Documentação Técnica

Oracle Cloud Project

Sumário

Resumo

Este projeto de BI é voltado para o **Campeonato de Futebol Brasileiro**. O objetivo é coletar, processar e visualizar dados relevantes para fornecer insights valiosos sobre o campeonato. O projeto abrange desde a criação de um Data Mart até a criação de painéis para refletir os dados.

Processo de ETL:

O processo de Extração, Transformação e Carga (ETL) é realizado da seguinte maneira:

1. **Coleta de Dados de Futebol**: Os dados são coletados por um script em Python. Este script é responsável por extrair os dados relevantes necessários para a análise.
2. **Gravação de Dados**: Após a coleta, os dados são gravados em um Bucket na Oracle Cloud Infrastructure (OCI).
3. **Carga no Data Mart**: Após a gravação dos dados, inicia-se o processo de ETL para carga no Data Mart. Este processo é dividido em três etapas:
    - Os dados são carregados em uma stage por uma procedure que busca o arquivo no bucket.
    - Os dados são tratados para serem carregados em suas respectivas dimensões.
    - Por fim, é feita a carga da tabela FATO.

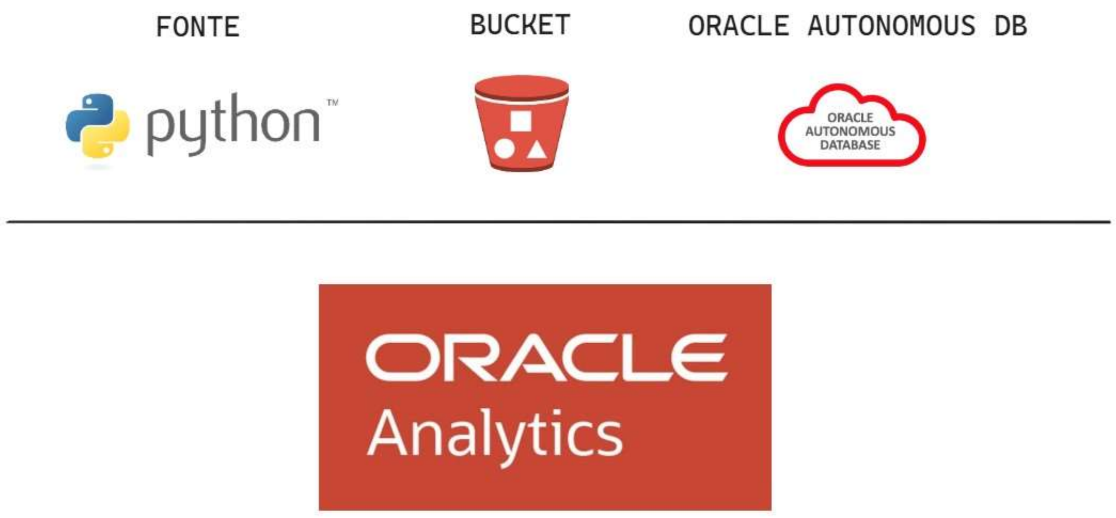
Todo este processo de ETL foi realizado utilizando o Autonomous Data Warehouse da Oracle.

Por fim todo o consumo dessas informações será feito através do **Oracle Cloud Analytics** por meio de um Dashboard.

Arquitetura da Solução

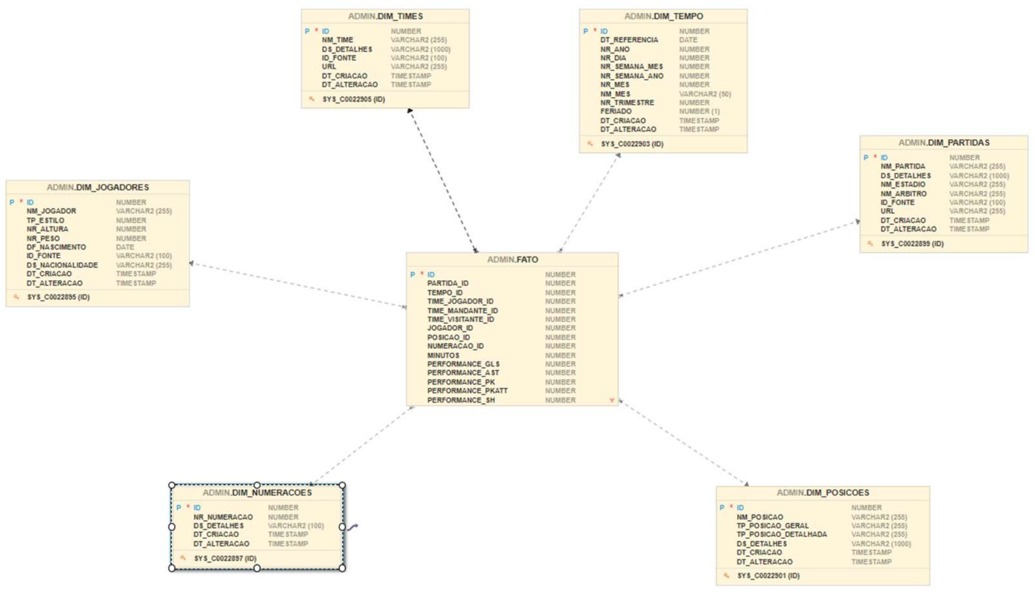
A arquitetura foi definida da seguinte forma:

- Web Scraping
- Bucket Store
- Data Mart
- Analytics



Data Mart

O DataMart foi modelado seguindo os padrões da **modelagem dimensional Star Schema**, com ela conseguiremos obter alta velocidade para consulta.



Toda **dimensão** possui diversos atributos e a tabela Fato é responsável por integrar todo as dimensões. O objetivo era tentar ir até a granularidade de dados de jogador por partida.

As dimensões foram:

- Tempo
- Partida
- Jogador
- Times
- Numerações
- Posições

A tabela Fato contou com mais de 100 colunas (features) que podem ser usadas para analisar os dados posteriormente.

ETL

O ETL foi realizado inteiramente por scripts PL/SQL e a parte com Python de Webscrapping.

Para a carga stage do nosso Data Mart o Banco deveria acessar os arquivos do Bucket. Posteriormente as dimensões seriam carregadas depois de tratadas, assim como a tabela Fato.

Webscrapping

A raspagem de dados de um site geralmente não é um mecanismo muito eficiente, dado que se houver quaisquer mudanças podem fazer o script não funcionar como anteriormente. Como não havia um provedor de dados para essa tarefa, foi realizada a raspagem dos dados do site FBREF.

Código

```
"""
Docstring
"""
from io import StringIO
from io import BytesIO
import pandas as pd
import requests
import loguru
from bs4 import BeautifulSoup
import boto3
from botocore.client import Config


class Fbref():
    """
    This Class is responsible for handling data from FbRef.
    """

    URL_BASE = "https://fbref.com/"

    def __init__(self, championship):
        """
        """
        df = self.season_championship(championship)
        df_final = self.get_season_data(df)
        self.df_intermediate = df_final
        self.df_initial = df
        df_partidas = pd.read_html(StringIO(str(self.df_initial)))[0]
        df_fim = df_partidas[df_partidas["Match Report"] == "Match
Report"].reset_index(drop=True)
        df_fim["PARTIDAS_FINAL"] = df_final["PARTIDAS_FINAL"].to_list()
        df_fim["ID_PARTIDA"] = df_final["ID_PARTIDA"].to_list()
        self.df_final = df_fim

    def get_all_championship(self):
        """
        get all champions games.
        """
        list_all = []
        list_home = []
        list_away = []
        for number, _ in enumerate(self.df_final.ID_PARTIDA):
            dados,team_home_id,team_away_id =
self.get_brasileirao_championship_game(
                game_url=self.df_final.PARTIDAS_FINAL[number],
                match_id_=self.df_final.ID_PARTIDA[number]
            )
            list_all.append(dados)
            list_home.append(team_home_id)
            list_away.append(team_away_id)
            loguru.logger.debug(
                f"Game number: {str(number)}, match:
{self.df_final.PARTIDAS_FINAL[number]}")
        return list_all,list_home,list_away

    def compilate_all(self):
        """
        Compilate all necessary steps.
        """
        list_dfs, list_home,list_away =self.get_all_championship()
        self.df_final["ID_TIME_CASA"] = list_home
        self.df_final["ID_TIME_VISITANTE"] = list_away
        return pd.concat(list_dfs)

    def season_championship(self, url: str):
        """
        This function will pick all season championship games and
        will retrieve table games id readable.

        Params
        -------
        url: str

        """

        req = requests.get(url,timeout=10)
        if req.status_code == 200:
            content = req.content
        else:
            loguru.logger.error(f"Error request: {str(req.status_code)}")
            raise requests.exceptions.HTTPError("No more reqauests for a
while.")
        soup = BeautifulSoup(content, 'html.parser')
        # Find the table element with class "my-table"
        table = soup.select_one('.stats_table')

        # Extract the id attribute
        if table:
            table_id = table.get('id')
            print("Table ID:", table_id)
        else:
            table_id = None
            print("Table not found.")

        tb = soup.find(id=table_id)
        return tb

    def get_season_data(self, tb):
        """
        It gets all season data from a html table

        Params
        --------
        tb: html table

        Returns
        --------
        df: pandas.DataFrame

        """
        s1 = [str(i) for i in tb.find_all("a")]
        s2 = [str(i.get_text('href')) for i in tb.find_all("a")]
        s4 = [i.replace('<a href="', '').replace('</a>', '') for i in s1]
        s5 = [i if "matches" in i else None for i in s1]
        s6 = [i.replace('<a href="/en/squads/', '')[0:8]
              if '<a href="/en/squads/' in i else None for i in s1]
        s9 = [i if "Match Report" in i else None for i in s4]

        base = pd.DataFrame(list(zip(s1, s2, s4, s5, s6, s9)), columns=[
                                    'CODES', 'ID', 'URL_FINAL', 'PARTIDAS',
"TEAM_CODE", "PARTIDAS_FINAL"])
        df_final = base[base["ID"] == "Match
Report"].reset_index(drop=True)
        df_final["ID_PARTIDA"] = [
            i.split("/")[3] for i in df_final.PARTIDAS_FINAL.to_list()]

        return df_final

    def get_teams_ids_from_match(self,soup):
        """
        This funtion will return home and away teams id

        Params
        ---------
        game_url: str
        Returns
        ---------
        id_team_home: str
        id_team_away: str
        """
        scoreboxes = soup.find_all(class_="scorebox")
        ids_list = [str(url).split("/")[3] for url in
scoreboxes[0].find_all('a') if "squads" in str(url)]
        id_team_home = ids_list[0]
        id_team_away = ids_list[1]
        return id_team_home,id_team_away


    def get_brasileirao_championship_game(self, game_url: str, match_id_:
str):
        """
        This function scrapes the site in order to get data from the game
analysed.

        Params
        ---------
        game_url: string

        Returns
        ---------
        list_df: list

        """
        soup = self._make_request(Fbref.URL_BASE + game_url)
        result_set = self._read(
            "table_wrapper tabbed",
            soup
        )

        # goalkeppers_set = self._read("table_wrapper",Fbref.URL_BASE +
        # game_url) #not used

        team_home_id,team_away_id =self.get_teams_ids_from_match(soup)

        team_1 = str(result_set[0])
        team_2 = str(result_set[1])

        players_home_team = self._extract_player_ids(result_set[0])
        players_away_team = self._extract_player_ids(result_set[1])

        # shots = str(result_set[2]) #not used
        # goalkeppers_1 =
self._read_goalkeppers_stats(str(goalkeppers_set[1])) #not used right now
```

```
        # goalkeppers_2 =
self._read_goalkeppers_stats(str(goalkeppers_set[3]))
        # #not used right now

        team_1_summary, team_1_passing, team_1_passing_types,
team_1_defensive, team_1_possession, team_1_miscellaneous =
self._read_teams_stats(
            team_1)
        team_2_summary, team_2_passing, team_2_passing_types,
team_2_defensive, team_2_possession, team_2_miscellaneous =
self._read_teams_stats(
            team_2)
        # shots_summary = self._read_shots_stats(shots) #not used right
now

        players_home_team = players_home_team[:len(team_1_summary) - 1]
        players_away_team = players_away_team[:len(team_2_summary) - 1]

        def add_column_from_list(
            df, column_name, lst): return
df.copy().assign(**{column_name: lst})

        list_df_home_team = self._treat_all_dfs(
            team_home_id,
            match_id_,
            add_column_from_list(
                team_1_summary[:len(team_1_summary) - 1], "JOGADOR_ID",
players_home_team),
            add_column_from_list(
                team_1_passing[:len(team_1_passing) - 1], "JOGADOR_ID",
players_home_team),
            add_column_from_list(team_1_passing_types[:len(
                team_1_passing_types) - 1], "JOGADOR_ID",
players_home_team),
            add_column_from_list(team_1_defensive[:len(
                team_1_defensive) - 1], "JOGADOR_ID", players_home_team),
            add_column_from_list(team_1_possession[:len(
                team_1_possession) - 1], "JOGADOR_ID",
players_home_team),
            add_column_from_list(team_1_miscellaneous[:len(
                team_1_miscellaneous) - 1], "JOGADOR_ID",
players_home_team)
        )

        list_df_away_team = self._treat_all_dfs(
            team_away_id,
            match_id_,
            add_column_from_list(
                team_2_summary[:len(team_2_summary) - 1], "JOGADOR_ID",
players_away_team),
            add_column_from_list(
                team_2_passing[:len(team_2_passing) - 1], "JOGADOR_ID",
players_away_team),
            add_column_from_list(team_2_passing_types[:len(
                team_2_passing_types) - 1], "JOGADOR_ID",
players_away_team),
            add_column_from_list(team_2_defensive[:len(
                team_2_defensive) - 1], "JOGADOR_ID", players_away_team),
            add_column_from_list(team_2_possession[:len(
                team_2_possession) - 1], "JOGADOR_ID",
players_away_team),
            add_column_from_list(team_2_miscellaneous[:len(
                team_2_miscellaneous) - 1], "JOGADOR_ID",
players_away_team)
        )

        home_team_df = self._treat_list_dfs(list_df_home_team)
        away_team_df = self._treat_list_dfs(list_df_away_team)

        return pd.concat([home_team_df,
away_team_df]).reset_index(drop=True),team_home_id,team_away_id

    def _treat_list_dfs(self, list_df):
        """
        Trat all dataframes dropping unnecessary collumns.

        Params
        ---------
        list_df:list[pd.DataFrame]

        """
        df_compiled = pd.concat(list_df, axis=1)
        df = df_compiled.loc[:, ~df_compiled.columns.duplicated()]
        df_final = df.copy()
        for i in list(df.columns):
            if ("__" in i) or ("_1" in i and "Carries" not in i):
                df_final.drop(columns=[i], axis=1, inplace=True)
        return df_final

    def _read_goalkeppers_stats(self, shots_str: str):
        """
        This function read the content of the page and returns a
Beautiful Soup Result.Set.

        Params
        ----------
        shots_str: string
        Returns
        ----------
        result: bs4.element.ResultSet

        """

        shots_summary = pd.read_html(StringIO(shots_str))[0]
        return shots_summary

    def _read_shots_stats(self, shots_str: str):
        """
        This function read the content of the page and returns a
Beautiful Soup Result.Set.

        Params
        ----------
        shots_str: string

        Returns
        ----------
        result: bs4.element.ResultSet

        """

        shots_summary = pd.read_html(StringIO(shots_str))[0]

        return shots_summary

    def _read_teams_stats(self, team_str: str):
        """
        This function read the content of the page and returns a
Beautiful Soup Result.Set.

        Params
        ----------
        team_str: string

        Returns
        ----------
        result: bs4.element.ResultSet

        """

        team_summary = pd.read_html(StringIO(team_str))[0]
        team_passing = pd.read_html(StringIO(team_str))[1]
        team_passing_types = pd.read_html(StringIO(team_str))[2]
        team_defensive = pd.read_html(StringIO(team_str))[3]
        team_possession = pd.read_html(StringIO(team_str))[4]
        team_miscellaneous = pd.read_html(StringIO(team_str))[5]

        return team_summary, team_passing, team_passing_types,
team_defensive, team_possession, team_miscellaneous

    def _make_request(self,url:str):
        """
        This function makes requests

        Params
        --------
        url: str

        Return:
        --------
        content : bs4.soup
        """
        req = requests.get(url,timeout=10)  # AJUSTAR
        content = req.content
        if req.status_code == 200:
            soup = BeautifulSoup(content, 'html.parser')
        else:
            loguru.logger.error(f"Error request: {str(req.status_code)}")
            raise requests.exceptions.HTTPError("No more requests for a
while.")
        return soup

    def _read(self, class_id, soup):
        """
        This function read the content of the page and returns a
Beautiful Soup Result.Set.

        Params
        ----------
        class_id: string
        url: string

        Returns
        ----------
        result: bs4.element.ResultSet

        """
        result = soup.find_all(class_=class_id)
        return result

    def _treat_all_dfs(self, team_id: str, match_id: str, *args):
        """
        This function treats all dataframes and returns a list with all
dataframes.
```

```
        Params
        ---------
        *args: pandas.DataFrames

        Returns
        ---------
        returns_df_list: list

        """
        returns_df_list = []
        for arg in args:
            df_arg = self._treat_columns_df(arg)
            df_arg_final = df_arg.copy()
            df_arg_final["TIME_ID"] = team_id
            df_arg_final["PARTIDA_ID"] = match_id
            returns_df_list.append(df_arg_final)
        return returns_df_list

    def _treat_columns_df(self, df):
        """
        As all dataframes are with multi-index columns, this function
normalize columns names and
        returns the same dataframe with new columns.

        Params
        ----------
        df: pandas.DataFrame

        Returns
        ----------
        df: pandas.DataFrame

        """

        new_names_list = []
        for column_name in df.columns:
            if "Unnamed" in str(column_name[0]):
                new_names_list.append(str(column_name[1]))
            else:
                new_names_list.append(
                    str(column_name[0]) + "_" + str(column_name[1]))

        df.columns = [
            i.replace(
                "%",
                "_Percentage").replace(
                "#O",
                "O").replace(
                "#",
                "Number").replace(
                    " ",
                "_") for i in new_names_list]

        return df

    def _extract_player_ids(self, table):
        """
        Extract player id from fbref.

        Params
        --------
        table: bs4.Element

        Returns
        --------
        pd.DataFrame
        """
        player_hrefs = []
        rows = table.find_all('tr')
        for row in rows:
            player_link = row.find('a')
            if player_link:
                player_href = player_link.get('href')
                player_href = player_href.split("/")
                player_hrefs.append(player_href[3])
        return pd.DataFrame(player_hrefs)


    def get_s3_client(self):
        """
        This function will generate s3 client.

        """
        oci_access_key_id = '-'
        oci_secret_access_key = '-'
        bucket_name = 'bucket-20240426-1658'
        bucket_url = 'https://objectstorage.us-ashburn-
1.oraclecloud.com/p/pgjlHHHp-
hMwg8KhvUgmmXlktPdjqeknT82TaZ0b97t1sSun4WtRa7FXA9nbmLDc/n/idpqeeodnr4t/b/
bucket-20240426-1658/o/'


        s3_client = boto3.client(
            's3',
            endpoint_url=bucket_url,
            aws_access_key_id=oci_access_key_id,
            aws_secret_access_key=oci_secret_access_key,
            config=Config(signature_version='s3v4')
        )
        return s3_client


    def post_data_oci(self, df, s3_client, df_name):
        """
        This function will save dataframe into OCI

        Params
        -------
        df: pd.DataFrame
        s3_client: boto.s3_client
        df_name: str

        """
        csv_buffer = BytesIO()
        df.to_csv(csv_buffer, index=False)
        csv_buffer.seek(0)

        metadata = {
            'description': 'FuteLab',
        }

        s3_client.upload_fileobj(
            Fileobj=csv_buffer,
            Bucket='ARQUIVOS_AUTOMATIZADOS',
            Key= df_name,
            ExtraArgs={'Metadata': metadata}
        )
        loguru.logger.debug(f"Arquivo {df_name} CSV salvo na OCI.")


if __name__ == "__main__":
    fbref = Fbref("https://fbref.com/en/comps/24/2024/schedule/2024-
Serie-A-Scores-and-Fixtures")
    df_fim = fbref.compilate_all()
    boto = fbref.get_s3_client()
    fbref.post_data_oci(fbref.df_final,boto,"BRASILEIRAO_JOGOS_2024.csv")
    fbref.post_data_oci(df_fim,boto,"BRASILEIRAO_TOTAL_2024.csv")
```

Bucket Store

O script acima deve ser capaz de salvar o arquivo no Bucket automaticamente, sendo assim o bucket funciona como nossa camada RAW, ou nossa STAGE, armazenando o dado bruto.

Procedures

SP_IMPORTAR_DADOS_STAGE_JOGOS

```
CREATE OR REPLACE EDITIONABLE PROCEDURE
"ADMIN"."SP_IMPORTAR_DADOS_STAGE_JOGOS" (
    PARAMETER_OBJECT VARCHAR2,
    PARAMETER_TABELA VARCHAR2
) AS


    BEGIN
        execute immediate 'TRUNCATE TABLE ' || PARAMETER_TABELA;

        DBMS_CLOUD.COPY_DATA[1](
            table_name => PARAMETER_TABELA,
            file_uri_list => 'https://objectstorage.us-ashburn-
1.oraclecloud.com/p/vEcxcJBg64xQ8_lWziDrstU8VNDUm-F-
wTZ8RuoweMA3hGWfrgA0XWab_GnhQRRw/n/idpqeeodnr4t/b/bucket-20240426-1658/o/' ||
PARAMETER_OBJECT,
            format => json_object(
                'delimiter' value ',',
                'skipheaders' value 1,
                'enablelogs' value FALSE
            )
        );
    END;
```

SP_IMPORTAR_DADOS_STAGE

```
CREATE OR REPLACE EDITIONABLE PROCEDURE "ADMIN"."SP_IMPORTAR_DADOS_STAGE" (
    PARAMETER_OBJECT VARCHAR2,
    PARAMETER_TABELA VARCHAR2
) AS


    BEGIN
        execute immediate 'TRUNCATE TABLE ' || PARAMETER_TABELA;

        DBMS_CLOUD.COPY_DATA (
```

[1] Oracle Autonomous Database on Dedicated Exadata Infrastructure

```
            table_name => PARAMETER_TABELA,
            file_uri_list => 'https://objectstorage.us-ashburn-
1.oraclecloud.com/p/vEcxcJBg64xQ8_lWziDrstU8VNDUm-F-
wTZ8RuoweMA3hGWfrgA0XWab_GnhQRRw/n/idpqeeodnr4t/b/bucket-20240426-1658/o/' ||
PARAMETER_OBJECT,
            format => json_object(
                'delimiter' value ',',
                'skipheaders' value 1,
                'blankasnull' value true,
                'ignoremissingcolumns' value true,
                'quote' value '"',
                'endquote' value '"',
                'enablelogs' value FALSE
            )
        );
    END;
```

SP_FATO

```
CREATE OR REPLACE EDITIONABLE PROCEDURE "ADMIN"."SP_FATO" IS
BEGIN
    INSERT INTO TB_LOGS(NM_TABELA, DS_EVENTO, ST_STATUS)
    VALUES ('FATO', 'MERGE', 'INICIADO');




BEGIN

    MERGE INTO FATO dest
    USING (

SELECT
    TO_NUMBER(DIM_PARTIDAS.ID) AS PARTIDA_ID,
    TO_NUMBER(DIM_JOGADORES.ID) AS JOGADOR_ID,
    TO_NUMBER(DIM_NUMERACOES.ID) AS NUMERACAO_ID,
    TO_NUMBER(DIM_POSICOES.ID) AS POSICAO_ID,
    TO_NUMBER(MANDANTES.ID) AS TIME_MANDANTE_ID,
    TO_NUMBER(VISITANTES.ID) AS TIME_VISITANTE_ID,
    TO_NUMBER(DIM_TIMES.ID) AS TIME_JOGADOR_ID,
    TO_NUMBER(DIM_TEMPO.ID) AS TEMPO_ID,
    TO_NUMBER(COALESCE(MIN,'0')) AS MINUTOS,
    TO_NUMBER(COALESCE(PERFORMANCE_GLS,'0')) AS PERFORMANCE_GLS,
    TO_NUMBER(COALESCE(PERFORMANCE_AST,'0')) AS PERFORMANCE_AST,
    TO_NUMBER(COALESCE(PERFORMANCE_PK,'0')) AS PERFORMANCE_PK,
    TO_NUMBER(COALESCE(PERFORMANCE_PKATT,'0')) AS PERFORMANCE_PKATT,
    TO_NUMBER(COALESCE(PERFORMANCE_SH,'0')) AS PERFORMANCE_SH,
    TO_NUMBER(COALESCE(PERFORMANCE_SOT,'0')) AS PERFORMANCE_SOT,
    TO_NUMBER(COALESCE(PERFORMANCE_CRDY,'0')) AS PERFORMANCE_CRDY,
    TO_NUMBER(COALESCE(PERFORMANCE_CRDR,'0')) AS PERFORMANCE_CRDR,
    TO_NUMBER(COALESCE(PERFORMANCE_TOUCHES,'0')) AS PERFORMANCE_TOUCHES,
    TO_NUMBER(COALESCE(PERFORMANCE_TKL,'0')) AS PERFORMANCE_TKL,
    TO_NUMBER(COALESCE(PERFORMANCE_INT, '0')) AS PERFORMANCE_INT,
    TO_NUMBER(COALESCE(PERFORMANCE_BLOCKS, '0')) AS PERFORMANCE_BLOCKS,
    TO_NUMBER(COALESCE(PERFORMANCE_2CRDY, '0')) AS PERFORMANCE_2CRDY,
    TO_NUMBER(COALESCE(PERFORMANCE_FLS, '0')) AS PERFORMANCE_FLS,
    TO_NUMBER(COALESCE(PERFORMANCE_FLD, '0')) AS PERFORMANCE_FLD,
    TO_NUMBER(COALESCE(PERFORMANCE_OFF, '0')) AS PERFORMANCE_OFF,
    TO_NUMBER(COALESCE(PERFORMANCE_CRS, '0')) AS PERFORMANCE_CRS,
    TO_NUMBER(COALESCE(PERFORMANCE_TKLW, '0')) AS PERFORMANCE_TKLW,
    TO_NUMBER(COALESCE(PERFORMANCE_PKWON, '0')) AS PERFORMANCE_PKWON,
    TO_NUMBER(COALESCE(PERFORMANCE_PKCON, '0')) AS PERFORMANCE_PKCON,
    TO_NUMBER(COALESCE(PERFORMANCE_OG, '0')) AS PERFORMANCE_OG,
    TO_NUMBER(COALESCE(PERFORMANCE_RECOV, '0')) AS PERFORMANCE_RECOV,
    TO_NUMBER(COALESCE(SCA_SCA,'0')) AS SCA_SCA,
    TO_NUMBER(COALESCE(SCA_GCA,'0')) AS SCA_GCA,
    TO_NUMBER(COALESCE(PASSES_CMP,'0')) AS PASSES_CMP,
    TO_NUMBER(COALESCE(PASSES_ATT,'0')) AS PASSES_ATT,
    TO_NUMBER(COALESCE(PASSES_PRGP,'0')) AS PASSES_PRGP,
    TO_NUMBER(COALESCE(CARRIES_CARRIES,'0')) AS CARRIES_CARRIES,
    TO_NUMBER(COALESCE(CARRIES_PRGC,'0')) AS CARRIES_PRGC,
    TO_NUMBER(COALESCE(TOTAL_CMP,'0')) AS PASSING_TOTAL_CMP,
    TO_NUMBER(COALESCE(TOTAL_ATT,'0')) AS PASSING_TOTAL_ATT,
    TO_NUMBER(COALESCE(TOTAL_TOTDIST,'0')) AS PASSING_TOTAL_TOTDIST,
    TO_NUMBER(COALESCE(TOTAL_PRGDIST,'0')) AS PASSING_TOTAL_PRGDIST,
    TO_NUMBER(COALESCE(SHORT_CMP,'0')) AS PASSING_SHORT_CMP,
    TO_NUMBER(COALESCE(SHORT_ATT,'0')) AS PASSING_SHORT_ATT,
    TO_NUMBER(COALESCE(MEDIUM_CMP,'0')) AS PASSING_MEDIUM_CMP,
    TO_NUMBER(COALESCE(MEDIUM_ATT,'0')) AS PASSING_MEDIUM_ATT,
    TO_NUMBER(COALESCE(LONG_CMP,'0')) AS PASSING_LONG_CMP,
    TO_NUMBER(COALESCE(LONG_ATT,'0')) AS PASSING_LONG_ATT,
    TO_NUMBER(COALESCE(AST,'0')) AS PASSING_AST,
    TO_NUMBER(COALESCE(ATT,'0')) AS ATT ,
    TO_NUMBER(COALESCE(PASS_TYPES_LIVE,'0')) AS PASS_TYPES_LIVE ,
    TO_NUMBER(COALESCE(PASS_TYPES_DEAD,'0')) AS PASS_TYPES_DEAD ,
    TO_NUMBER(COALESCE(PASS_TYPES_FK,'0')) AS PASS_TYPES_FK ,
    TO_NUMBER(COALESCE(PASS_TYPES_TB,'0')) AS PASS_TYPES_TB ,
    TO_NUMBER(COALESCE(PASS_TYPES_SW,'0')) AS PASS_TYPES_SW ,
    TO_NUMBER(COALESCE(PASS_TYPES_CRS,'0')) AS PASS_TYPES_CRS ,
    TO_NUMBER(COALESCE(PASS_TYPES_TI,'0')) AS PASS_TYPES_TI ,
    TO_NUMBER(COALESCE(PASS_TYPES_CK,'0')) AS PASS_TYPES_CK ,
    TO_NUMBER(COALESCE(CORNER_KICKS_IN,'0')) AS CORNER_KICKS_IN ,
    TO_NUMBER(COALESCE(CORNER_KICKS_OUT,'0')) AS CORNER_KICKS_OUT ,
    TO_NUMBER(COALESCE(CORNER_KICKS_STR,'0')) AS CORNER_KICKS_STR ,
    TO_NUMBER(COALESCE(OUTCOMES_CMP,'0')) AS OUTCOMES_CMP ,
    TO_NUMBER(COALESCE(OUTCOMES_OFF,'0')) AS OUTCOMES_OFF ,
    TO_NUMBER(COALESCE(OUTCOMES_BLOCKS,'0')) AS OUTCOMES_BLOCKS ,
    TO_NUMBER(COALESCE(TACKLES_TKL,'0')) AS TACKLES_TKL ,
    TO_NUMBER(COALESCE(TACKLES_TKLW,'0')) AS TACKLES_TKLW ,
    TO_NUMBER(COALESCE(TACKLES_DEF_3RD,'0')) AS TACKLES_DEF_3RD ,
    TO_NUMBER(COALESCE(TACKLES_MID_3RD,'0')) AS TACKLES_MID_3RD ,
    TO_NUMBER(COALESCE(TACKLES_ATT_3RD,'0')) AS TACKLES_ATT_3RD ,
    TO_NUMBER(COALESCE(CHALLENGES_TKL,'0')) AS CHALLENGES_TKL ,
    TO_NUMBER(COALESCE(CHALLENGES_ATT,'0')) AS CHALLENGES_ATT ,
    TO_NUMBER(COALESCE(CHALLENGES_LOST,'0')) AS CHALLENGES_LOST ,
    TO_NUMBER(COALESCE(BLOCKS_BLOCKS,'0')) AS BLOCKS_BLOCKS ,
    TO_NUMBER(COALESCE(BLOCKS_SH,'0')) AS BLOCKS_SH ,
    TO_NUMBER(COALESCE(BLOCKS_PASS,'0')) AS BLOCKS_PASS ,
    TO_NUMBER(COALESCE(TKL_INT,'0')) AS TKL_INT ,
    TO_NUMBER(COALESCE(CLR,'0')) AS CLR ,
    TO_NUMBER(COALESCE(ERR,'0')) AS ERR ,
    TO_NUMBER(COALESCE(TOUCHES_TOUCHES,'0')) AS TOUCHES_TOUCHES ,
    TO_NUMBER(COALESCE(TOUCHES_DEF_PEN,'0')) AS TOUCHES_DEF_PEN ,
    TO_NUMBER(COALESCE(TOUCHES_DEF_3RD,'0')) AS TOUCHES_DEF_3RD ,
    TO_NUMBER(COALESCE(TOUCHES_MID_3RD,'0')) AS TOUCHES_MID_3RD ,
    TO_NUMBER(COALESCE(TOUCHES_ATT_3RD,'0')) AS TOUCHES_ATT_3RD ,
    TO_NUMBER(COALESCE(TOUCHES_ATT_PEN,'0')) AS TOUCHES_ATT_PEN ,
    TO_NUMBER(COALESCE(TOUCHES_LIVE,'0')) AS TOUCHES_LIVE ,
    TO_NUMBER(COALESCE(CARRIES_TOTDIST,'0')) AS CARRIES_TOTDIST ,
    TO_NUMBER(COALESCE(CARRIES_PRGDIST,'0')) AS CARRIES_PRGDIST ,
    TO_NUMBER(COALESCE(CARRIES_1_3,'0')) AS CARRIES_1_3 ,
    TO_NUMBER(COALESCE(CARRIES_CPA,'0')) AS CARRIES_CPA ,
    TO_NUMBER(COALESCE(CARRIES_MIS,'0')) AS CARRIES_MIS ,
    TO_NUMBER(COALESCE(CARRIES_DIS,'0')) AS CARRIES_DIS ,
    TO_NUMBER(COALESCE(RECEIVING_REC,'0')) AS RECEIVING_REC ,
    TO_NUMBER(COALESCE(RECEIVING_PRGR,'0')) AS RECEIVING_PRGR ,
    TO_NUMBER(COALESCE(AERIAL_DUELS_WON,'0')) AS AERIAL_DUELS_WON ,
    TO_NUMBER(COALESCE(AERIAL_DUELS_LOST,'0')) AS AERIAL_DUELS_LOST

FROM BRASILEIRAO_TOTAL BRASILEIRAO
LEFT JOIN DIM_PARTIDAS ON BRASILEIRAO.PARTIDA_ID = DIM_PARTIDAS.ID_FONTE
LEFT JOIN BRASILEIRAO_JOGOS JOGOS ON JOGOS.ID_PARTIDA  =
BRASILEIRAO.PARTIDA_ID
LEFT JOIN DIM_NUMERACOES ON BRASILEIRAO.NUMBER_RW =
DIM_NUMERACOES.NR_NUMERACAO
LEFT JOIN DIM_POSICOES ON UPPER(SUBSTR(BRASILEIRAO.POS, 1, 2)) =
DIM_POSICOES.NM_POSICAO
LEFT JOIN DIM_TIMES MANDANTES ON JOGOS.ID_TIME_CASA  = MANDANTES.ID_FONTE
LEFT JOIN DIM_TIMES VISITANTES ON JOGOS.ID_TIME_VISITANTE  =
VISITANTES.ID_FONTE
LEFT JOIN DIM_JOGADORES ON BRASILEIRAO.JOGADOR_ID_ = DIM_JOGADORES.ID_FONTE
LEFT JOIN DIM_TIMES ON BRASILEIRAO.TIME_ID = DIM_TIMES.ID_FONTE
LEFT JOIN DIM_TEMPO
    ON
        EXTRACT(DAY FROM DIM_TEMPO.DT_REFERENCIA) = EXTRACT(DAY FROM
TO_DATE(JOGOS.DATE_RW,'YYYY/MM/DD')) AND
        EXTRACT(MONTH FROM DIM_TEMPO.DT_REFERENCIA) = EXTRACT(MONTH FROM
TO_DATE(JOGOS.DATE_RW,'YYYY/MM/DD')) AND
        EXTRACT(YEAR FROM DIM_TEMPO.DT_REFERENCIA) =  EXTRACT(YEAR FROM
TO_DATE(JOGOS.DATE_RW,'YYYY/MM/DD'))


    ) src
    ON (
        src.PARTIDA_ID = dest.PARTIDA_ID AND
        src.JOGADOR_ID = dest.JOGADOR_ID AND
        src.NUMERACAO_ID = dest.NUMERACAO_ID AND
        src.POSICAO_ID = dest.POSICAO_ID AND
        src.TIME_MANDANTE_ID = dest.TIME_MANDANTE_ID AND
        src.TIME_VISITANTE_ID = dest.TIME_VISITANTE_ID AND
        src.TEMPO_ID = dest.TEMPO_ID
        )
    WHEN MATCHED THEN
        UPDATE SET
            dest.MINUTOS = src.MINUTOS,
            dest.PERFORMANCE_GLS = src.PERFORMANCE_GLS,
            dest.PERFORMANCE_AST = src.PERFORMANCE_AST,
            dest.PERFORMANCE_PK = src.PERFORMANCE_PK,
            dest.PERFORMANCE_PKATT = src.PERFORMANCE_PKATT,
            dest.PERFORMANCE_SH = src.PERFORMANCE_SH,
            dest.PERFORMANCE_SOT = src.PERFORMANCE_SOT,
            dest.PERFORMANCE_CRDY = src.PERFORMANCE_CRDY,
            dest.PERFORMANCE_CRDR = src.PERFORMANCE_CRDR,
            dest.PERFORMANCE_TOUCHES = src.PERFORMANCE_TOUCHES,
            dest.PERFORMANCE_TKL = src.PERFORMANCE_TKL,
            dest.PERFORMANCE_INT = src.PERFORMANCE_INT,
            dest.PERFORMANCE_BLOCKS = src.PERFORMANCE_BLOCKS,
            dest.PERFORMANCE_2CRDY = src.PERFORMANCE_2CRDY,
            dest.PERFORMANCE_FLS = src.PERFORMANCE_FLS,
            dest.PERFORMANCE_FLD = src.PERFORMANCE_FLD,
            dest.PERFORMANCE_OFF = src.PERFORMANCE_OFF,
            dest.PERFORMANCE_CRS = src.PERFORMANCE_CRS,
            dest.PERFORMANCE_TKLW = src.PERFORMANCE_TKLW,
            dest.PERFORMANCE_PKWON = src.PERFORMANCE_PKWON,
            dest.PERFORMANCE_PKCON = src.PERFORMANCE_PKCON,
            dest.PERFORMANCE_OG = src.PERFORMANCE_OG,
            dest.PERFORMANCE_RECOV = src.PERFORMANCE_RECOV,
            dest.SCA_SCA = src.SCA_SCA,
            dest.SCA_GCA = src.SCA_GCA,
            dest.PASSES_CMP = src.PASSES_CMP,
            dest.PASSES_ATT = src.PASSES_ATT,
            dest.PASSES_PRGP = src.PASSES_PRGP,
            dest.CARRIES_CARRIES = src.CARRIES_CARRIES,
            dest.CARRIES_PRGC = src.CARRIES_PRGC,
            dest.PASSING_TOTAL_CMP = src.PASSING_TOTAL_CMP,
            dest.PASSING_TOTAL_ATT = src.PASSING_TOTAL_ATT,
            dest.PASSING_TOTAL_TOTDIST = src.PASSING_TOTAL_TOTDIST,
            dest.PASSING_TOTAL_PRGDIST = src.PASSING_TOTAL_PRGDIST,
            dest.PASSING_SHORT_CMP = src.PASSING_SHORT_CMP,
            dest.PASSING_SHORT_ATT = src.PASSING_SHORT_ATT,
            dest.PASSING_MEDIUM_CMP = src.PASSING_MEDIUM_CMP,
            dest.PASSING_MEDIUM_ATT = src.PASSING_MEDIUM_ATT,
            dest.PASSING_LONG_CMP = src.PASSING_LONG_CMP,
            dest.PASSING_LONG_ATT = src.PASSING_LONG_ATT,
            dest.PASSING_AST = src.PASSING_AST,
            dest.ATT = src.ATT,
            dest.PASS_TYPES_LIVE = src.PASS_TYPES_LIVE,
            dest.PASS_TYPES_DEAD = src.PASS_TYPES_DEAD,
            dest.PASS_TYPES_FK = src.PASS_TYPES_FK,
            dest.PASS_TYPES_TB = src.PASS_TYPES_TB,
            dest.PASS_TYPES_SW = src.PASS_TYPES_SW,
            dest.PASS_TYPES_CRS = src.PASS_TYPES_CRS,
            dest.PASS_TYPES_TI = src.PASS_TYPES_TI,
            dest.PASS_TYPES_CK = src.PASS_TYPES_CK,
            dest.CORNER_KICKS_IN = src.CORNER_KICKS_IN,
            dest.CORNER_KICKS_OUT = src.CORNER_KICKS_OUT,
            dest.CORNER_KICKS_STR = src.CORNER_KICKS_STR,
            dest.OUTCOMES_CMP = src.OUTCOMES_CMP,
            dest.OUTCOMES_OFF = src.OUTCOMES_OFF,
            dest.OUTCOMES_BLOCKS = src.OUTCOMES_BLOCKS,
            dest.TACKLES_TKL = src.TACKLES_TKL,
            dest.TACKLES_TKLW = src.TACKLES_TKLW,
            dest.TACKLES_DEF_3RD = src.TACKLES_DEF_3RD,
            dest.TACKLES_MID_3RD = src.TACKLES_MID_3RD,
            dest.TACKLES_ATT_3RD = src.TACKLES_ATT_3RD,
            dest.CHALLENGES_TKL = src.CHALLENGES_TKL,
            dest.CHALLENGES_ATT = src.CHALLENGES_ATT,
            dest.CHALLENGES_LOST = src.CHALLENGES_LOST,
            dest.BLOCKS_BLOCKS = src.BLOCKS_BLOCKS,
            dest.BLOCKS_SH = src.BLOCKS_SH,
            dest.BLOCKS_PASS = src.BLOCKS_PASS,
            dest.TKL_INT = src.TKL_INT,
            dest.CLR = src.CLR,
            dest.ERR = src.ERR,
            dest.TOUCHES_TOUCHES = src.TOUCHES_TOUCHES,
            dest.TOUCHES_DEF_PEN = src.TOUCHES_DEF_PEN,
            dest.TOUCHES_DEF_3RD = src.TOUCHES_DEF_3RD,
            dest.TOUCHES_MID_3RD = src.TOUCHES_MID_3RD,
            dest.TOUCHES_ATT_3RD = src.TOUCHES_ATT_3RD,
            dest.TOUCHES_ATT_PEN = src.TOUCHES_ATT_PEN,
            dest.TOUCHES_LIVE = src.TOUCHES_LIVE,
            dest.CARRIES_TOTDIST = src.CARRIES_TOTDIST,
            dest.CARRIES_PRGDIST = src.CARRIES_PRGDIST,
            dest.CARRIES_1_3 = src.CARRIES_1_3,
            dest.CARRIES_CPA = src.CARRIES_CPA,
            dest.CARRIES_MIS = src.CARRIES_MIS,
            dest.CARRIES_DIS = src.CARRIES_DIS,
            dest.RECEIVING_REC = src.RECEIVING_REC,
            dest.RECEIVING_PRGR = src.RECEIVING_PRGR,
            dest.AERIAL_DUELS_WON = src.AERIAL_DUELS_WON,
            dest.AERIAL_DUELS_LOST = src.AERIAL_DUELS_LOST ,
            dest.DT_ALTERACAO = CURRENT_TIMESTAMP

    WHEN NOT MATCHED THEN
        INSERT (

            PARTIDA_ID,
            JOGADOR_ID,
            NUMERACAO_ID,
            POSICAO_ID,
            TIME_MANDANTE_ID,
            TIME_VISITANTE_ID,
            TIME_JOGADOR_ID,
            TEMPO_ID,
            MINUTOS,
            PERFORMANCE_GLS,
            PERFORMANCE_AST,
            PERFORMANCE_PK,
            PERFORMANCE_PKATT,
            PERFORMANCE_SH,
            PERFORMANCE_SOT,
            PERFORMANCE_CRDY,
            PERFORMANCE_CRDR,
            PERFORMANCE_TOUCHES,
            PERFORMANCE_TKL,
            PERFORMANCE_INT,
            PERFORMANCE_BLOCKS,
            PERFORMANCE_2CRDY,
            PERFORMANCE_FLS,
            PERFORMANCE_FLD,
            PERFORMANCE_OFF,
            PERFORMANCE_CRS,
            PERFORMANCE_TKLW,
            PERFORMANCE_PKWON,
            PERFORMANCE_PKCON,
            PERFORMANCE_OG,
            PERFORMANCE_RECOV,
            SCA_SCA,
            SCA_GCA,
            PASSES_CMP,
            PASSES_ATT,
            PASSES_PRGP,
            CARRIES_CARRIES,
            CARRIES_PRGC,
            PASSING_TOTAL_CMP,
            PASSING_TOTAL_ATT,
            PASSING_TOTAL_TOTDIST,
            PASSING_TOTAL_PRGDIST,
            PASSING_SHORT_CMP,
            PASSING_SHORT_ATT,
            PASSING_MEDIUM_CMP,
            PASSING_MEDIUM_ATT,
            PASSING_LONG_CMP,
            PASSING_LONG_ATT,
            PASSING_AST,
            ATT,
            PASS_TYPES_LIVE,
            PASS_TYPES_DEAD,
            PASS_TYPES_FK,
            PASS_TYPES_TB,
            PASS_TYPES_SW,
            PASS_TYPES_CRS,
            PASS_TYPES_TI,
            PASS_TYPES_CK,
            CORNER_KICKS_IN,
            CORNER_KICKS_OUT,
            CORNER_KICKS_STR,
            OUTCOMES_CMP,
            OUTCOMES_OFF,
            OUTCOMES_BLOCKS,
            TACKLES_TKL,
            TACKLES_TKLW,
            TACKLES_DEF_3RD,
            TACKLES_MID_3RD,
            TACKLES_ATT_3RD,
            CHALLENGES_TKL,
            CHALLENGES_ATT,
            CHALLENGES_LOST,
            BLOCKS_BLOCKS,
            BLOCKS_SH,
            BLOCKS_PASS,
            TKL_INT,
            CLR,
            ERR,
            TOUCHES_TOUCHES,
            TOUCHES_DEF_PEN,
            TOUCHES_DEF_3RD,
            TOUCHES_MID_3RD,
            TOUCHES_ATT_3RD,
            TOUCHES_ATT_PEN,
            TOUCHES_LIVE,
            CARRIES_TOTDIST,
            CARRIES_PRGDIST,
            CARRIES_1_3,
            CARRIES_CPA,
            CARRIES_MIS,
            CARRIES_DIS,
            RECEIVING_REC,
            RECEIVING_PRGR,
            AERIAL_DUELS_WON,
            AERIAL_DUELS_LOST

        )
        VALUES (
            src.PARTIDA_ID,
            src.JOGADOR_ID,
            src.NUMERACAO_ID,
            src.POSICAO_ID,
            src.TIME_MANDANTE_ID,
            src.TIME_VISITANTE_ID,
            src.TIME_JOGADOR_ID,
            src.TEMPO_ID,
            src.MINUTOS,
            src.PERFORMANCE_GLS,
            src.PERFORMANCE_AST,
            src.PERFORMANCE_PK,
            src.PERFORMANCE_PKATT,
            src.PERFORMANCE_SH,
            src.PERFORMANCE_SOT,
            src.PERFORMANCE_CRDY,
            src.PERFORMANCE_CRDR,
            src.PERFORMANCE_TOUCHES,
            src.PERFORMANCE_TKL,
            src.PERFORMANCE_INT,
            src.PERFORMANCE_BLOCKS,
            src.PERFORMANCE_2CRDY,
            src.PERFORMANCE_FLS,
            src.PERFORMANCE_FLD,
            src.PERFORMANCE_OFF,
            src.PERFORMANCE_CRS,
            src.PERFORMANCE_TKLW,
            src.PERFORMANCE_PKWON,
            src.PERFORMANCE_PKCON,
            src.PERFORMANCE_OG,
            src.PERFORMANCE_RECOV,
            src.SCA_SCA,
            src.SCA_GCA,
            src.PASSES_CMP,
            src.PASSES_ATT,
            src.PASSES_PRGP,
            src.CARRIES_CARRIES,
            src.CARRIES_PRGC,
            src.PASSING_TOTAL_CMP,
            src.PASSING_TOTAL_ATT,
            src.PASSING_TOTAL_TOTDIST,
            src.PASSING_TOTAL_PRGDIST,
            src.PASSING_SHORT_CMP,
            src.PASSING_SHORT_ATT,
            src.PASSING_MEDIUM_CMP,
            src.PASSING_MEDIUM_ATT,
            src.PASSING_LONG_CMP,
            src.PASSING_LONG_ATT,
            src.PASSING_AST,
            src.ATT,
            src.PASS_TYPES_LIVE,
            src.PASS_TYPES_DEAD,
            src.PASS_TYPES_FK,
            src.PASS_TYPES_TB,
            src.PASS_TYPES_SW,
            src.PASS_TYPES_CRS,
            src.PASS_TYPES_TI,
            src.PASS_TYPES_CK,
            src.CORNER_KICKS_IN,
            src.CORNER_KICKS_OUT,
            src.CORNER_KICKS_STR,
            src.OUTCOMES_CMP,
            src.OUTCOMES_OFF,
            src.OUTCOMES_BLOCKS,
            src.TACKLES_TKL,
            src.TACKLES_TKLW,
            src.TACKLES_DEF_3RD,
            src.TACKLES_MID_3RD,
            src.TACKLES_ATT_3RD,
            src.CHALLENGES_TKL,
            src.CHALLENGES_ATT,
            src.CHALLENGES_LOST,
            src.BLOCKS_BLOCKS,
            src.BLOCKS_SH,
            src.BLOCKS_PASS,
            src.TKL_INT,
            src.CLR,
            src.ERR,
            src.TOUCHES_TOUCHES,
            src.TOUCHES_DEF_PEN,
            src.TOUCHES_DEF_3RD,
            src.TOUCHES_MID_3RD,
            src.TOUCHES_ATT_3RD,
            src.TOUCHES_ATT_PEN,
            src.TOUCHES_LIVE,
            src.CARRIES_TOTDIST,
            src.CARRIES_PRGDIST,
            src.CARRIES_1_3,
            src.CARRIES_CPA,
            src.CARRIES_MIS,
            src.CARRIES_DIS,
            src.RECEIVING_REC,
            src.RECEIVING_PRGR,
            src.AERIAL_DUELS_WON,
            src.AERIAL_DUELS_LOST
        );

UPDATE TB_LOGS
SET
    ST_STATUS = 'FINALIZADO',
    DT_ALTERACAO = CURRENT_TIMESTAMP,
    NR_LINHAS = (SELECT COUNT(*) FROM (
    SELECT
        TO_NUMBER(DIM_PARTIDAS.ID) AS PARTIDA_ID,
        TO_NUMBER(DIM_JOGADORES.ID) AS JOGADOR_ID,
        TO_NUMBER(DIM_NUMERACOES.ID) AS NUMERACAO_ID,
        TO_NUMBER(DIM_POSICOES.ID) AS POSICAO_ID,
        TO_NUMBER(MANDANTES.ID) AS TIME_MANDANTE_ID,
            TO_NUMBER(VISITANTES.ID) AS TIME_VISITANTE_ID,
            TO_NUMBER(DIM_TIMES.ID) AS TIME_JOGADOR_ID,
            TO_NUMBER(DIM_TEMPO.ID) AS TEMPO_ID,
            TO_NUMBER(COALESCE(MIN,'0')) AS MINUTOS,
            TO_NUMBER(COALESCE(PERFORMANCE_GLS,'0')) AS PERFORMANCE_GLS,
            TO_NUMBER(COALESCE(PERFORMANCE_AST,'0')) AS PERFORMANCE_AST,
            TO_NUMBER(COALESCE(PERFORMANCE_PK,'0')) AS PERFORMANCE_PK,
            TO_NUMBER(COALESCE(PERFORMANCE_PKATT,'0')) AS PERFORMANCE_PKATT,
            TO_NUMBER(COALESCE(PERFORMANCE_SH,'0')) AS PERFORMANCE_SH,
            TO_NUMBER(COALESCE(PERFORMANCE_SOT,'0')) AS PERFORMANCE_SOT,
            TO_NUMBER(COALESCE(PERFORMANCE_CRDY,'0')) AS PERFORMANCE_CRDY,
            TO_NUMBER(COALESCE(PERFORMANCE_CRDR,'0')) AS PERFORMANCE_CRDR,
            TO_NUMBER(COALESCE(PERFORMANCE_TOUCHES,'0')) AS
PERFORMANCE_TOUCHES,
            TO_NUMBER(COALESCE(PERFORMANCE_TKL,'0')) AS PERFORMANCE_TKL,
            TO_NUMBER(COALESCE(PERFORMANCE_INT, '0')) AS PERFORMANCE_INT,
            TO_NUMBER(COALESCE(PERFORMANCE_BLOCKS, '0')) AS
PERFORMANCE_BLOCKS,
            TO_NUMBER(COALESCE(PERFORMANCE_2CRDY, '0')) AS PERFORMANCE_2CRDY,
            TO_NUMBER(COALESCE(PERFORMANCE_FLS, '0')) AS PERFORMANCE_FLS,
            TO_NUMBER(COALESCE(PERFORMANCE_FLD, '0')) AS PERFORMANCE_FLD,
            TO_NUMBER(COALESCE(PERFORMANCE_OFF, '0')) AS PERFORMANCE_OFF,
            TO_NUMBER(COALESCE(PERFORMANCE_CRS, '0')) AS PERFORMANCE_CRS,
            TO_NUMBER(COALESCE(PERFORMANCE_TKLW, '0')) AS PERFORMANCE_TKLW,
            TO_NUMBER(COALESCE(PERFORMANCE_PKWON, '0')) AS PERFORMANCE_PKWON,
            TO_NUMBER(COALESCE(PERFORMANCE_PKCON, '0')) AS PERFORMANCE_PKCON,
            TO_NUMBER(COALESCE(PERFORMANCE_OG, '0')) AS PERFORMANCE_OG,
            TO_NUMBER(COALESCE(PERFORMANCE_RECOV, '0')) AS PERFORMANCE_RECOV,
            TO_NUMBER(COALESCE(SCA_SCA,'0')) AS SCA_SCA,
            TO_NUMBER(COALESCE(SCA_GCA,'0')) AS SCA_GCA,
            TO_NUMBER(COALESCE(PASSES_CMP,'0')) AS PASSES_CMP,
            TO_NUMBER(COALESCE(PASSES_ATT,'0')) AS PASSES_ATT,
            TO_NUMBER(COALESCE(PASSES_PRGP,'0')) AS PASSES_PRGP,
            TO_NUMBER(COALESCE(CARRIES_CARRIES,'0')) AS CARRIES_CARRIES,
            TO_NUMBER(COALESCE(CARRIES_PRGC,'0')) AS CARRIES_PRGC,
            TO_NUMBER(COALESCE(TOTAL_CMP,'0')) AS PASSING_TOTAL_CMP,
            TO_NUMBER(COALESCE(TOTAL_ATT,'0')) AS PASSING_TOTAL_ATT,
            TO_NUMBER(COALESCE(TOTAL_TOTDIST,'0')) AS PASSING_TOTAL_TOTDIST,
            TO_NUMBER(COALESCE(TOTAL_PRGDIST,'0')) AS PASSING_TOTAL_PRGDIST,
            TO_NUMBER(COALESCE(SHORT_CMP,'0')) AS PASSING_SHORT_CMP,
            TO_NUMBER(COALESCE(SHORT_ATT,'0')) AS PASSING_SHORT_ATT,
            TO_NUMBER(COALESCE(MEDIUM_CMP,'0')) AS PASSING_MEDIUM_CMP,
            TO_NUMBER(COALESCE(MEDIUM_ATT,'0')) AS PASSING_MEDIUM_ATT,
            TO_NUMBER(COALESCE(LONG_CMP,'0')) AS PASSING_LONG_CMP,
            TO_NUMBER(COALESCE(LONG_ATT,'0')) AS PASSING_LONG_ATT,
            TO_NUMBER(COALESCE(AST,'0')) AS PASSING_AST,
            TO_NUMBER(COALESCE(ATT,'0')) AS ATT ,
            TO_NUMBER(COALESCE(PASS_TYPES_LIVE,'0')) AS PASS_TYPES_LIVE ,
            TO_NUMBER(COALESCE(PASS_TYPES_DEAD,'0')) AS PASS_TYPES_DEAD ,
            TO_NUMBER(COALESCE(PASS_TYPES_FK,'0')) AS PASS_TYPES_FK ,
            TO_NUMBER(COALESCE(PASS_TYPES_TB,'0')) AS PASS_TYPES_TB ,
            TO_NUMBER(COALESCE(PASS_TYPES_SW,'0')) AS PASS_TYPES_SW ,
            TO_NUMBER(COALESCE(PASS_TYPES_CRS,'0')) AS PASS_TYPES_CRS ,
            TO_NUMBER(COALESCE(PASS_TYPES_TI,'0')) AS PASS_TYPES_TI ,
            TO_NUMBER(COALESCE(PASS_TYPES_CK,'0')) AS PASS_TYPES_CK ,
            TO_NUMBER(COALESCE(CORNER_KICKS_IN,'0')) AS CORNER_KICKS_IN ,
            TO_NUMBER(COALESCE(CORNER_KICKS_OUT,'0')) AS CORNER_KICKS_OUT ,
            TO_NUMBER(COALESCE(CORNER_KICKS_STR,'0')) AS CORNER_KICKS_STR ,
            TO_NUMBER(COALESCE(OUTCOMES_CMP,'0')) AS OUTCOMES_CMP ,
            TO_NUMBER(COALESCE(OUTCOMES_OFF,'0')) AS OUTCOMES_OFF ,
            TO_NUMBER(COALESCE(OUTCOMES_BLOCKS,'0')) AS OUTCOMES_BLOCKS ,
            TO_NUMBER(COALESCE(TACKLES_TKL,'0')) AS TACKLES_TKL ,
            TO_NUMBER(COALESCE(TACKLES_TKLW,'0')) AS TACKLES_TKLW ,
            TO_NUMBER(COALESCE(TACKLES_DEF_3RD,'0')) AS TACKLES_DEF_3RD ,
            TO_NUMBER(COALESCE(TACKLES_MID_3RD,'0')) AS TACKLES_MID_3RD ,
            TO_NUMBER(COALESCE(TACKLES_ATT_3RD,'0')) AS TACKLES_ATT_3RD ,
            TO_NUMBER(COALESCE(CHALLENGES_TKL,'0')) AS CHALLENGES_TKL ,
            TO_NUMBER(COALESCE(CHALLENGES_ATT,'0')) AS CHALLENGES_ATT ,
            TO_NUMBER(COALESCE(CHALLENGES_LOST,'0')) AS CHALLENGES_LOST ,
            TO_NUMBER(COALESCE(BLOCKS_BLOCKS,'0')) AS BLOCKS_BLOCKS ,
            TO_NUMBER(COALESCE(BLOCKS_SH,'0')) AS BLOCKS_SH ,
            TO_NUMBER(COALESCE(BLOCKS_PASS,'0')) AS BLOCKS_PASS ,
            TO_NUMBER(COALESCE(TKL_INT,'0')) AS TKL_INT ,
            TO_NUMBER(COALESCE(CLR,'0')) AS CLR ,
            TO_NUMBER(COALESCE(ERR,'0')) AS ERR ,
            TO_NUMBER(COALESCE(TOUCHES_TOUCHES,'0')) AS TOUCHES_TOUCHES ,
            TO_NUMBER(COALESCE(TOUCHES_DEF_PEN,'0')) AS TOUCHES_DEF_PEN ,
            TO_NUMBER(COALESCE(TOUCHES_DEF_3RD,'0')) AS TOUCHES_DEF_3RD ,
            TO_NUMBER(COALESCE(TOUCHES_MID_3RD,'0')) AS TOUCHES_MID_3RD ,
            TO_NUMBER(COALESCE(TOUCHES_ATT_3RD,'0')) AS TOUCHES_ATT_3RD ,
            TO_NUMBER(COALESCE(TOUCHES_ATT_PEN,'0')) AS TOUCHES_ATT_PEN ,
            TO_NUMBER(COALESCE(TOUCHES_LIVE,'0')) AS TOUCHES_LIVE ,
            TO_NUMBER(COALESCE(CARRIES_TOTDIST,'0')) AS CARRIES_TOTDIST ,
            TO_NUMBER(COALESCE(CARRIES_PRGDIST,'0')) AS CARRIES_PRGDIST ,
            TO_NUMBER(COALESCE(CARRIES_1_3,'0')) AS CARRIES_1_3 ,
            TO_NUMBER(COALESCE(CARRIES_CPA,'0')) AS CARRIES_CPA ,
            TO_NUMBER(COALESCE(CARRIES_MIS,'0')) AS CARRIES_MIS ,
            TO_NUMBER(COALESCE(CARRIES_DIS,'0')) AS CARRIES_DIS ,
            TO_NUMBER(COALESCE(RECEIVING_REC,'0')) AS RECEIVING_REC ,
            TO_NUMBER(COALESCE(RECEIVING_PRGR,'0')) AS RECEIVING_PRGR ,
            TO_NUMBER(COALESCE(AERIAL_DUELS_WON,'0')) AS AERIAL_DUELS_WON ,
            TO_NUMBER(COALESCE(AERIAL_DUELS_LOST,'0')) AS AERIAL_DUELS_LOST

        FROM BRASILEIRAO_TOTAL BRASILEIRAO
        LEFT JOIN DIM_PARTIDAS ON BRASILEIRAO.PARTIDA_ID =
DIM_PARTIDAS.ID_FONTE
        LEFT JOIN BRASILEIRAO_JOGOS JOGOS ON JOGOS.ID_PARTIDA  =
BRASILEIRAO.PARTIDA_ID
        LEFT JOIN DIM_NUMERACOES ON BRASILEIRAO.NUMBER_RW =
DIM_NUMERACOES.NR_NUMERACAO
        LEFT JOIN DIM_POSICOES ON UPPER(SUBSTR(BRASILEIRAO.POS, 1, 2)) =
DIM_POSICOES.NM_POSICAO
        LEFT JOIN DIM_TIMES MANDANTES ON JOGOS.ID_TIME_CASA  =
MANDANTES.ID_FONTE
        LEFT JOIN DIM_TIMES VISITANTES ON JOGOS.ID_TIME_VISITANTE  =
VISITANTES.ID_FONTE
        LEFT JOIN DIM_JOGADORES ON BRASILEIRAO.JOGADOR_ID_ =
DIM_JOGADORES.ID_FONTE
        LEFT JOIN DIM_TIMES ON BRASILEIRAO.TIME_ID = DIM_TIMES.ID_FONTE
        LEFT JOIN DIM_TEMPO
            ON
                EXTRACT(DAY FROM DIM_TEMPO.DT_REFERENCIA) = EXTRACT(DAY FROM
TO_DATE(JOGOS.DATE_RW,'YYYY/MM/DD')) AND
                EXTRACT(MONTH FROM DIM_TEMPO.DT_REFERENCIA) = EXTRACT(MONTH
FROM TO_DATE(JOGOS.DATE_RW,'YYYY/MM/DD')) AND
```

```
                EXTRACT(YEAR FROM DIM_TEMPO.DT_REFERENCIA) =  EXTRACT(YEAR
FROM TO_DATE(JOGOS.DATE_RW,'YYYY/MM/DD'))

            ))
    WHERE ID = (SELECT MAX(ID) FROM TB_LOGS WHERE NM_TABELA = 'FATO' AND
DS_EVENTO = 'MERGE');
END;
EXCEPTION
    WHEN OTHERS THEN
        UPDATE TB_LOGS
        SET
            ST_STATUS = 'ERRO',
            DT_ALTERACAO = CURRENT_TIMESTAMP
        WHERE ID = (SELECT MAX(ID) FROM TB_LOGS WHERE NM_TABELA = 'FATO' AND
DS_EVENTO = 'MERGE');
END SP_FATO;
```

SP_DIM_TIMES

```
CREATE OR REPLACE EDITIONABLE PROCEDURE "ADMIN"."SP_DIM_TIMES" IS
BEGIN
    INSERT INTO TB_LOGS(NM_TABELA, DS_EVENTO, ST_STATUS)
    VALUES ('DIM_TIMES', 'MERGE', 'INICIADO');

    MERGE INTO DIM_TIMES dest
    USING (
        SELECT
            ID_TIME_CASA AS ID_FONTE,
            HOME AS NM_TIME

        FROM
            ADMIN.BRASILEIRAO_JOGOS
        UNION
        SELECT
            ID_TIME_VISITANTE AS ID_FONTE,
            AWAY AS NM_TIME
        FROM
            ADMIN.BRASILEIRAO_JOGOS

    ) src
    ON (src.ID_FONTE = dest.ID_FONTE)
    WHEN MATCHED THEN
        UPDATE SET
            dest.NM_TIME = src.NM_TIME,
            dest.DT_ALTERACAO = CURRENT_TIMESTAMP

    WHEN NOT MATCHED THEN
        INSERT (
            ID_FONTE,
            NM_TIME
        )
        VALUES (
            src.ID_FONTE,
            src.NM_TIME
        );
    UPDATE TB_LOGS
    SET
        ST_STATUS = 'FINALIZADO',
        DT_ALTERACAO = CURRENT_TIMESTAMP,
        NR_LINHAS = (SELECT COUNT(*) FROM (
        SELECT
            ID_TIME_CASA AS ID_FONTE,
            HOME AS NM_TIME

        FROM
            ADMIN.BRASILEIRAO_JOGOS
        UNION
        SELECT
            ID_TIME_VISITANTE AS ID_FONTE,
            AWAY AS NM_TIME
        FROM
            ADMIN.BRASILEIRAO_JOGOS

    ))
    WHERE ID = (SELECT MAX(ID) FROM TB_LOGS WHERE NM_TABELA = 'DIM_TIMES' AND
DS_EVENTO = 'MERGE');

EXCEPTION
    WHEN OTHERS THEN
        UPDATE TB_LOGS
        SET
            ST_STATUS = 'ERRO',
            DT_ALTERACAO = CURRENT_TIMESTAMP
        WHERE ID = (SELECT MAX(ID) FROM TB_LOGS WHERE NM_TABELA = 'DIM_TIMES'
AND DS_EVENTO = 'MERGE');
END SP_DIM_TIMES;
```

SP_DIM_POSICOES

```
CREATE OR REPLACE EDITIONABLE PROCEDURE "ADMIN"."SP_DIM_POSICOES" IS
BEGIN
    INSERT INTO TB_LOGS(NM_TABELA, DS_EVENTO, ST_STATUS)
    VALUES ('DIM_POSICOES', 'MERGE', 'INICIADO');

    MERGE INTO DIM_POSICOES dest
    USING (
        SELECT DISTINCT
            UPPER(SUBSTR(POS, 1, 2))  AS NM_POSICAO,
            CASE UPPER(SUBSTR(POS, 1, 2))
                WHEN 'GK' THEN 'GOLEIRO'
                WHEN 'CB' THEN 'ZAGUEIRO CENTRAL'
                WHEN 'LB' THEN 'LATERAL ESQUERDO'
                WHEN 'RB' THEN 'LATERAL DIREITO'
                WHEN 'CM' THEN 'MEIO-CAMPISTA CENTRAL'
                WHEN 'LM' THEN 'MEIO-CAMPISTA ESQUERDO'
                WHEN 'RM' THEN 'MEIO-CAMPISTA DIREITO'
                WHEN 'MF' THEN 'MEIO-CAMPISTA'
                WHEN 'FW' THEN 'ATACANTE'
                WHEN 'DF' THEN 'DEFENSOR'
                WHEN 'DM' THEN 'MEIO-CAMPISTA DEFENSIVO'
                WHEN 'RW' THEN 'ALA DIREITA'
                WHEN 'LW' THEN 'ALA ESQUERDA'
                WHEN 'AM' THEN 'MEIA AVANÇADO'
                ELSE 'POSICAO NAO ESPECIFICADA'
            END AS TP_POSICAO_DETALHADA,
                CASE UPPER(SUBSTR(POS, 1, 2))
                WHEN 'GK' THEN 'DEFESA'
                WHEN 'CB' THEN 'DEFESA'
                WHEN 'LB' THEN 'DEFESA'
                WHEN 'RB' THEN 'DEFESA'
                WHEN 'CM' THEN 'MEIO CAMPO'
                WHEN 'LM' THEN 'MEIO CAMPO'
                WHEN 'RM' THEN 'MEIO CAMPO'
                WHEN 'MF' THEN 'MEIO CAMPO'
                WHEN 'FW' THEN 'ATAQUE'
                WHEN 'DF' THEN 'DEFESA'
                WHEN 'DM' THEN 'MEIO CAMPO'
                WHEN 'RW' THEN 'ATAQUE'
                WHEN 'LW' THEN 'ATAQUE'
                WHEN 'AM' THEN 'MEIO CAMPO'
                ELSE 'POSICAO NAO ESPECIFICADA'
            END AS TP_POSICAO_GERAL,
            NULL AS DS_DETALHES
        FROM
            ADMIN.BRASILEIRAO_TOTAL
) src
ON (src.NM_POSICAO = dest.NM_POSICAO)
WHEN MATCHED THEN
    UPDATE SET
        dest.TP_POSICAO_GERAL = src.TP_POSICAO_GERAL,
        dest.TP_POSICAO_DETALHADA = src.TP_POSICAO_DETALHADA,
        dest.DS_DETALHES = src.DS_DETALHES,
        dest.DT_ALTERACAO = CURRENT_TIMESTAMP

WHEN NOT MATCHED THEN
    INSERT (
        NM_POSICAO,
        TP_POSICAO_GERAL,
        TP_POSICAO_DETALHADA,
        DS_DETALHES
    )
    VALUES (
        src.NM_POSICAO,
        src.TP_POSICAO_GERAL,
        src.TP_POSICAO_DETALHADA,
        src.DS_DETALHES
    );

UPDATE TB_LOGS
SET
    ST_STATUS = 'FINALIZADO',
    DT_ALTERACAO = CURRENT_TIMESTAMP,
    NR_LINHAS = (SELECT COUNT(*) FROM (
    SELECT DISTINCT
        UPPER(SUBSTR(POS, 1, 2))  AS NM_POSICAO,
        CASE UPPER(SUBSTR(POS, 1, 2))
            WHEN 'GK' THEN 'GOLEIRO'
            WHEN 'CB' THEN 'ZAGUEIRO CENTRAL'
```

```
                WHEN 'LB' THEN 'LATERAL ESQUERDO'
                WHEN 'RB' THEN 'LATERAL DIREITO'
                WHEN 'CM' THEN 'MEIO-CAMPISTA CENTRAL'
                WHEN 'LM' THEN 'MEIO-CAMPISTA ESQUERDO'
                WHEN 'RM' THEN 'MEIO-CAMPISTA DIREITO'
                WHEN 'MF' THEN 'MEIO-CAMPISTA'
                WHEN 'FW' THEN 'ATACANTE'
                WHEN 'DF' THEN 'DEFENSOR'
                WHEN 'DM' THEN 'MEIO-CAMPISTA DEFENSIVO'
                WHEN 'RW' THEN 'ALA DIREITA'
                WHEN 'LW' THEN 'ALA ESQUERDA'
                WHEN 'AM' THEN 'MEIA AVANÇADO'
                ELSE 'POSICAO NAO ESPECIFICADA'
            END AS TP_POSICAO_DETALHADA,
                CASE UPPER(SUBSTR(POS, 1, 2))
                WHEN 'GK' THEN 'DEFESA'
                WHEN 'CB' THEN 'DEFESA'
                WHEN 'LB' THEN 'DEFESA'
                WHEN 'RB' THEN 'DEFESA'
                WHEN 'CM' THEN 'MEIO CAMPO'
                WHEN 'LM' THEN 'MEIO CAMPO'
                WHEN 'RM' THEN 'MEIO CAMPO'
                WHEN 'MF' THEN 'MEIO CAMPO'
                WHEN 'FW' THEN 'ATAQUE'
                WHEN 'DF' THEN 'DEFESA'
                WHEN 'DM' THEN 'MEIO CAMPO'
                WHEN 'RW' THEN 'ATAQUE'
                WHEN 'LW' THEN 'ATAQUE'
                WHEN 'AM' THEN 'MEIO CAMPO'
                ELSE 'POSICAO NAO ESPECIFICADA'
            END AS TP_POSICAO_GERAL,
            NULL AS DS_DETALHES
        FROM
            ADMIN.BRASILEIRAO_TOTAL
    ))
    WHERE ID = (SELECT MAX(ID) FROM TB_LOGS WHERE NM_TABELA = 'DIM_POSICOES'
AND DS_EVENTO = 'MERGE');

EXCEPTION
    WHEN OTHERS THEN
        UPDATE TB_LOGS
        SET
            ST_STATUS = 'ERRO',
            DT_ALTERACAO = CURRENT_TIMESTAMP
        WHERE ID = (SELECT MAX(ID) FROM TB_LOGS WHERE NM_TABELA =
'DIM_POSICOES' AND DS_EVENTO = 'MERGE');
END SP_DIM_POSICOES;
```

SP_DIM_PARTIDAS

```
CREATE OR REPLACE EDITIONABLE PROCEDURE "ADMIN"."SP_DIM_PARTIDAS" IS
BEGIN
    INSERT INTO TB_LOGS(NM_TABELA, DS_EVENTO, ST_STATUS)
    VALUES ('DIM_PARTIDAS', 'MERGE', 'INICIADO');
    MERGE INTO DIM_PARTIDAS dest
    USING (
        SELECT
            ID_PARTIDA AS ID_FONTE,
            HOME || ' X ' || AWAY AS NM_PARTIDA,
            NULL AS DS_DETALHES,
            PARTIDAS_FINAL AS URL,
            REFEREE AS NM_ARBITRO,
            VENUE AS NM_ESTADIO
        FROM ADMIN.BRASILEIRAO_JOGOS
    ) src
    ON (src.ID_FONTE = dest.ID_FONTE)
    WHEN MATCHED THEN
        UPDATE SET
            dest.NM_PARTIDA = src.NM_PARTIDA,
            dest.DS_DETALHES = src.DS_DETALHES,
            dest.URL = src.URL,
            dest.NM_ARBITRO = src.NM_ARBITRO,
            dest.NM_ESTADIO = src.NM_ESTADIO,
            dest.DT_ALTERACAO = CURRENT_TIMESTAMP

    WHEN NOT MATCHED THEN
        INSERT (
            ID_FONTE,
            NM_PARTIDA,
            DS_DETALHES,
            URL,
            NM_ARBITRO,
            NM_ESTADIO
        )
        VALUES (
            src.ID_FONTE,
            src.NM_PARTIDA,
            src.DS_DETALHES,
            src.URL,
            src.NM_ARBITRO,
            src.NM_ESTADIO
        );

    UPDATE TB_LOGS
    SET
        ST_STATUS = 'FINALIZADO',
        DT_ALTERACAO = CURRENT_TIMESTAMP,
        NR_LINHAS = (SELECT COUNT(*) FROM (
        SELECT
            ID_PARTIDA AS ID_FONTE,
            HOME || ' X ' || AWAY AS NM_PARTIDA,
            NULL AS DS_DETALHES,
            PARTIDAS_FINAL AS URL,
            REFEREE AS NM_ARBITRO,
            VENUE AS NM_ESTADIO
        FROM ADMIN.BRASILEIRAO_JOGOS
    ))
    WHERE ID = (SELECT MAX(ID) FROM TB_LOGS WHERE NM_TABELA = 'DIM_PARTIDAS'
AND DS_EVENTO = 'MERGE');

EXCEPTION
    WHEN OTHERS THEN
```

```
        UPDATE TB_LOGS
        SET
            ST_STATUS = 'ERRO',
            DT_ALTERACAO = CURRENT_TIMESTAMP
        WHERE ID = (SELECT MAX(ID) FROM TB_LOGS WHERE NM_TABELA =
'DIM_PARTIDAS' AND DS_EVENTO = 'MERGE');
END SP_DIM_PARTIDAS;
```

SP_DIM_NUMERACOES

```
CREATE OR REPLACE EDITIONABLE PROCEDURE "ADMIN"."SP_DIM_NUMERACOES" IS
BEGIN
    INSERT INTO TB_LOGS(NM_TABELA, DS_EVENTO, ST_STATUS)
    VALUES ('DIM_NUMERACOES', 'MERGE', 'INICIADO');

    MERGE INTO DIM_NUMERACOES dest
    USING (
        SELECT DISTINCT
            NUMBER_RW as NR_NUMERACAO
        FROM
            ADMIN.BRASILEIRAO_TOTAL
    ) src
    ON (src.NR_NUMERACAO = dest.NR_NUMERACAO)
    WHEN MATCHED THEN
        UPDATE SET
            dest.DT_ALTERACAO = CURRENT_TIMESTAMP

    WHEN NOT MATCHED THEN
        INSERT (
            NR_NUMERACAO
        )
        VALUES (
            src.NR_NUMERACAO
        );

    UPDATE TB_LOGS
    SET
        ST_STATUS = 'FINALIZADO',
        DT_ALTERACAO = CURRENT_TIMESTAMP,
        NR_LINHAS = (SELECT COUNT(*) FROM (
        SELECT DISTINCT
            NUMBER_RW as NR_NUMERACAO
        FROM
            ADMIN.BRASILEIRAO_TOTAL
    ))
    WHERE ID = (SELECT MAX(ID) FROM TB_LOGS WHERE NM_TABELA = 'DIM_NUMERACOES'
AND DS_EVENTO = 'MERGE');

EXCEPTION
    WHEN OTHERS THEN
        UPDATE TB_LOGS
        SET
            ST_STATUS = 'ERRO',
            DT_ALTERACAO = CURRENT_TIMESTAMP
        WHERE ID = (SELECT MAX(ID) FROM TB_LOGS WHERE NM_TABELA =
'DIM_NUMERACOES' AND DS_EVENTO = 'MERGE');
```

```
END SP_DIM_NUMERACOES;
```

SP_DIM_JOGADORES

```
CREATE OR REPLACE EDITIONABLE PROCEDURE "ADMIN"."SP_DIM_JOGADORES" IS
BEGIN
    INSERT INTO TB_LOGS(NM_TABELA, DS_EVENTO, ST_STATUS)
    VALUES ('DIM_JOGADORES', 'MERGE', 'INICIADO');

    MERGE INTO DIM_JOGADORES dest
    USING (
        SELECT  DISTINCT
            JOGADOR_ID_ AS ID_FONTE,
            UPPER(PLAYER) AS NM_JOGADOR,
            UPPER(NATION) AS DS_NACIONALIDADE
        FROM
            ADMIN.BRASILEIRAO_TOTAL

    ) src
    ON (src.ID_FONTE = dest.ID_FONTE)
    WHEN MATCHED THEN
        UPDATE SET
            dest.NM_JOGADOR = src.NM_JOGADOR,
            dest.DS_NACIONALIDADE = src.DS_NACIONALIDADE,
            dest.DT_ALTERACAO = CURRENT_TIMESTAMP

    WHEN NOT MATCHED THEN
        INSERT (
            ID_FONTE,
            NM_JOGADOR,
            DS_NACIONALIDADE
        )
        VALUES (
            src.ID_FONTE,
            src.NM_JOGADOR,
            src.DS_NACIONALIDADE
        );

    UPDATE TB_LOGS
    SET
        ST_STATUS = 'FINALIZADO',
        DT_ALTERACAO = CURRENT_TIMESTAMP,
        NR_LINHAS = (SELECT COUNT(*) FROM (
        SELECT  DISTINCT
            JOGADOR_ID_ AS ID_FONTE,
            UPPER(PLAYER) AS NM_JOGADOR,
            UPPER(NATION) AS DS_NACIONALIDADE
        FROM
            ADMIN.BRASILEIRAO_TOTAL

    ))
    WHERE ID = (SELECT MAX(ID) FROM TB_LOGS WHERE NM_TABELA = 'DIM_JOGADORES'
AND DS_EVENTO = 'MERGE');

EXCEPTION
    WHEN OTHERS THEN
        UPDATE TB_LOGS
        SET
            ST_STATUS = 'ERRO',
            DT_ALTERACAO = CURRENT_TIMESTAMP
        WHERE ID = (SELECT MAX(ID) FROM TB_LOGS WHERE NM_TABELA =
'DIM_JOGADORES' AND DS_EVENTO = 'MERGE');
END SP_DIM_JOGADORES;
```

EXECUTE_ALL_IMPORT_PROCEDURES

```
CREATE OR REPLACE EDITIONABLE PROCEDURE
"ADMIN"."EXECUTE_ALL_IMPORT_PROCEDURES" AS
BEGIN

    SP_IMPORTAR_DADOS_STAGE('BRASILEIRAO_TOTAL_2024.csv','BRASILEIRAO_TOTAL');
    SP_IMPORTAR_DADOS_STAGE_JOGOS('BRASILEIRAO_JOGOS_2024.csv','BRASILEIRAO_JO
GOS');
    ADMIN.SP_DIM_JOGADORES;
    ADMIN.SP_DIM_NUMERACOES;
    ADMIN.SP_DIM_PARTIDAS;
    ADMIN.SP_DIM_POSICOES;
    ADMIN.SP_DIM_TIMES;
    ADMIN.SP_FATO;
END;
```

Scheduler

Foi criado um Job dentro do Autonomous BD para realizar a execução da procedure.

JOB's DDL

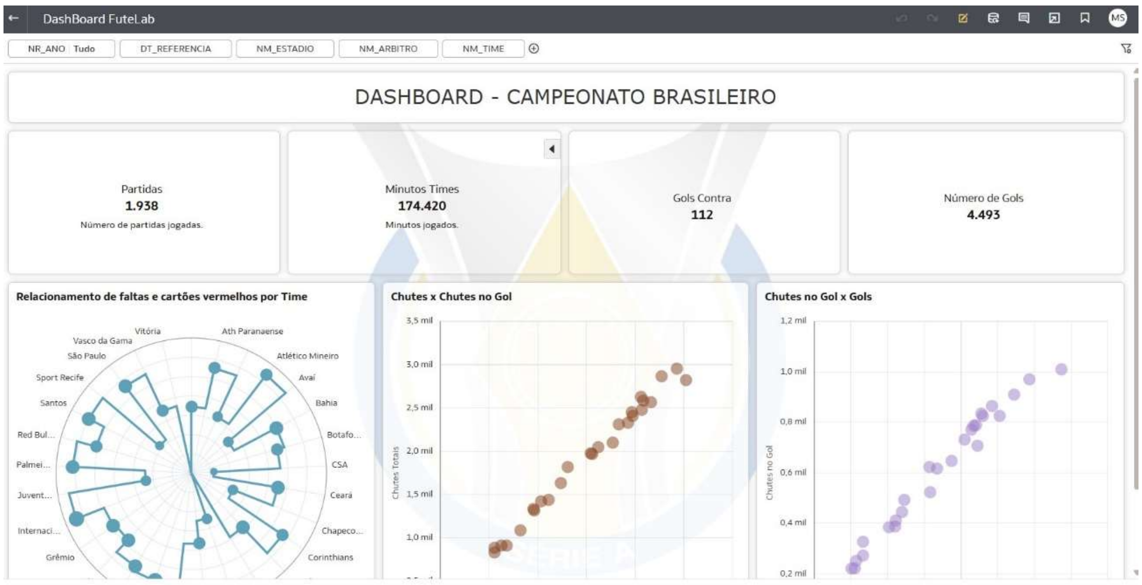
```
BEGIN
    SYS.DBMS_SCHEDULER.CREATE_JOB (
        job_name => 'ADMIN.FUTELAB_JOB_FINAL',
        job_type => 'PLSQL_BLOCK',
        job_action => 'BEGIN ADMIN.EXECUTE_ALL_IMPORT_PROCEDURES(); END;'
    );

    SYS.DBMS_SCHEDULER.SET_ATTRIBUTE(
        name => 'ADMIN.FUTELAB_JOB_FINAL',
        attribute => 'START_DATE',
        value => TO_TIMESTAMP_TZ('2024-05-03T07:56:42 +00:00','YYYY-MM-
DD"T"HH24:MI:SS.FF TZR'));
    SYS.DBMS_SCHEDULER.SET_ATTRIBUTE(
        name => 'ADMIN.FUTELAB_JOB_FINAL',
        attribute => 'REPEAT_INTERVAL',
        value => 'FREQ=DAILY; BYHOUR=2');
    SYS.DBMS_SCHEDULER.SET_ATTRIBUTE(
        name => 'ADMIN.FUTELAB_JOB_FINAL',
        attribute => 'STORE_OUTPUT',
        value => TRUE);
    SYS.DBMS_SCHEDULER.SET_ATTRIBUTE(
        name => 'ADMIN.FUTELAB_JOB_FINAL',
        attribute => 'NLS_ENV',
        value => 'NLS_LANGUAGE=''BRAZILIAN PORTUGUESE''
NLS_TERRITORY=''AMERICA'' NLS_CURRENCY=''$'' NLS_ISO_CURRENCY=''AMERICA''
NLS_NUMERIC_CHARACTERS=''.,'' NLS_CALENDAR=''GREGORIAN'' NLS_DATE_FORMAT=''DD-
MON-RR'' NLS_DATE_LANGUAGE=''BRAZILIAN PORTUGUESE'' NLS_SORT=''WEST_EUROPEAN''
NLS_TIME_FORMAT=''HH.MI.SSXFF AM'' NLS_TIMESTAMP_FORMAT=''DD-MON-RR
HH.MI.SSXFF AM'' NLS_TIME_TZ_FORMAT=''HH.MI.SSXFF AM TZR''
NLS_TIMESTAMP_TZ_FORMAT=''DD-MON-RR HH.MI.SSXFF AM TZR''
NLS_DUAL_CURRENCY=''$'' NLS_COMP=''BINARY'' NLS_LENGTH_SEMANTICS=''BYTE''
NLS_NCHAR_CONV_EXCP=''FALSE''');


    SYS.DBMS_SCHEDULER.enable(name => 'ADMIN.FUTELAB_JOB_FINAL');
END;
/
```

OCI Analytics

Foi criado um DASHBOARD no Oracle Cloud Analytics.



Link Apresentação