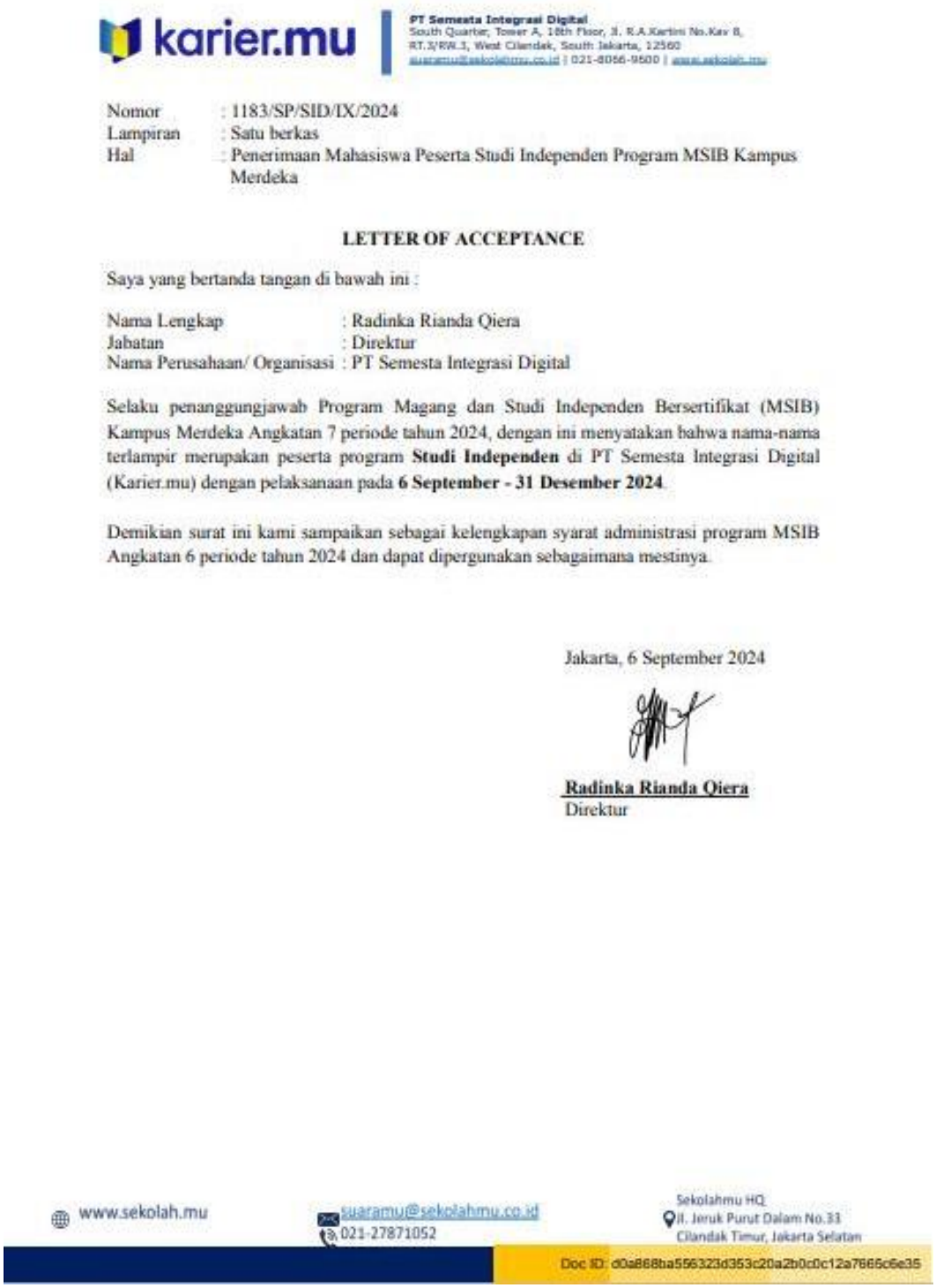
LAMPIRAN

Lampiran 1. LoA Penerimaan Mitra PKL



karier.mu

PT Semesta Integrasi Digital
South Quarter, Tower A, 18th Floor, Jl. R.A.Kartini No.Kav 8,
RT.5/RW.3, West Cilandak, South Jakarta, 12560
suaramu@sekolahmu.co.id | 021-8056-9600 | www.sekolah.mu

Nomor : 1183/SP/SID/IX/2024
Lampiran : Satu berkas
Hal : Penerimaan Mahasiswa Peserta Studi Independen Program MSIB Kampus Merdeka

LETTER OF ACCEPTANCE

Saya yang bertanda tangan di bawah ini :

Nama Lengkap : Radinka Rianda Qiera
Jabatan : Direktur
Nama Perusahaan/ Organisasi : PT Semesta Integrasi Digital

Selaku penanggungjawab Program Magang dan Studi Independen Bersertifikat (MSIB) Kampus Merdeka Angkatan 7 periode tahun 2024, dengan ini menyatakan bahwa nama-nama terlampir merupakan peserta program **Studi Independen** di PT Semesta Integrasi Digital (Karier.mu) dengan pelaksanaan pada **6 September - 31 Desember 2024**.

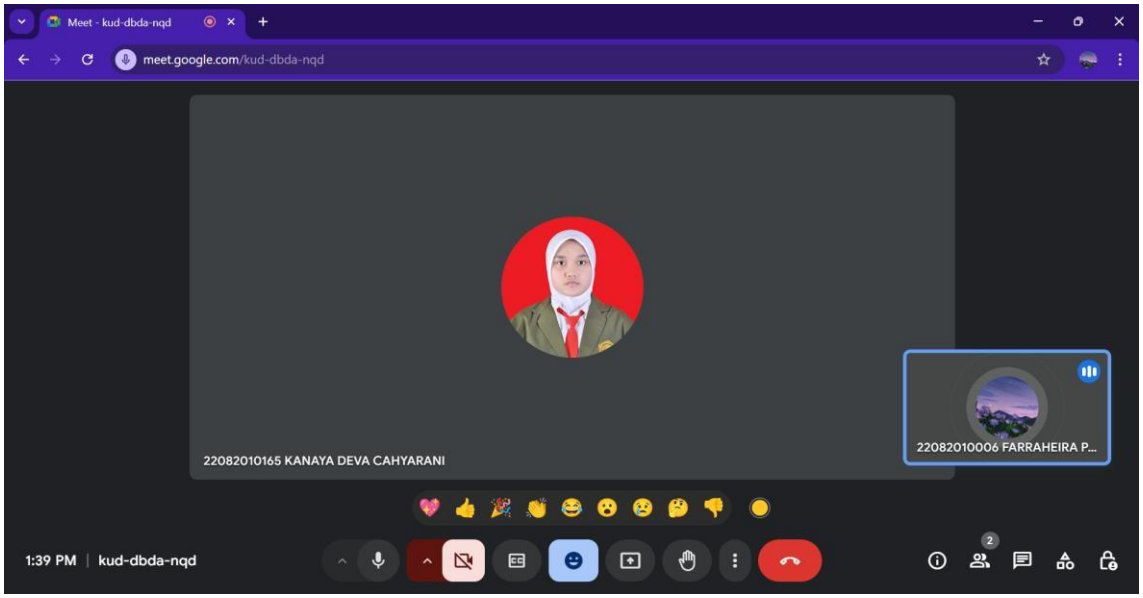
Demikian surat ini kami sampaikan sebagai kelengkapan syarat administrasi program MSIB Angkatan 6 periode tahun 2024 dan dapat dipergunakan sebagaimana mestinya.

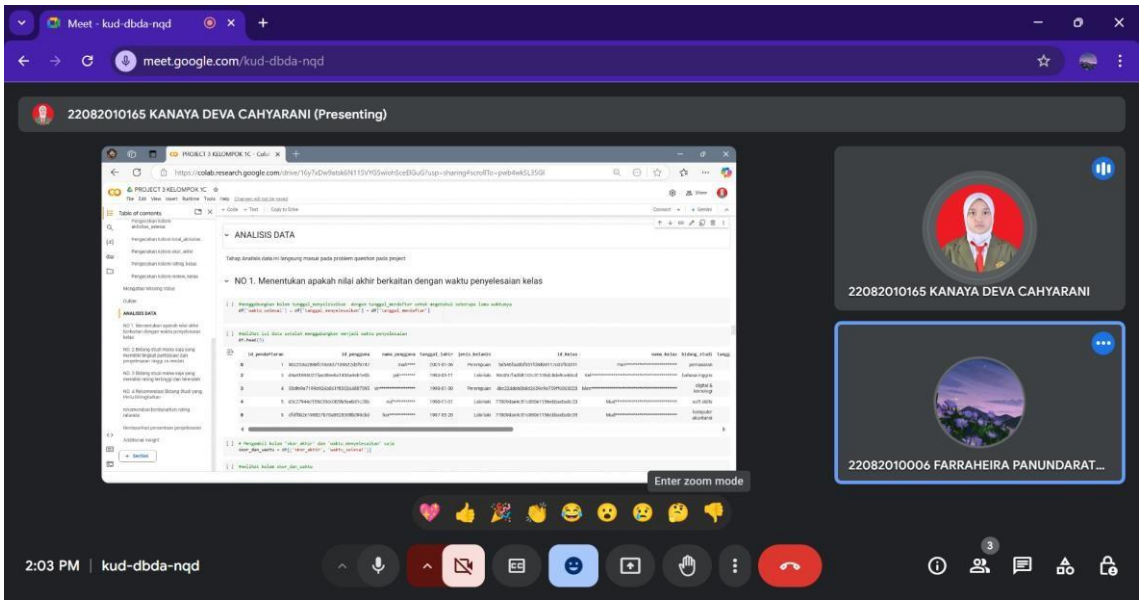
Jakarta, 6 September 2024

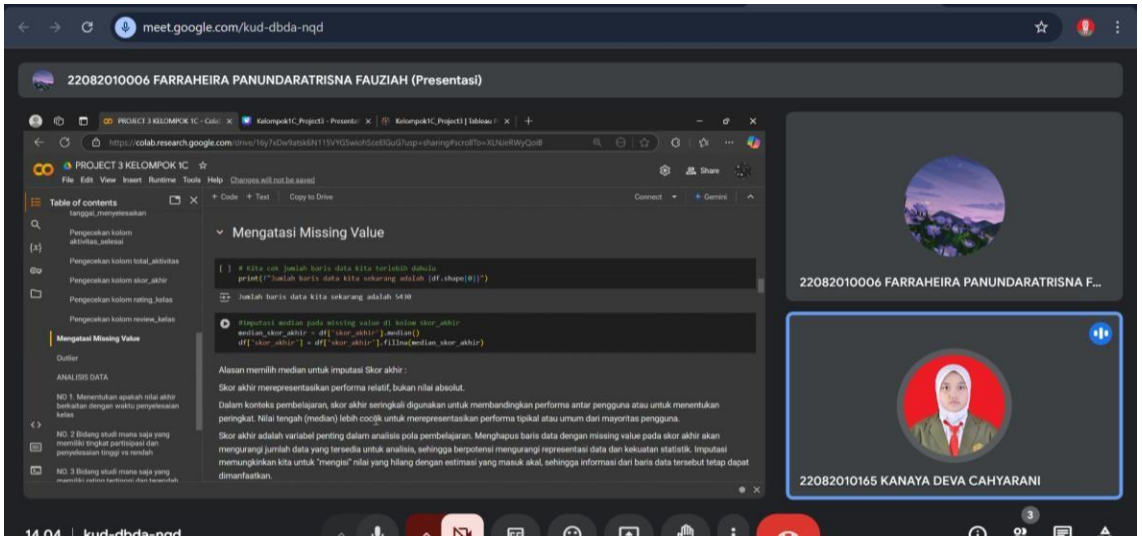
Radinka Rianda Qiera
Direktur

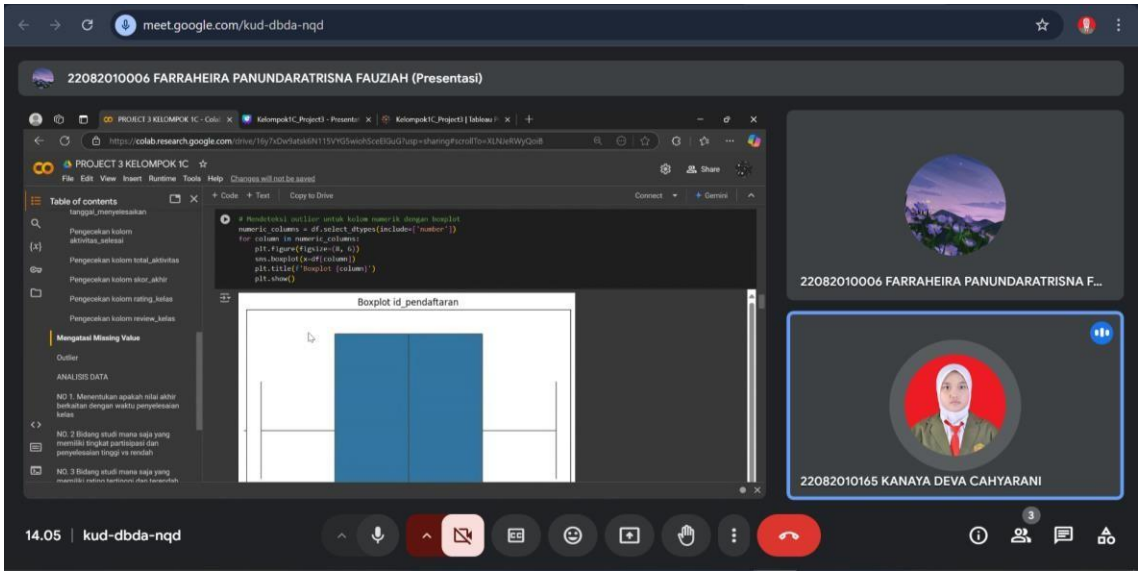
www.sekolah.mu
suaramu@sekolahmu.co.id
021-27871052
Sekolahmu HQ
Jl. Jeruk Purut Dalam No.33
Cilandak Timur, Jakarta Selatan
Doc ID: d0a868ba556323d353c20a2b0c0c12a7666c6e35

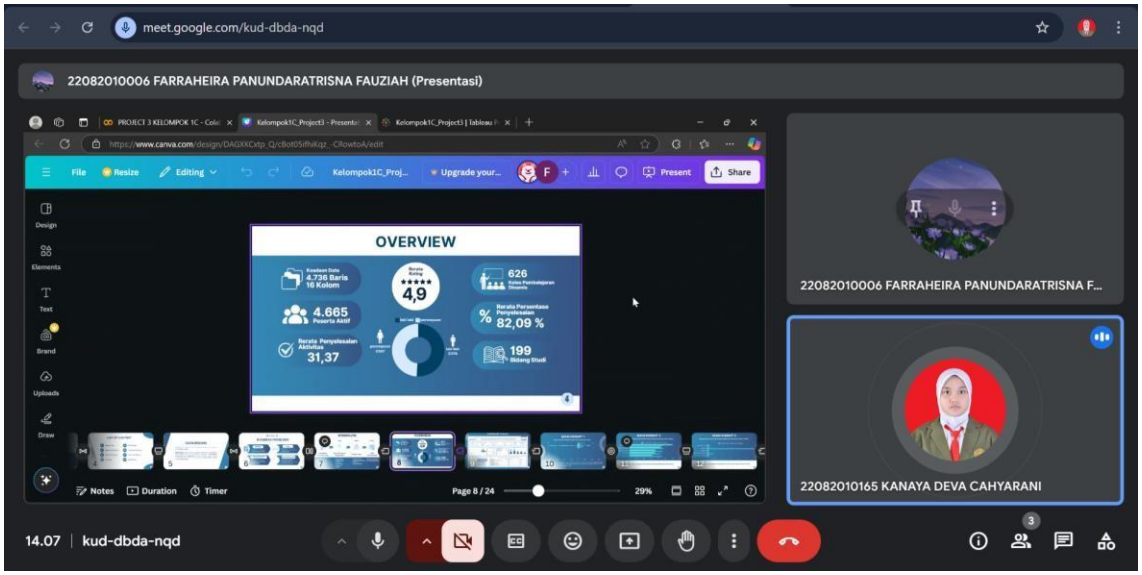
Lampiran 2. Dokumentasi Kegiatan






meet.google.com/kud-dbda-nqd
22082010006 FARRAHEIRA PANUNDARATRISNA FAUZIAH (Presentasi)
PROJECT 3 KELOMPOK 1C
Mengatasi Missing Value
Alasan memilih median untuk imputasi Skor akhir :
Skor akhir merepresentasikan performa relatif, bukan nilai absolut.
Dalam konteks pembelajaran, skor akhir seringkali digunakan untuk membandingkan performa antar pengguna atau untuk menentukan peringkat. Nilai tengah (median) lebih cocjik untuk merepresentasikan performa tipikal atau umum dari mayoritas pengguna.
Skor akhir adalah variabel penting dalam analisis pola pembelajaran. Menghapus baris data dengan missing value pada skor akhir akan mengurangi jumlah data yang tersedia untuk analisis, sehingga berpotensi mengurangi representasi data dan kekuatan statistik. Imputasi memungkinkan kita untuk "mengisi" nilai yang hilang dengan estimasi yang masuk akal, sehingga informasi dari baris data tersebut tetap dapat dimanfaatkan.
22082010006 FARRAHEIRA PANUNDARATRISNA F...
22082010165 KANAYA DEVA CAHYARANI


meet.google.com/kud-dbda-nqd
22082010006 FARRAHEIRA PANUNDARATRISNA FAUZIAH (Presentasi)
PROJECT 3 KELOMPOK 1C
Boxplot id_pendaftaran
22082010006 FARRAHEIRA PANUNDARATRISNA F...
22082010165 KANAYA DEVA CAHYARANI
14.05 | kud-dbda-nqd


meet.google.com/kud-dbda-nqd
22082010006 FARRAHEIRA PANUNDARATRISNA FAUZIAH (Presentasi)
OVERVIEW
4.736 Baris 16 Kolom
4,9
626
4.665
82,09 %
31,37
199
22082010006 FARRAHEIRA PANUNDARATRISNA F...
22082010165 KANAYA DEVA CAHYARANI
14.07 | kud-dbda-nqd

Lampiran 3. Source Code

1. Library

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Input dan Screening Data Awal

```
df = pd.read_excel('/content/Project 3 _ Analisis pola pembelajaran pengguna kelas pelatihan kerja pada platform MasaDepan.ku..xlsx')
# tampilkan jumlah baris dan data
df.shape
# tampilkan 5 baris pertama
df.head()
# Melihat distribusi dari kolom numerik
numeric_columns =
df.select_dtypes(include=['number']).columns

for col in numeric_columns:
    plt.figure(figsize=(10, 5))
    sns.histplot(df[col], kde=True)
    plt.title(f'Distribusi Kolom {col}')
    plt.xlabel(col)
    plt.ylabel('Frekuensi')
    plt.show()
```

3. Missing Value

```
# hitung jumlah missing value di tiap kolom
df.isna().sum()
# Kita cek jumlah baris data kita terlebih dahulu
```

```
print(f"Jumlah baris data kita sekarang adalah {df.shape[0]}")
#imputasi median pada missing value di kolom skor_akhir
median_skor_akhir = df['skor_akhir'].median()
df['skor_akhir'] = df['skor_akhir'].fillna(median_skor_akhir)
#imputasi missing value pada kolom review_kelas dengan "pengguna tidak memberikan review"
df['review_kelas'].fillna('pengguna tidak memberikan review', inplace=True)
#imputasi missing value pada kolom tanggal_menyelesaikan dengan median
median_tanggal_menyelesaikan = df['tanggal_menyelesaikan'].median()
df['tanggal_menyelesaikan'] = df['tanggal_menyelesaikan'].fillna(median_tanggal_menyelesaikan)
# mengecek kembali jumlah missing value
df.isnull().sum()
```

4. Typos

```
#mengubah 'design' menjadi 'desain' agar seragam
df['bidang_studi'] = df['bidang_studi'].replace('design', 'desain')

#mengubah 'perecanaan bisnis' menjadi 'perencanaan bisnis'
df['bidang_studi'] = df['bidang_studi'].replace('perecanaan bisnis', 'perencanaan bisnis')
```

5. Outlier

```
# Mendeteksi outlier untuk kolom numerik dengan boxplot
numeric_columns = df.select_dtypes(include=['number'])
for column in numeric_columns:
    plt.figure(figsize=(8, 6))
    sns.boxplot(x=df[column])
    plt.title(f'Boxplot {column}')
```

```
    plt.show()
# Mendeteksi outlier untuk kolom numerik dengan boxplot
numeric_columns = df.select_dtypes(include=['number'])
for column in numeric_columns:
    plt.figure(figsize=(8, 6))
    sns.boxplot(x=df[column])
    plt.title(f'Boxplot {column}')
    plt.show()
# Menghapus Outlier
def remove_outliers_iqr(df, column):
  Q1 = df[column].quantile(0.25)
  Q3 = df[column].quantile(0.75)
  IQR = Q3 - Q1
  lower_bound = Q1 - 1.5 * IQR
  upper_bound = Q3 + 1.5 * IQR
  df = df[(df[column] >= lower_bound)    &  (df[column]  <=
upper_bound)]
  return df

# Kolom yang mau dihapus outlier nya
for column in ['aktivitas_selesai', 'total_aktivitas',
'skor_akhir']:
  df = remove_outliers_iqr(df, column)

# Memastikan apakah masih terdapat outlier
def count_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df[(df[column] < lower_bound) | (df[column]
> upper_bound)]
    return len(outliers)

for  column  in  ['aktivitas_selesai',  'total_aktivitas',
'skor_akhir']:
```

```
    num_outliers = count_outliers_iqr(df, column)
    print(f"Number of outliers in column {column}:
{num_outliers}")
# mengecek outlier setelah kita menghapus outlier
menggunakan boxplot
numeric_columns = df.select_dtypes(include=['number'])
for column in numeric_columns:
    plt.figure(figsize=(8, 6))
    sns.boxplot(x=df[column])
    plt.title(f'Boxplot {column}')
    plt.show()
from scipy.stats.mstats import winsorize

# Winsorize 'skor_akhir' column at the 5th and 95th
percentiles
df['skor_akhir'] = winsorize(df['skor_akhir'],
limits=[0.05, 0.05])

# Mengecek apakah masih terdapat oulier
num_outliers = count_outliers_iqr(df, 'skor_akhir')
print(f"Number of outliers in column skor_akhir:
{num_outliers}")
# mendeteksi outlier untuk kolom numerik dengan boxplot
setelah melakukan treatment winsorize pada kolom
skor_akhir
numeric_columns = df.select_dtypes(include=['number'])
for column in numeric_columns:
    plt.figure(figsize=(8, 6))
    sns.boxplot(x=df[column])
    plt.title(f'Boxplot {column}')
    plt.show()
```

6. Eksport Data Bersih

```
df.shape
# export data ke excel
df.to_excel('data_clean.xlsx', index=False)
```

7. Data Analytics: BQ1

```
#menggabungkan kolom tanggal_menyelesaikan dengan
tanggal_mendaftar untuk megetahui seberapa lama waktunya
df['waktu_selesai'] = df['tanggal_menyelesaikan'] -
df['tanggal_mendaftar']
#melihat isi data setelah menggabungkan menjadi waktu
penyelesaian
df.head(5)
# Mengambil kolom 'skor_akhir' dan 'waktu_menyelesaikan'
saja
skor_dan_waktu = df[['skor_akhir', 'waktu_selesai']]
#melihat kolom skor_dan_waktu
print(skor_dan_waktu)
# Ubah kolom 'waktu_selesai' menjadi tipe timedelta
skor_dan_waktu['waktu_selesai'] =
pd.to_timedelta(skor_dan_waktu['waktu_selesai'])

# Konversi waktu menjadi hari dalam bentuk numerik
skor_dan_waktu['waktu_selesai_hari'] =
skor_dan_waktu['waktu_selesai'].dt.total_seconds() /
(3600 * 24)

# Tampilkan hasilnya
print(skor_dan_waktu[['waktu_selesai',
'waktu_selesai_hari']])
#melihat kolom skor_dan_waktu
print(skor_dan_waktu)
skor_dan_waktu.info()
#mengimport library yang diperlukan untuk uji normalitas
from scipy import stats
from scipy.stats import levene, normaltest, spearmanr
import scipy.stats as stats
import statsmodels.formula.api as smf
from statsmodels.stats.diagnostic import het_breuschpagan
import statsmodels.formula.api as smf
```

```
#melakukan uji normalitas dengan kolmogorov-smirnov
skor_akhir = stats.kstest(skor_dan_waktu['skor_akhir'],
'norm')
waktu_selesai_hari =
stats.kstest(skor_dan_waktu['waktu_selesai_hari'],
'norm')

#menampilkan hasil uji
print("Kolmogorov-Smirnov Test - Skor Akhir:")
print("statistic:", skor_akhir.statistic)
print("p-value:", skor_akhir.pvalue)

print("Kolmogorov-Smirnov Test - Waktu Selesai hari:")
print("statistic:", waktu_selesai_hari.statistic)
print("p-value:", waktu_selesai_hari.pvalue)

if skor_akhir.pvalue > 0.05:
    print("Skor Akhir memiliki distribusi normal.")
else:
    print("Skor Akhir tidak memiliki distribusi normal.")
#melakukan uji heteroskedastisitas (levene's test)
statistic_levene, p_value_levene =
levene(skor_dan_waktu['skor_akhir'],
skor_dan_waktu['waktu_selesai_hari'])

print("uji heterokedastisitas")
print("statistic:", statistic_levene)
print("p-value:", p_value_levene)

#mencetak hasil uji heterokedastisitas
if p_value_levene < 0.05:
    print("Data tidak homogen, ada heteroskedastisitas
atau varian tidak sama.")
else:
    print("Data homogen, tidak ada heteroskedastisitas
atau varian sama.")
from scipy.stats import spearmanr
```

```
# Melakukan uji korelasi Spearman antara dua variabel
korelasi_spearman, p_value_spearman =
spearmanr(skor_dan_waktu['skor_akhir'],
skor_dan_waktu['waktu_selesai_hari'])

# Menampilkan hasil korelasi dan p-value
print("Korelasi Spearman:", korelasi_spearman)
print("P-value:", p_value_spearman)

if p_value_spearman < 0.05:
    print("Tidak terdapat korelasi antara skor_akhir dan
waktu_selesai_hari.")
else:
    print("Ada korelasi antara skor_akhir dan
waktu_selesai_hari.")
# Buat scatter plot menggunakan seaborn
plt.figure(figsize=(8,6))
sns.scatterplot(x=skor_dan_waktu['skor_akhir'],
y=skor_dan_waktu['waktu_selesai_hari'])
plt.title('Korelasi antara Skor Akhir dan Waktu Selesai')
plt.xlabel('Skor Akhir')
plt.ylabel('Waktu Selesai (hari)')

# Tampilkan plot
plt.show()
```

8. Data Analytics: BQ2

```
from tabulate import tabulate
# 1. Menghitung Partisipasi (Jumlah Siswa per Bidang Studi)
partisipasi =
df.groupby('bidang_studi')['id_pengguna'].count().reset_i
ndex()
partisipasi.rename(columns={'id_pengguna':
'jumlah_partisipasi'}, inplace=True)
```

```
# 2. Menghitung Penyelesaian (Rata-Rata Persentase Siswa
yang Menyelesaikan per Bidang Studi)
df['persentase_penyelesaian'] = df.apply(
    lambda      row:      (row['aktivitas_selesai']      /
row['total_aktivitas']) * 100 if row['total_aktivitas'] >
0 else 0, axis=1
)
penyelesaian                                              =
df.groupby('bidang_studi')['persentase_penyelesaian'].mea
n().reset_index()
penyelesaian.rename(columns={'persentase_penyelesaian':
'rata_rata_penyelesaian'}, inplace=True)

# 3. Menggabungkan Data Partisipasi dan Penyelesaian
data_gabungan   =   pd.merge(partisipasi,   penyelesaian,
on='bidang_studi')

# 4. Menentukan Rata-Rata untuk Klasifikasi
rata_rata_partisipasi                                     =
data_gabungan['jumlah_partisipasi'].mean()
rata_rata_penyelesaian                                    =
data_gabungan['rata_rata_penyelesaian'].mean()

# 5. Klasifikasi Bidang Studi Berdasarkan Partisipasi dan
Penyelesaian
data_gabungan['kategori_partisipasi']                      =
data_gabungan['jumlah_partisipasi'].apply(lambda        x:
'Tinggi' if x >= rata_rata_partisipasi else 'Rendah')
data_gabungan['kategori_penyelesaian']                     =
data_gabungan['rata_rata_penyelesaian'].apply(lambda    x:
'Tinggi' if x >= rata_rata_penyelesaian else 'Rendah')

# 6. Urutkan berdasarkan Partisipasi dan Penyelesaian
#  Bidang  Studi  dengan  Partisipasi  dan  Penyelesaian
Tertinggi
bidang_studi_tinggi_terurut = data_gabungan[
    (data_gabungan['kategori_partisipasi'] == 'Tinggi') &
```

```
    (data_gabungan['kategori_penyelesaian'] == 'Tinggi')
].sort_values(by=['jumlah_partisipasi',
'rata_rata_penyelesaian'],                ascending=[False,
False]).head(5)

# Bidang Studi dengan Partisipasi dan Penyelesaian Terendah
bidang_studi_rendah_terurut                                =
data_gabungan.sort_values(by=['jumlah_partisipasi',
'rata_rata_penyelesaian'],                 ascending=[True,
True]).head(5)

# 7. Tampilkan Hasil dengan Format yang Rapi
print("📈 5 Bidang Studi dengan Partisipasi dan
Penyelesaian Tertinggi:")
print(tabulate(bidang_studi_tinggi_terurut[['bidang_studi
', 'jumlah_partisipasi', 'rata_rata_penyelesaian',
'kategori_partisipasi',          'kategori_penyelesaian']],
headers='keys', tablefmt='fancy_grid'))

print("\n📉 5 Bidang Studi dengan Partisipasi dan
Penyelesaian Terendah:")
print(tabulate(bidang_studi_rendah_terurut[['bidang_studi
', 'jumlah_partisipasi', 'rata_rata_penyelesaian',
'kategori_partisipasi',          'kategori_penyelesaian']],
headers='keys', tablefmt='fancy_grid'))
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors

# Gabungkan Data untuk Partisipasi dan Penyelesaian
top_bottom_combined                                        =
pd.concat([bidang_studi_tinggi_terurut,
bidang_studi_rendah_terurut])

# Normalisasi untuk Colormap
norm                                                       =
mcolors.Normalize(vmin=top_bottom_combined['rata_rata_pen
yelesaian'].min(),
```

```
vmax=top_bottom_combined['rata_rata_penyelesaian'].max())
cmap = plt.cm.Blues  # Colormap biru

# Membuat Horizontal Bar Chart
fig, ax = plt.subplots(figsize=(10, 8))

# Warna batang berdasarkan rata-rata penyelesaian
colors = [cmap(norm(value)) for value in
top_bottom_combined['rata_rata_penyelesaian']]
ax.barh(top_bottom_combined['bidang_studi'],
top_bottom_combined['jumlah_partisipasi'], color=colors)

# Balikkan urutan sumbu Y untuk menampilkan data dari
tertinggi ke terendah
ax.invert_yaxis()

# Tambahkan judul, label, dan grid
ax.set_title('5 Bidang Studi dengan Partisipasi dan
Penyelesaian Tertinggi dan Terendah', fontsize=16)
ax.set_xlabel('Jumlah Partisipasi', fontsize=12)
ax.set_ylabel('Bidang Studi', fontsize=12)
ax.grid(axis='x', linestyle='--', alpha=0.7)

# Tambahkan legenda colormap untuk rata-rata penyelesaian
sm = plt.cm.ScalarMappable(cmap=cmap, norm=norm)
sm.set_array([])
cbar = plt.colorbar(sm, ax=ax, orientation='vertical')
cbar.set_label('Rata-rata Penyelesaian', fontsize=12)

# Menampilkan Grafik
plt.tight_layout()
plt.show()
```

9. Data Analytics: BQ3

```
# Menghitung rata-rata rating per bidang studi
```

```
rating_summary                                                =
df.groupby('bidang_studi')['rating_kelas'].mean().reset_i
ndex()
rating_summary.columns = ['bidang_studi', 'rata2_rating']
rate    =    rating_summary.sort_values(by='rata2_rating',
ascending=False)

# Menampilkan tabel dengan tabulate
print("Bidang Studi dengan rating tertinggi:")
print(tabulate(rate.head(5),                headers='keys',
tablefmt='fancy_grid'))

print("\nBidang Studi dengan rating terendah:")
print(tabulate(rate.tail(5),                headers='keys',
tablefmt='fancy_grid'))
import matplotlib.colors as mcolors

# Menghitung rata-rata rating per bidang studi
rating_summary                                                =
df.groupby('bidang_studi')['rating_kelas'].mean().reset_i
ndex()
rating_summary.columns = ['bidang_studi', 'rata2_rating']

# Urutkan berdasarkan rata-rata rating
rate    =    rating_summary.sort_values(by='rata2_rating',
ascending=False)

# Ambil 5 tertinggi dan 5 terendah
rate_top5 = rate.head(5)
rate_bottom5 = rate.tail(5)

# Gabungkan data 5 tertinggi dan 5 terendah
rate_combined = pd.concat([rate_top5, rate_bottom5])

# Normalisasi untuk warna
```

```
norm = mcolors.Normalize(vmin=rate_combined['rata2_rating'].min(), vmax=rate_combined['rata2_rating'].max())
cmap = plt.cm.Blues  # Colormap biru

# Membuat horizontal bar chart
fig, ax = plt.subplots(figsize=(10, 6))

# Plot data dengan warna sesuai colormap
colors = [cmap(norm(value)) for value in rate_combined['rata2_rating']]
ax.barh(rate_combined['bidang_studi'], rate_combined['rata2_rating'], color=colors)

# Balikkan urutan sumbu Y untuk menampilkan dari rating tertinggi ke terendah
ax.invert_yaxis()

# Tambahkan judul, label, dan grid
ax.set_title('5 Bidang Studi dengan Rating Tertinggi dan Terendah', fontsize=16)
ax.set_xlabel('Rata-rata Rating', fontsize=12)
ax.set_ylabel('Bidang Studi', fontsize=12)
ax.grid(axis='x', linestyle='--', alpha=0.7)

# Menampilkan grafik
plt.tight_layout()
plt.show()
```

10. Data Analytics: BQ4

```
# Hitung rata-rata rating kelas per bidang studi
rata_rata_rating_kelas = df.groupby('bidang_studi')['rating_kelas'].mean().reset_index()

# Hitung presentase penyelesaian
```

```
df['presentase_penyelesaian'] = (df['aktivitas_selesai'] /
df['total_aktivitas']) * 100

# Hitung rata-rata presentase penyelesaian per bidang studi
rata_rata_penyelesaian                                    =
df.groupby('bidang_studi')['presentase_penyelesaian'].mea
n().reset_index()

# Hitung jumlah partisipasi per bidang studi
jumlah_partisipasi                                        =
df.groupby('bidang_studi')['id_pengguna'].count().reset_i
ndex()
jumlah_partisipasi.columns        =        ['bidang_studi',
'jumlah_partisipasi']

# Gabungkan DataFrame pertama dan kedua
result          =          pd.merge(rata_rata_rating_kelas,
rata_rata_penyelesaian, on='bidang_studi')

# Gabungkan DataFrame hasil dengan DataFrame ketiga
result      =      pd.merge(result,      jumlah_partisipasi,
on='bidang_studi')

# Ubah nama kolom
result.columns              =              ['bidang_studi',
'rata_rating_kelas','presentase_penyelesaian','jumlah_par
tisipasi']

result
def tentukan_kategori(row):
    if      row['rata_rating_kelas']      <      4      or
row['presentase_penyelesaian'] < 50:
        return 'Perlu Ditingkatkan'
    else:
        return 'Tidak Perlu Ditingkatkan'

# Menambahkan kolom kategori ke DataFrame 'result'
```

```
result['kategori']    =    result.apply(tentukan_kategori,
axis=1)

# Tampilkan DataFrame dengan kategori
result
kategori_perlu_ditingkatkan                                =
result.loc[result['kategori'] == 'Perlu Ditingkatkan']

kategori_perlu_ditingkatkan
result.sort_values(by=['rata_rating_kelas',
'presentase_penyelesaian'], ascending=True)
#    Urutkan    berdasarkan    'rata_rating_kelas'    dan
'presentase_penyelesaian' dalam urutan menurun
top_rating                                                 =
result.sort_values(by=['rata_rating_kelas','presentase_pe
nyelesaian'], ascending=[False, False]).head(5)

#                   Urutkan                   berdasarkan
'j'presentase_penyelesaian',rata_rating_kelas'  d'  dalam
urutan naik
low_rating           =           result.sort_values(by=[
'rata_rating_kelas','presentase_penyelesaian'],
ascending=[False, False]).tail(5)

# Menggabungkan hasil
hasil_gabungan3  =  pd.concat([top_rating,  low_rating],
ignore_index=True)

hasil_gabungan3
result.sort_values(by=['presentase_penyelesaian','rata_ra
ting_kelas'], ascending=True)
#    Urutkan    berdasarkan    'rata_rating_kelas'    dan
'presentase_penyelesaian' dalam urutan menurun
top_presentase2                                            =
result.sort_values(by=['presentase_penyelesaian','rata_ra
ting_kelas'], ascending=[False, False]).head(5)
```

```
# Urutkan berdasarkan 'j'presentase_penyelesaian',rata_rating_kelas' d' dalam urutan naik
low_presentase2 = result.sort_values(by=['presentase_penyelesaian','rata_rating_kelas'], ascending=[False, False]).tail(5)
# Menggabungkan hasil
hasil_gabungan4 = pd.concat([top_presentase2, low_presentase2], ignore_index=True)

Hasil_gabungan4
# Menggabungkan hasil
hasil_gabungan_final2 = pd.concat([low_rating, low_presentase2], ignore_index=True)
#Low rating dan low presentasi karena insight mencari bidang studi paling bawah dri segi rating dan tingkat penyelesaian
hasil_gabungan_final2

kategori_perlu_ditingkatkan = result.loc[result['kategori'] == 'Perlu Ditingkatkan']

Kategori_perlu_ditingkatkan

def plot_count(data, x=None, y=None, palette=None, figsize=(12, 6), **kwargs):
    column_order = data[x or y].value_counts().index

    plt.figure(figsize=figsize)
    sns.countplot(
        x=x, y=y, data=data, order=column_order,
        palette=palette, **kwargs
    )
    plt.grid(False)


MAIN_COLOR = ["#29B5BF"]    # warna utama untuk visualisasi
```

```
OTHER_COLOR = ["#777777"]          # warna lainnya untuk
visualisasi
#library yang dibutuhkan
!pip install scikit-learn-extra
from sklearn.cluster import KMeans, MiniBatchKMeans
from sklearn.linear_model import LinearRegression
from    sklearn.preprocessing    import    OneHotEncoder,
StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.impute import SimpleImputer
from sklearn.metrics import silhouette_score
from sklearn_extra.cluster import KMedoids
from wordcloud import WordCloud
from matplotlib.colors import LinearSegmentedColormap

# One-hot encoding untuk kolom 'bidang_studi'
encoder = OneHotEncoder()
X_cat = result[['bidang_studi']]
encoder.fit(X_cat)
X_onehot                                                =
pd.DataFrame(encoder.transform(X_cat).toarray(),
columns=encoder.get_feature_names_out())

# Menggabungkan kolom numerik
X_num            =            result[['rata_rating_kelas',
'presentase_penyelesaian', 'jumlah_partisipasi']]
X_preprocessed = pd.concat([X_num, X_onehot], axis=1)

# Standarisasi kolom numerik
scaler = StandardScaler()
scaler.fit(X_preprocessed)
X_scaled = pd.DataFrame(scaler.transform(X_preprocessed),
columns=X_preprocessed.columns)

# Menampilkan data yang telah diproses
X_scaled.head()
```

```
X_scaled.tail()

# Membuat objek SimpleImputer dengan strategi median
imputer = SimpleImputer(strategy='median')

# Mengisi nilai NaN dengan nilai median dari setiap kolom
X_scaled_clean                                                =
pd.DataFrame(imputer.fit_transform(X_scaled),
columns=X_scaled.columns)

# Menampilkan DataFrame yang telah diimputasi
(X_scaled_clean)
X_scaled_clean.head()

from time import time
from scipy.stats import pearsonr

kmedoids_2 = KMedoids(n_clusters=2, random_state=42)

start = time()
kmedoids_2.fit(X_scaled_clean)
print(len(X_scaled_clean))
print(f"Done fitting KMedoids in {time()-start:.3f}s")

kmedoids_3 = KMedoids(n_clusters=3, random_state=42)

start = time()
kmedoids_3.fit(X_scaled_clean)
print(len(X_scaled_clean))
print(f"Done fitting KMedoids in {time()-start:.3f}s")

kmedoids_5 = KMedoids(n_clusters=5, random_state=42)

start = time()
kmedoids_5.fit(X_scaled_clean)
print(len(X_scaled_clean))
print(f"Done fitting KMedoids in {time()-start:.3f}s")
```

```
kmedoids_15 = KMedoids(n_clusters=42)

start = time()
kmedoids_15.fit(X_scaled_clean)
print(len(X_scaled_clean))
print(f"Done fitting KMedoids in {time()-start:.3f}s")

result = result.assign(
    clusters_of_2=kmedoids_2.predict(X_scaled),
    clusters_of_3=kmedoids_3.predict(X_scaled),
    clusters_of_5=kmedoids_5.predict(X_scaled),
    clusters_of_15=kmedoids_15.predict(X_scaled)
)

with pd.option_context("display.max_columns", None):
    display(result.head())

#memvisualisasikan distribusi data pada setiap cluster
yang dihasilkan oleh model K-Medoids
fig, axis = plt.subplots(1, 4, figsize=(18, 6))  # Mengubah
jumlah subplot menjadi 3
cluster_names           =["clusters_of_2","clusters_of_3",
"clusters_of_5", "clusters_of_15"]

for ax, preds in zip(axis, cluster_names):
    sns.countplot(x=preds,       data=result,       ax=ax,
palette="Paired")
    ax.set_title(preds)

plt.suptitle("Cluster Cardinality")
plt.show()

# sum of squared distances
ssd = []
```

```
# may take a longer time to run (takes up to approximately
6 min)
for k in range(1, 20):
    model = KMedoids(n_clusters=k, random_state=11)
    print(f"Clustering with n_clusters={k}")
    start = time()
    model.fit(X_scaled_clean)
    print(f"Done clustering in {time()-start:.3f}s")

    ssd.append(model.inertia_)

plt.figure(figsize=(12, 6))
sns.lineplot(x=range(1, 20), y=ssd)
sns.scatterplot(x=range(1, 20), y=ssd)
plt.xticks(range(1, 20))
plt.show()

# Code for the whole dataset
silhouette_scores = []

# may take a longer time to run
for k in range(2, 21):
    model = KMedoids(n_clusters=k, random_state=11)
    print(f"Clustering with n_clusters={k}")
    start = time()
    model.fit(X_scaled_clean)
    print(f"Done clustering in {time()-start:.3f}s")

    print("Calculating silhouette coefficient..")
    start = time()

silhouette_scores.append(silhouette_score(X_scaled_clean,
model.labels_))
    print(f"Done calculating in {time()-start:.3f}s")

X_scaled_sample                                          =
X_scaled_clean[0:int(len(X_scaled_clean)*0.2)]
```

```
silhouette_scores = []

# INFORMATION
# It may take a longer time to run, the sample data took
around 46m 30s to be completed

for k in range(2, 21):
    model = KMedoids(n_clusters=k, random_state=11)
    print(f"Clustering with n_clusters={k}")
    start = time()
    model.fit(X_scaled_sample)
    print(f"Done clustering in {time()-start:.3f}s")

    print("Calculating silhouette coefficient..")
    start = time()

silhouette_scores.append(silhouette_score(X_scaled_sample
, model.labels_))
    print(f"Done calculating in {time()-start:.3f}s")

plt.figure(figsize=(12, 6))
sns.lineplot(x=range(2, 21), y=silhouette_scores)
sns.scatterplot(x=range(2, 21), y=silhouette_scores)
plt.xticks(range(1, 20))
plt.show()

df_agg = result.groupby("clusters_of_2").agg(
    avg_rata_rating_kelas=("rata_rating_kelas", np.mean),

avg_presentase_penyelesaian=("presentase_penyelesaian",
np.mean),
    modus_bidang_studi=("bidang_studi",      lambda      x:
x.mode().iloc[0] if len(x.mode()) > 0 else None)
).reset_index()

df_agg
```

```
# Mencari cluster dengan presentase penyelesaian terendah
yang lebih rendah dari nilai di Cluster 0
lowest_completion_cluster                                =
df_agg[(df_agg['avg_presentase_penyelesaian']            <
df_agg.loc[0,      'avg_presentase_penyelesaian'])       &
(df_agg['clusters_of_2']                                 !=
0)].sort_values(by='avg_presentase_penyelesaian').head(1)

print("Cluster dengan presentase penyelesaian terendah
yang lebih rendah dari Cluster 0:")
(lowest_completion_cluster)

# Mendapatkan nilai rata-rata rating kelas terendah dalam
Cluster 1
lowest_rating_cluster_1 = df_agg[df_agg['clusters_of_2']
== 1]['avg_rata_rating_kelas'].min()

# Mencari data dengan rata-rata rating kelas lebih rendah
dari nilai di Cluster 1
lowest_rating_data = result[(result['clusters_of_2'] != 1)
&              (result['rata_rating_kelas']              <
lowest_rating_cluster_1)].sort_values(by='rata_rating_kel
as').head(1)

print("\nData dengan rata-rata rating kelas terendah di
luar Cluster 1:")
(lowest_rating_data)

plt.figure(figsize=(10, 6))
sns.scatterplot(data=result,       x='rata_rating_kelas',
y='presentase_penyelesaian',         hue='clusters_of_2',
palette='viridis')
plt.xlabel('Rata Rating Kelas')
plt.ylabel('Presentase Penyelesaian')
plt.title('Visualisasi Hasil Clustering')
plt.show()
```

11. Additional Insight

```
from datetime import date

def calculate_age(born):
    today = date.today()
    return today.year - born.year - ((today.month,
today.day) < (born.month, born.day))

df['usia'] = df['tanggal_lahir'].apply(calculate_age)
# Menyesuaikan kategori umur sesuai rentang usia yang
ditentukan
import numpy as np
df['kategori_umur'] = np.where((df['usia'] >= 17) &
(df['usia'] <= 40),
                               'Generasi Muda
(Millennial/Gen Z)',
                               'Generasi Dewasa (Gen
X/Baby Boomer)')

studi_per_umur = df.groupby(['kategori_umur',
'bidang_studi'])['id_pengguna'].count().reset_index()
studi_per_umur.rename(columns={'id_pengguna':
'jumlah_peserta'}, inplace=True)
top_5_studi =
studi_per_umur.groupby('kategori_umur').apply(lambda x:
x.nlargest(5, 'jumlah_peserta')).reset_index(drop=True)
print(top_5_studi)
```

Lampiran 4. Lembar Bimbingan PKL

Program Studi Sistem Informasi
Fakultas Ilmu Komputer UPN "Veteran" Jawa Timur

PKL_3

LEMBAR BIMBINGAN, PENILAIAN, & PERSETUJUAN PKL

Kelompok Proyek :

1. FARRAHEIRA PANUNDARATRISNA FAUZIAH NPM. 22082010006
2. KANAYA DEVA CAHYARANI NPM. 22082010165

Tempat PKL : PT. Semesta Integrasi Digital
Waktu Pelaksanaan : 04 November s/d 18 Desember 2024
Judul PKL : Analisis Pola Pembelajaran Pengguna Kelas Pelatihan Kerja pada Platform MasaDepan.ku. menggunakan EDA dan Tableau

Tanggal	Materi Bimbingan	TTD Dosen Pembimbing
08 - 01 - 2025	Konsultasi BAB I - III (Revisi latar belakang, tinjauan pustaka, workflow)	
10 - 01 - 2025	Presentasi Revisi BAB I - III Konsultasi BAB IV - V (Revisi penjelasan missing value, daftpus, format)	
14 - 01 - 2025	Presentasi Revisi BAB IV - V	

* Bimbingan PKL dengan Dosen Pembimbing MINIMAL 5 kali & dilakukan oleh SEMUA anggota kelompok proyek.

No	Penilaian	Bobot	Mhs 1	Mhs 2
1	Kehadiran dan Keaktifan	25 %	85	81
2	Etika dan Kesopanan	25 %	80	80
3	Kreativitas dan Kelayakan Hasil	25 %	81	81
4	Isi dan Penulisan Buku Laporan	25 %	84	84
Nilai Akhir			82,5	81,5

* Keterangan Nilai

80 – 100	: A	72 – 75,9	: B+	58 – 63,9	: C+	46 – 49,9	: D+
76 – 79,9	: A-	68 – 71,9	: B	54 – 57,9	: C	42 – 45,9	: D
		64 – 67,9	: B-	50 – 53,9	: C-	0 – 41,9	: E

Dengan ini mahasiswa tersebut di atas **DINYATAKAN** telah menyelesaikan PKL.

Surabaya, 14 JANUARI 2025
Dosen Pembimbing,

Prasasti Karunia F.A., S.Kom., M.Kom., M.IM.
NIP/NPT.19970704 202406 2 001