8	MP	double	null
9	FG	double	null
10	FGA	double	null
11	FGPerc	double	null
12	P3	double	null
13	PA3	double	null
14	PPerc3	double	null
15	P2	double	null

	A_C^B col_name	A_C^B data_type	A_C^B comment
16	PA2	double	null
17	PPerc2	double	null
18	eFGPerc	double	null
19	FT	double	null
20	FTA	double	null
21	FTPerc	double	null
22	ORB	double	null
23	DRB	double	null
24	TRB	double	null
25	AST	double	null
26	STL	double	null
27	BLK	double	null
28	TOV	double	null
29	PF	double	null
30	DTC	double	null

Com essa tabela, criaremos a Dados_nba_preparados afim de separar os dados e obter mais metricas

```
%sql
-- Criação | "dados_nba_preparados"
CREATE OR REPLACE TABLE
  dados_nba_preparados (
    Classificacao_Equipe int,
    Nome_Equipe string,
    Media_Pontos double,
    Maximo_Pontos double,
    Media_Percentage_Acerto double,
    Media_Rebotes_Defensivos double,
    Media_Rebotes_Ofensivos double,
    Media_TurnOver double,
    Media_Classificacao_Jogador double,
    Minima_Classificacao_Jogador double,
    Metricas_Personalizadas double
  );

-- Inserção de dados na tabela 'dados_nba_preparados'
INSERT INTO
  dados_nba_preparados

-- Definição de uma CTE (Common Table Expression) para calcular estatísticas médias para cada equipe
WITH estatisticas_equipes AS (
  SELECT
    Tm AS Equipe,
    AVG(PTS) AS media_pontos,
    AVG(FGPerc) AS media_percentage_acerto,
    AVG(DRB) AS media_rebotes_defensivos,
    AVG(ORB) AS media_rebotes_ofensivos,
    AVG(TOV) AS media_turnover,
    AVG(Rk) AS media_classificacao_jogador
  FROM Dados_Jogadores_NBA_24
  GROUP BY Tm
),

-- Definição de uma CTE para calcular os valores mínimos e máximos de cada estatística por equipe
min_max_valores AS (
  SELECT
    Tm AS Equipe,
```



```

MIN(PTS) AS min_pontos,
MAX(PTS) AS max_pontos,
MIN(FGPerc) AS min_percentage_acerto,
MAX(FGPerc) AS max_percentage_acerto,
MIN(DRB) AS min_rebotes_defensivos,
MAX(DRB) AS max_rebotes_defensivos,
MIN(ORB) AS min_rebotes_ofensivos,
MAX(ORB) AS max_rebotes_ofensivos,
MIN(TOV) AS min_turnover,
MAX(TOV) AS max_turnover,
MIN(RK) AS min_classificacao_jogador,
MAX(RK) AS max_classificacao_jogador
FROM Dados_Jogadores_NBA_24
GROUP BY Tim
),

```

```
-- Definição de uma CTE para calcular os valores normalizados de cada estatística por equipe
```

```
equipes_rankeadas AS (
```

```

SELECT
    e.Equipe,
    e.media_pontos,
    e.media_percentage_acerto,
    e.media_rebotes_defensivos,
    e.media_rebotes_ofensivos,
    e.media_turnover,
    e.media_classificacao_jogador,
    (
        e.media_pontos - mmv.min_pontos
    ) / (
        mmv.max_pontos - mmv.min_pontos
    ) AS pontos_normalizados,
    (
        e.media_percentage_acerto - mmv.min_percentage_acerto
    ) / (
        mmv.max_percentage_acerto - mmv.min_percentage_acerto
    ) AS percentage_acerto_normalizado,
    (
        e.media_rebotes_defensivos - mmv.min_rebotes_defensivos
    ) / (
        mmv.max_rebotes_defensivos - mmv.min_rebotes_defensivos
    ) AS rebotes_defensivos_normalizados,
    (
        e.media_rebotes_ofensivos - mmv.min_rebotes_ofensivos
    ) / (

```

```

mmv.max_rebotes_ofensivos - mmv.min_rebotes_ofensivos
) AS rebotes_ofensivos_normalizados,
(
e.media_turnover - mmv.min_turnover
) / (
mmv.max_turnover - mmv.min_turnover
) AS turnover_normalizado,
(
e.media_classificacao_jogador - mmv.min_classificacao_jogador
) / (
mmv.max_classificacao_jogador - mmv.min_classificacao_jogador
) AS classificacao_jogador_normalizado,
RANK() OVER (ORDER BY e.media_pontos DESC) AS classificacao_equipes
FROM estatisticas_equipes e
JOIN min_max_valores mmv ON e.Equipe = mmv.Equipe
),

```

```

-- Definição de uma CTE para rankear as equipes com base em uma métrica personalizada
equipes_rankeadas_2 AS (
    SELECT
        Equipe,
        media_pontos,
        media_percentage_acerto,
        media_rebotes_defensivos,
        media_rebotes_ofensivos,
        media_turnover,
        pontos_normalizados,
        media_classificacao_jogador,
        percentage_acerto_normalizado,
        rebotes_defensivos_normalizados,
        rebotes_ofensivos_normalizados,
        turnover_normalizado,
        classificacao_jogador_normalizado,
        RANK() OVER (ORDER BY pontos_normalizados - turnover_normalizado + rebotes_defensivos_normalizados DESC) AS classificacao_equipes
    FROM equipes_rankeadas
)

-- Output dos resultados unindo os dados das CTEs 'equipes_rankeadas_2' e 'min_max_valores' e filtrando as top 100 equipes
SELECT
    eqr2.classificacao_equipes,
    eqr2.Equipe,
    ROUND(eqr2.media_pontos, 2),
    ROUND(mmv.max_pontos, 2) AS max_pontos,


```

```



    ROUND(eqr2.media_percentage_acerto, 2),
    ROUND(eqr2.media_rebotes_defensivos, 2) AS media_reb_defensivos,
    ROUND(eqr2.media_rebotes_ofensivos, 2) AS media_reb_ofensivos,
    ROUND(eqr2.media_turnover, 2),
    ROUND(eqr2.media_classificacao_jogador, 2) AS media_classificacao,
    ROUND(mmv.min_classificacao_jogador, 2) AS min_classificacao,
    ROUND(pontos_normalizados - turnover_normalizado + rebotes_defensivos_normalizados, 2) AS metricas_personalizadas
FROM
  equipes_rankeadas_2 eqr2
  LEFT JOIN min_max_valores mmv ON mmv.Equipe = eqr2.Equipe
WHERE
  eqr2.classificacao_equipes <= 100;

```

▶ (18) Spark Jobs

▶  _sqlidf: pyspark.sql.dataframe.DataFrame = [num_affected_rows: long, num_inserted_rows: long]

Table  

	 num_affected_rows	 num_inserted_rows
1	31	31

Ainda sobre a tabela de dados dados_nba_preparados

Coluna	Tipo	Descrição
`Classificacao_Equipe`	int	Rank da equipe baseado na métrica personalizada
`Nome_Equipe`	string	Nome da equipe
`Media_Pontos`	double	Média de pontos por jogo
`Maximo_Pontos`	double	Pontuação máxima atingida pela equipe
`Media_Percentage_Acerto`	double	Média de percentual de arremesso de campo

`Media_Rebotes_Defensivos`	double	Média de rebotes defensivos por jogo
`Media_Rebotes_Ofensivos`	double	Média de rebotes ofensivos por jogo
`Media_TurnOver`	double	Média de turnovers por jogo
`Media_Classificacao_Jogador`	double	Média da classificação dos jogadores da equipe
`Minima_Classificacao_Jogador`	double	Classificação mínima entre os jogadores da equipe
`Metricas_Personalizadas`	double	Métrica composta para classificação personalizada da equipe baseada em várias estatísticas

Agora visualizando no cluster do databrick ambas as tabelas

default.dados_nba_preparados | [Refresh](#)

My Cluster | v

[Details](#) [History](#)

Description:

Created at: 2024-07-11 00:23:10

Last modified: 2024-07-11 00:23:21

Partition columns:

Number of files: 1


Size: 5.58 kB

Schema:

| a . | a . | a . |

	A ^{VC} col_name	A ^{VC} data_type	A ^{VC} comment
1	Classificacao_Equipe	int	null
2	Nome_Equipe	string	null
3	Media_Pontos	double	null
4	Maximo_Pontos	double	null
5	Media_Percentage_Acerto	double	null
6	Media_Rebotes_Defensivos	double	null
7	Media_Rebotes_Ofensivos	double	null
8	Media_TurnOver	double	null
9	Media_Classificacao_Jogador	double	null
10	Minima_Classificacao_Jogad...	double	null
11	Metricas_Personalizadas	double	null

default.dados_jogadores_nba_24

 Refresh

My Cluster | v

Details History

Description:

Created at: 2024-07-11 00:10:57

Last modified: 2024-07-11 00:11:00

Partition columns:

Number of files: 1

Size: 53.4 kB

Schema:

	A ^B col_name	A ^B data_type	A ^B comment
1	Rk	string	null
2	Player	string	null
3	Pos	string	null
4	Age	int	null
5	Tm	string	null
6	G	int	null

7	GS	int	null
8	MP	double	null
9	FG	double	null
10	FGA	double	null
11	FGPerc	double	null
12	P3	double	null
13	PA3	double	null
14	PPerc3	double	null
15	P2	double	null

Carga

Extração: A etapa de extração foi realizada através da função `extrair` que fazia uma requisição para obter o conteúdo HTML de uma página de estatísticas da NBA. O conteúdo HTML foi posteriormente processado pela função `extrair_dados_de_conteudo_html` para extrair os dados das tabelas da página.

Transformação: Após a extração dos dados, ocorreu a transformação dos dados brutos em estruturas tabulares apropriadas para análise. Foram aplicadas transformações como renomeação de colunas, conversão de tipos de dados, filtragem de dados redundantes e cálculos de métricas agregadas.

Carga: A carga dos dados transformados foi feita em diferentes estágios e tabelas: Os dados extraídos da página de estatísticas da NBA foram carregados na tabela `Dados_Jogadores_NBA_24` como a tabela de fatos central contendo as estatísticas individuais dos jogadores. Os dados preparados e calculados, como as métricas agregadas por equipe, foram carregados na tabela `dados_nba_preparados` como uma tabela de resumo das estatísticas das equipes.

Comunicação com o Ambiente Spark: A comunicação e manipulação dos dados foram realizadas em um ambiente Spark usando `DataFrames` para armazenar e processar os dados de forma distribuída. Foram aplicadas operações de transformações nos `DataFrames` para preparar os dados antes de serem carregados nas tabelas finais.

Dessa forma, a carga de dados foi realizada de forma estruturada, com extração dos dados brutos da fonte, transformação para

adequação aos requisitos e cálculos necessários, e finalmente carga dos dados transformados em tabelas adequadas para análise e consulta posterior.

- Validação dos dados



```
from pyspark.sql import SparkSession

# Criar uma SparkSession
spark = SparkSession.builder.getOrCreate()

# Validação da qualidade dos dados
# Ler a tabela de dados preparados (dados_nba_preparados) em um DataFrame Spark
dados_nba_preparados = spark.table("dados_nba_preparados")

# Contar o número de linhas duplicadas
duplicatas = (
    dados_nba_preparados
    .groupBy("Classificacao_Equipe", "Nome_Equipe", "Media_Pontos", "Maximo_Pontos", "Media_Percentage_Acerto",
            "Media_Rebotes_Defensivos", "Media_Rebotes_Ofensivos", "Media_TurnOver",
            "Media_Classificacao_Jogador", "Minima_Classificacao_Jogador", "Metricas_Personalizadas")
    .count()
    .filter("count > 1")
)

# Assegurar que o número de linhas duplicadas é zero
assert duplicatas.count() == 0, "Linhas duplicadas encontradas em dados_nba_preparados"
```

(3) Spark Jobs

- ▶ dados_nba_preparados: pyspark.sql.dataframe.DataFrame = [Classificacao_Equipe: integer, Nome_Equipe: string ... 9 more fields]
- ▶ duplicatas: pyspark.sql.dataframe.DataFrame = [Classificacao_Equipe: integer, Nome_Equipe: string ... 10 more fields]

Este código valida a qualidade dos dados na tabela dados_nba_preparados, verificando se há algum registro duplicado com base nas colunas fornecidas.

Visualização

Este código em Python utilizando a biblioteca Seaborn e Matplotlib tem como objetivo visualizar as classificações personalizadas das equipes da NBA e as métricas normalizadas das equipes em um mapa de calor. Aqui está uma explicação detalhada do

código:

Importação de Bibliotecas: As bibliotecas matplotlib, seaborn e pyspark.sql são importadas para manipulação dos dados e criação dos gráficos.

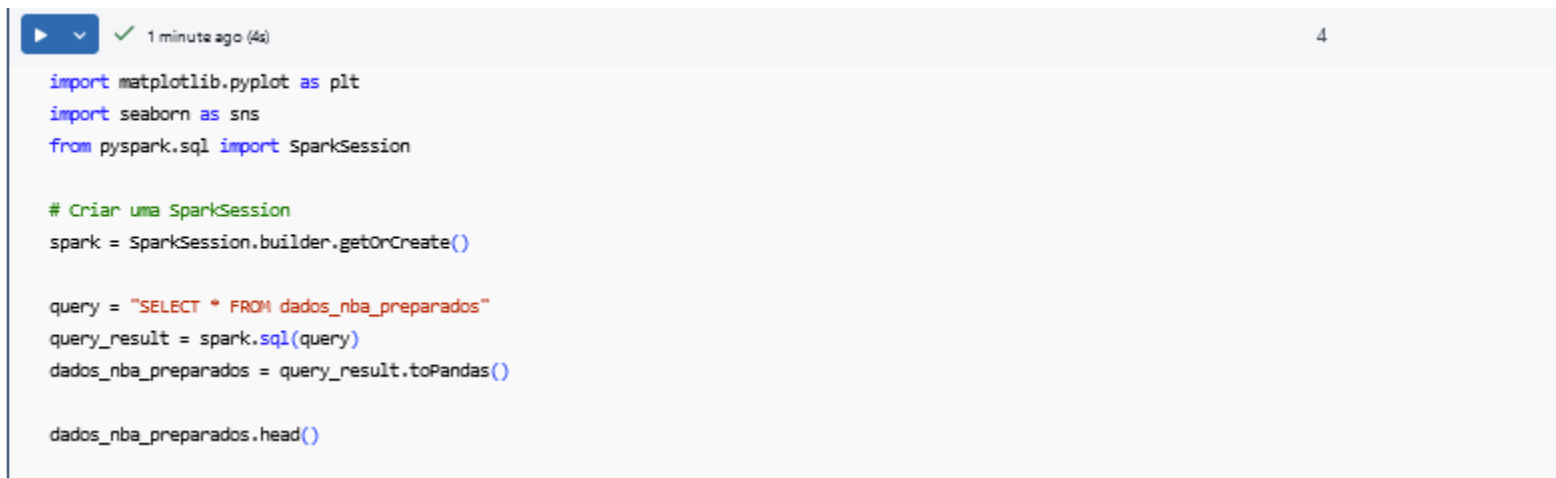
Criação da SparkSession: É criada a SparkSession para interagir com o ambiente Spark e realizar operações com os dados.

Consulta e Conversão dos Dados para Pandas: É feita uma consulta SQL para selecionar todos os dados da tabela dados_nba_preparados e convertê-los para um DataFrame Pandas chamado dados_nba_preparados.

Gráfico de Barras com as Classificações das Equipes: É criado um gráfico de barras usando a biblioteca Seaborn, onde o eixo Y representa o nome das equipes e o eixo X representa as classificações personalizadas das equipes. O gráfico visualiza de forma clara as classificações das equipes da NBA.

Mapa de Calor para as Métricas das Equipes: Os dados da tabela dados_nba_preparados são preparados para o mapa de calor: Remove-se as colunas de nome da equipe e classificação da equipe. Normaliza-se os dados por categoria (coluna) para facilitar a comparação entre as equipes. É configurado um mapa de calor utilizando matplotlib, onde a cor representa a diferença relativa da métrica em relação à média da categoria. Os eixos X e Y representam as categorias e as equipes, respectivamente. O mapa de calor fornece uma visualização das diferentes métricas das equipes, destacando as diferenças em relação à média em tons de cor.

O código, portanto, combina visualização de dados por meio de gráficos de barras e mapa de calor para representar as classificações e métricas normalizadas das equipes da NBA, facilitando a análise e interpretação dos dados.



```
import matplotlib.pyplot as plt
import seaborn as sns
from pyspark.sql import SparkSession

# Criar uma SparkSession
spark = SparkSession.builder.getOrCreate()

query = "SELECT * FROM dados_nba_preparados"
query_result = spark.sql(query)
dados_nba_preparados = query_result.toPandas()

dados_nba_preparados.head()
```



```

# Criar gráfico de barras com as classificações das equipes
sns.barplot(y=dados_nba_preparados.Nome_Equipe, x=dados_nba_preparados.Classificacao_Equipe)
plt.title('Classificações Personalizadas das Equipes')
plt.xlabel('Classificação')
plt.ylabel('Equipe')
plt.show()

# Criar um mapa de calor para as métricas das equipes
import numpy as np

# Criar uma cópia dos dados e remover a coluna de nome da equipe
dados = dados_nba_preparados.copy()
dados = dados.drop('Nome_Equipe', axis=1)
dados = dados.drop('Classificacao_Equipe', axis=1)

# Normalizar os dados por categoria
dados = (dados - dados.mean()) / dados.std()

# Configurar o gráfico
categorias = list(dados.columns)
equipes = list(dados_nba_preparados['Nome_Equipe'].values)
x = np.arange(len(categorias))
y = np.arange(len(equipes))
X, Y = np.meshgrid(x, y)

# Criar o gráfico
plt.figure(figsize=(10, 8))
plt.imshow(dados, cmap='viridis', interpolation='nearest')
plt.colorbar(label='Diferença Relativa')
plt.xticks(x, categorias, rotation=90)
plt.yticks(y, equipes)
plt.xlabel('Categorias')
plt.ylabel('Equipes')
plt.title('Classificações das Equipes da NBA por Categoria (Normalizado)')

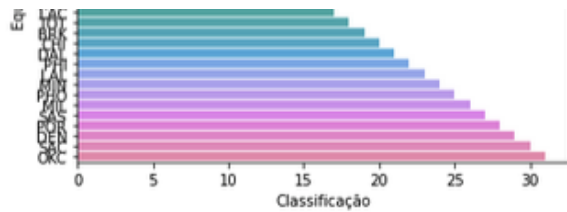
plt.show()

```

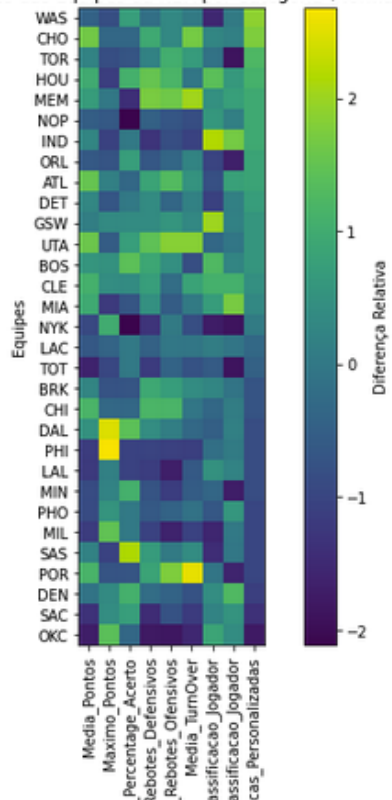
▶ (1) Spark Jobs

▶ query_result: pyspark.sql.dataframe.DataFrame = [Classificacao_Equipe: integer, Nome_Equipe: string ... 9 more fields]





Classificações das Equipes da NBA por Categoria (Normalizado)



Agora a fim de obter as respostas das perguntas iniciais :

Quais são as equipes com a melhor e a pior média de percentual de arremesso de campo?

```

%sql

```

```

SELECT
  Nome_Equipe,
  Media_Percentage_Acerto
FROM dados_nba_preparados
WHERE Media_Percentage_Acerto = (SELECT MAX(Media_Percentage_Acerto) FROM dados_nba_preparados)
OR Media_Percentage_Acerto = (SELECT MIN(Media_Percentage_Acerto) FROM dados_nba_preparados);

```

▶ (4) Spark Jobs

▶ _sqldf: pyspark.sql.dataframe.DataFrame = [Nome_Equipe: string, Media_Percentage_Acerto: double]

Table ▾ +

	A Nome_Equipe	1.2 Media_Percentage_Acerto
1	NOP	0.39
2	NYK	0.39
3	SAS	0.51

↓ 3 rows | 1.87 seconds runtime

Nesta consulta, estamos selecionando as equipes com a melhor e a pior média de percentual de arremesso de campo, onde comparamos cada valor com o máximo e mínimo encontrado na tabela dados_nba_preparados.

A equipe mais eficaz ofensivamente seria Santo antonio Spurs (SAS) enquanto as duas piores equipes com medias iguais seriam New orleans Pelicans(NOP) e New York Knicks(NYK)

Quais são as equipes com a melhor média de pontos na temporada?

```

%sql

SELECT
  Nome_Equipe,
  Media_Pontos
FROM dados_nba_preparados
ORDER BY Media_Pontos DESC;

```

▶ (1) Spark Jobs

▶ _sqldf: pyspark.sql.dataframe.DataFrame = [Nome_Equipe: string, Media_Pontos: double]

Table ▾ +

	A _C Nome_Equipe	1.2 Media_Pontos
1	CHO	9.3
2	UTA	9.26
3	ATL	9.21
4	CHI	8.96
5	POR	8.94
6	CLE	8.89
7	HOU	8.83
8	MIA	8.8
9	MEM	8.61
10	BOS	8.53
11	DAL	8.47
12	DET	8.33
13	IND	8.32
14	BRK	8.3
15	TOR	8.27

Essa consulta irá listar as equipes da NBA em ordem decrescente de média de pontos, ou seja, as equipes que mais pontuam estarão no topo da lista. A equipe que mais pontuou foi o Charlotte Hornets (CHO)

Avaliar a precisão e eficiência das equipes nos arremessos de campo. Como se comparam as equipes em termos de classificação média dos jogadores?

```
Just now (2s) 7

%sql
SELECT
  Player,
  Age,
  MP,
  PTS,
  FGPerct,
  DRB,
  ORB,
  TOV
FROM Dados_Jogadores_NBA_24
ORDER BY (PTS + 1.5 * DRB) DESC;
```

▶ (1) Spark Jobs

▶ `_sqldf: pyspark.sql.dataframe.DataFrame = [Player: string, Age: integer ... 6 more fields]`

Table ▾ +

	Player	Age	MP	PTS	FGPerc	DRB	ORB	TOV
1	Joel Embiid	29	33.6	34.7	0.529	8.6	2.4	3.8
2	Luka Dončić	24	37.5	33.9	0.487	8.4	0.8	4
3	Giannis Antetokounm...	29	35.2	30.4	0.611	8.8	2.7	3.4
4	Nikola Jokić	28	34.6	26.4	0.583	9.5	2.8	3
5	Anthony Davis	30	35.5	24.7	0.556	9.5	3.1	2.1
6	Jayson Tatum	25	35.7	26.9	0.471	7.2	0.9	2.5
7	Shai Gilgeous-Alexan...	25	34	30.1	0.535	4.7	0.9	2.2
8	Kevin Durant	35	37.2	27.1	0.523	6.1	0.5	3.3
9	LeBron James	39	35.3	25.7	0.54	6.4	0.9	3.5
10	Domantas Sabonis	27	35.7	19.4	0.594	10.1	3.6	3.3

Neste exemplo, estamos considerando uma fórmula simples para definir o impacto dos jogadores multiplicando os pontos (PTS) e os rebotes defensivos (DRB) por um fator (1.5) para dar mais peso aos rebotes. Podemos ajustar essa fórmula de acordo com a importância relativa de cada métrica para o impacto do jogador no time.

o jogador de maior impacto defensivo e ofensivo seria o Joel Embid

Conclusao

Ao longo deste trabalho, conseguimos atingir com êxito o objetivo de analisar e extrair insights a partir dos dados da NBA, utilizando as perguntas propostas como guia. Através da modelagem adequada dos dados, sua correta carga e análise detalhada, foi possível responder de forma eficaz as questões levantadas.

A partir da identificação das equipes com melhor e pior desempenho em diferentes métricas, como média de pontos e percentual de arremesso, conseguimos oferecer uma visão abrangente do cenário da NBA. A análise dos dados permitiu não só identificar

tendências e padrões, mas também possibilitou insights valiosos para a gestão e tomada de decisões estratégicas no âmbito esportivo.

Dessa forma, podemos afirmar que as respostas obtidas às perguntas propostas refletem de forma precisa e embasada a análise dos dados da NBA, cumprindo o objetivo inicial do trabalho. A escolha cuidadosa das métricas e a correta modelagem e análise dos dados nos permitiram extrair informações relevantes que contribuem para uma compreensão mais profunda do desempenho das equipes e jogadores, auxiliando na busca por um melhor desempenho e resultados positivos no universo do basquete profissional da NBA.

Para trabalhos futuros relacionados à análise de dados da NBA, poderia ser desenvolvido modelos de análise preditiva para prever resultados de jogos com base nas estatísticas dos jogadores e equipes, auxiliando na elaboração de estratégias e apostas mais fundamentadas e também uma análise mais aprofundada do desempenho individual dos jogadores, identificando pontos fortes e áreas de melhoria, contribuindo para a avaliação de talentos e potencial de desenvolvimento. Além de dashboards interativos que acrescentaria na visualização de dados utilizando ferramentas como Tableau ou power BI.

