By : Sermon Paskah Zagoto

Member of DQLab

Pendahuluan

Data Analytics Test

Data analytics test ini berisi 2 bagian, teori dan test coding yang masing-masing terdiri dari:

1. **Teori Konsep Dasar Data Analytics: Tes ini dimaksudkan untuk menguji pemahaman member tentang data analytics.**
2. **Coding Test**

- **Data preparation test: Tes ini dimaksudkan untuk menguji kemampuan member dalam melakukan ETL data.**
- **Data visualization test: Tes ini dimaksudkan untuk menguji kemampuan member dalam hal visualisasi data.**
- **Basic Stats Method test: Tes ini dimaksudkan untuk menguji kemampuan member dalam melakukan modeling data menggunakan statistika dasar.**

Theoritical Test

Pengantar

Jumat pagi, aku diundang oleh Senja untuk rapat kerja bersama Kroma. Rapat berlangsung cepat, kurang dari satu jam dan aku mendapat kepercayaan dari Kroma untuk menangani proyek market research. Jujur saja, ini tantangan baru buatku. Terlebih pekerjaan ini adalah rikues langsung dari Kroma.

“Sebelumnya, apakah kamu sudah di-brief oleh Senja terkait proyek ini? Apakah Senja sudah sempat membagikan padamu contoh yang pernah dilakukan senior tim terdahulu?” tanya Kroma saat rapat baru berakhir.

Aku mengangguk. “Sudah. Saya juga sudah mempelajari cara untuk coding-nya hingga penyajian visualisasi data di slide presentasi maupun dijadikan dashboard.”

“Baik, sepertinya sudah banyak yang sudah kamu pelajari di sini bersama Senja. Jadi, sudah siap untuk menangani proyek yang tadi saya sebutkan di rapat bukan? Saya ingin kamu melakukan churn analysis terhadap produk di salah satu cabang kita. Harapan saya adalah kamu bisa memberikan rekomendasi dan strategi untuk menurunkan churn dari pelanggan kita,” tukas Kroma seakan menguji kesiapanku.

“Pasti akan saya kerjakan dan berikan yang terbaik,” jawabku mantap.

Skill bagi Data Analyst

Skill apakah yang dibutuhkan oleh seorang data analyst?

1. **Business Understanding**
2. **Data Cleansing and Algorithm skills**
3. **Data Storytelling and Visualization**

Coding Test: Data Preparation

Market Research and Recommendation and Visualization Technique for Business Decision Making - Part 1

“Aksara, tadi saya forward studi kasus untuk proyek market research dari Kroma. Sudah masuk?” tanya Senja padaku begitu aku sampai di meja.

“Iya, notifikasinya sudah masuk. Segera kukerjakan ya.”

Aku pun membuka isi email yang dimaksud:

DQLab sport center adalah toko yang menjual berbagai kebutuhan olahraga seperti Jaket, Baju, Tas, dan Sepatu. Toko ini mulai berjualan sejak tahun 2013, sehingga sudah memiliki pelanggan tetap sejak lama, dan tetap berusaha untuk mendapatkan pelanggan baru sampai saat ini.

Di awal tahun 2019, manajer toko tersebut merekrut junior DA untuk membantu memecahkan masalah yang ada di tokonya, yaitu menurunnya pelanggan yang membeli kembali ke tokonya. Junior DA tersebut pun diberi kepercayaan mengolah data transaksi toko tersebut. Manajer toko mendefinisikan bahwa customer termasuk sudah bukan disebut pelanggan lagi (churn) ketika dia sudah tidak bertransaksi ke tokonya lagi sampai dengan 6 bulan terakhir dari update data terakhir yang tersedia.

Manajer toko pun memberikan data transaksi dari tahun 2013 sampai dengan 2019 dalam bentuk csv (comma separated value) dengan data_retail.csv dengan jumlah baris 100.000 baris data.

Berikut tampilan datanya:

	no	Row_Num	Customer_ID	Product	First_Transaction	Last_Transaction	Average_Transaction_Amount	Count_Transaction
0	1	1	29531	Jaket	1466304274396	1538718482608	1467681	22
1	2	2	29531	Sepatu	1406077331494	1545735761270	1269337	41
2	3	3	141526	Tas	1493349147000	1548322802000	310915	30
3	4	4	141526	Jaket	1493362372547	1547643603911	722632	27
4	5	5	37545	Sepatu	1429178498531	1542891221530	1775036	25

Field yang ada pada data tersebut antara lain:

1. No
2. Row_Num
3. Customer_ID
4. Product
5. First_Transaction
6. Last_Transaction
7. Average_Transaction_Amount
8. Count_Transaction

Market Research and Recommendation and Visualization Technique for Business Decision Making - Part 2

“Sudah kamu baca?” tanya Senja lagi.

Aku mengangguk seraya menyimak kembali isi studi kasus yang perlu kudapatkan solusinya.

“Jadi, manajer toko dan junior DA di salah satu cabang kita minta bantuan kamu untuk mengurusi riset pasar mereka dengan data dan persoalan tadi. Ada baiknya sembari kamu mengerjakan, buat laporan kerjamu langkah per langkah agar bisa diketahui proses analisis datanya untuk mengatasi kasus ini.”

“Siap!” Dengan saran dari Senja, aku pun membuat catatan dari awal pengerjaan hingga selesai agar lebih sistematis untuk dilaporkan pada Kroma nanti, seperti ini:

1. Data preparation test

- Importing data: Melakukan import data_retail.csv ke python environment.
- Cleansing data: Melakukan pembersihan dan modifikasi data sehingga siap digunakan untuk analisis lebih lanjut.

1. Data visualization test: Mendapatkan insight dari hasil visualisasi yang telah dibuat.
2. Basic stats method test: Mendapatkan insight dari model dan evaluasi model yang sudah dibuat dan diuji.

Importing Data dan Inspection

Importlah dataset dari https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/data_retail.csv dan kemudian inspeksilah dataset tersebut dengan

1. mencetak lima data teratas saja,
2. mencetak info dataset.

In [1]:

```
import pandas as pd

df = pd.read_csv('https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/data_retail.csv',
sep = ';')

print('Lima data teratas:')
print(df.head())

print('\nInfo dataset:')
print(df.info())
```

```
Lima data teratas:
   no  Row_Num  ...  Average_Transaction_Amount Count_Transaction
0   1        1  ...                     1467681                22
1   2        2  ...                     1269337                41
2   3        3  ...                      310915                30
3   4        4  ...                      722632                27
4   5        5  ...                     1775036                25

[5 rows x 8 columns]

Info dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 8 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   no                          100000 non-null  int64
 1   Row_Num                     100000 non-null  int64
 2   Customer_ID                 100000 non-null  int64
 3   Product                     100000 non-null  object
 4   First_Transaction           100000 non-null  int64
 5   Last_Transaction            100000 non-null  int64
 6   Average_Transaction_Amount  100000 non-null  int64
 7   Count_Transaction           100000 non-null  int64
dtypes: int64(7), object(1)
memory usage: 6.1+ MB
None
```

Data Cleansing

Dua kolom yang menunjukkan terjadinya transaksi tidak bertipe datetime, maka ubahlah kedua kolom tersebut ke tipe data datetime. Kemudian cetaklah kembali 5 data teratas dari dataframe df dan juga tipe data masing-masing kolomnya.

In [2]:

```
# Kolom First_Transaction
df['First_Transaction'] = pd.to_datetime(df['First_Transaction']/1000, unit='s', origin=
'1970-01-01')
# Kolom Last_Transaction
df['Last_Transaction'] = pd.to_datetime(df['Last_Transaction']/1000, unit='s', origin='1
970-01-01')

print('Lima data teratas:')
print(df.head())

print('\nInfo dataset:')
print(df.info())
```

```
Lima data teratas:
   no  Row_Num  ...  Average_Transaction_Amount Count_Transaction
0   1        1  ...                     1467681                22
1   2        2  ...                     1269337                41
2   3        3  ...                      310915                30
3   4        4  ...                      722632                27
4   5        5  ...                     1775036                25
```

```
[5 rows x 8 columns]

Info dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 8 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   no                          100000 non-null  int64
 1   Row_Num                     100000 non-null  int64
 2   Customer_ID                 100000 non-null  int64
 3   Product                     100000 non-null  object
 4   First_Transaction           100000 non-null  datetime64[ns]
 5   Last_Transaction            100000 non-null  datetime64[ns]
 6   Average_Transaction_Amount  100000 non-null  int64
 7   Count_Transaction           100000 non-null  int64
dtypes: datetime64[ns](2), int64(5), object(1)
memory usage: 6.1+ MB
None
```

Churn Customers

Untuk menentukan churn customers sesuai definisi yang telah diberikan, carilah

1. **transaksi paling terakhir kapan dilakukan**
2. **klasifikasikanlah mana customer yang berstatus churn dan mana yang tidak. Setelah itu cetak lima data teratas dan informasi dataset.**

In [3]:

```
# Pengecekan transaksaksi terakhir dalam dataset
print(max(df['Last_Transaction']))

# Klasifikasikan customer yang berstatus churn atau tidak dengan boolean
df.loc[df['Last_Transaction'] <= '2018-08-01', 'is_churn'] = True
df.loc[df['Last_Transaction'] > '2018-08-01', 'is_churn'] = False

print('Lima data teratas:')
print(df.head())

print('\nInfo dataset:')
print(df.info())
```

```
2019-02-01 23:57:57.286000013
Lima data teratas:
   no  Row_Num  ...  Count_Transaction is_churn
0   1        1  ...                 22    False
1   2        2  ...                 41    False
2   3        3  ...                 30    False
3   4        4  ...                 27    False
4   5        5  ...                 25    False

[5 rows x 9 columns]

Info dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 9 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   no                          100000 non-null  int64
 1   Row_Num                     100000 non-null  int64
 2   Customer_ID                 100000 non-null  int64
 3   Product                     100000 non-null  object
 4   First_Transaction           100000 non-null  datetime64[ns]
 5   Last_Transaction            100000 non-null  datetime64[ns]
 6   Average_Transaction_Amount  100000 non-null  int64
 7   Count_Transaction           100000 non-null  int64
 8   is_churn                    100000 non-null  object
dtypes: datetime64[ns](2), int64(5), object(2)
```

```
memory usage: 6.9+ MB
None
```

In [4]:

```
df['is_churn']
```

Out[4]:

```
0         False
1         False
2         False
3         False
4         False
          ...
99995      True
99996      True
99997      True
99998      True
99999     False
Name: is_churn, Length: 100000, dtype: object
```

Mengubah value kolom is_churn menjadi numerik menggunakan Label Encoder

In [5]:

```
import numpy as np
from sklearn.preprocessing import LabelEncoder

# Convert feature/column 'Month'
LE = LabelEncoder()
df['is_churn'] = LE.fit_transform(df['is_churn'])
print(LE.classes_)
print(np.sort(df['is_churn'].unique()))
print('')
```

```
[False True]
[0 1]
```

In [6]:

```
df['is_churn']
```

Out[6]:

```
0         0
1         0
2         0
3         0
4         0
         ..
99995     1
99996     1
99997     1
99998     1
99999     0
Name: is_churn, Length: 100000, dtype: int64
```

Menghapus kolom yang tidak diperlukan

Menghapus kolom yang tida diperlukan yaitu kolom 'no' dan 'Row_Num'

In [7]:

```
# Hapus kolom-kolom yang tidak diperlukan
del df['no']
del df['Row_Num']

# Cetak lima data teratas
print(df.head())
```

```
   Customer_ID Product  ... Count_Transaction is_churn
0        29531   Jaket  ...                22        0
1        29531  Sepatu  ...                41        0
2       141526     Tas  ...                30        0
3       141526   Jaket  ...                27        0
4        37545  Sepatu  ...                25        0

[5 rows x 7 columns]
```

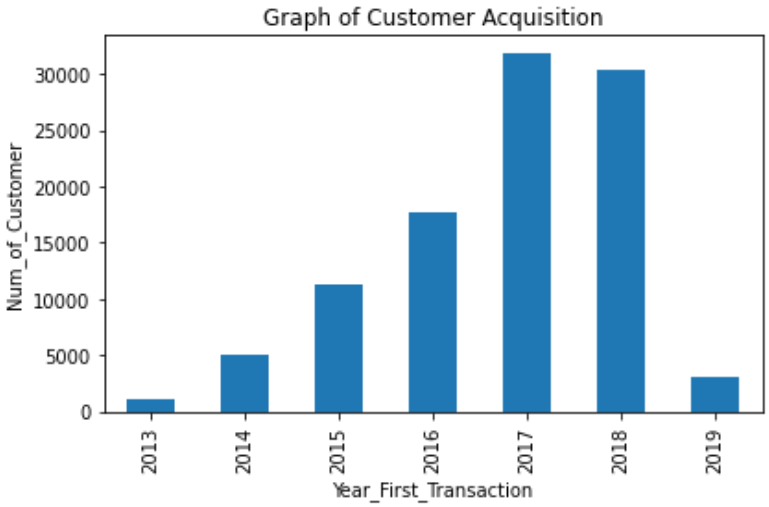
Coding Test: Data Visualization

Customer acquisition by year

Setelah semuanya lancar, langkah berikutnya adalah membuat visualisasi data berupa trend of customer acquisition by year dengan meggunakan bar chart. Untuk itu buatlah feature/kolom tambahan yang merupakan tahun dari First_Transaction dan tahun dari Last_Transaction masing-masingnya dengan nama Year_First_Transaction dan Year_Last_Transaction sebelum melakukan visualisasi.

In [8]:

```
import matplotlib.pyplot as plt

# Kolom tahun transaksi pertama
df['Year_First_Transaction'] = df['First_Transaction'].dt.year
# Kolom tahun transaksi terakhir
df['Year_Last_Transaction'] = df['Last_Transaction'].dt.year

df_year = df.groupby(['Year_First_Transaction'])['Customer_ID'].count()
df_year.plot(x='Year_First_Transaction', y='Customer_ID', kind='bar', title='Graph of Cus
tomer Acquisition')
plt.xlabel('Year_First_Transaction')
plt.ylabel('Num_of_Customer')
plt.tight_layout()
plt.show()
```



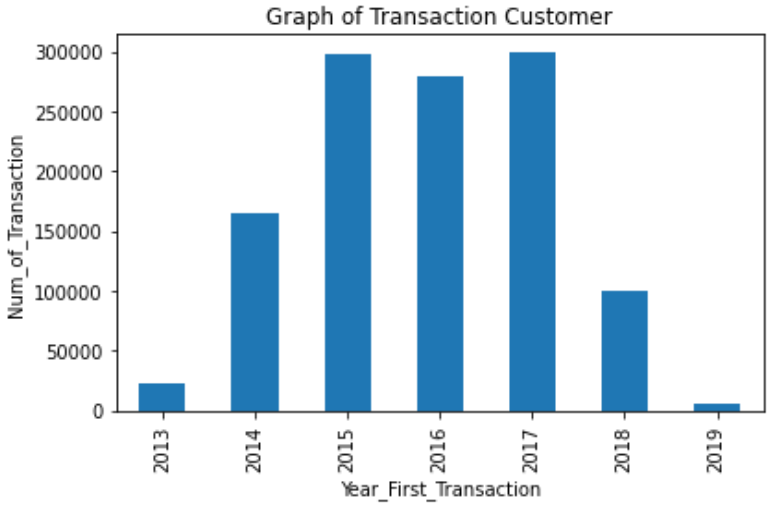
Transaction by year

Visualisasikanlah trend jumlah transaksi per tahunnya dengan menggunakan bar chart.

In [9]:

```
import matplotlib.pyplot as plt

plt.clf()
df_year = df.groupby(['Year_First_Transaction'])['Count_Transaction'].sum()
df_year.plot(x='Year_First_Transaction', y='Count_Transaction', kind='bar', title='Graph
of Transaction Customer')
plt.xlabel('Year_First_Transaction')
plt.ylabel('Num_of_Transaction')
plt.tight_layout()
```

```
plt.show()
```

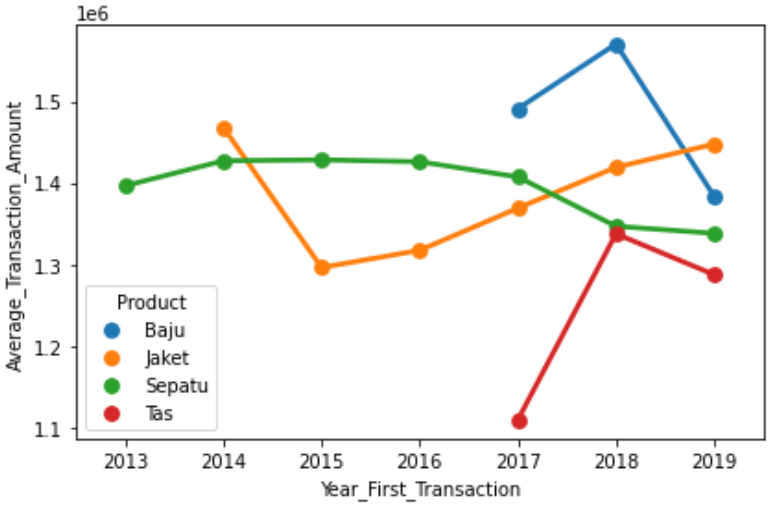


Average transaction amount by year

Dengan menggunakan seaborn pointplot, visualisasikanlah tren dari tahun ke tahun rata-rata jumlah transaksi untuk tiap-tiap produknya.

In [10]:

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

plt.clf()
sns.pointplot(data = df.groupby(['Product', 'Year_First_Transaction']).mean().reset_inde
x(),
x='Year_First_Transaction',
y='Average_Transaction_Amount',
hue='Product')
plt.tight_layout()
plt.show()
```



Proporsi churned customer untuk setiap produk

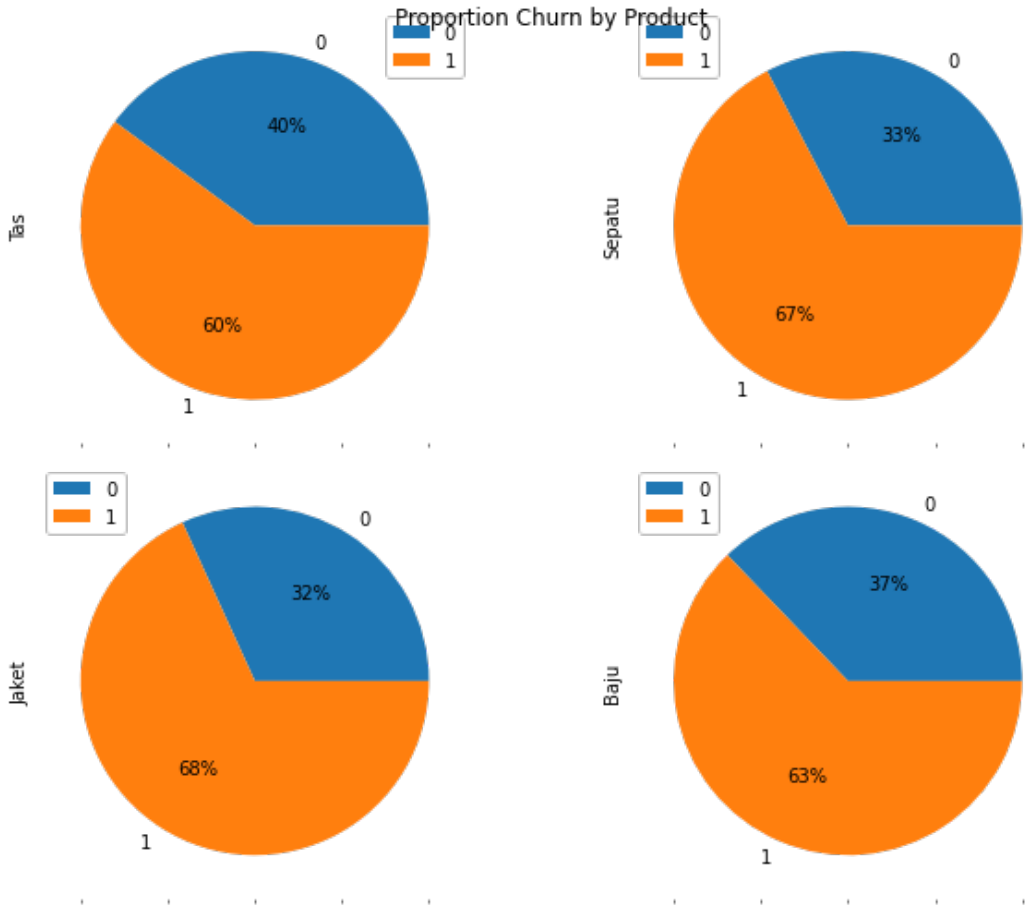
Dari sisi churned customer, khususnya untuk melihat seberapa besar proporsi churned customer untuk tiap-tiap produk dapat diketahui insight-nya melalui pie chart. Visualisasikan pie chartnya untuk keempat produk yang dimaksudkan.

In [11]:

```
import matplotlib.pyplot as plt

plt.clf()

# Melakukan pivot data
```

```
df_piv = df.pivot_table(index='is_churn',
columns='Product',
values='Customer_ID',
aggfunc='count',
fill_value=0)

# Mendapatkan Proportion Churn by Product
plot_product = df_piv.count().sort_values(ascending=False).head(5).index

# Plot pie chartnya
df_piv = df_piv.reindex(columns=plot_product)
df_piv.plot.pie(subplots=True,
figsize=(10, 7),
layout=(-1, 2),
autopct='%1.0f%%',
title='Proportion Churn by Product')
plt.tight_layout()
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



Distribusi kategorisasi count transaction

Selanjutnya akan melakukan visualisasi dari distribusi kategorisasi count transaction. Kategorisasi ini dilakukan dengan mengelompokkan jumlah transaksi seperti yang diperlihatkan oleh tabel berikut:

Rentang jumlah transaksi Kategori

s/d 1 1. 1

2 s/d 3 2. 2 - 3

4 s/d 6 3. 4 - 6

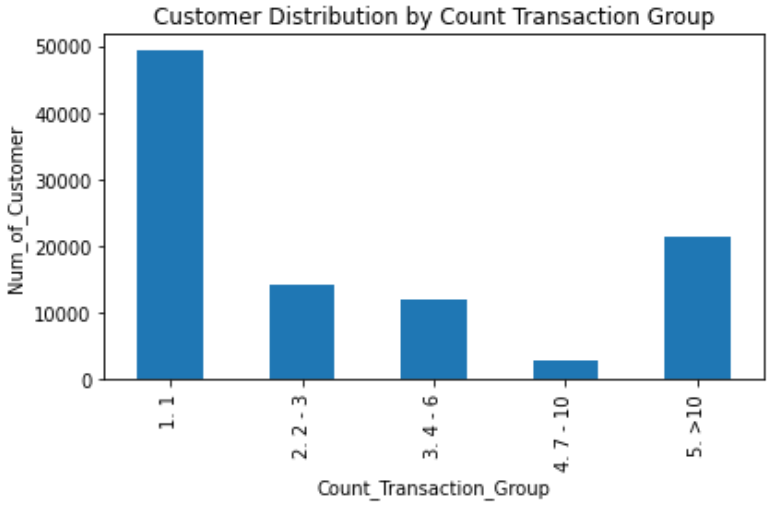
7 s/d 10 4. 7 - 10

10 5. > 10

Setelah menambahkan kolom baru untuk kategori ini dengan nama Count_Transaction_Group, maka visualisasikanlah dengan bar chart.

In [12]:

```
import matplotlib.pyplot as plt

plt.clf()
# Kategorisasi jumlah transaksi
def func(row):
    if row['Count_Transaction'] == 1:
        val = '1. 1'
    elif (row['Count_Transaction'] >1 and row['Count_Transaction'] <= 3):
        val ='2. 2 - 3'
    elif (row['Count_Transaction'] >3 and row['Count_Transaction'] <= 6):
        val ='3. 4 - 6'
    elif (row['Count_Transaction'] >6 and row['Count_Transaction'] <= 10):
        val ='4. 7 - 10'
    else:
        val ='5. >10'
    return val
# Tambahkan kolom baru
df['Count_Transaction_Group'] = df.apply(func, axis=1)

df_year = df.groupby(['Count_Transaction_Group'])['Customer_ID'].count()
df_year.plot(x='Count_Transaction_Group', y='Customer_ID', kind='bar', title='Customer Di
stribution by Count Transaction Group')
plt.xlabel('Count_Transaction_Group')
plt.ylabel('Num_of_Customer')
plt.tight_layout()
plt.show()
```



Distribusi kategorisasi average transaction amount

Selanjutnya, akan melakukan visualisasi dari distribusi kategorisasi average transaction amount. Kategorisasi ini dilakukan dengan mengelompokkan rata-rata besar transaksi seperti yang diperlihatkan oleh tabel berikut:

Rentang rata-rata besar transaksi Kategori 100.000 s/d 200.000 1. 100.000 - 250.000

250.000 s/d 500.000 2. >250.000 - 500.000

500.000 s/d 750.000 3. >500.000 - 750.000

750.000 s/d 1.000.000 4. >750.000 - 1.000.000

1.000.000 s/d 2.500.000 5. >1.000.000 - 2.500.000

2.500.000 s/d 5.000.000 6. >2.500.000 - 5.000.000

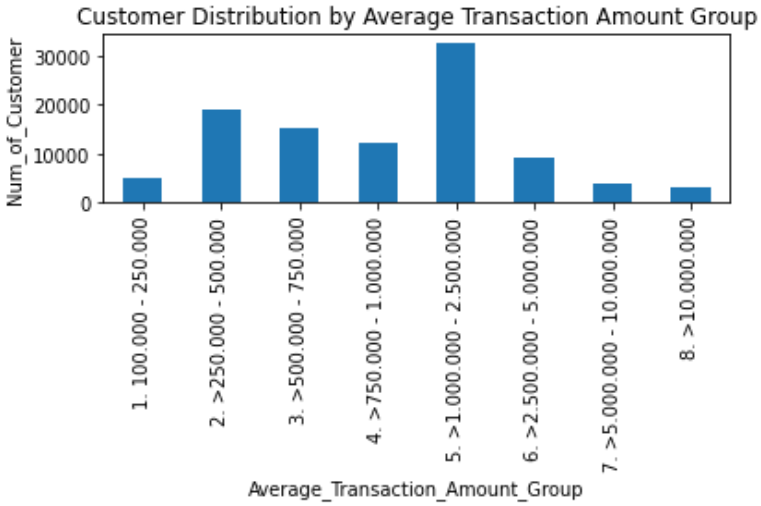
5.000.000 s/d 10.000.000 7. >5.000.000 - 10.000.000

10.000.000 8. >10.000.000

Setelah ditambahkan kolom baru untuk kategori ini dengan nama Average_Transaction_Amount_Group, maka visualisasikanlah dengan bar chart.

In [13]:

```
import matplotlib.pyplot as plt

plt.clf()
# Kategorisasi rata-rata besar transaksi
def f(row):
    if (row['Average_Transaction_Amount'] >= 100000 and row['Average_Transaction_Amount']
<=200000):
        val ='1. 100.000 - 250.000'
    elif (row['Average_Transaction_Amount'] >250000 and row['Average_Transaction_Amount']
<= 500000):
        val ='2. >250.000 - 500.000'
    elif (row['Average_Transaction_Amount'] >500000 and row['Average_Transaction_Amount']
<= 750000):
        val ='3. >500.000 - 750.000'
    elif (row['Average_Transaction_Amount'] >750000 and row['Average_Transaction_Amount']
<= 1000000):
        val ='4. >750.000 - 1.000.000'
    elif (row['Average_Transaction_Amount'] >1000000 and row['Average_Transaction_Amount'
] <= 2500000):
        val ='5. >1.000.000 - 2.500.000'
    elif (row['Average_Transaction_Amount'] >2500000 and row['Average_Transaction_Amount'
] <= 5000000):
        val ='6. >2.500.000 - 5.000.000'
    elif (row['Average_Transaction_Amount'] >5000000 and row['Average_Transaction_Amount'
] <= 10000000):
        val ='7. >5.000.000 - 10.000.000'
    else:
        val ='8. >10.000.000'
    return val

# Tambahkan kolom baru
df['Average_Transaction_Amount_Group'] = df.apply(f, axis=1)

df_year = df.groupby(['Average_Transaction_Amount_Group'])['Customer_ID'].count()
df_year.plot(x='Average_Transaction_Amount_Group', y='Customer_ID',kind='bar', title='Cus
tomer Distribution by Average Transaction Amount Group')
plt.xlabel('Average_Transaction_Amount_Group')
plt.ylabel('Num_of_Customer')
plt.tight_layout()
plt.show()
```



Coding Test: Modelling

Feature Columns dan Target

Di bagian ini, selanjutnya akan menentukan feature columns dari dataset yang dimiliki, di sini dipilih kolom Average_Transaction_Amount, Count_Transaction, dan Year_Diff. Akan tetapi, kolom terakhir belum ada. Silakan dicreate dahulu kolom Year_Diff ini dan kemudian assign dataset dengan feature columns ini sebagai variabel independent X.

Untuk target tentunya persoalan costumer dengan kondisi churn atau tidak, assign dataset untuk target ini ke dalam variabe dependent y.

In [14]:

```
# Feature column: Year_Diff
df['Year_Diff'] = df['Year_Last_Transaction'] - df['Year_First_Transaction']

# Nama-nama feature columns
feature_columns = ['Average_Transaction_Amount', 'Count_Transaction', 'Year_Diff']

# Features variable
X = df[feature_columns]

# Target variable
y = df['is_churn']
```

In [16]:

```
print('Kolom Feature', X)
print('\nKolom Target', y)
```

```
Kolom Feature        Average_Transaction_Amount  Count_Transaction  Year_Diff
0                                      1467681                 22          2
1                                      1269337                 41          4
2                                       310915                 30          2
3                                       722632                 27          2
4                                      1775036                 25          3
...                                        ...                ...        ...
99995                                   298662                  1          0
99996                                   349556                  1          0
99997                                   598013                  1          0
99998                                  1208591                  1          0
99999                                   486710                  1          0

[100000 rows x 3 columns]

Kolom Target 0        0
1         0
2         0
3         0
4         0
         ..
99995     1
99996     1
99997     1
99998     1
99999     0
Name: is_churn, Length: 100000, dtype: int64
```

Split X dan y ke dalam bagian training dan testing

Setelah variabel independent X dan variabel dependent y selesai dilakukan, maka pecahlah X dan y ke dalam bagian training dan testing. Bagian testing 25% dari jumlah entri data.

In [17]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0
)
```

Train, predict dan evaluate

Langkah selanjutnya akan membuat model menggunakan Linear Regression, inisialisasilah model, fit, dan kemudian evaluasi model dengan menggunakan confusion matrix.

In [18]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix

# Inisiasi model logreg
logreg = LogisticRegression()

# fit the model with data
logreg.fit(X_train, y_train)

# Predict model
y_pred=logreg.predict(X_test)

# Evaluasi model menggunakan confusion matrix
cnf_matrix = confusion_matrix(y_test, y_pred)
print('Confusion Matrix:\n', cnf_matrix)
```

```
Confusion Matrix:
 [[    1  8330]
 [    3 16666]]
```

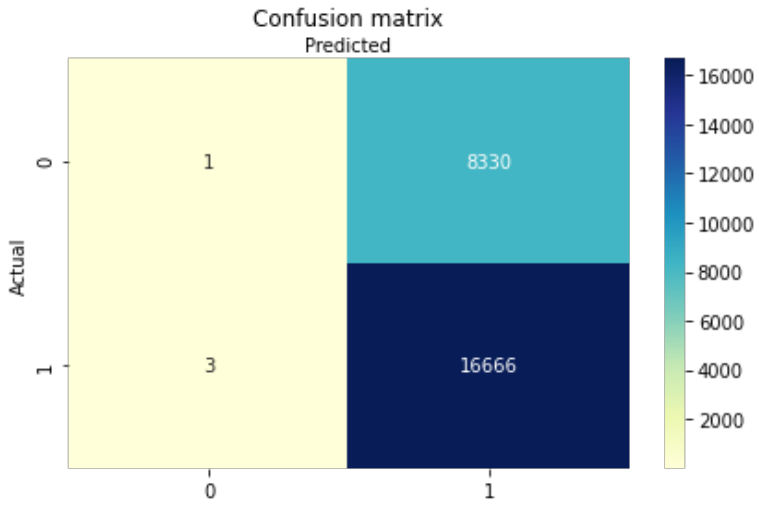
Visualisasi Confusion Matrix

Confusion matrix yang telah dihitung sebelumnya dapat divisualisasikan dengan menggunakan heatmap dari seaborn. Untuk itu tampilkanlah visualisasi dari confusion matrix ini.

In [19]:

```
# import required modules
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

plt.clf()
# name of classes
class_names = [0, 1]
fig, ax = plt.subplots()

tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)

# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap='YlGnBu', fmt='g')
ax.xaxis.set_label_position('top')
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.tight_layout()
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



Accuracy, Precision, dan Recall

Kemudian, hitunglah nilai accuracy, precission dan recall berdasarkan nilai target sesungguhnya dan nilai target hasil prediksi.

In [20]:

```
from sklearn.metrics import accuracy_score, precision_score, recall_score

#Menghitung Accuracy, Precision, dan Recall
print('Accuracy :', accuracy_score(y_test, y_pred))
print('Precision:', precision_score(y_test, y_pred, average='micro'))
print('Recall   :', recall_score(y_test, y_pred, average='micro'))
```

```
Accuracy : 0.66668
Precision: 0.66668
Recall   : 0.66668
```

Penutup

Ingatan akan perkataan Senja

“Fiuh! Selesai juga untuk studi kasus proyek ini. Bener juga kata Senja, dengan bikin catatan proses kerja dari awal sampai akhir seperti ini, aku jadi bisa meninjau ulang pekerjaanku juga.”

Aku pun merapikan catatanku untuk kuperlihatkan dulu kepada Senja sebelum menghadap Kroma. Sejauh ini sih, lancar dan sukses!