```
!pip install pyspark
!pip install pandas
!pip install matplotlib
```

```
Collecting pyspark
  Downloading pyspark-3.5.0.tar.gz (316.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 316.9/316.9 MB 2.2 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.9.7)
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.5.0-py2.py3-none-any.whl size=317425345 sha256=1fd8c5c59f2d758953bf5b5e0f6d82a4c753210527b82e64d28c935256c7925e
  Stored in directory: /root/.cache/pip/wheels/41/4e/10/c2cf2467f71c678cfc8a6b9ac9241e5e44a01940da8fbb17fc
Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.5.0
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (1.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.3.post1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.23.5)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.47.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.23.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (23.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

```
import os
import warnings
warnings.filterwarnings('ignore')
from pyspark.sql import SparkSession
from pyspark.sql.types import StructField, StructType, StringType, IntegerType
from pyspark.sql.functions import split, count,  when, isnan, col, regexp_replace
from pyspark.ml.feature import OneHotEncoder, StringIndexer
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
from pyspark.sql.functions import max
from pyspark.sql.functions import min
```

```
spark = SparkSession.builder.appName('First Session').getOrCreate()

print('Spark Version: {}'.format(spark.version))
```

```
Spark Version: 3.5.0
```

```
file_path = 'data-pelaporan-kependudukan-walikota-jakarta-timur-bulan-februari-tahun-2022.csv'

data = spark.read.csv(file_path,
                      header = True,
                     inferSchema = False,
                      nanValue = '?')

data.show(5)
```

```
+-----+---------------+-------------------------+---------+---------------+-------------+------------+
|bulan|jenis_pelaporan|keterangan_jenis_pelaporan|kecamatan|      kelurahan|jenis kelamin|jumlah_orang|
+-----+---------------+-------------------------+---------+---------------+-------------+------------+
|    2|          LAHIR|      TGL LAHIR BULAN I...| MATRAMAN|  PISANGAN BARU|  Laki - laki|           4|
|    2|          LAHIR|      TGL LAHIR BULAN I...| MATRAMAN|UTAN KAYU UTARA|  Laki - laki|           2|
|    2|          LAHIR|      TGL LAHIR BULAN I...| MATRAMAN|     KAYU MANIS|  Laki - laki|           2|
|    2|          LAHIR|      TGL LAHIR BULAN I...| MATRAMAN|     PAL MERIAM|  Laki - laki|           3|
|    2|          LAHIR|      TGL LAHIR BULAN I...| MATRAMAN|  KEBON MANGGIS|  Laki - laki|           2|
+-----+---------------+-------------------------+---------+---------------+-------------+------------+
only showing top 5 rows
```

```
data.describe().show()
```

```
+-------+-----+---------------+-------------------------+----------+---------------+-------------+-----------------+
|summary|bulan|jenis_pelaporan|keterangan_jenis_pelaporan| kecamatan|      kelurahan|jenis kelamin|     jumlah_orang|
+-------+-----+---------------+-------------------------+----------+---------------+-------------+-----------------+
|  count| 1040|           1040|                     1040|      1040|           1040|         1040|             1040|
|   mean|  2.0|           NULL|                     NULL|      NULL|           NULL|         NULL|9.966346153846153|
| stddev|  0.0|           NULL|                     NULL|      NULL|           NULL|         NULL|13.92195426316521|
|    min|    2|          LAHIR|      TAHUN KEMATIAN DI...|    CAKUNG|   BALE KAMBANG|  Laki - laki|                0|
|    max|    2|           MATI|      TGL LAHIR DI BULA...|PULOGADUNG|UTAN KAYU UTARA|    Perempuan|               96|
+-------+-----+---------------+-------------------------+----------+---------------+-------------+-----------------+
```

```
from pyspark.sql.types import IntegerType

# Mengubah tipe data kolom 'jumlah_orang' menjadi IntegerType
data = data.withColumn("jumlah_orang", data["jumlah_orang"].cast(IntegerType()))

data = data.withColumn("bulan", data["bulan"].cast(IntegerType()))
```

```
data.printSchema()
```

```
root
 |-- bulan: integer (nullable = true)
 |-- jenis_pelaporan: string (nullable = true)
 |-- keterangan_jenis_pelaporan: string (nullable = true)
 |-- kecamatan: string (nullable = true)
 |-- kelurahan: string (nullable = true)
 |-- jenis kelamin: string (nullable = true)
 |-- jumlah_orang: integer (nullable = true)
```

```
data.head(1)
```

```
[Row(bulan=2, jenis_pelaporan='LAHIR', keterangan_jenis_pelaporan='TGL LAHIR BULAN INI DIINPUT DI BULAN INI', kecamatan='MATRAMAN', kelurahan='PISANGAN BARU', jenis kelamin='Laki - laki',
jumlah_orang=4)]
```

```
from pyspark.ml.feature import VectorAssembler
```

```
data.columns
```

```
['bulan',
 'jenis_pelaporan',
 'keterangan_jenis_pelaporan',
 'kecamatan',
 'kelurahan',
 'jenis kelamin',
 'jumlah_orang']
```

```
assembler = VectorAssembler(inputCols=['bulan','jumlah_orang'],outputCol='features')
```

Dalam kasus ini, 'bulan' dan 'jumlah_orang' adalah kolom-kolom yang akan digunakan sebagai fitur untuk model.

```
output = assembler.transform(data)
```

```
from pyspark.ml.feature import StringIndexer
```

```
indexer = StringIndexer(inputCol='jenis_pelaporan',outputCol='jenis_pelaporanIndex')
```

```
output_fixed = indexer.fit(output).transform(output)
```

```
output_fixed.printSchema()
```

```
root
 |-- bulan: integer (nullable = true)
 |-- jenis_pelaporan: string (nullable = true)
 |-- keterangan_jenis_pelaporan: string (nullable = true)
 |-- kecamatan: string (nullable = true)
 |-- kelurahan: string (nullable = true)
 |-- jenis kelamin: string (nullable = true)
 |-- jumlah_orang: integer (nullable = true)
 |-- features: vector (nullable = true)
 |-- jenis_pelaporanIndex: double (nullable = false)
```

```
final_data = output_fixed.select('features','jenis_pelaporanIndex')
```

```
train_data, test_data = final_data.randomSplit([0.7,0.3])
```

```
from pyspark.ml.classification import (DecisionTreeClassifier,GBTClassifier,RandomForestClassifier)
```

```
from pyspark.ml import Pipeline
```

```
# menciptakan objek DecisionTreeClassifier yang nantinya akan digunakan untuk melatih model Decision Tree untuk tugas klasifikasi

dtc = DecisionTreeClassifier(labelCol='jenis_pelaporanIndex',
 featuresCol='features')
```

```
# menciptakan objek random forest classifier yang nantinya akan digunakan untuk melatih model random forest untuk tugas klasifikasi

rfc = RandomForestClassifier(labelCol='jenis_pelaporanIndex',
 featuresCol='features')
```

```
# menciptakan objek Gradient Boosted Trees Classifier yang nantinya akan digunakan untuk melatih model Gradient Boosted Trees untuk tugas klasifikasi

gbt = GBTClassifier(labelCol='jenis_pelaporanIndex',
 featuresCol='features')
```

```
dtc_model = dtc.fit(train_data)
```

```
rfc_model = rfc.fit(train_data)
```

```
gbt_model = gbt.fit(train_data)
```

```
dtc_preds = dtc_model.transform(test_data)
```

```
rfc_preds = rfc_model.transform(test_data)
```

```
gbt_preds = gbt_model.transform(test_data)
```

```
from pyspark.ml.evaluation import BinaryClassificationEvaluator
```

```
my_binary_eval = BinaryClassificationEvaluator(labelCol='jenis_pelaporanIndex')
```

```
# mencetak hasil evaluasi dari performa model
print('DTC')
print(my_binary_eval.evaluate(dtc_preds))
```

```
DTC
0.7329397167672101
```

Kinerja Prediksi Tinggi: Nilai evaluasi mendekati 1 menunjukkan bahwa model-model tersebut mampu memprediksi kelas atau label dengan akurasi yang tinggi.

```
# mencetak hasil evaluasi dari performa model

print('RFC')
print(my_binary_eval.evaluate(rfc_preds))
```

```
RFC
0.7956402686860908
```

Kinerja Prediksi Tinggi: Nilai evaluasi mendekati 1 menunjukkan bahwa model-model tersebut mampu memprediksi kelas atau label dengan akurasi yang tinggi.

```
gbt_preds.printSchema()
```

```
root
 |-- features: vector (nullable = true)
 |-- jenis_pelaporanIndex: double (nullable = false)
 |-- rawPrediction: vector (nullable = true)
 |-- probability: vector (nullable = true)
 |-- prediction: double (nullable = false)
```

```
rfc_preds.printSchema()
```

```
root
 |-- features: vector (nullable = true)
 |-- jenis_pelaporanIndex: double (nullable = false)
 |-- rawPrediction: vector (nullable = true)
 |-- probability: vector (nullable = true)
 |-- prediction: double (nullable = false)
```

```
my_binary_eval2 = BinaryClassificationEvaluator(labelCol='jenis_pelaporanIndex', rawPredictionCol='jenis_pelaporan')
```

```
# mencetak hasil evaluasi dari performa model

print('GBT')
print(my_binary_eval.evaluate(gbt_preds))
```

```
GBT
0.7878321139776665
```

Kinerja Prediksi Tinggi: Nilai evaluasi mendekati 1 menunjukkan bahwa model-model tersebut mampu memprediksi kelas atau label dengan akurasi yang tinggi.

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

```
# mengevaluasi performa model klasifikasi multikelas (multiclass classification)

acc_eval = MulticlassClassificationEvaluator(labelCol='jenis_pelaporanIndex', metricName='accuracy')
```

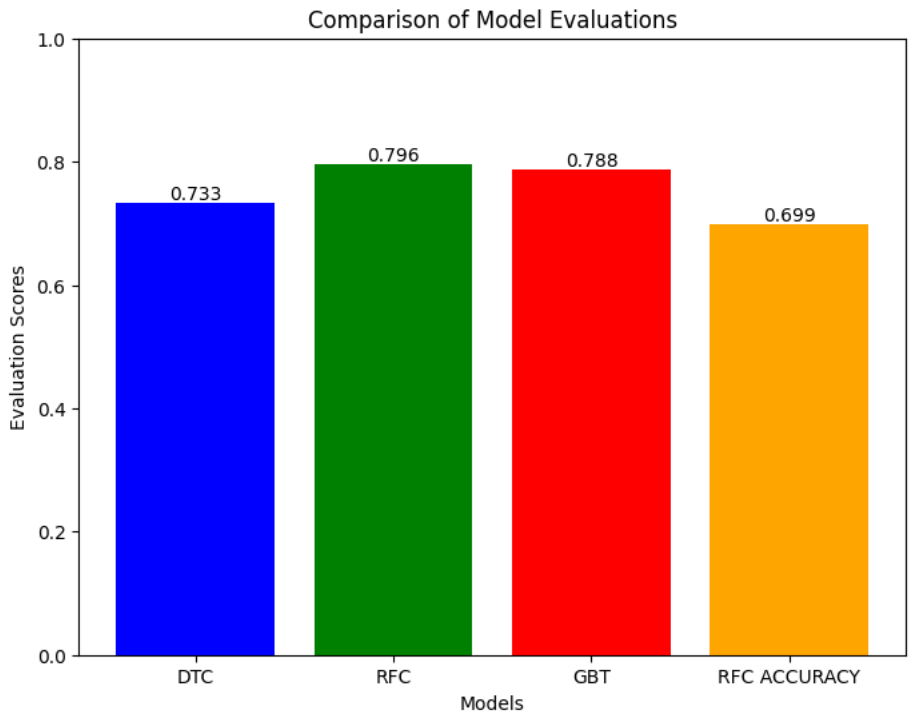
```
# menghitung akurasi model

rfc_acc = acc_eval.evaluate(rfc_preds)
```

```
# mencetak hasil akurasi model

rfc_acc
```

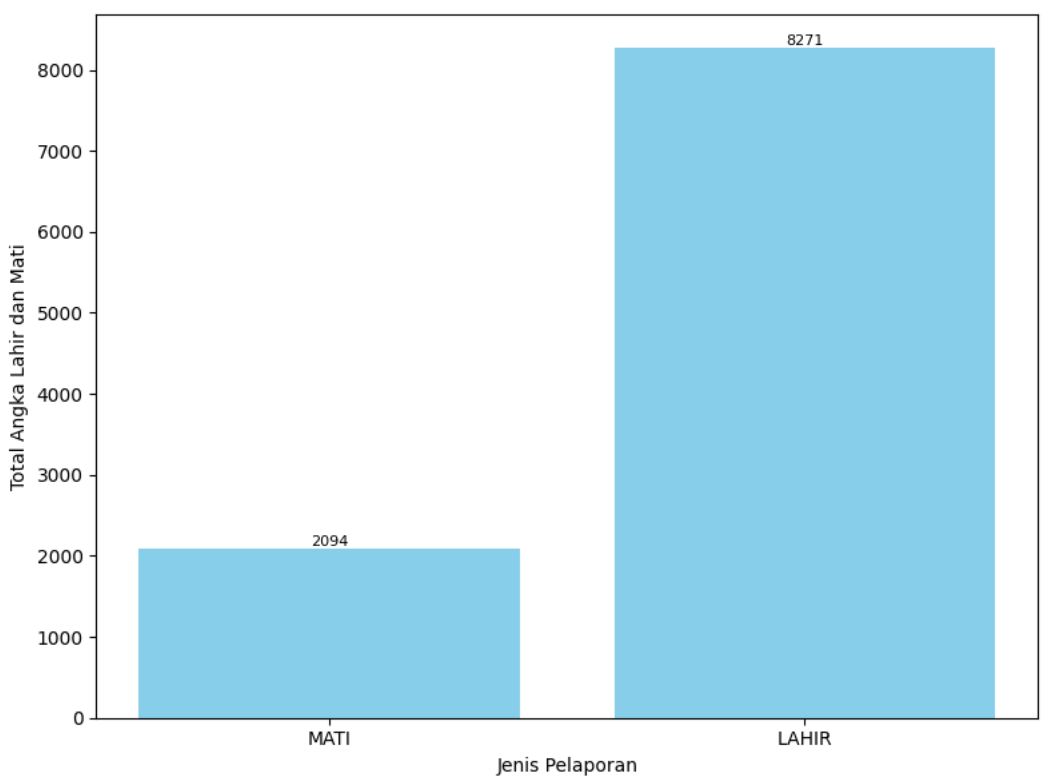
```
0.6993464052287581
```

```

```

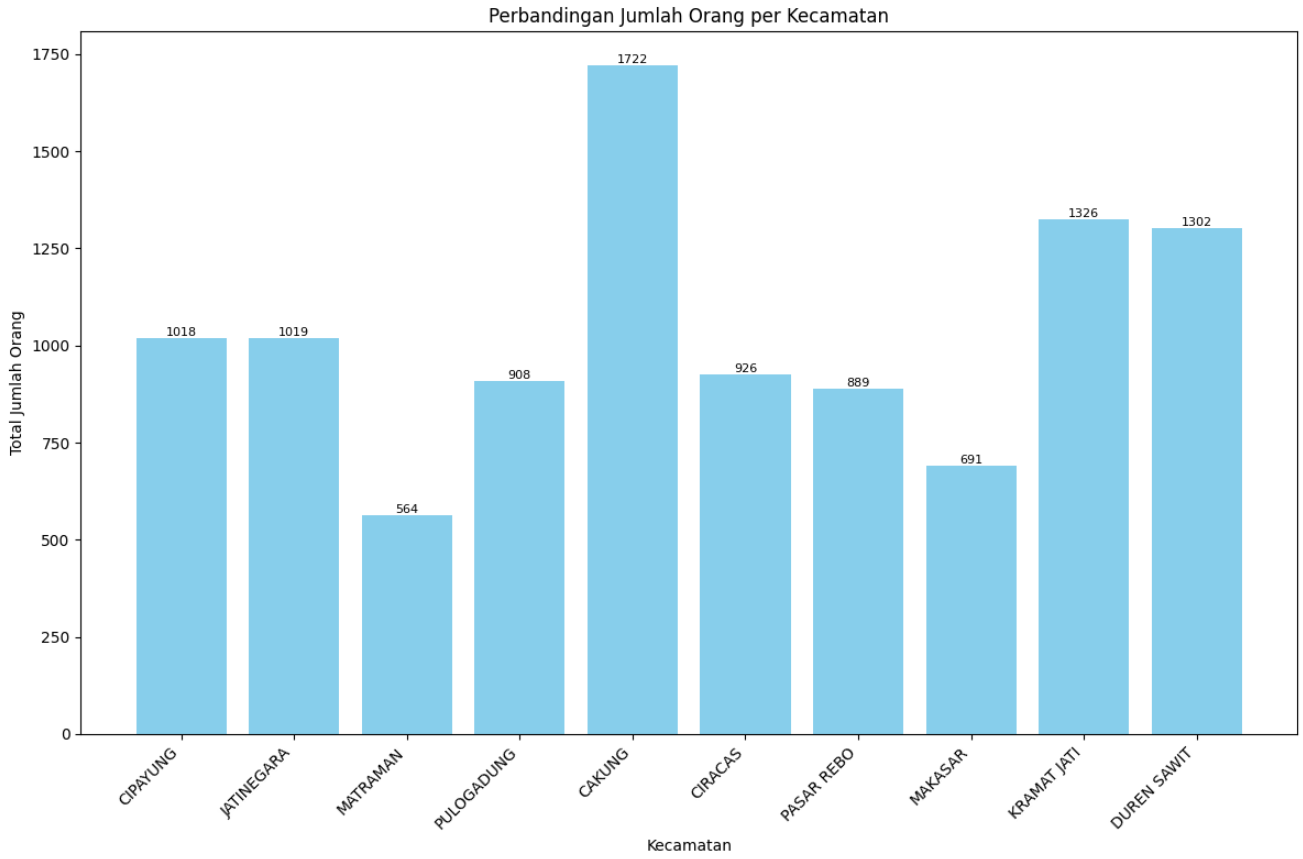
```
import matplotlib.pyplot as plt

# Data evaluasi dari model-model yang telah dilatih
model_labels = ['DTC', 'RFC', 'GBT','RFC ACCURACY']
model_scores = [my_binary_eval.evaluate(dtc_preds),
                my_binary_eval.evaluate(rfc_preds),
                my_binary_eval.evaluate(gbt_preds),
                acc_eval.evaluate(rfc_preds)]

# Membuat bar plot untuk perbandingan evaluasi model
plt.figure(figsize=(8, 6))
bars = plt.bar(model_labels, model_scores, color=['blue', 'green', 'red','orange'])
plt.xlabel('Models')
plt.ylabel('Evaluation Scores')
plt.title('Comparison of Model Evaluations')
plt.ylim(0, 1)  # Rentang nilai dari 0 hingga 1

# Menambahkan nilai di atas setiap batang diagram
for bar, score in zip(bars, model_scores):
    plt.text(bar.get_x() + bar.get_width() / 1.6 - 0.1, bar.get_height() - 0.001, f'{score:.3f}', ha='center', va='bottom', color='black')

plt.show()
```




```
import matplotlib.pyplot as plt

# Menghitung total angka lahir dan mati berdasarkan jenis pelaporan
total_lahir_mati = data.groupBy('jenis_pelaporan').sum('jumlah_orang').toPandas()

# Membuat grafik dari hasil menggunakan Matplotlib
plt.figure(figsize=(8, 6))
bars = plt.bar(total_lahir_mati['jenis_pelaporan'], total_lahir_mati['sum(jumlah_orang)'], color='skyblue')
plt.xlabel('Jenis Pelaporan')
plt.ylabel('Total Angka Lahir dan Mati')

# Menambahkan nilai-nilai di atas setiap bar
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval), ha='center', va='bottom', fontsize=8)

plt.tight_layout()
plt.show()
```




```
# Import library yang dibutuhkan
import matplotlib.pyplot as plt
from pyspark.sql.functions import sum
import pandas as pd

# Menghitung total jumlah orang untuk setiap kecamatan
total_jumlah_orang_per_kecamatan = data.groupBy('kecamatan').agg(sum('jumlah_orang').alias('total_jumlah_orang'))

# Mengambil data ke dalam Pandas DataFrame untuk memudahkan visualisasi
total_jumlah_orang_per_kecamatan_pd = total_jumlah_orang_per_kecamatan.toPandas()

# Membuat grafik bar untuk perbandingan jumlah orang per kecamatan
plt.figure(figsize=(12, 8))
bars = plt.bar(total_jumlah_orang_per_kecamatan_pd['kecamatan'], total_jumlah_orang_per_kecamatan_pd['total_jumlah_orang'], color='skyblue')
plt.xlabel('Kecamatan')
plt.ylabel('Total Jumlah Orang')
plt.title('Perbandingan Jumlah Orang per Kecamatan')
plt.xticks(rotation=45, ha='right')  # Rotasi label kecamatan agar mudah dibaca

# Menambahkan nilai-nilai di atas setiap bar
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval), ha='center', va='bottom', fontsize=8)

plt.tight_layout()
plt.show()
```




```
# Import library yang dibutuhkan
import matplotlib.pyplot as plt
from pyspark.sql.functions import sum
import pandas as pd

# Filter data untuk kelahiran
data_kelahiran = data.filter(data['jenis_pelaporan'] == 'LAHIR')
total_kelahiran_per_kecamatan = data_kelahiran.groupBy('kecamatan').agg(sum('jumlah_orang').alias('total_kelahiran'))

# Filter data untuk kematian
data_kematian = data.filter(data['jenis_pelaporan'] == 'MATI')
total_kematian_per_kecamatan = data_kematian.groupBy('kecamatan').agg(sum('jumlah_orang').alias('total_kematian'))

# Menggabungkan data kelahiran dan kematian berdasarkan kecamatan
merged_data = total_kelahiran_per_kecamatan.join(total_kematian_per_kecamatan, 'kecamatan', 'outer').fillna(0)

# Membuat grafik bar untuk perbandingan kelahiran dan kematian per kecamatan
plt.figure(figsize=(12, 8))
bar_width = 0.4
index = merged_data.toPandas().index  # Menggunakan toPandas() untuk mendapatkan indeks

bars_kelahiran = plt.bar(index, merged_data.toPandas()['total_kelahiran'], color='skyblue', width=bar_width, label='Kelahiran')
bars_kematian = plt.bar(index, merged_data.toPandas()['total_kematian'], color='lightcoral', width=bar_width, label='Kematian', align='edge')

plt.xlabel('Kecamatan')
plt.ylabel('Jumlah Orang')
plt.title('Perbandingan Kelahiran dan Kematian per Kecamatan')
plt.xticks(index, merged_data.toPandas()['kecamatan'], rotation=45, ha='right')  # Rotasi label kecamatan agar mudah dibaca
plt.legend()

# Menambahkan nilai-nilai di atas setiap bar
for bar_kelahiran, bar_kematian in zip(bars_kelahiran, bars_kematian):
    yval_kelahiran = bar_kelahiran.get_height()
    yval_kematian = bar_kematian.get_height()
    plt.text(bar_kelahiran.get_x() + bar_width / 2, yval_kelahiran, round(yval_kelahiran), ha='center', va='bottom', fontsize=8)
    plt.text(bar_kematian.get_x() + bar_width * 1.5, yval_kematian, round(yval_kematian), ha='center', va='bottom', fontsize=8)

plt.tight_layout()
plt.show()
```