PROJETO ANALYTICS ENGINEER

Autor da Solução: Matheus Adão de Souza e Silva

Contexto:

Link: https://github.com/clicksign/data-analytics-engineer-test

De posse do link, entenda o repositório e se atente ao contexto em que está envolvido. No repositório você encontrará dentro do diretório data, em um arquivo compactados data.rar os arquivos .csv disponiblizados como sample de dados, esses são os únicos e exclusivos dados para a realização desse teste. Boa sorte!

Questão 1 - Utilize Python

Como primeiro passo do desafio, faça a leitura/importação desses arquivos via python e realize o upload/carregamento desses dados em um banco **SQLite**.

- Crie um schema chamado: **test_analytics_engineer**
- Crie as tabelas com os mesmos nomes dos arquivos .csv
- Respeite a tipagem e os nomes das colunas dos arquivos .csv

Questão 2 - Utilize SQL

Estabeleça uma relação entre as tabelas **Player** e **Player_Attributes** em uma nova tabela, chamada **'Player_Attributes_Modified'**, onde cada key do json é uma nova coluna. Faça o mesmo para a relação **Team** e **Team_Attributes**, como o nome para a tabela **'Team_Attributes_Modified'**

Crie uma tabela chamada **Match_Modified** cuja coluna seja representada como JSON, onde as chaves precisam ser referentes às colunas da tabela Match, sendo elas : **id,match_api_id, home_team_api_id, away_team_api_id .**

Questão 3 - Utilize SQL ou Python

Faça uma análise exploratória dos dados no sentido de validar a qualidade dos dados destes datasets. Use sua criatividade e imaginação para buscar "sujeiras" na base de dados.

Lembre-se que queremos gerar insights com dados, então realize relações com tabelas que nos forneçam alguma informação relevante para os dados tratados analisados. Crie análises exploratórias dos dados.

Exemplos:

- Qual a proporção entre jogadores destros e canhotos? Quais os seus nomes?
- Qual o nome do país com maior saldo de gols.

Questão 4 - Utilize SQL

Encontre uma relação de dados entre as tabelas League, Country, Team_Attributes e Player, crie uma nova tabela chamada **'Relations'**

Exemplo:

Esquematize uma relação entre a **altura** e **peso** dos jogadores de forma isolada, com seu rendimento.

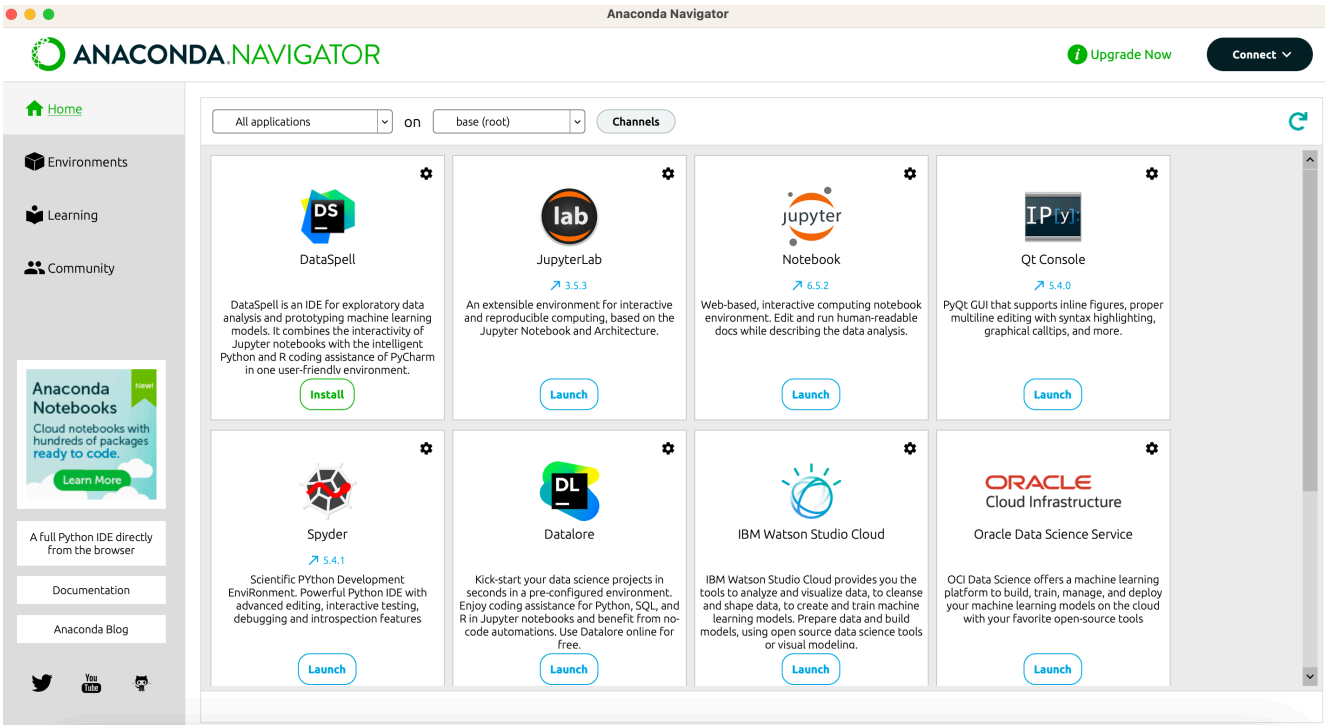
Dica: Criar 3 intervalos entre os 6 valores com maiores contagens para estes dois parâmetros.

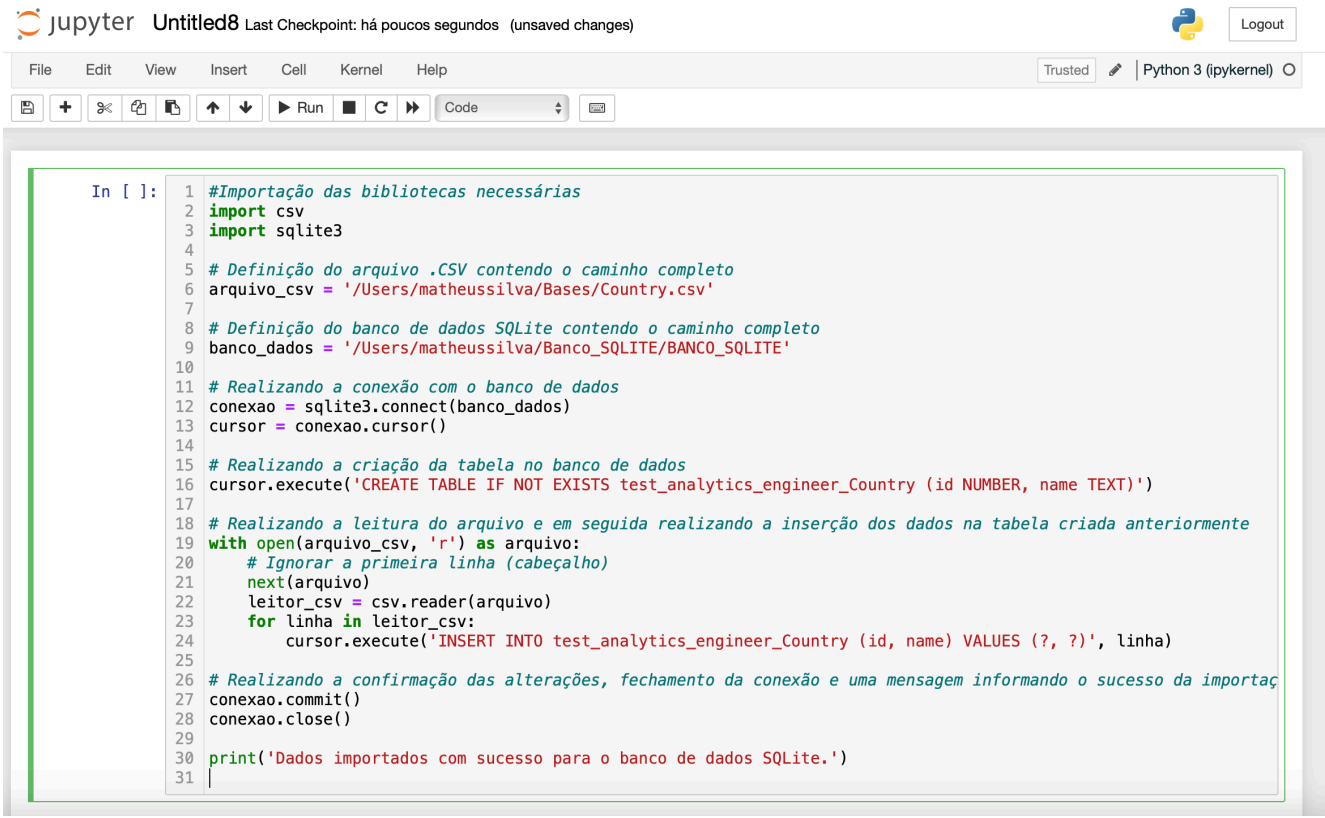
Questão 5 - Utiize SQL-CTE

Semanalmente o gerente da Fifa solicita a média de gols dos times mandantes, porém como você gosta de entregar mais do que lhe pedem, você resolveu montar uma CTE para entregar outras métricas para seu gerente. Como você faria, em SQL, para salvar ou automatizar essa query? Envie-nos seu código.

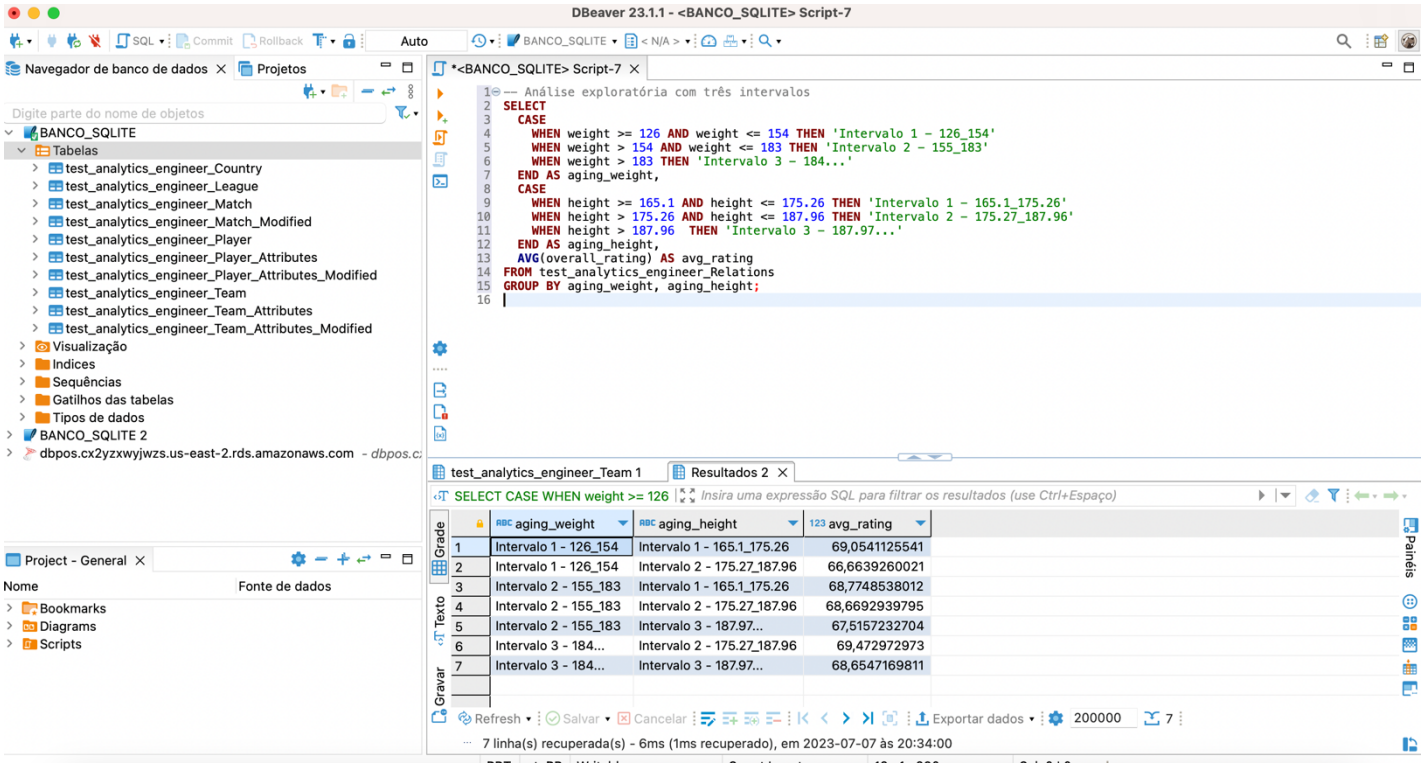
Bônus: realize a criação de um projeto no DBT e faça análises exploratórias mais aprofundadas.

Projeto desenvolvido em Python e SQL, para as etapas de Python foi utilizado como apoio a ferramenta Anaconda juntamente com o Jupyter.





Para bancos de dados foi utilizado o banco de dados SQLite juntamente com a ferramenta DBeaver.



Para análise exploratória foi desenvolvido um dashboard .

Link :
https://app.powerbi.com/view?r=eyJrIjoiODMyMmMyMTQtNmZiNC00MDRiLTk3ZmQtM2ZmMjcyMWUzNDA1IiwidCI6IjE0Y2JkNWE3LWVjOTQtNDZiYS1iMzE0LWNjMGZjOTcyYTE2MSIsImMiOjh9

PROPOSTA DE SOLUÇAO – QUESTÃO 01:

```
#Importação das bibliotecas necessárias
import csv
import sqlite3

# Definição do arquivo .CSV contendo o caminho completo
arquivo_csv = '/Users/matheussilva/Bases/Country.csv'

# Definição do banco de dados SQLite contendo o caminho completo
banco_dados = '/Users/matheussilva/Banco_SQLITE/BANCO_SQLITE'

# Realizando a conexão com o banco de dados
conexao = sqlite3.connect(banco_dados)
cursor = conexao.cursor()

# Realizando a criação da tabela no banco de dados
cursor.execute('CREATE TABLE IF NOT EXISTS
test_analytics_engineer_Country (id NUMBER, name TEXT)')

# Realizando a leitura do arquivo e em seguida realizando a inserção dos
dados na tabela criada anteriormente
with open(arquivo_csv, 'r') as arquivo:
    # Ignorar a primeira linha (cabeçalho)
    next(arquivo)
    leitor_csv = csv.reader(arquivo)
    for linha in leitor_csv:
        cursor.execute('INSERT INTO test_analytics_engineer_Country (id,
name) VALUES (?, ?)', linha)

# Realizando a confirmação das alterações, fechamento da conexão e uma
mensagem informando o sucesso da importação
conexao.commit()
conexao.close()

print('Dados importados com sucesso para o banco de dados SQLite.')
```

```
#Importação das bibliotecas necessárias
import csv
import sqlite3

# Definição do arquivo .CSV contendo o caminho completo
arquivo_csv = '/Users/matheussilva/Bases/League.csv'

# Definição do banco de dados SQLite contendo o caminho completo
banco_dados = '/Users/matheussilva/Banco_SQLITE/BANCO_SQLITE'

# Realizando a conexão com o banco de dados
conexao = sqlite3.connect(banco_dados)
cursor = conexao.cursor()

# Realizando a criação da tabela no banco de dados
cursor.execute('CREATE TABLE IF NOT EXISTS test_analytics_engineer_League
(id INT, country_id INT, name TEXT)')
```

```
# Realizando a leitura do arquivo e em seguida realizando a inserção dos
dados na tabela criada anteriormente
with open(arquivo_csv, 'r') as arquivo:
    # Ignorar a primeira linha (cabeçalho)
    next(arquivo)
    leitor_csv = csv.reader(arquivo)
    for linha in leitor_csv:
        cursor.execute('INSERT INTO test_analytics_engineer_League (id,
country_id, name) VALUES (?,?,?)', linha)

# Realizando a confirmação das alterações, fechamento da conexão e uma
mensagem informando o sucesso da importação
conexao.commit()
conexao.close()

print('Dados importados com sucesso para o banco de dados SQLite.')
```

```
#Importação das bibliotecas necessárias
import csv
import sqlite3

# Definição do arquivo .CSV contendo o caminho completo
arquivo_csv = '/Users/matheussilva/Bases/Match.csv'

# Definição do banco de dados SQLite contendo o caminho completo
banco_dados = '/Users/matheussilva/Banco_SQLITE/BANCO_SQLITE'

# Realizando a conexão com o banco de dados
conexao = sqlite3.connect(banco_dados)
cursor = conexao.cursor()

# Realizando a criação da tabela no banco de dados
cursor.execute('CREATE TABLE IF NOT EXISTS test_analytics_engineer_Match
(id INT,country_id INT,league_id INT,season TEXT,stage INT,date
DATE,match_api_id INT,home_team_api_id INT,away_team_api_id
INT,home_team_goal INT,away_team_goal INT,home_player_X1
INT,home_player_X2 INT,home_player_X3 INT,home_player_X4
INT,home_player_X5 INT,home_player_X6 INT,home_player_X7
INT,home_player_X8 INT,home_player_X9 INT,home_player_X10
INT,home_player_X11 INT,away_player_X1 INT,away_player_X2
INT,away_player_X3 INT,away_player_X4 INT,away_player_X5
INT,away_player_X6 INT,away_player_X7 INT,away_player_X8
INT,away_player_X9 INT,away_player_X10 INT,away_player_X11
INT,home_player_Y1 INT,home_player_Y2 INT,home_player_Y3
INT,home_player_Y4 INT,home_player_Y5 INT,home_player_Y6
INT,home_player_Y7 INT,home_player_Y8 INT,home_player_Y9
INT,home_player_Y10 INT,home_player_Y11 INT,away_player_Y1
INT,away_player_Y2 INT,away_player_Y3 INT,away_player_Y4
INT,away_player_Y5 INT,away_player_Y6 INT,away_player_Y7
INT,away_player_Y8 INT,away_player_Y9 INT,away_player_Y10
INT,away_player_Y11 INT,home_player_1 INT,home_player_2 INT,home_player_3
INT,home_player_4 INT,home_player_5 INT,home_player_6 INT,home_player_7
INT,home_player_8 INT,home_player_9 INT,home_player_10 INT,home_player_11
INT,away_player_1 INT,away_player_2 INT,away_player_3 INT,away_player_4
INT,away_player_5 INT,away_player_6 INT,away_player_7 INT,away_player_8
INT,away_player_9 INT,away_player_10 INT,away_player_11 INT,goal
```

```
TEXT,shoton TEXT,shotoff TEXT,foulcommit TEXT,card TEXT,cross TEXT,corner
TEXT,possession TEXT,B365H NUMERIC,B365D NUMERIC,B365A NUMERIC,BWH
NUMERIC,BWD NUMERIC,BWA NUMERIC,IWH NUMERIC,IWD NUMERIC,IWA NUMERIC,LBH
NUMERIC,LBD NUMERIC,LBA NUMERIC,PSH NUMERIC,PSD NUMERIC,PSA NUMERIC,WHH
NUMERIC,WHD NUMERIC,WHA NUMERIC,SJH NUMERIC,SJD NUMERIC,SJA NUMERIC,VCH
NUMERIC,VCD NUMERIC,VCA NUMERIC,GBH NUMERIC,GBD NUMERIC,GBA NUMERIC,BSH
NUMERIC,BSD NUMERIC,BSA NUMERIC)')

# Realizando a leitura do arquivo e em seguida realizando a inserção dos
dados na tabela criada anteriormente
with open(arquivo_csv, 'r') as arquivo:
    # Ignorar a primeira linha (cabeçalho)
    next(arquivo)
    leitor_csv = csv.reader(arquivo)
    for linha in leitor_csv:
        cursor.execute('INSERT INTO test_analytics_engineer_Match
(id,country_id,league_id,season,stage,date,match_api_id,home_team_api_id,
away_team_api_id,home_team_goal,away_team_goal,home_player_X1,home_player
_X2,home_player_X3,home_player_X4,home_player_X5,home_player_X6,home_play
er_X7,home_player_X8,home_player_X9,home_player_X10,home_player_X11,away_
player_X1,away_player_X2,away_player_X3,away_player_X4,away_player_X5,awa
y_player_X6,away_player_X7,away_player_X8,away_player_X9,away_player_X10,
away_player_X11,home_player_Y1,home_player_Y2,home_player_Y3,home_player_
Y4,home_player_Y5,home_player_Y6,home_player_Y7,home_player_Y8,home_playe
r_Y9,home_player_Y10,home_player_Y11,away_player_Y1,away_player_Y2,away_p
layer_Y3,away_player_Y4,away_player_Y5,away_player_Y6,away_player_Y7,away
_player_Y8,away_player_Y9,away_player_Y10,away_player_Y11,home_player_1,h
ome_player_2,home_player_3,home_player_4,home_player_5,home_player_6,home
_player_7,home_player_8,home_player_9,home_player_10,home_player_11,away_
player_1,away_player_2,away_player_3,away_player_4,away_player_5,away_pla
yer_6,away_player_7,away_player_8,away_player_9,away_player_10,away_playe
r_11,goal,shoton,shotoff,foulcommit,card,cross,corner,possession,B365H,B3
65D,B365A,BWH,BWD,BWA,IWH,IWD,IWA,LBH,LBD,LBA,PSH,PSD,PSA,WHH,WHD,WHA,SJH
,SJD,SJA,VCH,VCD,VCA,GBH,GBD,GBA,BSH,BSD,BSA) VALUES
(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
?,?,?,?,?,?)', linha)

# Realizando a confirmação das alterações, fechamento da conexão e uma
mensagem informando o sucesso da importação
conexao.commit()
conexao.close()

print('Dados importados com sucesso para o banco de dados SQLite.')
```

```
#Importação das bibliotecas necessárias
import csv
import sqlite3

# Definição do arquivo .CSV contendo o caminho completo
arquivo_csv = '/Users/matheussilva/Bases/Player_Attributes.csv'

# Definição do banco de dados SQLite contendo o caminho completo
banco_dados = '/Users/matheussilva/Banco_SQLITE/BANCO_SQLITE'
```

```
# Realizando a conexão com o banco de dados
conexao = sqlite3.connect(banco_dados)
cursor = conexao.cursor()

# Realizando a criação da tabela no banco de dados
cursor.execute('CREATE TABLE IF NOT EXISTS
test_analytics_engineer_Player_Attributes (coluna1 INTEGER,
player_attributes TEXT)')

# Realizando a leitura do arquivo e em seguida realizando a inserção dos
dados na tabela criada anteriormente
with open(arquivo_csv, 'r') as arquivo:
    leitor_csv = csv.reader(arquivo)
    next(leitor_csv)  # Pular a primeira linha (cabeçalho)

    for linha in leitor_csv:
        coluna1 = int(linha[0])  # Valor da primeira coluna (inteiro)
        player_attributes = linha[1].split(',')  # Separar os valores por
vírgula

        cursor.execute('INSERT INTO
test_analytics_engineer_Player_Attributes (coluna1, player_attributes)
VALUES (?, ?)', (coluna1, ','.join(player_attributes)))

# Realizando a confirmação das alterações, fechamento da conexão e uma
mensagem informando o sucesso da importação
conexao.commit()
conexao.close()

print('Dados importados com sucesso para o banco de dados SQLite.')
```

```
#Importação das bibliotecas necessárias
import csv
import sqlite3

# Definição do arquivo .CSV contendo o caminho completo
arquivo_csv = '/Users/matheussilva/Bases/Player.csv'

# Definição do banco de dados SQLite contendo o caminho completo
banco_dados = '/Users/matheussilva/Banco_SQLITE/BANCO_SQLITE'

# Realizando a conexão com o banco de dados
conexao = sqlite3.connect(banco_dados)
cursor = conexao.cursor()

# Realizando a criação da tabela no banco de dados
cursor.execute('CREATE TABLE IF NOT EXISTS test_analytics_engineer_Player
(id INT, player_api_id INT, player_name TEXT, player_fifa_api_id INT,
birthday DATE, height NUMBER, weight INT)')

# Realizando a leitura do arquivo e em seguida realizando a inserção dos
dados na tabela criada anteriormente
with open(arquivo_csv, 'r') as arquivo:
    # Ignorar a primeira linha (cabeçalho)
    next(arquivo)
```

```
    leitor_csv = csv.reader(arquivo)
    for linha in leitor_csv:
        cursor.execute('INSERT INTO test_analytics_engineer_Player (id,
player_api_id, player_name, player_fifa_api_id, birthday, height, weight)
VALUES (?, ?, ?, ?, ?, ?, ?)', linha)

# Realizando a confirmação das alterações, fechamento da conexão e uma
mensagem informando o sucesso da importação
conexao.commit()
conexao.close()

print('Dados importados com sucesso para o banco de dados SQLite.')
```

```
#Importação das bibliotecas necessárias
import csv
import sqlite3

# Definição do arquivo .CSV contendo o caminho completo
arquivo_csv = '/Users/matheussilva/Bases/Team_Attributes.csv'

# Definição do banco de dados SQLite contendo o caminho completo
banco_dados = '/Users/matheussilva/Banco_SQLITE/BANCO_SQLITE'

# Realizando a conexão com o banco de dados
conexao = sqlite3.connect(banco_dados)
cursor = conexao.cursor()

# Realizando a criação da tabela no banco de dados
cursor.execute('CREATE TABLE IF NOT EXISTS
test_analytics_engineer_Team_Attributes (coluna1 INTEGER, team_attributes
TEXT)')

# Realizando a leitura do arquivo e em seguida realizando a inserção dos
dados na tabela criada anteriormente
with open(arquivo_csv, 'r') as arquivo:
    leitor_csv = csv.reader(arquivo)
    next(leitor_csv)  # Pular a primeira linha (cabeçalho)

    for linha in leitor_csv:
        coluna1 = int(linha[0])  # Valor da primeira coluna (inteiro)
        team_attributes = linha[1].split(',')  # Separar os valores por
vírgula

        cursor.execute('INSERT INTO
test_analytics_engineer_Team_Attributes (coluna1, team_attributes) VALUES
(?, ?)', (coluna1, ','.join(team_attributes)))

# Realizando a confirmação das alterações, fechamento da conexão e uma
mensagem informando o sucesso da importação
conexao.commit()
conexao.close()

print('Dados importados com sucesso para o banco de dados SQLite.')
```

```
#Importação das bibliotecas necessárias
import csv
import sqlite3

# Definição do arquivo .CSV contendo o caminho completo
arquivo_csv = '/Users/matheussilva/Bases/Team.csv'

# Definição do banco de dados SQLite contendo o caminho completo
banco_dados = '/Users/matheussilva/Banco_SQLITE/BANCO_SQLITE'

# Realizando a conexão com o banco de dados
conexao = sqlite3.connect(banco_dados)
cursor = conexao.cursor()

# Realizando a criação da tabela no banco de dados
cursor.execute('CREATE TABLE IF NOT EXISTS test_analytics_engineer_Team
(id INT, team_api_id INT, team_fifa_api_id INT, team_long_name TEXT,
team_Short_name TEXT)')

# Realizando a leitura do arquivo e em seguida realizando a inserção dos
dados na tabela criada anteriormente
with open(arquivo_csv, 'r') as arquivo:
    # Ignorar a primeira linha (cabeçalho)
    next(arquivo)
    leitor_csv = csv.reader(arquivo)
    for linha in leitor_csv:
        cursor.execute('INSERT INTO test_analytics_engineer_Team (id,
team_api_id, team_fifa_api_id, team_long_name, team_Short_name) VALUES
(?, ?, ?, ?, ?)', linha)

# Realizando a confirmação das alterações, fechamento da conexão e uma
mensagem informando o sucesso da importação
conexao.commit()
conexao.close()

print('Dados importados com sucesso para o banco de dados SQLite.')
```

PROPOSTA DE SOLUÇAO – QUESTÃO 02:

```
#Importação das bibliotecas necessárias
import json
import sqlite3

# Definição do banco de dados SQLite contendo o caminho completo
banco_dados = '/Users/matheussilva/Banco_SQLITE/BANCO_SQLITE'

# Realizando a conexão com o banco de dados
conexao = sqlite3.connect(banco_dados)
cursor = conexao.cursor()

# Realizando a criação da tabela no banco de dados
cursor.execute('CREATE TABLE IF NOT EXISTS
test_analytics_engineer_Player_Attributes_Modified (coluna1 INTEGER
PRIMARY KEY, '
               'id INTEGER, player_fifa_api_id INTEGER, player_api_id
INT, date DATE, overall_rating NUMERIC, potential NUMERIC, preferred_foot
TEXT, attacking_work_rate TEXT, defensive_work_rate TEXT, crossing
NUMERIC, finishing NUMERIC, heading_accuracy NUMERIC, short_passing
NUMERIC, volleys NUMERIC, dribbling NUMERIC, curve NUMERIC,
free_kick_accuracy NUMERIC, long_passing NUMERIC, ball_control NUMERIC,
acceleration NUMERIC, sprint_speed NUMERIC, agility NUMERIC, reactions
NUMERIC, balance NUMERIC, shot_power NUMERIC, jumping NUMERIC, stamina
NUMERIC, strength NUMERIC, long_shots NUMERIC, aggression NUMERIC,
interceptions NUMERIC, positioning NUMERIC, vision NUMERIC, penalties
NUMERIC, marking NUMERIC, standing_tackle NUMERIC, sliding_tackle
NUMERIC, gk_diving NUMERIC, gk_handling NUMERIC, gk_kicking NUMERIC,
gk_positioning NUMERIC, gk_reflexes NUMERIC)')

# Selecionando os dados da tabela
test_analytics_engineer_Player_Attributes e inserindo na tabela
test_analytics_engineer_Player_Attributes_Modified

cursor.execute('SELECT A.id, B.player_attributes FROM
test_analytics_engineer_Player A INNER JOIN
test_analytics_engineer_Player_Attributes B ON A.id = B.coluna1')
rows = cursor.fetchall()
for row in rows:
    coluna1 = row[0]
    attributes_json = row[1]

    attributes = json.loads(attributes_json)

    id= attributes.get('id')
    player_fifa_api_id= attributes.get('player_fifa_api_id')
    player_api_id= attributes.get('player_api_id')
    date= attributes.get('date')
    overall_rating= attributes.get('overall_rating')
    potential= attributes.get('potential')
    preferred_foot= attributes.get('preferred_foot')
    attacking_work_rate= attributes.get('attacking_work_rate')
    defensive_work_rate= attributes.get('defensive_work_rate')
    crossing= attributes.get('crossing')
    finishing= attributes.get('finishing')
```

```
    heading_accuracy= attributes.get('heading_accuracy')
    short_passing= attributes.get('short_passing')
    volleys= attributes.get('volleys')
    dribbling= attributes.get('dribbling')
    curve= attributes.get('curve')
    free_kick_accuracy= attributes.get('free_kick_accuracy')
    long_passing= attributes.get('long_passing')
    ball_control= attributes.get('ball_control')
    acceleration= attributes.get('acceleration')
    sprint_speed= attributes.get('sprint_speed')
    agility= attributes.get('agility')
    reactions= attributes.get('reactions')
    balance= attributes.get('balance')
    shot_power= attributes.get('shot_power')
    jumping= attributes.get('jumping')
    stamina= attributes.get('stamina')
    strength= attributes.get('strength')
    long_shots= attributes.get('long_shots')
    aggression= attributes.get('aggression')
    interceptions= attributes.get('interceptions')
    positioning= attributes.get('positioning')
    vision= attributes.get('vision')
    penalties= attributes.get('penalties')
    marking= attributes.get('marking')
    standing_tackle= attributes.get('standing_tackle')
    sliding_tackle= attributes.get('sliding_tackle')
    gk_diving= attributes.get('gk_diving')
    gk_handling= attributes.get('gk_handling')
    gk_kicking= attributes.get('gk_kicking')
    gk_positioning= attributes.get('gk_positioning')
    gk_reflexes= attributes.get('gk_reflexes')



    cursor.execute('INSERT INTO
test_analytics_engineer_Player_Attributes_Modified
(coluna1,id,player_fifa_api_id,player_api_id,date,overall_rating,potentia
l,preferred_foot,attacking_work_rate,defensive_work_rate,crossing,finishi
ng,heading_accuracy,short_passing,volleys,dribbling,curve,free_kick_accur
acy,long_passing,ball_control,acceleration,sprint_speed,agility,reactions
,balance,shot_power,jumping,stamina,strength,long_shots,aggression,interc
eptions,positioning,vision,penalties,marking,standing_tackle,sliding_tack
le,gk_diving,gk_handling,gk_kicking,gk_positioning,gk_reflexes) '
                    'VALUES
(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
?,?,?,?,?,?,?)',
(coluna1,id,player_fifa_api_id,player_api_id,date,overall_rating,potentia
l,preferred_foot,attacking_work_rate,defensive_work_rate,crossing,finishi
ng,heading_accuracy,short_passing,volleys,dribbling,curve,free_kick_accur
acy,long_passing,ball_control,acceleration,sprint_speed,agility,reactions
,balance,shot_power,jumping,stamina,strength,long_shots,aggression,interc
eptions,positioning,vision,penalties,marking,standing_tackle,sliding_tack
le,gk_diving,gk_handling,gk_kicking,gk_positioning,gk_reflexes))

# Realizando a confirmação das alterações, fechamento da conexão e uma
mensagem informando o sucesso da importação
conexao.commit()
```

```
conexao.close()

print('Relação estabelecida entre as tabelas. Tabela
test_analytics_engineer_Player_Attributes_Modified criada com sucesso.')

-----------------------------------------------------------------------------------------------------------------------

#Importação das bibliotecas necessárias

import json
import sqlite3

# Definição do banco de dados SQLite contendo o caminho completo
banco_dados = '/Users/matheussilva/Banco_SQLITE/BANCO_SQLITE'

# Realizando a conexão com o banco de dados
conexao = sqlite3.connect(banco_dados)
cursor = conexao.cursor()

# Realizando a criação da tabela no banco de dados
cursor.execute('CREATE TABLE IF NOT EXISTS
test_analytics_engineer_Team_Attributes_Modified (coluna1 INTEGER PRIMARY
KEY, '
               'id INT,team_fifa_api_id INT,team_api_id INT,date
DATE,buildUpPlaySpeed INT,buildUpPlaySpeedClass TEXT,buildUpPlayDribbling
TEXT,buildUpPlayDribblingClass TEXT,buildUpPlayPassing
INT,buildUpPlayPassingClass TEXT,buildUpPlayPositioningClass
TEXT,chanceCreationPassing INT,chanceCreationPassingClass
TEXT,chanceCreationCrossing INT,chanceCreationCrossingClass
TEXT,chanceCreationShooting INT,chanceCreationShootingClass
TEXT,chanceCreationPositioningClass TEXT,defencePressure
INT,defencePressureClass TEXT,defenceAggression
INT,defenceAggressionClass TEXT,defenceTeamWidth
INT,defenceTeamWidthClass TEXT,defenceDefenderLineClass TEXT)')

# Selecionando os dados da tabela test_analytics_engineer_Team e
test_analytics_engineer_Team_Attributes para inserirmos na tabela
test_analytics_engineer_Team_Attributes_Modified

cursor.execute('SELECT A.id, B.team_attributes FROM
test_analytics_engineer_Team A INNER JOIN
test_analytics_engineer_Team_Attributes B ON A.id = B.coluna1')
rows = cursor.fetchall()
for row in rows:
    coluna1 = row[0]
    attributes_json = row[1]

    attributes = json.loads(attributes_json)

    id=attributes.get('id')
    team_fifa_api_id=attributes.get('team_fifa_api_id')
    team_api_id=attributes.get('team_api_id')
    date=attributes.get('date')
    buildUpPlaySpeed=attributes.get('buildUpPlaySpeed')
    buildUpPlaySpeedClass=attributes.get('buildUpPlaySpeedClass')
    buildUpPlayDribbling=attributes.get('buildUpPlayDribbling')
    buildUpPlayDribblingClass=attributes.get('buildUpPlayDribblingClass')
    buildUpPlayPassing=attributes.get('buildUpPlayPassing')
```

```
    buildUpPlayPassingClass=attributes.get('buildUpPlayPassingClass')

buildUpPlayPositioningClass=attributes.get('buildUpPlayPositioningClass')
    chanceCreationPassing=attributes.get('chanceCreationPassing')

chanceCreationPassingClass=attributes.get('chanceCreationPassingClass')
    chanceCreationCrossing=attributes.get('chanceCreationCrossing')

chanceCreationCrossingClass=attributes.get('chanceCreationCrossingClass')
    chanceCreationShooting=attributes.get('chanceCreationShooting')

chanceCreationShootingClass=attributes.get('chanceCreationShootingClass')

chanceCreationPositioningClass=attributes.get('chanceCreationPositioningC
lass')
    defencePressure=attributes.get('defencePressure')
    defencePressureClass=attributes.get('defencePressureClass')
    defenceAggression=attributes.get('defenceAggression')
    defenceAggressionClass=attributes.get('defenceAggressionClass')
    defenceTeamWidth=attributes.get('defenceTeamWidth')
    defenceTeamWidthClass=attributes.get('defenceTeamWidthClass')
    defenceDefenderLineClass=attributes.get('defenceDefenderLineClass')




    cursor.execute('INSERT INTO
test_analytics_engineer_Team_Attributes_Modified
(coluna1,id,team_fifa_api_id,team_api_id,date,buildUpPlaySpeed,buildUpPla
ySpeedClass,buildUpPlayDribbling,buildUpPlayDribblingClass,buildUpPlayPas
sing,buildUpPlayPassingClass,buildUpPlayPositioningClass,chanceCreationPa
ssing,chanceCreationPassingClass,chanceCreationCrossing,chanceCreationCro
ssingClass,chanceCreationShooting,chanceCreationShootingClass,chanceCreat
ionPositioningClass,defencePressure,defencePressureClass,defenceAggressio
n,defenceAggressionClass,defenceTeamWidth,defenceTeamWidthClass,defenceDe
fenderLineClass) '
                     'VALUES
(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)',
(coluna1,id,team_fifa_api_id,team_api_id,date,buildUpPlaySpeed,buildUpPla
ySpeedClass,buildUpPlayDribbling,buildUpPlayDribblingClass,buildUpPlayPas
sing,buildUpPlayPassingClass,buildUpPlayPositioningClass,chanceCreationPa
ssing,chanceCreationPassingClass,chanceCreationCrossing,chanceCreationCro
ssingClass,chanceCreationShooting,chanceCreationShootingClass,chanceCreat
ionPositioningClass,defencePressure,defencePressureClass,defenceAggressio
n,defenceAggressionClass,defenceTeamWidth,defenceTeamWidthClass,defenceDe
fenderLineClass))

# Realizando a confirmação das alterações, fechamento da conexão e uma
mensagem informando o sucesso da importação
conexao.commit()
conexao.close()

print('Relação estabelecida entre as tabelas. Tabela
test_analytics_engineer_Team_Attributes_Modified foi criada com
sucesso.')
```

```
#Importação das bibliotecas necessárias
import sqlite3
import json

# Definição do banco de dados SQLite contendo o caminho completo
banco_dados = '/Users/matheussilva/Banco_SQLITE/BANCO_SQLITE'

# Realizando a conexão com o banco de dados
conexao = sqlite3.connect(banco_dados)
cursor = conexao.cursor()

# Realizando a criação da tabela no banco de dados
cursor.execute('CREATE TABLE IF NOT EXISTS
test_analytics_engineer_Match_Modified (id INTEGER PRIMARY KEY
AUTOINCREMENT, match_json TEXT)')

# Selecionando os dados da tabela Match
cursor.execute('SELECT  id,match_api_id, home_team_api_id,
away_team_api_id FROM test_analytics_engineer_Match')
matches = cursor.fetchall()

# Inserindo os dados na tabela test_analytics_engineer_Match_Modified
for match in matches:
    match_dict = {
        'id': match[0],
        'match_api_id': match[1],
        'home_team_api_id': match[2],
        'away_team_api_id': match[3]
    }
    match_json = json.dumps(match_dict)

    cursor.execute('INSERT INTO test_analytics_engineer_Match_Modified
(match_json) VALUES (?)', (match_json,))

# Realizando a confirmação das alterações, fechamento da conexão e uma
mensagem informando o sucesso da importação
conexao.commit()
conexao.close()

print('Criação das colunas como JSON na tabela
test_analytics_engineer_Match_Modified realizada com sucesso.')
```

PROPOSTA DE SOLUÇAO – QUESTÃO 03:

PRINCIPAIS SUJEIRAS NAS BASES:

```
--Campos vazios
SELECT *
FROM test_analytics_engineer_Country
WHERE ID = ''

--ID duplicado representando o mesmo nome porém escrito em outra língua, além de campos com nomes vazios
SELECT id, COUNT(id) AS Repeticoes_ID
FROM test_analytics_engineer_Country
GROUP BY id
HAVING COUNT(id) > 1;

--Campos com nomes vazios

SELECT name, COUNT(id) AS Repeticoes_ID
FROM test_analytics_engineer_Country
GROUP BY name
HAVING COUNT(name) > 1;

--Problemas em alguns campos no momento da importação da base da API fez com que alguns campos não possuía o valor esperado para o mesmo

SELECT
goal,shoton,shotoff,foulcommit,card,"cross",corner,possession,count(*)
FROM test_analytics_engineer_Match
group by goal,shoton,shotoff,foulcommit,card,"cross",corner,possession

--Cabeçalho faltando para a primeira coluna na base, dessa forma atribuído um nome aleatório

SELECT *
from test_analytics_engineer_Player_Attributes

--Campo team_fifa_api_id com repetições

SELECT team_fifa_api_id, COUNT(team_fifa_api_id) AS Repeticoes_ID
FROM test_analytics_engineer_Team
GROUP BY team_fifa_api_id
HAVING COUNT(team_fifa_api_id) > 1;

--Nas tabelas utilizadas para criar as "_Modified" temos ids que constam na tabela que contém a coluna com o JSON e na tabela correspondente o id não existe

SELECT A.coluna1
FROM test_analytics_engineer_Team_Attributes A
LEFT JOIN test_analytics_engineer_Team B ON A.coluna1 = B.id
WHERE B.id IS NULL;
```

ALGUMAS ANALISES EXPLORÁTORIAS:

```sql
--Quantidade de país e ligas que compõem as bases

SELECT 'Country' AS tabela, COUNT(*) AS quantidade FROM
test_analytics_engineer_Country
UNION ALL
SELECT 'League' AS tabela, COUNT(*) AS quantidade FROM
test_analytics_engineer_League;


--Temporada mais recente que contém na tabela
test_analytics_engineer_Match

SELECT MAX(season) AS "Temporada mais Recente" FROM
test_analytics_engineer_Match;


--Quantidade de gols dentro de casa, fora de casa e média de gols por
país

SELECT sum(home_team_goal) AS QTD_HOME_TEAM_GOAL,
sum(away_team_goal) AS QTD_AWAY_TEAM_GOAL,
sum(goal) QTD_GOAL,
AVG(home_team_goal + away_team_goal) AS MEDIA_GOAL,
B.name AS NAME_COUNTRY

FROM test_analytics_engineer_Match A
LEFT JOIN (SELECT DISTINCT id,
CASE WHEN name = 'Bélgica' THEN 'Belgium'
      WHEN name = 'Inglaterra' THEN 'England'
      WHEN name = 'França' THEN 'France'
      WHEN name = 'Alemanha' THEN 'Germany'
      WHEN name = 'Itália' THEN 'Italy'
ELSE name END AS name
FROM test_analytics_engineer_Country) B ON B.id = A.country_id

GROUP BY B.NAME

--Quantidade de gols dentro de casa, fora de casa e média de gols por
liga

SELECT sum(home_team_goal) AS QTD_HOME_TEAM_GOAL,
sum(away_team_goal) AS QTD_AWAY_TEAM_GOAL,
sum(goal) QTD_GOAL,
AVG(home_team_goal + away_team_goal) AS MEDIA_GOAL,
B.name AS NAME_LEAGUE

FROM test_analytics_engineer_Match A
LEFT JOIN test_analytics_engineer_League B ON A.league_id  = B.id

GROUP BY B.NAME
```

```sql
--Quantidade de gols dentro de casa, fora de casa e média de gols por temporada

SELECT sum(home_team_goal) AS QTD_HOME_TEAM_GOAL,
sum(away_team_goal) AS QTD_AWAY_TEAM_GOAL,
sum(goal) QTD_GOAL,
AVG(home_team_goal + away_team_goal) AS MEDIA_GOAL,
a.season AS SEASON

FROM test_analytics_engineer_Match A

GROUP BY a.season

--Quantidade de gols dentro de casa, fora de casa e média de gols por ano e mês

SELECT sum(home_team_goal) AS QTD_HOME_TEAM_GOAL,
sum(away_team_goal) AS QTD_AWAY_TEAM_GOAL,
sum(goal) QTD_GOAL,
AVG(home_team_goal + away_team_goal) AS MEDIA_GOAL,
strftime('%Y-%m', a.date) AS ANO_MES

FROM test_analytics_engineer_Match A

GROUP BY strftime('%Y-%m', a.date)

--Quantidade de jogadores pela altura ordenado em forma decrescente

SELECT height AS HEIGHT_PLAYER,
COUNT(*) AS QTD_PLAYER
FROM test_analytics_engineer_Player
GROUP BY height
ORDER BY QTD_PLAYER DESC;

--Quantidade de jogadores pelo peso ordenado em forma decrescente

SELECT weight AS WEIGHT_PLAYER,
COUNT(*) AS QTD_PLAYER
FROM test_analytics_engineer_Player
GROUP BY weight
ORDER BY QTD_PLAYER DESC;

--Média da avalição geral dos jogadores por ano

SELECT strftime('%Y', date) AS YEAR,
AVG(overall_rating) AS AVG_RATING
FROM test_analytics_engineer_Player_Attributes_Modified
GROUP BY year
ORDER BY year;
```

```sql
--Média da avalição geral por jogador

SELECT B.player_name AS PLAYER_NAME,
AVG(overall_rating) AS AVG_RATING
FROM test_analytics_engineer_Player_Attributes_Modified A
LEFT JOIN test_analytics_engineer_Player B ON A.player_api_id =
B.player_api_id
GROUP BY B.player_name
ORDER BY B.player_name;


--Quantidade de jogadores destros e canhotos contendo também os nomes

SELECT B.player_name AS PLAYER_NAME,
preferred_foot AS PREFERRED_FOOT,
count(DISTINCT A.player_api_id) AS QTD_PLAYER_API_ID

FROM test_analytics_engineer_Player_Attributes_Modified A
LEFT JOIN test_analytics_engineer_Player B ON A.player_api_id =
B.player_api_id
GROUP BY B.player_name,preferred_foot
ORDER BY B.player_name;

--Potencial médio por jogador

SELECT B.player_name AS PLAYER_NAME,
AVG(potential) AS AVG_POTENTIAL
FROM test_analytics_engineer_Player_Attributes_Modified A
LEFT JOIN test_analytics_engineer_Player B ON A.player_api_id =
B.player_api_id
GROUP BY B.player_name
ORDER BY B.player_name;

--Nível de ataque por jogador

SELECT B.player_name AS PLAYER_NAME,
attacking_work_rate as ATTACKING_WORK_RATE
FROM test_analytics_engineer_Player_Attributes_Modified A
LEFT JOIN test_analytics_engineer_Player B ON A.player_api_id =
B.player_api_id
GROUP BY B.player_name
ORDER BY B.player_name;


--Nível de defesa por jogador

SELECT B.player_name AS PLAYER_NAME,
 defensive_work_rate as DEFENSIVE_WORK_RATE
FROM test_analytics_engineer_Player_Attributes_Modified A
LEFT JOIN test_analytics_engineer_Player B ON A.player_api_id =
B.player_api_id
GROUP BY B.player_name
ORDER BY B.player_name;
```

```sql
--Velocidade das jogadas pelas equipes

SELECT buildUpPlaySpeedClass AS BUILD_UP_PLAY_SPEED_CLASS,
COUNT(distinct team_api_id) AS QTD
FROM test_analytics_engineer_Team_Attributes_Modified
GROUP BY buildUpPlaySpeedClass
ORDER BY QTD DESC;


--Defesa de pressão por equipes

SELECT A.team_long_name AS TEAM_NAME,
B.defencePressure AS DEFENCE_PRESSURE
FROM test_analytics_engineer_Team A
LEFT JOIN test_analytics_engineer_Team_Attributes_Modified B ON
A.team_api_id = B.team_api_id
ORDER BY B.defencePressure DESC;

--Agressão de defesa por equipes

SELECT A.team_long_name AS TEAM_NAME,
B.defenceAggression AS DEFENCE_AGRESSION
FROM test_analytics_engineer_Team A
LEFT JOIN test_analytics_engineer_Team_Attributes_Modified B ON
A.team_api_id = B.team_api_id
ORDER BY B.defenceAggression DESC;
```

PROPOSTA DE SOLUÇAO – QUESTÃO 04:

```sql
-- Criacao da tabela test_analytics_engineer_Relations

CREATE TABLE test_analytics_engineer_Relations AS
SELECT p.player_name, p.height, p.weight, pa.overall_rating
FROM test_analytics_engineer_Player p
JOIN test_analytics_engineer_Player_Attributes_Modified pa ON
p.player_api_id = pa.player_api_id;


-- Análise exploratória com três intervalos
SELECT
  CASE
    WHEN weight >= 126 AND weight <= 154 THEN 'Intervalo 1 - 126_154'
    WHEN weight > 154 AND weight <= 183 THEN 'Intervalo 2 - 155_183'
    WHEN weight > 183 THEN 'Intervalo 3 - 184...'
  END AS aging_weight,
  CASE
    WHEN height >= 165.1 AND height <= 175.26 THEN 'Intervalo 1 -
165.1_175.26'
    WHEN height > 175.26 AND height <= 187.96 THEN 'Intervalo 2 -
175.27_187.96'
    WHEN height > 187.96  THEN 'Intervalo 3 - 187.97...'
  END AS aging_height,
  AVG(overall_rating) AS avg_rating
FROM test_analytics_engineer_Relations
GROUP BY aging_weight, aging_height;
```

PROPOSTA DE SOLUÇAO – QUESTÃO 05:

```
#Aquivo python etapa_5.py
#Importação das bibliotecas necessárias
import pandas as pd
import sqlite3

# Definição do banco de dados SQLite contendo o caminho completo
banco_dados = '/Users/matheussilva/Banco_SQLITE/BANCO_SQLITE'

# Realizando a conexão com o banco de dados
conexao = sqlite3.connect(banco_dados)
cursor = conexao.cursor()


# Executando a consulta SQL 1
query_1 = '''
WITH dados_consolidados AS (
    SELECT SUM(home_team_goal) AS QTD_HOME_TEAM_GOAL,
           SUM(away_team_goal) AS QTD_AWAY_TEAM_GOAL,
           SUM(goal) AS QTD_GOAL,
           home_team_goal + away_team_goal AS QTD_HOME_AWAY,
           A.league_id AS league_id,
           A.country_id AS country_id,
           A.season as SEASON,
           A.date AS DATA
    FROM test_analytics_engineer_Match A
    GROUP BY A.league_id, A.country_id, A.season, A.date
)

-- Visão por Liga

SELECT sum(QTD_HOME_TEAM_GOAL) AS QTD_HOME_TEAM_GOAL
,sum(QTD_AWAY_TEAM_GOAL) AS QTD_AWAY_TEAM_GOAL
,sum(QTD_GOAL) AS QTD_GOAL
,AVG(QTD_HOME_AWAY) AS AVG_HOME_AWAY
,B.name
FROM dados_consolidados A
LEFT JOIN test_analytics_engineer_League B ON B.id = A.league_id
GROUP BY B.name
ORDER BY B.name  ASC

'''

# Executando a consulta SQL 2
query_2 = '''
WITH dados_consolidados AS (
    SELECT SUM(home_team_goal) AS QTD_HOME_TEAM_GOAL,
           SUM(away_team_goal) AS QTD_AWAY_TEAM_GOAL,
           SUM(goal) AS QTD_GOAL,
           home_team_goal + away_team_goal AS QTD_HOME_AWAY,
           A.league_id AS league_id,
           A.country_id AS country_id,
           A.season as SEASON,
           A.date AS DATA
    FROM test_analytics_engineer_Match A
    GROUP BY A.league_id, A.country_id, A.season, A.date
```

```
)

-- Visão por país

SELECT sum(QTD_HOME_TEAM_GOAL) AS QTD_HOME_TEAM_GOAL
,sum(QTD_AWAY_TEAM_GOAL) AS QTD_AWAY_TEAM_GOAL
,sum(QTD_GOAL) AS QTD_GOAL
,AVG(QTD_HOME_AWAY) AS AVG_HOME_AWAY
,C.name
FROM dados_consolidados A
LEFT JOIN (SELECT DISTINCT id,
CASE WHEN name = 'Bélgica' THEN 'Belgium'
      WHEN name = 'Inglaterra' THEN 'England'
      WHEN name = 'França' THEN 'France'
      WHEN name = 'Alemanha' THEN 'Germany'
      WHEN name = 'Itália' THEN 'Italy'
ELSE name END AS name
FROM test_analytics_engineer_Country) C ON C.id = A.country_id
GROUP BY C.name
ORDER BY C.name  ASC


'''

# Executando a consulta SQL 3
query_3 = '''
WITH dados_consolidados AS (
    SELECT SUM(home_team_goal) AS QTD_HOME_TEAM_GOAL,
           SUM(away_team_goal) AS QTD_AWAY_TEAM_GOAL,
           SUM(goal) AS QTD_GOAL,
           home_team_goal + away_team_goal AS QTD_HOME_AWAY,
           A.league_id AS league_id,
           A.country_id AS country_id,
           A.season as SEASON,
           A.date AS DATA
    FROM test_analytics_engineer_Match A
    GROUP BY A.league_id, A.country_id, A.season, A.date
)

-- Visão por temporada

SELECT sum(QTD_HOME_TEAM_GOAL) AS QTD_HOME_TEAM_GOAL
,sum(QTD_AWAY_TEAM_GOAL) AS QTD_AWAY_TEAM_GOAL
,sum(QTD_GOAL) AS QTD_GOAL
,AVG(QTD_HOME_AWAY) AS AVG_HOME_AWAY
,SEASON
FROM dados_consolidados A
GROUP BY SEASON
ORDER BY SEASON  ASC

'''

# Executando a consulta SQL 4
query_4 = '''
WITH dados_consolidados AS (
    SELECT SUM(home_team_goal) AS QTD_HOME_TEAM_GOAL,
           SUM(away_team_goal) AS QTD_AWAY_TEAM_GOAL,
           SUM(goal) AS QTD_GOAL,
```

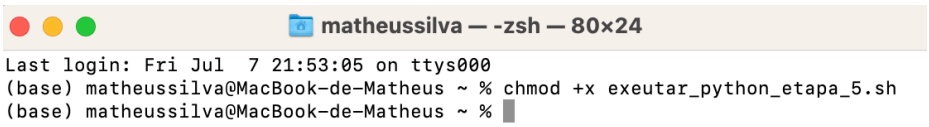
```
            home_team_goal + away_team_goal AS QTD_HOME_AWAY,
            A.league_id AS league_id,
            A.country_id AS country_id,
            A.season as SEASON,
            A.date AS DATA
    FROM test_analytics_engineer_Match A
    GROUP BY A.league_id, A.country_id, A.season, A.date
)

-- Visão por ano e mês

SELECT sum(QTD_HOME_TEAM_GOAL) AS QTD_HOME_TEAM_GOAL
,sum(QTD_AWAY_TEAM_GOAL) AS QTD_AWAY_TEAM_GOAL
,sum(QTD_GOAL) AS QTD_GOAL
,AVG(QTD_HOME_AWAY) AS AVG_HOME_AWAY
,strftime('%Y-%m', a.DATA) AS ANO_MES
FROM dados_consolidados A
GROUP BY strftime('%Y-%m', a.DATA)
ORDER BY strftime('%Y-%m', a.DATA)  ASC


'''



# Executando a consulta SQL 5
query_5 = '''
WITH dados_consolidados AS (
    SELECT SUM(home_team_goal) AS QTD_HOME_TEAM_GOAL,
            SUM(away_team_goal) AS QTD_AWAY_TEAM_GOAL,
            SUM(goal) AS QTD_GOAL,
            home_team_goal + away_team_goal AS QTD_HOME_AWAY,
            A.league_id AS league_id,
            A.country_id AS country_id,
            A.season as SEASON,
            A.date AS DATA
    FROM test_analytics_engineer_Match A
    GROUP BY A.league_id, A.country_id, A.season, A.date
)

-- Visão consolidada unificada

SELECT sum(QTD_HOME_TEAM_GOAL) AS QTD_HOME_TEAM_GOAL
,sum(QTD_AWAY_TEAM_GOAL) AS QTD_AWAY_TEAM_GOAL
,sum(QTD_GOAL) AS QTD_GOAL
,AVG(QTD_HOME_AWAY) AS AVG_HOME_AWAY
,B.name
,C.name
,SEASON
,strftime('%Y-%m', a.DATA) AS ANO_MES

FROM dados_consolidados A
LEFT JOIN test_analytics_engineer_League B ON B.id = A.league_id
LEFT JOIN (SELECT DISTINCT id,
CASE WHEN name = 'Bélgica' THEN 'Belgium'
      WHEN name = 'Inglaterra' THEN 'England'
      WHEN name = 'França' THEN 'France'
      WHEN name = 'Alemanha' THEN 'Germany'
      WHEN name = 'Itália' THEN 'Italy'
```

```
ELSE name END AS name
FROM test_analytics_engineer_Country) C ON C.id = A.country_id
GROUP BY B.name
,C.name
,SEASON
,strftime('%Y-%m', a.DATA)
ORDER BY B.name  ASC

'''

df_1 = pd.read_sql_query(query_1, conexao)
df_2 = pd.read_sql_query(query_2, conexao)
df_3 = pd.read_sql_query(query_3, conexao)
df_4 = pd.read_sql_query(query_4, conexao)
df_5 = pd.read_sql_query(query_5, conexao)


# Salve o resultado eM arquivos arquivos .CSV
df_1.to_csv('/Users/matheussilva/Base_league.csv', index=False)
df_2.to_csv('/Users/matheussilva/Base_Country.csv', index=False)
df_3.to_csv('/Users/matheussilva/Base_Season.csv', index=False)
df_4.to_csv('/Users/matheussilva/Base_YearMonth.csv', index=False)
df_5.to_csv('/Users/matheussilva/Base_Unified.csv', index=False)


# Realizando a confirmação das alterações e fechamento da conexão
conexao.commit()
conexao.close()
```

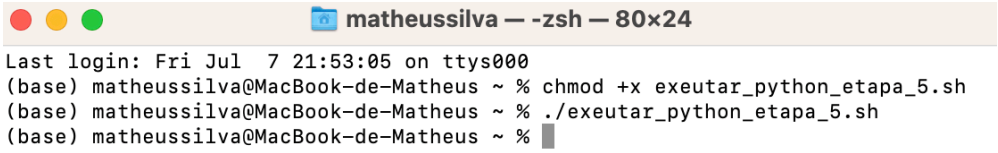
```
--Batch para automatização no Mac OS e em seguida definição do horário no
agendador de tarefas (alterar o diretório onde os arquivos estão
localizados)
--Criar um arquivo com a extensão .sh contendo o código abaixo

#!/bin/bash
python /Users/matheussilva/etapa_5.py
```

--Fluxo para executar no terminal e em seguida inserir no agendador
--No terminal entrar no diretório que contém o arquivo .sh
cd /Users/matheussilva/

-- Inserir o comando para permitir a execução caso não esteja liberado ainda
chmod +x exeutar_python_etapa_5.sh



--Comando para executar o arquivo .sh
./exeutar_python_etapa_5.sh



--Batch para automatização no Windows e em seguida definição do horário no agendador de tarefas (alterar o diretório onde os arquivos estão localizados)
--Criar um arquivo com a extensão .bat contendo o código abaixo

```
@echo off
python /Users/matheussilva/etapa_5.py
exit
```

--Clicar com o botão direito para executar e em seguida configurar a tarefa com a definição do horário no agendador de tarefas para ser executado de forma automática