```
import pandas as pd
import seaborn as sns
df = pd.read_csv('fifa_player_performance_market_value.csv')

df.head(10)
```

```
   player_id player_name  age  nationality             club
position  \
0          1    Player_1   23      Germany        Liverpool       ST

1          2    Player_2   36      England     FC Barcelona       ST

2          3    Player_3   31       France         Juventus       RB

3          4    Player_4   27     Portugal  Manchester City       LW

4          5    Player_5   24       Brazil        Liverpool      CDM

5          6    Player_6   37    Argentina  Manchester City       CM

6          7    Player_7   23  Netherlands        Liverpool       RB

7          8    Player_8   35        Spain    Bayern Munich       LW

8          9    Player_9   39       Brazil     FC Barcelona       GK

9         10   Player_10   27      England        Liverpool       LB


   overall_rating  potential_rating  matches_played  goals  assists  \
0              65                87               8      6       14
1              90                76              19      3       18
2              75                91              34     12       15
3              90                86              35     18       13
4              84                96              41      6        6
5              92                91              35      9        7
6              72                66              53     24        6
7              69                97               8     34       17
8              83                90              21     24       23
9              69                92               0     28        5

   minutes_played  market_value_million_eur  contract_years_left
injury_prone  \
0            2976                    122.51                    3
No
1            2609                     88.47                    5
No
2            1158                     20.24                    3
No
3             145                    164.29                    0
Yes
```

```
4             2226                      121.34                      4
No
5              263                       98.51                      5
Yes
6             4299                       67.69                      1
No
7             3101                       24.71                      0
No
8             2106                      127.50                      3
Yes
9             3080                      146.55                      1
No
```

```
  transfer_risk_level
0                 Low
1                High
2              Medium
3              Medium
4                 Low
5                 Low
6                 Low
7                 Low
8                High
9                High
```

```
df.info(10)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2800 entries, 0 to 2799
Data columns (total 16 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   player_id                2800 non-null   int64
 1   player_name              2800 non-null   object
 2   age                      2800 non-null   int64
 3   nationality              2800 non-null   object
 4   club                     2800 non-null   object
 5   position                 2800 non-null   object
 6   overall_rating           2800 non-null   int64
 7   potential_rating         2800 non-null   int64
 8   matches_played           2800 non-null   int64
 9   goals                    2800 non-null   int64
 10  assists                  2800 non-null   int64
 11  minutes_played           2800 non-null   int64
 12  market_value_million_eur  2800 non-null   float64
 13  contract_years_left      2800 non-null   int64
 14  injury_prone             2800 non-null   object
 15  transfer_risk_level      2800 non-null   object
dtypes: float64(1), int64(9), object(6)
memory usage: 350.1+ KB
```

```
df.describe()
```

```
         player_id          age  overall_rating  potential_rating  \
count  2800.000000  2800.000000     2800.000000       2800.000000
mean   1400.500000    27.952500       76.866786         81.563929
std     808.434702     6.750192        9.921113          9.755799
min       1.000000    17.000000       60.000000         65.000000
25%     700.750000    22.000000       68.000000         73.000000
50%    1400.500000    28.000000       77.000000         82.000000
75%    2100.250000    34.000000       85.000000         90.000000
max    2800.000000    39.000000       94.000000         98.000000

       matches_played        goals      assists  minutes_played  \
count     2800.000000  2800.000000  2800.000000     2800.000000
mean        27.135714    19.261786    12.015000     2250.101429
std         15.979627    11.567858     7.188459     1295.461829
min          0.000000     0.000000     0.000000        0.000000
25%         13.750000     9.000000     6.000000     1131.250000
50%         27.000000    19.000000    12.000000     2251.000000
75%         41.000000    30.000000    18.000000     3366.250000
max         54.000000    39.000000    24.000000     4497.000000

       market_value_million_eur  contract_years_left
count               2800.000000          2800.000000
mean                  90.565500             2.527857
std                   52.078881             1.699445
min                    0.670000             0.000000
25%                   45.355000             1.000000
50%                   89.170000             3.000000
75%                  136.682500             4.000000
max                  179.960000             5.000000
```

```
df.shape
```

```
(2800, 16)
```

```
df.isnull().sum()
```

```
player_id                   0
player_name                 0
age                         0
nationality                 0
club                        0
position                    0
overall_rating              0
potential_rating            0
matches_played              0
goals                       0
assists                     0
minutes_played              0
market_value_million_eur    0
```

```
contract_years_left         0
injury_prone                0
transfer_risk_level         0
dtype: int64
```

```
df.columns
```

```
Index(['player_id', 'player_name', 'age', 'nationality', 'club',
'position',
       'overall_rating', 'potential_rating', 'matches_played',
'goals',
       'assists', 'minutes_played', 'market_value_million_eur',
       'contract_years_left', 'injury_prone', 'transfer_risk_level'],
      dtype='object')
```

```
df['nationality'].value_counts()
```

```
nationality
Brazil         380
France         371
Spain          358
England        357
Germany        345
Netherlands    338
Argentina      329
Portugal       322
Name: count, dtype: int64
```

```
df.nunique()
```

```
player_id                   2800
player_name                 2800
age                           23
nationality                    8
club                           7
position                       9
overall_rating                35
potential_rating              34
matches_played                55
goals                         40
assists                       25
minutes_played              2085
market_value_million_eur    2593
contract_years_left            6
injury_prone                   2
transfer_risk_level            3
dtype: int64
```

```
df.sample(10)
```

```
      player_id  player_name  age nationality             club
position  \
1105       1106  Player_1106   34       Spain      Real Madrid
RB
1116       1117  Player_1117   21     Germany        Liverpool
CB
442         443   Player_443   19       Spain         Juventus
CM
1577       1578  Player_1578   28    Portugal    Bayern Munich
GK
1304       1305  Player_1305   38       Spain        Liverpool
CDM
1934       1935  Player_1935   30       Spain  Manchester City
CM
1033       1034  Player_1034   17       Spain              PSG
ST
684         685   Player_685   36     Germany        Liverpool
CB
2788       2789  Player_2789   38   Argentina     FC Barcelona
ST
769         770   Player_770   27   Argentina         Juventus
CDM

      overall_rating  potential_rating  matches_played  goals  assists
\
1105              77                79              45     28       22

1116              72                90              42     30       17

442               75                70              52      6       24

1577              68                87              16     30       23

1304              73                86              36     23       16

1934              78                71              32     33       16

1033              83                92               9     23        5

684               92                89              45     36       20

2788              70                88              53     36       22

769               91                83              30     19       11


      minutes_played  market_value_million_eur  contract_years_left  \
1105            3417                    151.77                    3
1116            1310                    103.51                    1
442             1286                     96.35                    1
1577             280                     52.28                    2
```

```
1304              473                    144.15                    1
1934             1256                     7.54                    2
1033             2934                     47.26                   3
684              3566                    103.43                   4
2788             1066                     2.78                    3
769              3041                    121.34                   2
```

```
     injury_prone transfer_risk_level
1105           No               Medium
1116          Yes                  Low
442           Yes               Medium
1577           No                  Low
1304           No               Medium
1934           No               Medium
1033           No                 High
684           Yes               Medium
2788          Yes               Medium
769            No                  Low
```

```
df.duplicated().sum()
```

```
np.int64(0)
```

```
import pandas as pd
import numpy as np

def clean_player_data(df):
    """
    Membersihkan dataset dari pemain yang tidak memiliki data
performa.
    """
    # Menghapus pemain dengan menit bermain 0 karena tidak relevan
untuk scouting
    df_cleaned = df[df['minutes_played'] > 0].copy()
    return df_cleaned

def perform_feature_engineering(df):
    """
    Membuat fitur baru (metrik) untuk membantu analisis scouting.
    """
    # 1. Menghitung sisa potensi (Gap antara rating saat ini dan
potensial)
    df['potential_growth'] = df['potential_rating'] -
df['overall_rating']

    # 2. Normalisasi performa per 90 menit (Standar industri sepak
bola)
    df['goals_per_90'] = (df['goals'] / df['minutes_played']) * 90
    df['assists_per_90'] = (df['assists'] / df['minutes_played']) * 90
```

```
    # 3. Scouting Score: Mengutamakan pemain muda dengan potensi
tinggi dan harga masuk akal
    # Rumus: (Potensi / Usia) + (Kontribusi Gol & Assist per 90)
    df['scouting_score'] = (df['potential_rating'] / df['age']) +
(df['goals_per_90'] + df['assists_per_90'])

    return df

def encode_categorical_data(df):
    """
    Mengubah kolom teks menjadi representasi numerik untuk pemrosesan
lebih lanjut.
    """
    # Mengubah injury_prone menjadi biner (0/1)
    df['is_injury_prone'] = df['injury_prone'].map({'Yes': 1, 'No':
0})

    # Ordinal Encoding untuk Transfer Risk Level
    risk_mapping = {'Low': 0, 'Medium': 1, 'High': 2}
    df['transfer_risk_numeric'] =
df['transfer_risk_level'].map(risk_mapping)

    return df

def get_top_scouting_targets(df, top_n=10):
    """
    Mengambil daftar pemain terbaik berdasarkan skor scouting.
    """
    return df.sort_values(by='scouting_score',
ascending=False).head(top_n)
```

PIPELINE

```
# 1. Tahap Preparation
df_prepared = clean_player_data(df)
df_prepared = perform_feature_engineering(df_prepared)
df_prepared = encode_categorical_data(df_prepared)

# 2. Hasil: Top 10 Wonderkids/Targets
top_targets = get_top_scouting_targets(df_prepared)

print("Data Preparation Selesai. Siap untuk tahap
Modeling/Evaluation.")
print(top_targets[['player_name', 'age', 'club', 'potential_growth',
'scouting_score']])
```

```
Data Preparation Selesai. Siap untuk tahap Modeling/Evaluation.
      player_name  age             club  potential_growth
scouting_score
2661  Player_2662   27        Liverpool                -8
1352.481481
1774  Player_1775   31    Bayern Munich                 9
956.903226
2576  Player_2577   20         Juventus                -3
423.950000
2402  Player_2403   34              PSG                16
289.698529
1688  Player_1689   31              PSG                11
250.241935
1279  Player_1280   21      Real Madrid                31
218.416667
1809  Player_1810   18              PSG                18
209.712121
1661  Player_1662   20    Bayern Munich                -3
201.450000
1685  Player_1686   17              PSG                19
198.445378
732    Player_733   26  Manchester City                 2
175.000000
```

```python
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

def prepare_features_for_modeling(df):
    """
    Memilih fitur kunci dan melakukan standarisasi (scaling).
    K-Means sangat sensitif terhadap skala data.
    """
    # Fitur untuk menentukan profile pemain
    features = [
        'age', 'overall_rating', 'potential_rating',
        'market_value_million_eur', 'scouting_score'
    ]

    scaler = StandardScaler()
    scaled_data = scaler.fit_transform(df[features])

    return scaled_data, features

def apply_player_clustering(df, n_clusters=3):
    """
    Mengelompokkan pemain menjadi beberapa kategori (Cluster).
    """
    scaled_data, _ = prepare_features_for_modeling(df)

    # Inisialisasi dan fit model KMeans
```

```
    kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init=10)
    df['player_cluster'] = kmeans.fit_predict(scaled_data)

    return df

def interpret_clusters(df):
    """
    Memberikan label manusiawi pada hasil cluster berdasarkan
karakteristiknya.
    """
    cluster_summary = df.groupby('player_cluster').agg({
        'age': 'mean',
        'overall_rating': 'mean',
        'market_value_million_eur': 'mean',
        'scouting_score': 'mean'
    }).sort_values('scouting_score', ascending=False)

    return cluster_summary
```

MODELING

```
# 1. Melakukan clustering pada data yang udah di-prepare sebelumnya
df_modeled = apply_player_clustering(df_prepared, n_clusters=3)

# 2. Interpretasi Hasil
print("Ringkasan Karakteristik Cluster:")
print(interpret_clusters(df_modeled))
```

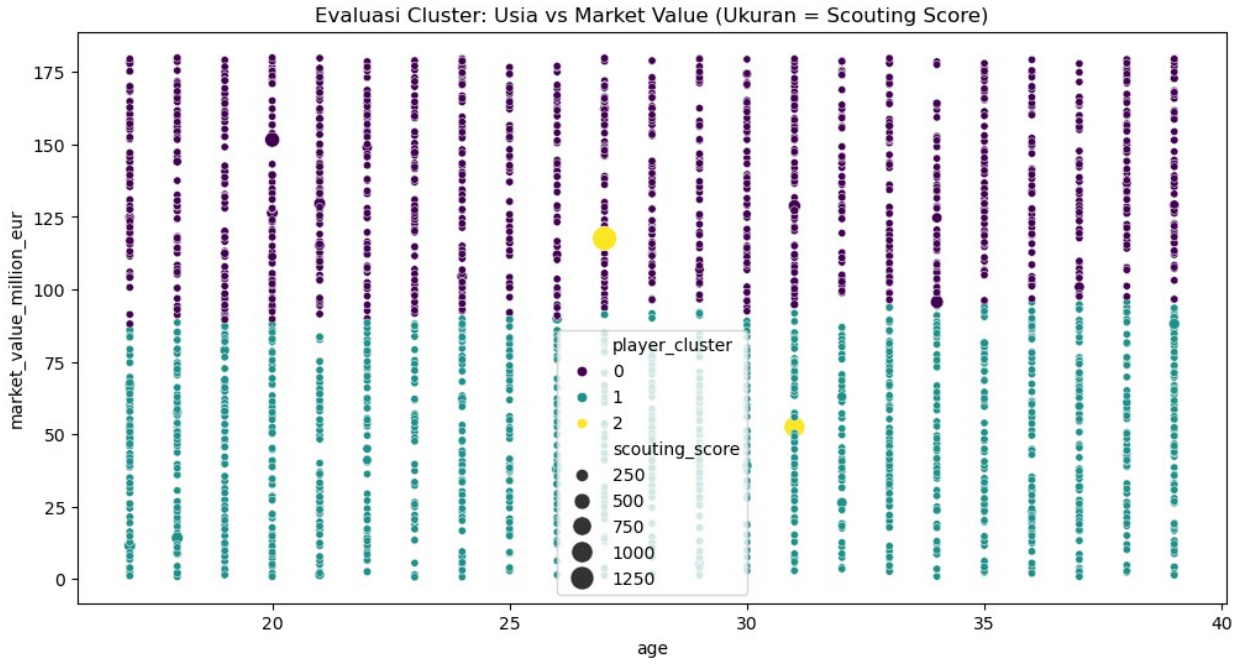
```
Ringkasan Karakteristik Cluster:
                      age  overall_rating  market_value_million_eur  \
player_cluster
2               29.000000       78.000000                 85.040000
0               27.616458       76.957384                137.052248
1               28.275766       76.782033                 46.568524

                scouting_score
player_cluster
2                 1154.692354
0                    7.972151
1                    7.224928
```

```
import matplotlib.pyplot as plt
import seaborn as sns

def evaluate_scouting_model(df):
    """
    Memvisualisasikan distribusi cluster untuk memastikan segmentasi
```

```
logis.
    """
    plt.figure(figsize=(12, 6))

    # Visualisasi hubungan antara Usia dan Market Value berdasarkan
Cluster
    sns.scatterplot(
        data=df,
        x='age',
        y='market_value_million_eur',
        hue='player_cluster',
        palette='viridis',
        size='scouting_score',
        sizes=(20, 200)
    )
    plt.title('Evaluasi Cluster: Usia vs Market Value (Ukuran =
Scouting Score)')
    plt.show()

def deploy_scouting_report(df, cluster_id,
filename='top_scouting_targets.csv'):
    """
    Menyimpan daftar target pemain terbaik ke dalam CSV.
    """
    # Mengambil pemain dari cluster yang paling menjanjikan (Cluster
2)
    target_players = df[df['player_cluster'] ==
cluster_id].sort_values(
        by='scouting_score', ascending=False
    )

    # Kolom terpilih untuk laporan manajemen
    report_columns = [
        'player_name', 'age', 'nationality', 'club', 'position',
        'overall_rating', 'potential_rating',
'market_value_million_eur',
        'scouting_score'
    ]

    final_report = target_players[report_columns]
    final_report.to_csv(filename, index=False)

    print(f"Laporan berhasil dikirim! {len(final_report)} pemain
tersimpan di {filename}")
    return final_report.head(10)
```

TAHAP AKHIR

```
# 1. Evaluasi model secara visual
evaluate_scouting_model(df_modeled)

# 2. Deployment: Ambil 10 besar dari Cluster 2 (The Scouting Gems)
print("\n DAFTAR TARGET UTAMA (DEPLOYMENT) ")
final_targets = deploy_scouting_report(df_modeled, cluster_id=2)
print(final_targets)
```




```
 DAFTAR TARGET UTAMA (DEPLOYMENT)
Laporan berhasil dikirim! 2 pemain tersimpan di
top_scouting_targets.csv
      player_name  age nationality           club position
overall_rating  \
2661  Player_2662   27    Portugal      Liverpool       RW
75
1774  Player_1775   31      France  Bayern Munich       RW
81

      potential_rating  market_value_million_eur  scouting_score
2661                67                    117.59     1352.481481
1774                90                     52.49      956.903226

def refined_data_preparation(df):
    """
    Menambah filter menit bermain untuk stabilitas statistik.
    """
```

```
    # Filtering: Minimal 900 menit bermain agar data statistik valid
    df_filtered = df[df['minutes_played'] >= 900].copy()

    # Menghitung ulang dari metrik dasar
    df_filtered['goals_per_90'] = (df_filtered['goals'] /
df_filtered['minutes_played']) * 90
    df_filtered['assists_per_90'] = (df_filtered['assists'] /
df_filtered['minutes_played']) * 90

    return df_filtered

def calculated_weighted_score(df):
    """
    Rumus skor yang lebih seimbang (Weighted Score).
    """
    # bobot: 70% Atribut Dasar, 30% Efisiensi Lapangan
    base_quality = (df['overall_rating'] * 0.4) +
(df['potential_rating'] * 0.6)
    efficiency = (df['goals_per_90'] + df['assists_per_90']) * 10 #
Skala disesuaikan

    # Penalti Usia: Pemain di atas 30 tahun mendapatkan sedikit
pengurangan skor scouting
    age_penalty = df['age'].apply(lambda x: 0.9 if x > 30 else 1.0)

    df['refined_scouting_score'] = (base_quality + efficiency) *
age_penalty
    return df
```

REFINEMENT

```
def prepare_features_for_modeling_refined(df):

    features = [
        'age', 'overall_rating', 'potential_rating',
        'market_value_million_eur', 'refined_scouting_score' # Nama
kolom udah sinkron
    ]

    scaler = StandardScaler()
    scaled_data = scaler.fit_transform(df[features])

    return scaled_data, features

def apply_player_clustering_refined(df, n_clusters=3):

    # Memastikan kita memanggil fungsi yang udah menggunakan fitur
terbaru
```

```
    scaled_data, _ = prepare_features_for_modeling_refined(df)

    kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init=10)
    df['player_cluster'] = kmeans.fit_predict(scaled_data)

    return df


def create_scouting_dashboard(df):
    """
    Visualisasi untuk pengambilan keputusan manajemen.
    """
    plt.figure(figsize=(12, 7))

    # Membuat Scatter Plot Harga vs Kualitas
    sns.scatterplot(
        data=df,
        x='market_value_million_eur',
        y='refined_scouting_score',
        hue='player_cluster',
        style='injury_prone', # Melihat faktor risiko cedera
        palette='viridis',
        alpha=0.7
    )

    plt.title('Scouting Map: Market Value vs. Refined Scouting Score')
    plt.xlabel('Market Value (Million EUR)')
    plt.ylabel('Scouting Score (Refined)')
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.show()

def finalize_scouting_project(df):
    """
    Menyimpan (2800 pemain) yang udah diberi skor dan cluster.
    """
    # Mengurutkan berdasarkan skor tertinggi
    df_final = df.sort_values(by='refined_scouting_score',
ascending=False)

    # Menyimpan hasil ke CSV
    df_final.to_csv('complete_scouting_database_2026.csv',
index=False)

    print(f"Deployment Berhasil: Database 2800 pemain telah
diperbarui.")
    return df_final.head(10)
```

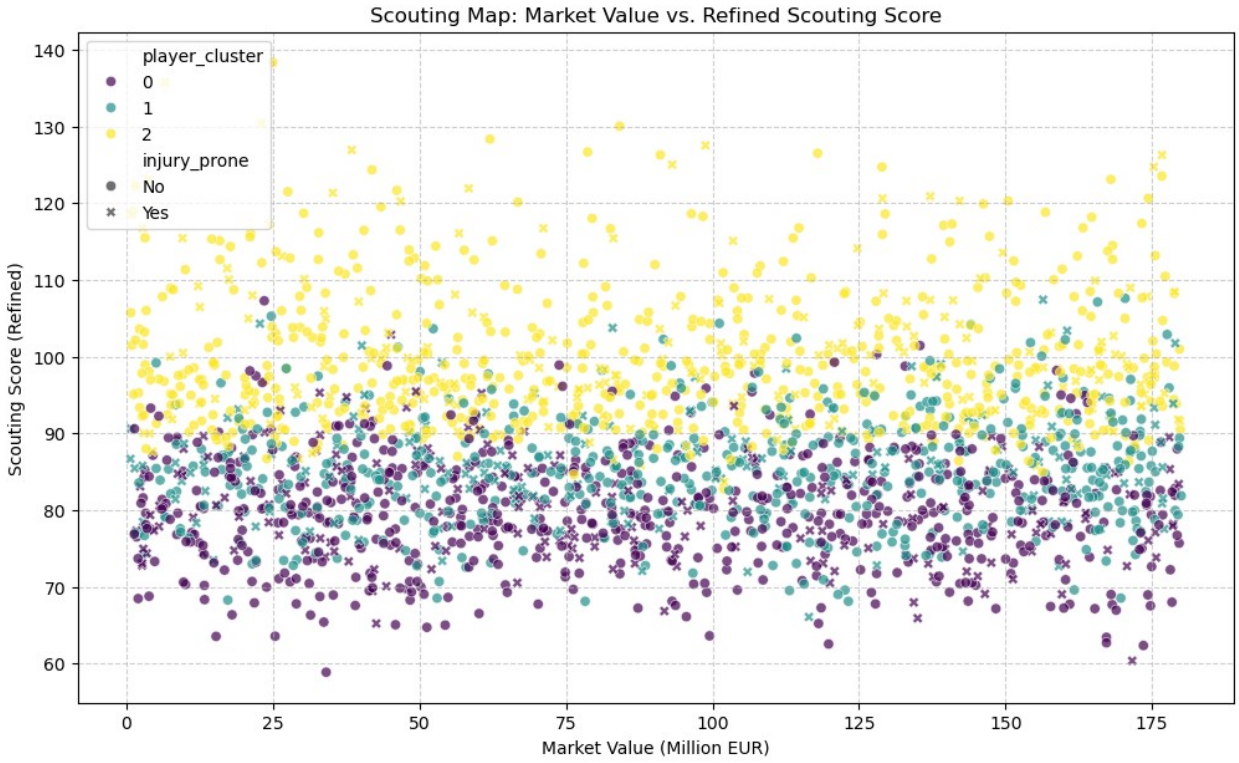
ULANG ALUR REFINEMENT

```
# 1. Preparation: Filter & Scoring (udah 900 menit)
df_refined = refined_data_preparation(df) # Fungsi dari chat
sebelumnya
df_refined = calculated_weighted_score(df_refined) # Menghasilkan
'refined_scouting_score'

# 2. Modeling: Jalankan Clustering dengan fitur yang sinkron
df_final_modeled = apply_player_clustering_refined(df_refined,
n_clusters=3)

# 3. Hasil: 10 Besar yang udah stabil (minimal 900 menit)
top_10_refined =
df_final_modeled.sort_values('refined_scouting_score',
ascending=False).head(10)

print("Berikut adalah Top 10 Target dengan data yang lebih stabil:")
print(top_10_refined[['player_name', 'age', 'overall_rating',
'potential_rating', 'refined_scouting_score']])
```

```
Berikut adalah Top 10 Target dengan data yang lebih stabil:
      player_name  age  overall_rating  potential_rating  \
2259  Player_2260   23              70                91
119    Player_120   24              83                84
79      Player_80   24              92                81
409    Player_410   19              93                87
240    Player_241   24              91                86
2185  Player_2186   26              80                96
2457  Player_2458   25              86                94
934    Player_935   20              90                78
2600  Player_2601   19              72                88
742    Player_743   29              70                77

      refined_scouting_score
2259              138.352212
119               135.734146
79                130.400000
409               130.049762
240               128.358744
2185              127.537220
2457              126.950628
934               126.680597
2600              126.516821
742               126.295808
```

DEPLOYMENT

```
# 1. Visualisasi untuk Presentasi
create_scouting_dashboard(df_final_modeled)

# 2. Finalisasi dan Pengiriman Laporan
top_recommendations = finalize_scouting_project(df_final_modeled)
print(top_recommendations[['player_name', 'club', 'position',
'refined_scouting_score']])
```




```
Deployment Berhasil: Database 2800 pemain telah diperbarui.
      player_name             club position  refined_scouting_score
2259  Player_2260    Bayern Munich       CB              138.352212
119    Player_120  Manchester City       RB              135.734146
79      Player_80         Juventus       LW              130.400000
409    Player_410    Bayern Munich       RB              130.049762
240    Player_241              PSG       ST              128.358744
2185  Player_2186      Real Madrid       RB              127.537220
2457  Player_2458        Liverpool      CDM              126.950628
934    Player_935  Manchester City       ST              126.680597
2600  Player_2601         Juventus       CB              126.516821
742    Player_743         Juventus       CM              126.295808
```

```
def generate_automated_insights(df_final):
    # Top 10
    top_10 = df_final.sort_values('refined_scouting_score',
```

```
ascending=False).head(10)

    print("INSIGHT")

    # Validasi Insight 1: Dominasi Usia
    avg_age = top_10['age'].mean()
    print(f"1. Rata-rata Usia Top Target: {avg_age:.1f} tahun (Usia
Muda)")

    # Validasi Insight 2: Sebaran Klub
    club_dist = top_10['club'].value_counts()
    print(f"2. Dominasi Klub di Top 10:\n{club_dist}")

    # Validasi Insight 3: Efisiensi per Posisi
    pos_dist = top_10['position'].value_counts()
    print(f"3. Distribusi Posisi Kuat: {pos_dist.index[0]}
({pos_dist.values[0]} pemain)")

generate_automated_insights(df_final_modeled)
```

```
INSIGHT
1. Rata-rata Usia Top Target: 23.3 tahun (Usia Muda)
2. Dominasi Klub di Top 10:
club
Juventus            3
Bayern Munich       2
Manchester City     2
PSG                 1
Real Madrid         1
Liverpool           1
Name: count, dtype: int64
3. Distribusi Posisi Kuat: RB (3 pemain)
```

WHAT DO I GET IN MY ASS DS PROJECT ?