

Data Analytics for Business 2024

Project Based Learning

Study Case

Paper Id

CS 09 - 5

- | | | |
|----|-------------------------|------------|
| 1. | Ni Putu Ana Rainita | KM-CS09408 |
| 2. | Naifa Ghaisani Syakira | KM-CS09135 |
| 3. | Roshan Syalwan Nurilham | KM-CS09156 |

Kata Pengantar

Dengan penuh rasa syukur kepada Tuhan Yang Maha Esa, penulis dengan bangga menyelesaikan Laporan Akhir Magang & Studi independen Bersertifikat Kampus Merdeka tepat pada waktunya. Laporan ini dibuat sebagai bagian dari persyaratan kelulusan kegiatan studi independen bersertifikat 2024, Penulis ingin menyampaikan penghargaan dan terima kasih kepada semua pihak yang telah turut serta memberikan bimbingan dan dukungan dalam proses penulisan laporan ini, diantaranya:

1. Tuhan Yang Maha Esa atas berkat dan rahmat-Nya sehingga penulis dapat menyelesaikan Program Studi Independen Bersertifikat Kampus Merdeka dengan baik dan lancar.
2. Direktorat Jenderal Pendidikan Tinggi (Dirjen Dikti) Kemendikbudristek yang telah meluncurkan Merdeka Belajar Kampus Merdeka sehingga dapat memberikan kesempatan bagi mahasiswa se-Indonesia dalam meningkatkan keterampilan yang dibutuhkan pada masa depan.
3. Kakak Al Mira Khozati selaku mentor yang senantiasa membimbing dalam program Studi Independen Bersertifikat Kampus Merdeka
4. Kakak Tania Amelinda selaku *student succes* yang senantiasa membantu menjembatani dalam proses komunikasi dan memberikan informasi selama kegiatan studi independen berlangsung.

Penulis berharap dengan selesainya laporan ini, dapat memberikan manfaat dan menjadi sumber referensi yang berharga bagi pembaca. Penulis menyadari bahwa laporan yang penulis kerjakan jauh dari kata sempurna. Sehingga penulis mengharapkan kritik dan saran yang membangun dari teman-teman pembaca sekalian. Terima kasih atas segala dukungan yang diberikan kepada penulis.

Daftar Isi

Kata Pengantar	1
Daftar Isi	2
BAB I	3
A. Latar Belakang	3
B. Identifikasi Masalah	5
BAB II	7
A. Data dan Sumber Data	7
B. Metode Analisis Data	10
BAB III	12
A. Gambaran Umum Objek Penelitian	12
B. Temuan Hasil Penelitian	14
C. Pembahasan	90
BAB IV	94
A. Simpulan	94
B. Saran	94
Lampiran	95

BAB I

Pendahuluan

A. Latar Belakang

Paper.id merupakan platform solusi digital terkemuka di Indonesia yang menyediakan layanan pembuatan faktur (invoicing) dan pembayaran secara terintegrasi. Dirancang khusus untuk kebutuhan bisnis di Indonesia, Paper.id memungkinkan pengguna untuk membuat faktur tanpa batas secara mudah, mengotomatisasi pengingat pembayaran, serta mengintegrasikan berbagai metode pembayaran seperti kartu kredit dan kode QR. Dengan fokus pada penyederhanaan transaksi keuangan, platform ini membantu bisnis mengelola proses faktur mereka dengan efisiensi dan akurasi yang lebih tinggi.

Seiring pertumbuhan pesat dalam penggunaan layanan, perusahaan menghadapi tantangan baru berupa meningkatnya transaksi penipuan. Hal ini tidak hanya berdampak pada penurunan pendapatan, tetapi juga merusak kepercayaan pelanggan terhadap platform. Untuk mengatasi tantangan ini, diperlukan analisis penipuan tingkat lanjut yang dapat membedakan pola perilaku penipuan serta mendeteksi hubungan mencurigakan antara pembeli dan penjual.

Sebagai bagian dari solusi, perusahaan memanfaatkan analitik mendalam dan kueri SQL tingkat lanjut untuk mengidentifikasi pola penipuan secara proaktif. Dengan mengolah jutaan data transaksi pembayaran digital, analisis ini bertujuan memberikan wawasan yang dapat ditindaklanjuti untuk mengurangi risiko penipuan. Melalui pendekatan ini, Paper.id terus berkomitmen untuk menyediakan layanan yang aman, efisien, dan dapat diandalkan bagi seluruh penggunanya, sekaligus mendukung pertumbuhan ekonomi digital di Indonesia.

1. Permasalahan dalam Sistem Paper Id

Pada latar belakang diatas, permasalahan yang diidentifikasi adalah sebagai berikut:

1. **Hubungan Tidak Normal antara Pembeli dan Penjual:** Adanya interaksi yang terlalu sering atau pola hubungan mencurigakan antara pembeli dan penjual tertentu, yang dapat mengindikasikan aktivitas penipuan seperti kolusi.
2. **Ketidaknormalan dalam Frekuensi Transaksi:** Lonjakan aktivitas transaksi dalam waktu singkat yang mungkin menunjukkan upaya penipuan secara masif.
3. **Penyalahgunaan Program Promosi:** Pengguna yang secara berulang menyalahgunakan kode promo atau diskon, yang dapat menyebabkan kerugian finansial bagi perusahaan.
4. **Pengelolaan Data yang Kompleks:** Pengolahan jutaan data transaksi pembayaran digital memerlukan pendekatan analitik yang mendalam dan teknologi mutakhir untuk mengidentifikasi ancaman secara efisien.

2. Tujuan Analisis dan Output yang Diharapkan

Berikut adalah tujuan analisis yang dirancang untuk menangani masalah yang ditemukan dan meningkatkan proses identifikasi fraud dengan fokus pada beberapa aspek utama:

1. **Mendeteksi Hubungan Mencurigakan antara Pembeli dan Penjual:** Mengidentifikasi interaksi yang tidak normal untuk mencegah kolusi atau aktivitas penipuan lainnya.
2. **Mengidentifikasi Ketidaknormalan dalam Frekuensi Transaksi:** Mengungkap pola transaksi tidak wajar, seperti lonjakan tiba-tiba atau jeda panjang, yang dapat menjadi indikasi penipuan.
3. **Mencegah Penyalahgunaan Program Promosi:** Menemukan dan mencegah pengguna yang memanfaatkan promosi secara tidak sah, seperti penggunaan berulang kode promo dengan cara yang melanggar aturan.

B. Identifikasi Masalah

Tabel di bawah ini menjelaskan proses identifikasi masalah dari analisis yang dilakukan terhadap dataset paper id yang mencakup informasi terkait *digital payment transaction data*, *digital payment request data*, *promotion data*, *company data*. Masalah utama yang akan dianalisis meliputi : hubungan tidak normal antara pembeli dan penjual, ketidaknormalan dalam frekuensi transaksi, penyalahgunaan program promosi.

Tabel 1. Identifikasi Permasalahan

Factor	Items	Code
Hubungan Tidak Normal antara Pembeli dan Penjual	Bagaimana cara mengidentifikasi dan menangani hubungan yang mencurigakan antara pembeli dan penjual yang mungkin mengindikasikan aktivitas penipuan atau kolusi?	H1
Ketidaknormalan dalam Frekuensi Transaksi	Apa saja indikator yang dapat menunjukkan ketidaknormalan dalam frekuensi transaksi, seperti lonjakan aktivitas tiba-tiba atau periode yang tidak biasa, yang dapat mengarah pada deteksi penipuan?	K1
Penyalahgunaan Program Promosi	Bagaimana cara mendeteksi dan mencegah pengguna yang menyalahgunakan kode promo atau diskon secara berulang untuk mendapatkan keuntungan yang tidak sah?	P1

Untuk mengidentifikasi permasalahan pada tabel di atas, analisis akan dilakukan dengan memanfaatkan dataset yang tersedia. Setiap permasalahan, seperti hubungan tidak normal antara pembeli dan penjual, ketidaknormalan dalam frekuensi transaksi, serta penyalahgunaan program promosi, akan dipecah menjadi faktor-faktor spesifik yang dapat diukur dan dianalisis. Dalam kasus hubungan tidak normal, faktor-faktor seperti frekuensi transaksi antara pembeli dan penjual tertentu, jumlah transaksi, serta pola waktu transaksi akan diperiksa untuk menemukan anomali atau indikasi kolusi.

Ketidaknormalan dalam frekuensi transaksi akan dianalisis dengan memperhatikan lonjakan tiba-tiba dalam aktivitas transaksi, pola pengulangan yang tidak biasa, serta waktu terjadinya transaksi. Dataset akan memberikan wawasan tentang rata-rata historis, pola harian, atau mingguan yang dapat dibandingkan untuk mendeteksi anomali. Sementara itu,

untuk penyalahgunaan program promosi, analisis akan fokus pada data penggunaan kode promo, termasuk jumlah penggunaan per akun, pola pengulangan kode, serta kemiripan atribut antara akun-akun pengguna. Dengan bantuan dataset, pola-pola ini dapat diidentifikasi untuk menemukan potensi penyalahgunaan. Melalui analisis data yang sistematis dan didukung dengan teknik data mining atau machine learning, faktor-faktor yang memengaruhi masing-masing permasalahan dapat dipetakan dengan jelas, sehingga solusi atau pencegahan yang efektif dapat dirancang.

BAB II

Metodologi

A. Data dan Sumber Data

Data yang dianalisis mencakup berbagai elemen penting yang dapat membantu dalam mendeteksi dan mengevaluasi potensi aktivitas fraud pada transaksi digital. Berikut adalah jenis dan sumber data yang digunakan pada fokus penelitian ini:

1. Company Data

Columns:

- company_id: ID unik perusahaan.
- company_kyc_status_name: Status verifikasi identitas pelanggan (KYC).
- company_kyb_status_name: Status verifikasi identitas bisnis (KYB).
- company_type_group: Kategori jenis perusahaan.
- company_phone_verified_flag: Status verifikasi nomor telepon.
- company_email_verified_flag: Status verifikasi email.
- user_fraud_flag: Indikator apakah pengguna pernah terlibat fraud.
- testing_account_flag: Indikator akun pengujian.
- blacklist_account_flag: Indikator akun yang masuk daftar hitam.
- company_registered_datetime: Waktu pendaftaran perusahaan.

2. Promotion Data

Columns:

- dpt_promotion_id: ID promosi unik.
- promotion_code: Kode promosi yang digunakan.
- promotion_name: Nama promosi.
- transaction_promo_cashback_amount: Jumlah cashback yang diberikan.

3. Digital Payment Request Data

Columns:

- dpt_id: ID transaksi unik (relasi dengan data transaksi).
- total_fee_amount: Jumlah total biaya transaksi.

- document_type_name: Jenis dokumen yang terkait dengan transaksi.

4. Digital Payment Transaction Data

Columns:

- dpt_id: ID transaksi unik.
- buyer_id: ID pembeli.
- seller_id: ID penjual.
- transaction_amount: Jumlah nilai transaksi.
- payment_method_name: Metode pembayaran yang digunakan.
- payment_provider_name: Penyedia layanan pembayaran.
- transaction_created_datetime: Tanggal dan waktu transaksi dibuat.
- transaction_updated_datetime: Tanggal dan waktu transaksi diperbarui.
- dpt_promotion_id: ID promosi yang digunakan dalam transaksi.

Sumber Data: Data yang digunakan diatas diperoleh dari perusahaan Paper Id yang bekerja sama dengan Bitlabs Academy untuk program *Project Based Learning*.

B. Alur Pengerjaan Study Case

1. Tools yang digunakan dalam proses menganalisis data

Dalam penelitian ini, berbagai tools digunakan untuk mendukung analisis data yang menyeluruh dan mendalam. Pemilihan tools didasarkan pada kemampuan dalam menangani data dalam jumlah besar, melakukan analisis secara mendalam, serta menyajikan hasil visualisasi yang interaktif dan mudah dipahami. Berikut adalah beberapa tools yang digunakan:

- **Python**

Python digunakan untuk berbagai tahapan analisis data, mulai dari data cleaning untuk menangani nilai yang hilang, outlier, anomali data, hingga proses feature engineering untuk menciptakan metrik transaksi dan skor hubungan antara buyer dan seller. Python dapat menangani seluruh proses analisis data secara efektif dan efisien mulai dari pemrosesan hingga visualisasi.

- **Google Colab**

Google Colab digunakan untuk menulis dan menjalankan skrip Python secara langsung di cloud. Platform ini memungkinkan pengguna untuk memanfaatkan pustaka analisis data seperti Pandas, NumPy, Matplotlib, dan Seaborn tanpa perlu instalasi perangkat lunak tambahan. Dengan fitur berbasis Google Drive, Colab memungkinkan kolaborasi secara real-time. Pengguna dapat berbagi notebook dan bekerja bersama secara langsung, yang membuatnya ideal untuk tim yang bekerja pada proyek data bersama-sama. Serta dapat membuat visualisasi data tanpa perlu menggunakan perangkat lunak tambahan.

- **SQL (Structured Query Language)**

SQL digunakan untuk mengelola dan mengekstrak data dari database relasional. Dengan SQL, proses analisis dapat dilakukan secara langsung dengan menulis query untuk mengidentifikasi anomali, kolusi, eksplorasi promosi dan pola transaksi yang mencurigakan. SQL juga memudahkan otomatisasi laporan dan analisis rutin, menjadikannya alat yang efisien untuk menangani data yang kompleks dalam sistem database relasional.

- **DBeaver**

DBeaver mendukung berbagai jenis sistem database, sehingga mempermudah pengguna untuk bekerja dengan berbagai sumber data tanpa perlu beralih antara banyak alat. DBeaver menyediakan grafis yang user-friendly untuk menjalankan query SQL, mengelola data, serta menjelajahi struktur database. DBeaver kompatibel dengan berbagai jenis database, yang mempermudah penggabungan analisis SQL dengan alat lain seperti Python, sehingga memperlancar dan mempercepat proses kerja.

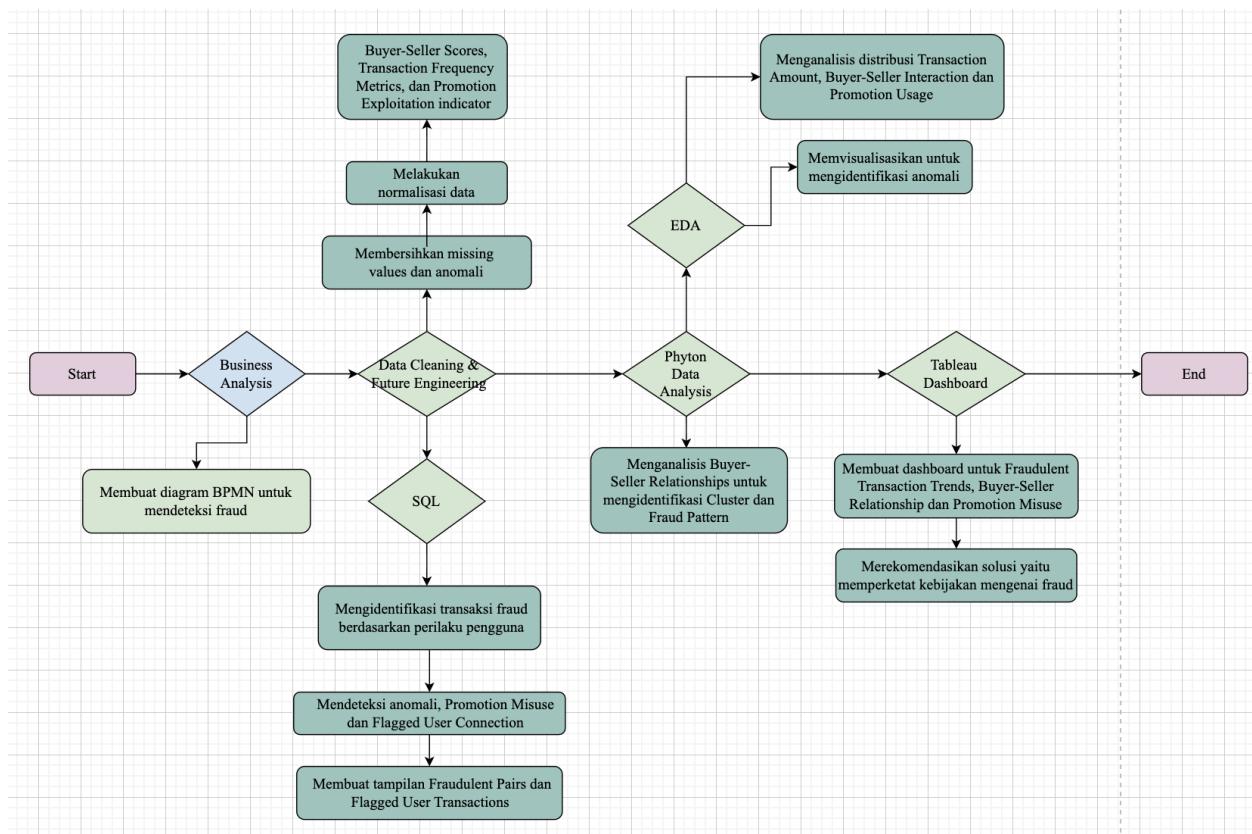
- **Tableau**

Tableau digunakan untuk pembuatan dashboard interaktif yang memudahkan pengguna untuk mengeksplorasi dan menganalisis data secara visual dengan dapat dilakukannya pemantauan tren fraud, analisis hubungan buyer-seller yang mencurigakan, dan pengamatan pola transaksi yang berkaitan dengan promosi. Dengan fitur filter dinamis, Tableau memungkinkan tim untuk melakukan eksplorasi data lebih mendalam dan menyajikan hasil analisis dengan cara yang

mudah dipahami. Sebagai alat visualisasi data, Tableau dapat membantu menyampaikan hasil analisis dengan format yang lebih mudah untuk dipahami.

Penggunaan Google Colab, Python, SQL, DBeaver, dan Tableau menciptakan alur kerja yang efisien dalam analisis data. Tools ini tidak hanya meningkatkan efisiensi dan fleksibilitas proses, tetapi juga memastikan hasil yang dapat diimplementasikan untuk mengurangi risiko fraud dalam transaksi digital secara efektif.

2. Proses Analisis Data



Gambar 1. Flowchart Proses Analisis Fraud pada Transaction.

Alur proses pendekripsi fraud dimulai dengan pemahaman terhadap permasalahan bisnis yang ada. Tahap pertama adalah pembuatan diagram proses (BPMN) untuk memetakan seluruh langkah yang diperlukan dalam mendekripsi potensi fraud. Setelah itu, proses pembersihan data dilakukan untuk menangani data yang hilang (missing values) atau adanya anomali yang dapat mengganggu analisis. Pada tahap berikutnya, dilakukan feature

engineering untuk menghasilkan atribut penting seperti skor hubungan antara pembeli dan penjual, metrik transaksi, serta indikator penyalahgunaan promosi. Fitur-fitur tersebut kemudian dinormalisasi agar analisis dapat dilakukan dengan lebih efisien.\

Setelah data siap, langkah berikutnya adalah melakukan query lanjutan menggunakan SQL untuk mengidentifikasi transaksi fraud, mengelompokkan pembeli dan penjual yang mencurigakan, serta pembuatan laporan tentang penyalahgunaan promosi. Dengan menggunakan Python, dilakukan eksplorasi data (EDA) untuk menemukan pola atau perilaku mencurigakan dalam transaksi, termasuk analisis jaringan untuk memeriksa hubungan antar pengguna. Hal ini membantu untuk menemukan cluster atau pola yang mendarah pada indikasi fraud.

Hasil dari analisis ini kemudian disajikan dalam bentuk dashboard interaktif menggunakan Tableau, yang memudahkan untuk pemantauan tren fraud, interaksi antar buyer dan seller, serta penyalahgunaan promosi. Dashboard ini memberikan wawasan yang berharga untuk pengambilan keputusan, termasuk saran kebijakan yang lebih ketat untuk mencegah terjadinya fraud di masa mendatang. Semua tahap dalam proses ini bekerja secara terintegrasi, membentuk solusi menyeluruh dalam mendeteksi dan mencegah terjadinya fraud.

BAB III

Hasil dan Pembahasan

A. Deskripsi Data Hasil Analisis Bisnis

1. Sumber dan Struktur Data

Dataset yang digunakan terdiri dari empat file utama, yaitu **transaction.csv**, **request.csv**, **promotion.csv**, dan **company.csv**. Berikut adalah rincian jumlah data dan struktur setiap file:

- **Digital Payment Transaction Data (transaction.csv)**
 - Jumlah Data: Sekitar 50.000 baris data
 - Kolom: Data ini berisi informasi transaksi pembayaran, termasuk ID transaksi, ID pembeli dan penjual, jumlah transaksi, metode pembayaran, penyedia pembayaran, serta ID promosi yang digunakan.
 - Kualitas Data: Tidak ditemukan nilai yang mencurigakan atau tidak konsisten dalam kolom **payment_method_name** dan **payment_provider_name**. Namun, terdapat beberapa transaksi yang terjadi dalam waktu sangat dekat (dalam rentang waktu yang singkat) oleh pembeli yang sama, yang berpotensi mencurigakan. Namun, jumlah transaksi tertinggi mencapai 20 juta yang masih dianggap wajar dan tidak perlu dihapus.
- **Digital Payment Request Data (request.csv)**
 - Kolom: Data ini mencakup informasi permintaan pembayaran seperti ID transaksi (**dpt_id**), jumlah biaya total (**total_fee_amount**), dan tipe dokumen (**document_type_name**).
 - Kualitas Data: Tidak ada data hilang atau tidak valid pada file ini, dan tidak ditemukan pola mencurigakan terkait kolom **total_fee_amount** dan **dpt_id**.
- **Promotion Data (promotion.csv)**

- Kolom: File ini berisi informasi tentang promosi yang digunakan dalam transaksi, termasuk ID promosi, kode promosi, nama promosi, dan jumlah cashback yang diberikan.
- Kualitas Data: Tidak ditemukan promo yang lebih sering digunakan dalam transaksi mencurigakan. Promo yang diberikan bergantung pada metode pembayaran yang digunakan oleh pembeli, dengan tujuan memberikan cashback atau diskon.
- **Company Data (company.csv)**
 - Kolom: Data ini berisi informasi tentang perusahaan, termasuk status verifikasi KYC (Know Your Customer), status verifikasi KYB (Know Your Business), jenis perusahaan, serta indikator terkait akun yang terindikasi fraud (misalnya, **user_fraud_flag**, **blacklist_account_flag**, **testing_account_flag**).
 - Kualitas Data: Perusahaan yang sudah terverifikasi cenderung lebih jarang terlibat dalam transaksi fraud. Namun, perusahaan yang memiliki status **user_fraud_flag** seringkali terlibat dalam lebih banyak transaksi yang mencurigakan.

2. Kualitas Data

Secara umum, dataset yang digunakan dalam analisis ini memiliki kualitas yang baik. Namun, beberapa hal yang perlu diperhatikan adalah sebagai berikut:

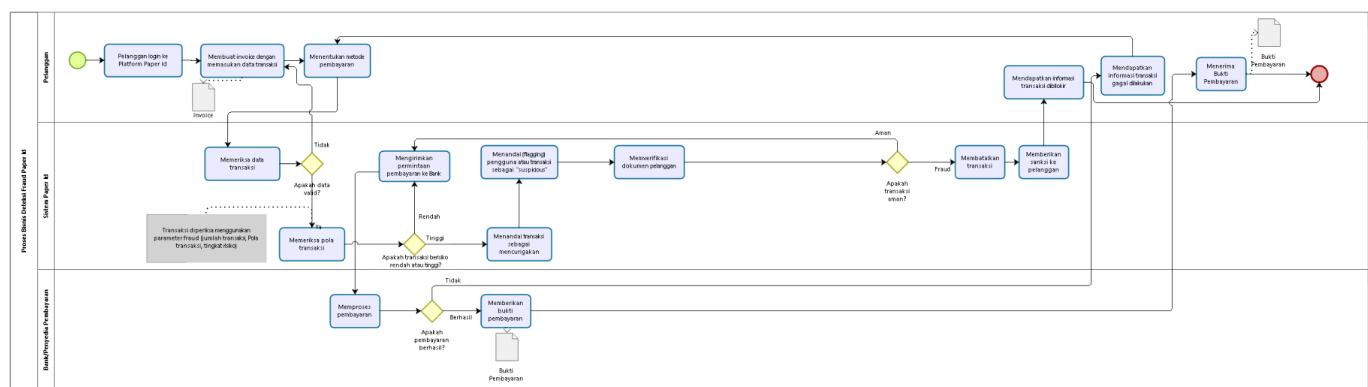
- **Data yang Hilang:** Tidak ditemukan data yang hilang pada sebagian besar kolom dalam dataset ini. Untuk beberapa kolom yang memiliki nilai hilang, langkah-langkah pembersihan akan dilakukan dengan menghapus baris yang memiliki data hilang mayoritas kolomnya, serta mengimputasi nilai yang hilang pada sebagian data.
- **Duplikasi dan Inkonsistensi:** Duplikasi data atau inkonsistensi dalam dataset tidak ditemukan secara signifikan, namun penggabungan antar dataset perlu memastikan kesesuaian ID seperti **dpt_id**, **company_id**, dan **dpt_promotion_id** untuk mencegah kesalahan dalam penggabungan data.
- **Outliers:** Beberapa transaksi memiliki jumlah yang sangat tinggi, tetapi berdasarkan analisis awal, jumlah transaksi yang mencapai angka miliaran masih dianggap wajar.

Namun, outliers akan terus diperiksa lebih lanjut untuk memastikan apakah mereka benar-benar mencurigakan atau dapat dipertahankan dalam analisis.

Dataset yang digunakan dalam analisis ini memiliki struktur yang jelas dan data yang relatif bersih. Beberapa transaksi dengan pola waktu yang sangat dekat antar satu sama lain menunjukkan potensi indikasi fraud, meskipun jumlah transaksi yang besar tidak terlihat mencurigakan. Selanjutnya, penggabungan data akan dilakukan dengan memperhatikan konsistensi antar ID untuk memastikan analisis yang akurat dan valid.

B. Hasil Analisis

1. Business Modeling and Fraud Flow Analysis



Gambar 2. Diagram BPMN Proses Deteksi Fraud.

Pada diagram BPMN diatas terdiri dari 3 *role* yang termasuk kedalam *swimlane* yaitu :

1. Pelanggan.
2. Sistem Paper Id.
3. Bank/ Penyedia Pembayaran.

Berikut merupakan penjelasan proses bisnis deteksi *fraud* dari diagram BPMN diatas:

1. Proses pertama diawali dengan *transaction initiation* yaitu pelanggan login ke platform Paper.id.
2. Pelanggan membuat invoice dengan memasukan data transaksi.
3. Pelanggan menentukan metode pembayaran.

4. Sistem memeriksa data transaksi.
5. Jika data valid maka sistem melanjutkan ke proses memeriksa pola transaksi.
6. Jika data tidak valid maka pelanggan kembali membuat *invoice* baru atau kembali ke proses nomer 2.
7. Data yang valid akan diperiksa oleh sistem secara *real-time* menggunakan parameter *fraud* (jumlah transaksim pola transaksi, da tingkat risiko).
8. Jika risiko transaksi rendah maka sistem akan mengirim permintaan pembayaran ke bank.
9. Jika risiko transaksi tinggi maka sistem akan menandai transaksi sebagai mencurigakan.
10. Sistem akan menandai atau melakukan *flagging* transaksi yang mencurigakan secara *real time* bahwa transaksi tersebut sebagai “*suspicious*”.
11. Sistem memverifikasi dokumen pelanggan yang mencurigakan.
12. Sistem menentukan keputusan apakah dokumen dari transaksi yang mencurigakan tersebut bersifat aman atau termasuk kedalam *fraud*.
13. Jika transaksi aman maka dilanjutkan dengan proses mengirim permintaan pembayaran
14. Jika transaksi termasuk ke dalam *fraud*, sistem akan membatalkan transaksi
15. Sistem memberikan sanksi ke pelanggan.
16. Pelanggan akan mendapatkan informasi transaksi diblokir .
17. Setelah sistem mengirimkan permintaan pembayaran ke bank untuk pelanggan dengan risiko transaksi rendah maka bank akan memproses pembayaran.
18. Jika pembayaran tidak berhasil maka pengguna akan mendapatkan bahwa informasi transaksi gagal dilakukan.
19. Pelanggan dapat menentukan kembali metode pembayaran yaitu proses no 3.
20. Jika pembayaran berhasil maka bank akan memberikan bukti pembayaran.
21. Pengguna akan menerima informasi dan bukti bahwa pembayaran berhasil dilakukan.
22. Proses selesai.

2. Data Cleaning, Feature Engineering, Scaling and Normalization

- **Data Cleaning**

Kode Program 3.1. Proses *data cleaning* pada Python.

```

import pandas as pd
import numpy as np

# Path ke file
file_path_company = '/dim_paper_company.csv'
file_path_promotion = '/dim_paper_promotion.csv'
file_path_payment_request =
'/fact_paper_digital_payment_request.csv'
file_path_payment_transaction =
'/fact_paper_digital_payment_transaction.csv'

# Membaca dataset
dt_company = pd.read_csv(file_path_company)
dt_promotion = pd.read_csv(file_path_promotion)
dt_digital_payment_request =
pd.read_csv(file_path_payment_request)
dt_digital_payment_transaction =
pd.read_csv(file_path_payment_transaction)

# Menampilkan informasi masing-masing dataset
print("==== dt_company.info() ===")
dt_company.info()

print("\n==== dt_promotion.info() ===")
dt_promotion.info()

print("\n==== dt_digital_payment_request.info() ===")
dt_digital_payment_request.info()

print("\n==== dt_digital_payment_transaction.info() ===")
dt_digital_payment_transaction.info()

==== dt_company.info() ===

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   company_id       50000 non-null   object  
 1   company_kyc_status_name 50000 non-null   object  
 2   company_kyb_status_name 50000 non-null   object  
 3   company_type_group    49999 non-null   object  
 4   company_phone_verified_flag 50000 non-null   float64 
 5   company_email_verified_flag 50000 non-null   float64 
 6   user_fraud_flag      50000 non-null   float64 
 7   testing_account_flag 50000 non-null   float64 
 8   blacklist_account_flag 50000 non-null   float64 
 9   package_active_name   50000 non-null   object  
 10  company_registered_datetime 50000 non-null   object  
dtypes: float64(5), object(6)
memory usage: 4.2+ MB

==== dt_promotion.info() ====
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   dpt_promotion_id 937 non-null    object  
 1   promotion_code     645 non-null    object  
 2   promotion_name     645 non-null    object  
 3   transaction_promo_cashback_amount 50000 non-null   float64 
dtypes: float64(1), object(3)
memory usage: 1.5+ MB

==== dt_digital_payment_request.info() ====
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Column           Non-Null Count  Dtype  
 1   Column           Non-Null Count  Dtype  
 2   Column           Non-Null Count  Dtype  

```

```

0    dpt_id              50000 non-null  object
1    total_fee_amount     50000 non-null  float64
2    document_type_name   50000 non-null  object
dtypes: float64(1), object(2)
memory usage: 1.1+ MB

==== dt_digital_payment_transaction.info() ====

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   dpt_id          50000 non-null  object 
 1   dpt_promotion_id 937 non-null   object 
 2   buyer_id         50000 non-null  object 
 3   seller_id        50000 non-null  object 
 4   transaction_amount 50000 non-null  float64
 5   payment_method_name 50000 non-null  object 
 6   payment_provider_name 50000 non-null  object 
 7   transaction_created_datetime 50000 non-null  object 
 8   transaction_updated_datetime 50000 non-null  object 
dtypes: float64(1), object(8)
memory usage: 3.4+ MB

```

● Handle Missing Values

Kode Program 3.2. Proses *handle missing values* pada Python.

```

# Mengecek nilai null di dataset dt_company
nulls_dt_company = dt_company.isnull().sum()

# Mengecek nilai null di dataset dt_promotion
nulls_dt_promotion = dt_promotion.isnull().sum()

# Mengecek nilai null di dataset dt_digital_payment_request
nulls_dt_payment_request =
dt_digital_payment_request.isnull().sum()

```

```
# Mengecek nilai null di dataset dt_digital_payment_transaction
nulls_dt_payment_transaction =
dt_digital_payment_transaction.isnull().sum()

# Menampilkan jumlah nilai null di setiap kolom untuk setiap
dataset
print("Nilai null di dt_company:\n", nulls_dt_company)
print("\nNilai null di dt_promotion:\n", nulls_dt_promotion)
print("\nNilai null di dt_digital_payment_request:\n",
nulls_dt_payment_request)
print("\nNilai null di dt_digital_payment_transaction:\n",
nulls_dt_payment_transaction)

Nilai null di dt_company:
company_id                      0
company_kyc_status_name          0
company_kyb_status_name          0
company_type_group                1
company_phone_verified_flag      0
company_email_verified_flag      0
user_fraud_flag                  0
testing_account_flag              0
blacklist_account_flag            0
package_active_name               0
company_registered_datetime       0
dtype: int64

Nilai null di dt_promotion:
dpt_promotion_id                 49063
promotion_code                    49355
promotion_name                    49355
transaction_promo_cashback_amount 0
dtype: int64

Nilai null di dt_digital_payment_request:
dpt_id                            0
total_fee_amount                  0
document_type_name                0
dtype: int64

Nilai null di dt_digital_payment_transaction:
dpt_id                            0
dpt_promotion_id                 49063
buyer_id                           0
seller_id                          0
transaction_amount                 0
payment_method_name                0
```

```

payment_provider_name          0
transaction_created_datetime   0
transaction_updated_datetime   0
dtype: int64

dt_promotion.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   dpt_promotion_id    937 non-null   object  
 1   promotion_code       645 non-null   object  
 2   promotion_name       645 non-null   object  
 3   transaction_promo_cashback_amount  50000 non-null  float64 
dtypes: float64(1), object(3)
memory usage: 1.5+ MB

dt_promotion = dt_promotion.dropna(subset=['dpt_promotion_id'])

dt_promotion.info()

<class 'pandas.core.frame.DataFrame'>
Index: 937 entries, 78 to 49970
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   dpt_promotion_id    937 non-null   object  
 1   promotion_code       645 non-null   object  
 2   promotion_name       645 non-null   object  
 3   transaction_promo_cashback_amount  937 non-null  float64 
dtypes: float64(1), object(3)
memory usage: 36.6+ KB

```

Pada dataset “dt_company”, hanya kolom “company_type_group” yang memiliki nilai null, sementara kolom lainnya lengkap tanpa data yang hilang. Pada dataset “dt_promotion”, sebagian besar kolom memiliki banyak nilai null, terutama pada kolom “dpt_promotion_id”, “promotion_code”, dan “promotion_name”, yang dapat mempengaruhi analisis lebih lanjut. Kolom “transaction_promo_cashback_amount” di dataset ini tidak memiliki nilai null. Sebagai langkah perbaikan, baris dengan nilai null pada “dpt_promotion_id” telah dihapus. Di dataset “dt_digital_payment_request”, tidak ada kolom yang memiliki nilai null, menunjukkan data yang lengkap. Sedangkan pada dataset “dt_digital_payment_transaction”, kolom “dpt_promotion_id” juga memiliki nilai

null. Secara keseluruhan, pembersihan data diperlukan pada dataset yang memiliki nilai null pada kolom-kolom penting untuk memastikan analisis yang lebih akurat.

- **Outliers (Cek outliers transaction_amount dari Digital Payment Transaction Data)**

Kode Program 3.3. Proses pengecekan *outliers* pada Python.

```
# Five-number summary
Q1 =
dt_digital_payment_transaction['transaction_amount'].quantile(0.25)
)
Q3 =
dt_digital_payment_transaction['transaction_amount'].quantile(0.75)
)
IQR = Q3 - Q1

# Definisi batas outlier
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Data outliers
outliers =
dt_digital_payment_transaction[(dt_digital_payment_transaction['transaction_amount'] < lower_bound) |
(dt_digital_payment_transaction['transaction_amount'] >
upper_bound)]
print(f"Jumlah outliers: {len(outliers)}")
print(outliers.head())
```

```
Jumlah outliers: 6777
dpt_id
dpt_promotion_id \
12 3994ae53eabf3ffe2d20340a45f3c75b99c8458815bc1c...
NaN
14 a6dcc743fe284523416b67914fe1c5b11bb26e70999b92...
NaN
17 2d2378e0f0a8786dccf6fc23e1ec68a68b5d2d818dc2cc...
NaN
22 890aa93fbe0e0edce8ab0bbe3d9fe06d684336991b2652...
```

```

NaN
31 8f110fe98176434656c2867d9249317973607c14fc9157...
NaN

           buyer_id \
12  4e748c692460aa2d2f08354ea0ef0b2146addf170d4c6a...
14  ffcb909cf0b66998e1f034d00ed93b65350b1defddd6e2...
17  ecad68c6f8f865f785578f9be9fde7916f2f598858c28d...
22  b958f0abcc3036f39ef281236179d70aa1f6e44c9775cf...
31  19fb72a33fb7d746ea5049eb55a7399479614bd8eb4eaf...

           seller_id
transaction_amount \
12  5d2233f5a1a6435891142442fac09a77809d0c16496f07...
9.999939e+07
14  5d2233f5a1a6435891142442fac09a77809d0c16496f07...
8.470973e+08
17  5d2233f5a1a6435891142442fac09a77809d0c16496f07...
1.015500e+08
22  5d2233f5a1a6435891142442fac09a77809d0c16496f07...
2.500000e+08
31  19fb72a33fb7d746ea5049eb55a7399479614bd8eb4eaf...
5.216400e+07

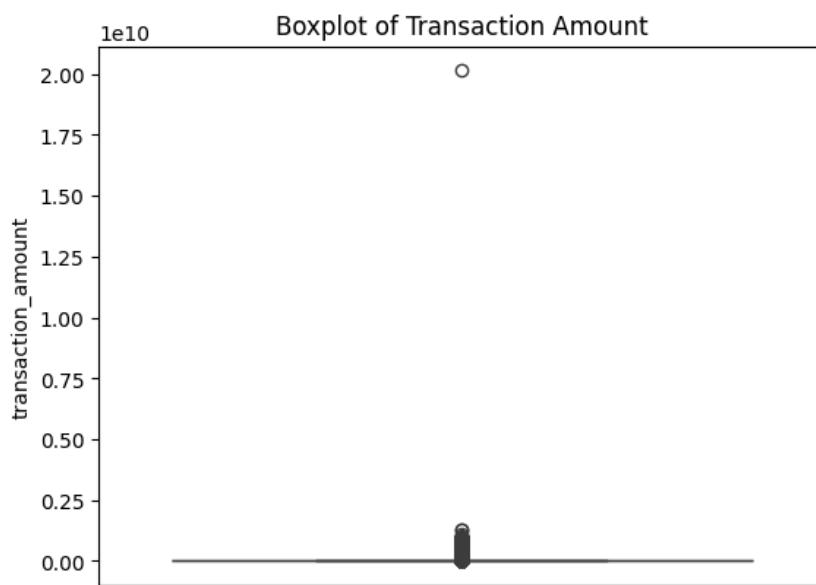
           payment_method_name payment_provider_name \
12  MITRA PEMBAYARAN DIGITAL                      BLIBLI
14  CREDIT CARD                                     VISA
17  MITRA PEMBAYARAN DIGITAL                      BLIBLI
22  BANK TRANSFER                                    BRI
31  CREDIT CARD                                     VISA

transaction_created_datetime transaction_updated_datetime
12  2023-04-03 14:04:58.678165    2023-04-03 14:24:36.286420
14  2023-11-12 10:56:42.301337    2023-11-12 10:59:53.169210
17  2023-06-19 22:32:58.382015    2023-06-19 22:40:25.571073
22  2023-11-22 13:19:38.824387    2023-11-22 13:21:00.217084
31  2023-09-22 16:17:49.459179    2023-09-22 16:20:41.017829

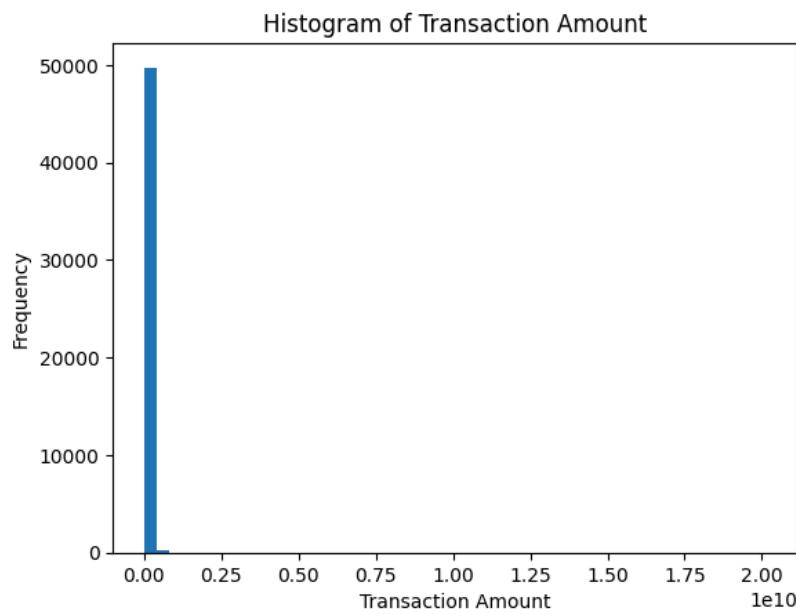
import seaborn as sns
import matplotlib.pyplot as plt

sns.boxplot(dt_digital_payment_transaction['transaction_amount'])
plt.title('Boxplot of Transaction Amount')
plt.show()

```



```
plt.hist(dt_digital_payment_transaction['transaction_amount'],
bins=50)
plt.title('Histogram of Transaction Amount')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()
```



Total outlier yang terdeteksi adalah 6.777 transaksi dengan jumlah yang sangat besar,

melebihi batas yang ditentukan oleh aturan IQR (Interquartile Range) pada kolom “transaction_amount”. Transaksi dengan nilai besar, meskipun secara statistik tergolong sebagai outlier, masih dapat dianggap wajar dalam konteks transaksi antar bisnis, terutama dalam lingkungan yang melibatkan jumlah uang dalam skala besar. Oleh karena itu, meskipun transaksi dengan jumlah yang sangat tinggi ini terlihat tidak biasa jika dibandingkan dengan transaksi lainnya, mereka tidak menunjukkan indikasi masalah atau kecurangan yang jelas dalam konteks data yang ada karena tidak ditemukan masalah signifikan pada kolom-kolom lain yang berkaitan dengan tanda kecurangan seperti “user_fraud_flag” atau “blacklist_account_flag”.

- **Outliers (Cek outliers total_fee_amount dari Digital Payment Request Data)**

Kode Program 3.4. Proses pengecekan *outliers* pada Python.

```
# Five-number summary
Q1 = dt_digital_payment_request['total_fee_amount'].quantile(0.25)
Q3 = dt_digital_payment_request['total_fee_amount'].quantile(0.75)
IQR = Q3 - Q1

# Definisi batas outlier
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Data outliers
outliers =
dt_digital_payment_request[(dt_digital_payment_request['total_fee_amount'] < lower_bound) |
                             (dt_digital_payment_request['total_fee_amount'] >
upper_bound) ]

print(f"Jumlah outliers: {len(outliers)}")
print(outliers.head())
```

```
Jumlah outliers: 6884
dpt_id
total_fee_amount \
12 3994ae53eabf3ffe2d20340a45f3c75b99c8458815bc1c...
1.429267e+06
14 a6dcc743fe284523416b67914fe1c5b11bb26e70999b92...
```

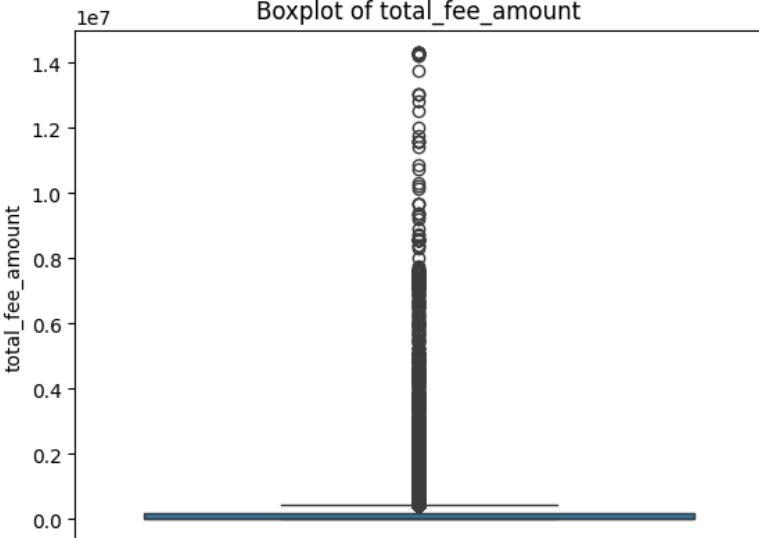
```

1.251868e+07
17 2d2378e0f0a8786dccf6fc23e1ec68a68b5d2d818dc2cc...
1.550000e+06
31 8f110fe98176434656c2867d9249317973607c14fc9157...
7.824600e+05
34 63013866b06a6bfdb1e3e046d42eaedd7fbf545c21d45a...
1.392000e+06

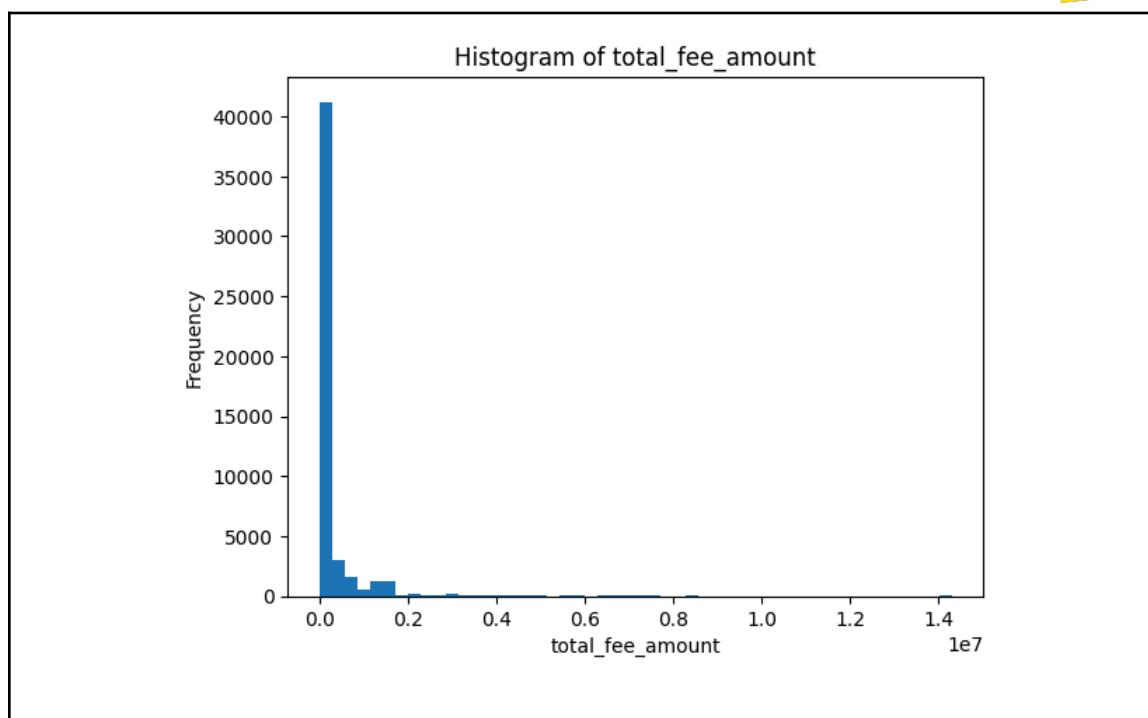
    document_type_name
12          PAY-OUT
14          PAY-OUT
17          PAY-OUT
31          PAY-IN
34          PAY-OUT

import seaborn as sns
import matplotlib.pyplot as plt

sns.boxplot(dt_digital_payment_request['total_fee_amount'])
plt.title('Boxplot of total_fee_amount')
plt.show()


plt.hist(dt_digital_payment_request['total_fee_amount'], bins=50)
plt.title('Histogram of total_fee_amount')
plt.xlabel('total_fee_amount')
plt.ylabel('Frequency')
plt.show()

```



Analisis pada kolom “total_fee_amount” menunjukkan 6.884 transaksi sebagai outlier, dengan nilai fee yang jauh lebih tinggi atau lebih rendah dari rentang normal. Meskipun terdeteksi sebagai outlier, transaksi ini mungkin masih sah, terutama jika terkait dengan jenis pembayaran besar seperti "PAY-OUT" atau "PAY-IN" dan berasal dari transaksi nyata bukan dari kesalahan input atau sistem, jadi baris-baris tersebut tetap di pertahankan sebagai mana aslinya.

- **Outliers (Cek outliers transaction_promo_cashback_amount dari Promotion Data)**

Kode Program 3.5. Proses pengecekan *outliers* pada Python.

```
# Five-number summary
Q1 =
dt_promotion['transaction_promo_cashback_amount'].quantile(0.25)
Q3 =
dt_promotion['transaction_promo_cashback_amount'].quantile(0.75)
IQR = Q3 - Q1

# Definisi batas outlier
```

```

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Data outliers
outliers =
dt_promotion[(dt_promotion['transaction_promo_cashback_amount'] <
lower_bound) |
(dt_promotion['transaction_promo_cashback_amount'] > upper_bound)]
```

```

print(f"Jumlah outliers: {len(outliers)}")
print(outliers.head())

```

```

Jumlah outliers: 67
dpt_promotion_id
promotion_code \
202 877cbc48ac832c9f872a12d06d949376f0d918ad48868d...
PPRMAYN2
266 0a77054fc75a9d6dbb33ae4be6e3420e557715ef8c3a2a...
PPRMARF1
947 6f4716fd7e3cbd6db611be014c647f08203b97c53c6a41...
NaN
2079 48dc720266358e4ef9aa83a8bd747bc6119b18a7f87141...
NaN
2228 92eb0e2ec24e64a91e40062bfdc2636e3452641f577c93...
PPRFEB2

promotion_name  transaction_promo_cashback_amount
202  Promo Loyalty Mei 2 450000.0
266  MERIAH-1 499800.0
947  NaN 1000000.0
2079  NaN 750000.0
2228  Feb-bastic 2 1000000.0

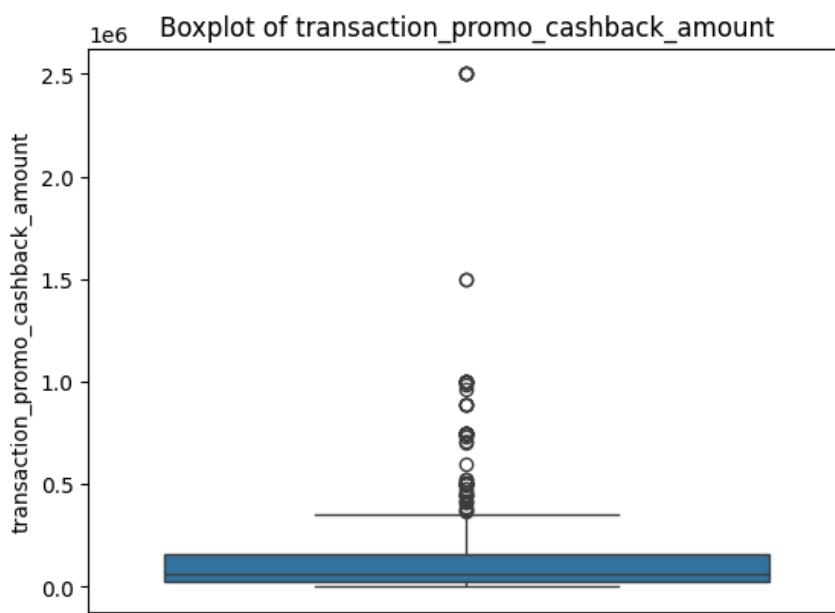
```

```

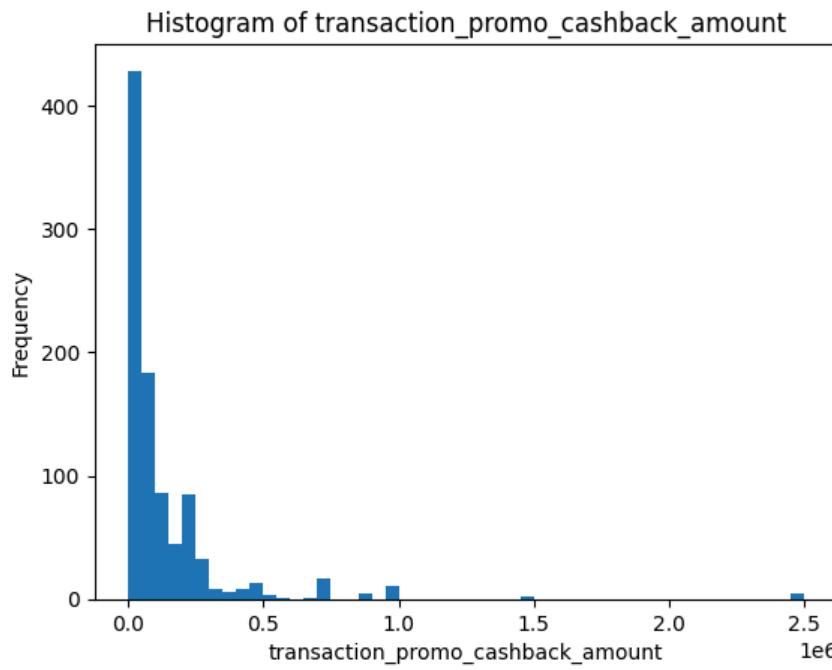
import seaborn as sns
import matplotlib.pyplot as plt

sns.boxplot(dt_promotion['transaction_promo_cashback_amount'])
plt.title('Boxplot of transaction_promo_cashback_amount')
plt.show()

```



```
plt.hist(dt_promotion['transaction_promo_cashback_amount'],
bins=50)
plt.title('Histogram of transaction_promo_cashback_amount')
plt.xlabel('transaction_promo_cashback_amount')
plt.ylabel('Frequency')
plt.show()
```



Hasil analisis pada kolom “transaction_promo_cashback_amount” menunjukkan adanya 67 transaksi yang terdeteksi sebagai outlier dengan nilai cashback yang tinggi. Namun, setelah ditinjau lebih lanjut, nilai-nilai ini masih dapat dianggap wajar dan realistik, terutama jika terkait dengan promosi besar. Tidak ditemukan indikasi bahwa nilai besar tersebut disebabkan oleh kesalahan input atau sistem, melainkan berasal dari transaksi asli. Meskipun terdapat beberapa entri dengan nilai NaN pada kolom “promotion_code” dan “promotion_name”, hal ini tidak mempengaruhi validitas nilai cashback itu sendiri. Oleh karena itu, nilai-nilai outlier pada kolom ini tidak perlu dihapus dari dataset karena tetap relevan dan mencerminkan kondisi transaksi yang sebenarnya. Namun, data dengan informasi promosi yang tidak lengkap harus diperiksa lebih lanjut untuk memastikan kelengkapan dan keakuratan informasi yang terkait.

- **Anomali (pada Digital Payment Transaction Data)**

Kode Program 3.6. Proses pengecekan *anomali* pada Python.

```
# Anomali 1: Buyer dan seller adalah akun yang sama
buyer_seller_anomalies =
dt_digital_payment_transaction[dt_digital_payment_transaction['buyer_id'] == dt_digital_payment_transaction['seller_id']]

# Anomali 2: Tanggal transaksi tidak valid
datetime_anomalies = dt_digital_payment_transaction[
    pd.to_datetime(dt_digital_payment_transaction['transaction_created_datetime']) >
    pd.to_datetime(dt_digital_payment_transaction['transaction_updated_datetime'])
]

print("Anomali Transaction Amount:\n", amount_anomalies.head())
print("Anomali Buyer-Seller:\n", buyer_seller_anomalies.head())
print("Anomali Tanggal Transaksi:\n", datetime_anomalies.head())
```

Terdapat anomali pada beberapa transaksi, di mana perusahaan yang tercatat sebagai pembeli “buyer_id” dan penjual “seller_id” memiliki ID yang sama. Hal ini menunjukkan adanya kemungkinan transaksi yang dilakukan antara dua pihak yang sebenarnya merupakan entitas yang sama. Hal ini sangat mencurigakan karena dapat mengindikasikan adanya potensi penyalahgunaan sistem, seperti manipulasi untuk mendapatkan keuntungan yang tidak sah, misalnya melalui cashback atau diskon yang hanya berlaku pada transaksi antar akun yang berbeda.

- **Anomali (pada Digital Payment Request Data)**

Kode Program 3.7. Proses pengecekan *anomali* pada Python.

```
# Anomali 1: Document type yang tidak valid dengan fee tertentu
# (contoh aturan logis)

invalid_document_fee = dt_digital_payment_request[
    (dt_digital_payment_request['document_type_name'] == 'Standard') &
    (dt_digital_payment_request['total_fee_amount'] > 1000)
]

print("Anomali Document Type:\n", invalid_document_fee.head())
```

Anomali Document Type:
Empty DataFrame
Columns: [dpt_id, total_fee_amount, document_type_name]
Index: []

Tidak ditemukan anomali yang berkaitan dengan tipe dokumen yang tidak valid dan biaya tertentu dalam dataset transaksi. Ketika mencari transaksi dengan tipe dokumen “Standard” yang memiliki “total_fee_amount” lebih dari 1000, hasilnya adalah DataFrame kosong, yang berarti semua transaksi dengan tipe dokumen 'Standard' telah sesuai dengan aturan atau kebijakan yang telah ditentukan, yaitu tidak ada transaksi yang memiliki total fee amount melebihi batas yang telah ditentukan untuk tipe dokumen tersebut.

- **Anomali (pada Promotion Data)**

Kode Program 3.8. Proses pengecekan *anomali* pada Python.

```
# Gabungkan data untuk mendapatkan informasi yang dibutuhkan
merged_data = pd.merge(dt_digital_payment_transaction,
dt_promotion, on='dpt_promotion_id', how='left')

# Anomali 1: Cashback lebih besar dari jumlah transaksi
cashback_anomalies =
merged_data[merged_data['transaction_promo_cashback_amount'] >
merged_data['transaction_amount']]

# Anomali 2: Promotion code digunakan berulang oleh akun
promo_usage = merged_data.groupby(['promotion_code',
'buyer_id']).size().reset_index(name='usage_count')
frequent_promo_users = promo_usage[promo_usage['usage_count'] >
10]

# Output hasil
print("Anomali Cashback (Cashback lebih besar dari transaksi):\n",
cashback_anomalies.head())
print("Penggunaan Promo Berulang (Lebih dari 10 kali):\n",
frequent_promo_users.head())
```

```
Anomali Cashback (Cashback lebih besar dari transaksi):
Empty DataFrame
Columns: [dpt_id, dpt_promotion_id, buyer_id, seller_id,
transaction_amount, payment_method_name, payment_provider_name,
transaction_created_datetime, transaction_updated_datetime,
promotion_code, promotion_name, transaction_promo_cashback_amount]
Index: []
Penggunaan Promo Berulang (Lebih dari 10 kali):
Empty DataFrame
Columns: [promotion_code, buyer_id, usage_count]
Index: []
```

Tidak ditemukan anomali mencurigakan terkait dengan cashback yang lebih besar dari jumlah transaksi maupun penggunaan kode promo yang berulang. Untuk anomali pertama, yaitu mencari transaksi dengan cashback lebih besar dari transaksi, hasilnya adalah DataFrame kosong, yang berarti tidak terdapat transaksi yang memenuhi kriteria tersebut, sehingga cashback yang diberikan berada dalam batas wajar. Kemudian untuk

anomali kedua yaitu penggunaan kode promo yang berulang oleh akun, hasilnya juga menunjukkan DataFrame kosong, yang berarti tidak terdapat akun yang menggunakan kode promo lebih dari 10 kali, menandakan bahwa penggunaan kode promo tetap dalam batas normal dan tidak ada indikasi penyalahgunaan.

- **Anomali (pada Data Perusahaan)**

Kode Program 3.9. Proses pengecekan *anomali* pada Python.

```
# Konversi kolom datetime menjadi format datetime
dt_company['registered_date'] =
pd.to_datetime(dt_company['company_registered_datetime'])
dt_company['date_only'] = dt_company['registered_date'].dt.date # Ambil hanya tanggal

# Menghitung jumlah pendaftaran per hari
batch_registration =
dt_company.groupby('date_only').size().reset_index(name='registration_count')

# Menghitung statistik deskriptif: Q1, Q3, IQR
Q1 = batch_registration['registration_count'].quantile(0.25)
Q3 = batch_registration['registration_count'].quantile(0.75)
IQR = Q3 - Q1

# Menentukan batas normal menggunakan IQR
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Mendeteksi batch registrasi yang mencurigakan (di luar batas normal)
suspicious_batches = batch_registration[
    (batch_registration['registration_count'] < lower_bound) | 
    (batch_registration['registration_count'] > upper_bound)
]

# Output hasil
print(f"Batas Normal: {lower_bound:.2f} hingga {upper_bound:.2f}")
print("Batch Registrasi Mencurigakan:\n",
```

```
suspicious_batches.head()

Batas Normal: -75.00 hingga 133.00
Batch Registrasi Mencurigakan:
    date_only  registration_count
117  2020-01-07              157
369  2021-09-29              218
393  2021-10-30              712
394  2021-10-31              146
414  2021-11-20              149
```

Terdapat beberapa batch registrasi yang mencurigakan berdasarkan jumlah pendaftaran perusahaan per-hari yang berada di luar batas normal. Dengan menggunakan IQR (Interquartile Range), ditemukan batas normal untuk jumlah pendaftaran perusahaan per hari antara -75 dan 133. Namun, terdapat beberapa tanggal yang mencatat jumlah pendaftaran yang jauh lebih tinggi dari batas atas (133), seperti pada 2021-09-29 terdapat 218 pendaftar, 2021-10-30 terdapat 712 pendaftar, dan 2021-10-31 terdapat 146 pendaftar. Angka-angka ini menunjukkan adanya lonjakan pendaftaran yang tidak wajar, yang dapat mengindikasikan adanya aktivitas yang mencurigakan.

- **Suspicious Missing Data or Inconsistencies in Key Fraud-Related**

Kode Program 3.10. Proses pengecekan *suspicious missing data* atau *inconsistencies in key fraud-related* pada Python.

```
# Memeriksa missing values pada kolom yang relevan
missing_data = dt_company[['user_fraud_flag',
                           'blacklist_account_flag']].isnull().sum()
print(f"Missing data pada 'user_fraud_flag' dan
      'blacklist_account_flag':\n{missing_data}")

Missing data pada 'user_fraud_flag' dan 'blacklist_account_flag':
user_fraud_flag          0
blacklist_account_flag    0
dtype: int64

# Memeriksa inkonsistensi antara 'user_fraud_flag' dan
# 'blacklist_account_flag'
inconsistent_data = dt_company[(dt_company['user_fraud_flag'] ==
                                 'true') &
```

```
(dt_company['blacklist_account_flag'] != 'true')]

print(f"Inkonsistensi data:\n{inconsistent_data}")

Inkonsistensi data:
Empty DataFrame
Columns: [company_id, company_kyc_status_name,
company_kyb_status_name, company_type_group,
company_phone_verified_flag, company_email_verified_flag,
user_fraud_flag, testing_account_flag, blacklist_account_flag,
package_active_name, company_registered_datetime, registered_date,
date_only]
Index: []

# Memeriksa apakah terdapat nilai yang tidak konsisten pada kolom
# flag
invalid_flags =
dt_company[dt_company['user_fraud_flag'].isin(['true', 'false'])]
print(f"Data tidak konsisten di
'user_fraud_flag':\n{invalid_flags}")

Data tidak konsisten di 'user_fraud_flag':
Empty DataFrame
Columns: [company_id, company_kyc_status_name,
company_kyb_status_name, company_type_group,
company_phone_verified_flag, company_email_verified_flag,
user_fraud_flag, testing_account_flag, blacklist_account_flag,
package_active_name, company_registered_datetime, registered_date,
date_only]
Index: []

# Pastikan kolom waktu transaksi dalam format datetime
dt_digital_payment_transaction['transaction_created_datetime'] =
pd.to_datetime(dt_digital_payment_transaction['transaction_created_datetime'])
dt_digital_payment_transaction['transaction_updated_datetime'] =
pd.to_datetime(dt_digital_payment_transaction['transaction_updated_datetime'])

# Memeriksa jika ada transaksi dengan waktu pembaruan lebih awal
# dari waktu pembuatan
invalid_time_data =
dt_digital_payment_transaction[dt_digital_payment_transaction['tra
```

```

nsaction_updated_datetime'] <
dt_digital_payment_transaction['transaction_created_datetime']]  

print(f"Transaksi dengan waktu pembaruan lebih awal dari waktu  

pembuatan:\n{invalid_time_data}")

Transaksi dengan waktu pembaruan lebih awal dari waktu pembuatan:  

Empty DataFrame  

Columns: [dpt_id, dpt_promotion_id, buyer_id, seller_id,  

transaction_amount, payment_method_name, payment_provider_name,  

transaction_created_datetime, transaction_updated_datetime,  

zscore, anomaly, amount_zscore]  

Index: []

# Memeriksa apakah setiap 'dpt_promotion_id' di transaction_data  

memiliki data terkait di promotion_data
missing_promotions =
dt_digital_payment_transaction[dt_digital_payment_transaction['dpt  

_promotion_id'].isin(dt_promotion['dpt_promotion_id'])]
print(f"Transaksi yang tidak memiliki promosi  

terkait:\n{missing_promotions}")

Transaksi yang tidak memiliki promosi terkait:  

Empty DataFrame  

Columns: [dpt_id, dpt_promotion_id, buyer_id, seller_id,  

transaction_amount, payment_method_name, payment_provider_name,  

transaction_created_datetime, transaction_updated_datetime,  

zscore, anomaly, amount_zscore]  

Index: []

# Memeriksa perusahaan dengan status KYC/KBV yang tidak valid tapi  

terlibat dalam transaksi
invalid_kyc_companies =
dt_company[(dt_company['company_kyc_status_name'] ==  

'AKUN_DIBEKUKAN') |  

(dt_company['company_kyb_status_name'] == 'AKUN_DIBEKUKAN')]

# Memastikan perusahaan ini tidak terlibat dalam transaksi
companies_with_transactions =
dt_digital_payment_transaction[dt_digital_payment_transaction['sel  

ler_id'].isin(invalid_kyc_companies['company_id'])]
```

```
print(f"Perusahaan dengan status KYC/KBV tidak valid yang terlibat dalam transaksi:\n{companies_with_transactions}")
```

Perusahaan dengan status KYC/KBV tidak valid yang terlibat dalam transaksi:

	dpt_id
55	b6b7962cdedcff94252b665ede6ba547dd7ec2c748104f...
NaN	
57	dd4bd7d7dd2bf1ec0d65e463331c504964b41ff603fcdf...
NaN	
71	06f4d1567d2ac261aa6af2467406e41704924c709348fe...
NaN	
72	8e977c69c62f0449f4e0a05a2fc6501c04c706fa29d5a...
NaN	
139	4e3039b0d73bf33566d21fb3cb357a2d191b919755f518...
NaN	
...	...
...	...
49848	74140921ac0ea87512a8982d22722a47f73d6053def4ac...
NaN	
49902	44b2f6125252eff4ebde3a7c43276943dad3acadb79849...
NaN	
49904	e7171f40afc9ed490be88c2078f6245c9e51d0206ff317...
NaN	
49935	dd83a26618c201c4dfe13b363f6b20cd2aed2802ffd92b...
NaN	
49954	a9c10f47c20715de54c22295a8b0e30f45977d76861c14...
NaN	
	buyer_id \
55	d76c958f4e0dd86f00a166da4c5d11d13f2a144a21f308...
57	25254fe39611d129f85367f800017c529900ec1b271206...
71	97e97173aebedc3df6d150745e6f9be87096d9b3194a5c...
72	ca828424979b71c9819e1110e5f2a0c3a13e7528de801d...
139	482665a02edd13a253c6a64917ae546ab9d34a2b92fb0d...
...	...
49848	a77ddf7b007c7f27700a9db82956ccbd89f950a2a854af...
49902	633a51a759692c477f63177f2ee592ac702c81390e6f74...
49904	eb8e6558e9f2d08f7d0ae158dce04445f0adfbdb3a1d191...
49935	f71664548c5df8e51c3c018ffb8b49642a314b298377c2...
49954	d6b762509e106d03205b7df124133b6a509b0741d5e801...
	seller_id
transaction_amount \	
55	d76c958f4e0dd86f00a166da4c5d11d13f2a144a21f308...
1.500000e+07	
57	25254fe39611d129f85367f800017c529900ec1b271206...
1.497862e+04	
71	97e97173aebedc3df6d150745e6f9be87096d9b3194a5c...
1.076003e+05	

```

72      ca828424979b71c9819e1110e5f2a0c3a13e7528de801d...
2.459500e+06
139     482665a02edd13a253c6a64917ae546ab9d34a2b92fb0d...
1.979250e+06
...
...
49848   a77ddf7b007c7f27700a9db82956ccbd89f950a2a854af...
2.000093e+04
49902   633a51a759692c477f63177f2ee592ac702c81390e6f74...
3.000000e+06
49904   eb8e6558e9f2d08f7d0ae158dce04445f0adfb3a1d191...
2.739150e+06
49935   f71664548c5df8e51c3c018ffb8b49642a314b298377c2...
1.499968e+04
49954   d6b762509e106d03205b7df124133b6a509b0741d5e801...
5.000000e+07

            payment_method_name  payment_provider_name \
55      MITRA_PAYMENT_DIGITAL          BLIBLI
57      MITRA_PAYMENT_DIGITAL          BLIBLI
71      MITRA_PAYMENT_DIGITAL          BLIBLI
72          CREDIT_CARD                VISA
139     MITRA_PAYMENT_DIGITAL          TOKOPEDIA_CREDIT_CARD
...
...
49848   MITRA_PAYMENT_DIGITAL          BLIBLI
49902       CREDIT_CARD                MASTERCARD
49904   MITRA_PAYMENT_DIGITAL          SHOPEE
49935   MITRA_PAYMENT_DIGITAL          BLIBLI
49954       CREDIT_CARD                MASTERCARD

            transaction_created_datetime transaction_updated_datetime
zscore \
55      2023-10-05 17:58:52.268863    2023-10-05 18:01:14.227600
-0.044004
57      2023-12-25 16:16:40.779303    2023-12-25 16:21:06.104634
-0.185775
71      2023-12-21 17:44:04.753258    2023-12-21 17:48:55.103081
-0.184898
72      2023-11-24 13:19:40.077983    2023-11-24 13:21:09.774015
-0.162648
139     2023-05-22 13:37:00.337889    2023-05-22 13:37:00.337889
-0.167191
...
...
49848   2023-12-12 04:51:32.975438    2023-12-12 04:52:21.004473
-0.185727
49902   2023-12-14 15:26:24.357234    2023-12-14 15:29:30.626949
-0.157534
49904   2023-10-28 17:16:31.716070    2023-10-28 17:28:12.037397
-0.160002
49935   2023-07-28 18:54:48.330895    2023-07-28 19:12:44.893223
-0.185775

```

```
49954 2023-09-21 17:11:07.656549 2023-09-21 17:12:21.785734
0.287123
```

	anomaly	amount_zscore
55	1	-0.044004
57	1	-0.185775
71	1	-0.184898
72	1	-0.162648
139	1	-0.167191
...
49848	1	-0.185727
49902	1	-0.157534
49904	1	-0.160002
49935	1	-0.185775
49954	1	0.287123

```
[3033 rows x 12 columns]
```

Pada proses pemeriksaan data yang dilakukan terhadap beberapa kolom yang relevan dengan potensi kecurangan dan inkonsistensi, tidak ditemukan adanya masalah signifikan. Pertama, dalam kolom “**user_fraud_flag**” dan “**blacklist_account_flag**”, yang seharusnya menunjukkan status kecurangan atau blacklist pada akun, tidak ditemukan adanya nilai yang hilang (missing values) yang menunjukkan bahwa data tersebut lengkap dan terisi dengan baik. Selain itu, tidak terdapat inkonsistensi antara nilai pada kedua kolom tersebut, yang berarti tidak terdapat kasus di mana akun yang terdeteksi sebagai fraud tidak terdaftar sebagai blacklist.

Selain itu, tidak terdapat transaksi dengan waktu pembaruan lebih awal daripada waktu pembuatan, serta semua transaksi memiliki referensi yang valid ke promosi “**dpt_promotion_id**”. Namun, ditemukan beberapa perusahaan dengan status **KYC/KBV** tidak valid (misalnya "AKUN_DIBEKUKAN") yang masih terlibat dalam transaksi. Hal ini mengindikasikan potensi inkonsistensi yang perlu ditindaklanjuti, karena perusahaan dengan status dibekukan seharusnya tidak dapat melakukan transaksi.

- **Future Engineering**

Buyer-Seller Relationship Score:

Kode Program 3.11. Proses pengecekan *buyer-seller relationship score* pada Python.

```
# Hitung jumlah transaksi antara setiap pasangan buyer-seller
buyer_seller_interaction =
dt_digital_payment_transaction.groupby(["buyer_id",
"seller_id"]).size().reset_index(name="transaction_count")

# Tampilkan hasil
print(buyer_seller_interaction.head())

          buyer_id \
0  00048ebf5503ef1dfb03eec05312583eeb2b57320ac682...
1  001046b5061e28476b83fe2335b04d3210bed72a2fee17...
2  00119737eef11ff1d30c2061dd1e19c06d963d5a125c92...
3  0012614e5a1366f102a3497b67f8ec9a8009c802aa6959...
4  0013cdaff46e67574660e0ddd214e5032e3ff5d94744e1...

                           seller_id
transaction_count
0  00048ebf5503ef1dfb03eec05312583eeb2b57320ac682...
2
1  5d2233f5a1a6435891142442fac09a77809d0c16496f07...
2
2  5d2233f5a1a6435891142442fac09a77809d0c16496f07...
1
3  5d2233f5a1a6435891142442fac09a77809d0c16496f07...
1
4  5d2233f5a1a6435891142442fac09a77809d0c16496f07...
13

# Normalisasi transaction_count agar berada di rentang [0, 1]
max_transaction =
buyer_seller_interaction["transaction_count"].max()
buyer_seller_interaction["relationship_score"] =
buyer_seller_interaction["transaction_count"] / max_transaction

# Tampilkan hasil
print(buyer_seller_interaction.head())

          buyer_id \
0  00048ebf5503ef1dfb03eec05312583eeb2b57320ac682...
1  001046b5061e28476b83fe2335b04d3210bed72a2fee17...
```

```

2 00119737eef11ff1d30c2061dd1e19c06d963d5a125c92...
3 0012614e5a1366f102a3497b67f8ec9a8009c802aa6959...
4 0013cdaff46e67574660e0ddd214e5032e3ff5d94744e1...

                                seller_id
transaction_count \
0 00048ebf5503ef1dfb03eec05312583eeb2b57320ac682...
2
1 5d2233f5a1a6435891142442fac09a77809d0c16496f07...
2
2 5d2233f5a1a6435891142442fac09a77809d0c16496f07...
1
3 5d2233f5a1a6435891142442fac09a77809d0c16496f07...
1
4 5d2233f5a1a6435891142442fac09a77809d0c16496f07...
13

    relationship_score
0          0.001580
1          0.001580
2          0.000790
3          0.000790
4          0.010269

# Tandai pasangan buyer-seller mencurigakan
threshold = 0.8
buyer_seller_interaction["suspicious"] =
buyer_seller_interaction["relationship_score"] > threshold

# Tampilkan pasangan mencurigakan
buyer_seller_interaction.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10354 entries, 0 to 10353
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   buyer_id         10354 non-null   object 
 1   seller_id        10354 non-null   object 
 2   transaction_count 10354 non-null   int64  
 3   relationship_score 10354 non-null   float64
 4   suspicious       10354 non-null   bool    
dtypes: bool(1), float64(1), int64(1), object(2)
memory usage: 333.8+ KB

# Gabungkan relationship_score dengan dataset DPTD
dt_digital_payment_transaction =
dt_digital_payment_transaction.merge(buyer_seller_interaction,

```

```

on=["buyer_id", "seller_id"], how="left")

# Tampilkan dataset yang sudah ditambahkan relationship_score
print(dt_digital_payment_transaction.head())

          dpt_id
dpt_promotion_id \
0 69e9566b3f4d6cb7db2216121b0cd0308c516e680e9c78...
NaN
1 961d6f7efde3622da8f35e76e2b53acd84c647a661de86...
NaN
2 6441defc089b4ae947cce529904a5c7db2326ede52bdaa...
NaN
3 64152dd86b2c5d1af6aaa911e7a229dc539273e0a7b8be...
NaN
4 ae4ddde99c8fe7f73fb3c2ee0e157e886b52417ece17bd...
NaN

          buyer_id \
0 bbce610a3267808752a7ec263a7ecfbe76a4987d529bcb...
1 09eb3b80abae1238ef39d50b66215e02e1ac9891ad6e8f...
2 25d0774533d69564d0deca724a55a76c693ed5f7ffa12a...
3 5b846313375cb4f4d065e50a05833dc3ac20ba3f532bbe...
4 5c19a13a9b229340b584f621b648f4dec7491e12368392...

          seller_id
transaction_amount \
0 5d2233f5a1a6435891142442fac09a77809d0c16496f07...
20380.0
1 5d2233f5a1a6435891142442fac09a77809d0c16496f07...
14673.6
2 5d2233f5a1a6435891142442fac09a77809d0c16496f07...
1012500.0
3 5b846313375cb4f4d065e50a05833dc3ac20ba3f532bbe...
30000.0
4 5c19a13a9b229340b584f621b648f4dec7491e12368392...
1000008.0

          payment_method_name payment_provider_name \
0 MITRA PEMBAYARAN DIGITAL                      BLIBLI
1 MITRA PEMBAYARAN DIGITAL                      BLIBLI
2 MITRA PEMBAYARAN DIGITAL                     TOKOPEDIA
3 MITRA PEMBAYARAN DIGITAL                      BLIBLI
4 MITRA PEMBAYARAN DIGITAL                     TOKOPEDIA

          transaction_created_datetime transaction_updated_datetime \
0 2023-08-16 09:00:53.297729    2023-08-16 10:24:56.875352
1 2023-06-09 15:22:49.867524    2023-06-09 15:23:42.718184
2 2023-10-08 10:45:24.139583    2023-10-09 11:47:23.938359
3 2023-05-26 13:41:27.133014    2023-05-26 13:47:40.595121

```

```
4 2023-05-26 17:29:34.201724 2023-05-26 17:30:04.688498
```

	transaction_count	relationship_score	suspicious
0	18	0.014218	False
1	13	0.010269	False
2	7	0.005529	False
3	11	0.008689	False
4	7	0.005529	False

Buyer-Seller Relationship Score mengukur seberapa sering pasangan pembeli dan penjual berinteraksi melalui transaksi dalam dataset. “transaction_count” menunjukkan jumlah transaksi antara pasangan *buyer* dan *seller*, yang dihitung berdasarkan pengelompokan “buyer_id” dan “seller_id”. Selanjutnya, “relationship_score” dihitung dengan membagi jumlah “transaction_count” pasangan tersebut dengan nilai transaksi maksimum di dataset, sehingga menghasilkan skor dalam rentang [0, 1]. Skor ini menggambarkan frekuensi transaksi pasangan tersebut dibandingkan dengan pasangan lainnya.

Kemudian menetapkan threshold untuk menandai pasangan yang dianggap mencurigakan jika skor mereka melebihi nilai tertentu, seperti 0,8. Kolom “suspicious” menunjukkan apakah pasangan tersebut terindikasi abnormal berdasarkan skor hubungan mereka. Dalam output, beberapa pasangan *buyer* dan *seller* memiliki "transaction_count" yang bervariasi, dengan skor rendah menunjukkan interaksi yang jarang, sementara skor lebih tinggi menunjukkan frekuensi transaksi yang lebih tinggi. Hal ini membantu pemantauan hubungan antara *buyer* dan *seller* untuk mendeteksi interaksi mencurigakan.

- **Transaction Frequency Metrics**

Kode Program 3.12. Proses pengecekan *transaction frequency metrics* pada Python.

```
dt_digital_payment_transaction['transaction_created_datetime'] =
pd.to_datetime(dt_digital_payment_transaction['transaction_created_
datetime'])

dt_digital_payment_transaction =
dt_digital_payment_transaction.sort_values(by=[ 'buyer_id',
'seller_id', 'transaction_created_datetime'])

# Buyer
```

```

dt_digital_payment_transaction['time_diff_buyer'] =
dt_digital_payment_transaction.groupby('buyer_id')['transaction_created_datetime'].diff().dt.total_seconds()

# Seller
dt_digital_payment_transaction['time_diff_seller'] =
dt_digital_payment_transaction.groupby('seller_id')['transaction_created_datetime'].diff().dt.total_seconds()

burst_threshold = 10 # dalam detik

# Flag untuk buyer
dt_digital_payment_transaction['buyer_burst_flag'] =
dt_digital_payment_transaction['time_diff_buyer'] <
burst_threshold

# Flag untuk seller
dt_digital_payment_transaction['seller_burst_flag'] =
dt_digital_payment_transaction['time_diff_seller'] <
burst_threshold

dt_digital_payment_transaction.info()

<class 'pandas.core.frame.DataFrame'>
Index: 50000 entries, 22038 to 38829
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   dpt_id          50000 non-null   object 
 1   dpt_promotion_id 937 non-null    object 
 2   buyer_id         50000 non-null   object 
 3   seller_id        50000 non-null   object 
 4   transaction_amount 50000 non-null   float64
 5   payment_method_name 50000 non-null   object 
 6   payment_provider_name 50000 non-null   object 
 7   transaction_created_datetime 50000 non-null   datetime64[ns]
 8   transaction_updated_datetime 50000 non-null   object 
 9   time_diff_buyer    40473 non-null   float64
 10  time_diff_seller   46081 non-null   float64
 11  buyer_burst_flag   50000 non-null   bool   
 12  seller_burst_flag   50000 non-null   bool  
dtypes: bool(2), datetime64[ns](1), float64(3), object(7)
memory usage: 4.7+ MB

```

Output ini menunjukkan hasil analisis frekuensi transaksi yang bertujuan untuk mengidentifikasi waktu transaksi yang tidak teratur, seperti lonjakan aktivitas atau kesenjangan waktu yang tidak biasa. Dataset diurutkan berdasarkan “buyer_id”, “seller_id”, dan “transaction_created_datetime”, sehingga perbedaan waktu antar transaksi dapat dihitung dengan lebih akurat. Kolom “time_diff_buyer” dan “time_diff_seller” menghitung selisih waktu antar transaksi berturut-turut untuk masing-masing pembeli dan penjual, dalam satuan detik. Jika selisih waktu antar transaksi lebih kecil dari nilai threshold (10 detik), maka transaksi tersebut dianggap sebagai lonjakan aktivitas yang tidak biasa.

Hasil analisis ini ditandai dengan kolom “buyer_burst_flag” dan “seller_burst_flag”, yang menunjukkan apakah terdapat lonjakan aktivitas transaksi pada pembeli atau penjual. Jika waktu antara dua transaksi berturut-turut lebih kecil dari 10 detik, flag ini akan bernilai ‘True’, menandakan adanya burst dalam aktivitas transaksi. Output ‘dt_digital_payment_transaction.info()’ juga menunjukkan bahwa terdapat beberapa nilai yang hilang (NaN) pada kolom perbedaan waktu, namun kolom burst flag tetap memberikan indikasi penting terkait lonjakan transaksi yang perlu diperiksa lebih lanjut untuk mendeteksi aktivitas yang tidak teratur atau mencurigakan.

- **Promotion Exploitation Indicator**

Kode Program 3.13. Proses pengecekan *promotion exploitation indicator* pada Python.

```
# 1. Gabungkan data transaksi dan promosi
merged_data = pd.merge(dt_digital_payment_transaction,
dt_promotion, on='dpt_promotion_id', how='inner')

# 2. Hitung frekuensi penggunaan promo oleh setiap buyer
promo_usage = merged_data.groupby(['buyer_id',
'promotion_code']).size().reset_index(name='promo_usage_count')

# 3. Identifikasi eksloitasi berdasarkan ambang batas
# Misalkan, penggunaan promo lebih dari 2 kali dianggap
eksloitasi
threshold = 2
```

```

promo_usage['is_exploitation'] = promo_usage['promo_usage_count']
> threshold

# 4. Tampilkan hasil
print(promo_usage)

merged_data.info()

dt_promotion.info()
dt_digital_payment_transaction.info()

Empty DataFrame
Columns: [buyer_id, promotion_code, promo_usage_count,
is_exploitation]
Index: []
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 0 entries
Data columns (total 20 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   dpt_id          0 non-null      object 
 1   dpt_promotion_id 0 non-null      object 
 2   buyer_id         0 non-null      object 
 3   seller_id        0 non-null      object 
 4   transaction_amount 0 non-null      float64
 5   payment_method_name 0 non-null      object 
 6   payment_provider_name 0 non-null      object 
 7   transaction_created_datetime 0 non-null      datetime64[ns] 
 8   transaction_updated_datetime 0 non-null      object 
 9   transaction_count    0 non-null      int64  
 10  relationship_score 0 non-null      float64
 11  suspicious        0 non-null      bool   
 12  time_diff_buyer   0 non-null      float64
 13  time_diff_seller  0 non-null      float64
 14  buyer_burst_flag  0 non-null      bool   
 15  seller_burst_flag 0 non-null      bool   
 16  time_diff        0 non-null      datetime64[ns] 
 17  promotion_code    0 non-null      object 
 18  promotion_name   0 non-null      object 
 19  transaction_promo_cashback_amount 0 non-null      float64
dtypes: bool(3), datetime64[ns](1), float64(5), int64(1),
object(9), timedelta64[ns](1)
memory usage: 124.0+ bytes
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 937 entries, 0 to 936
Data columns (total 4 columns):

```

```

#   Column           Non-Null Count Dtype
---  -----
0   dpt_promotion_id      937 non-null    object
1   promotion_code        645 non-null    object
2   promotion_name        645 non-null    object
3   transaction_promo_cashback_amount  937 non-null    float64
dtypes: float64(1), object(3)
memory usage: 29.4+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 17 columns):
 #   Column           Non-Null Count Dtype
---  -----
0   dpt_id            50000 non-null    object
1   dpt_promotion_id  937 non-null    object
2   buyer_id          50000 non-null    object
3   seller_id         50000 non-null    object
4   transaction_amount 50000 non-null    float64
5   payment_method_name 50000 non-null    object
6   payment_provider_name 50000 non-null    object
7   transaction_created_datetime 50000 non-null    datetime64[ns]
8   transaction_updated_datetime 50000 non-null    object
9   transaction_count    50000 non-null    int64
10  relationship_score   50000 non-null    float64
11  suspicious         50000 non-null    bool
12  time_diff_buyer    40473 non-null    float64
13  time_diff_seller   46081 non-null    float64
14  buyer_burst_flag   50000 non-null    bool
15  seller_burst_flag  50000 non-null    bool
16  time_diff          39646 non-null    timedelta64[ns]
dtypes: bool(3), datetime64[ns](1), float64(4), int64(1),
object(7), timedelta64[ns](1)
memory usage: 5.5+ MB

```

Output yang ditampilkan menunjukkan hasil analisis terhadap eksploitasi promosi dengan seringnya menggunakan kode promo yang sama. Pertama, data transaksi dan promosi digabungkan menggunakan “dpt_promotion_id”. Kemudian, frekuensi penggunaan setiap kode promo oleh masing-masing pembeli dihitung melalui “promo_usage_count”. Jika frekuensi penggunaan kode promo oleh pembeli melebihi batas ambang yang ditetapkan (seperti lebih dari 2 kali), maka transaksi tersebut dianggap sebagai eksploitasi promosi, yang ditandai dengan kolom is_exploitation.

Setelah mencoba untuk menggabungkan dataset transaksi dengan data promosi dan menghitung frekuensi penggunaan promo, terjadi kesalahan yang menyebabkan analisis tidak dapat dilakukan. Hal ini disebabkan oleh data tidak sesuai pada “promotion_id”

dalam dataset ‘promotion’ dan “dpt_promotion_id” dalam dataset ‘transaction’, yang menyebabkan penggabungan data tidak dapat dilakukan. Sehingga analisis eksplorasi promosi tidak dapat dilakukan sesuai yang diharapkan.

- **Scaling and Normalization**

Kode Program 3.14. Proses pengecekan *scaling* dan *normalization* pada Python.

```
# Menampilkan hanya kolom Primary Key dan kolom normalisasi
def tampilan_kolom_tertentu(df, primary_key, normalized_columns):
    selected_columns = [primary_key] + normalized_columns
    return df[selected_columns]

# Dataset Digital Payment Transaction
primary_key_transaction = 'dpt_id' # Ganti dengan nama Primary
Key sebenarnya
columns_normalized_transaction = ['transaction_amount',
'transaction_created_datetime']
print("Digital Payment Transaction:")
print(tampilan_kolom_tertentu(dt_digital_payment_transaction,
primary_key_transaction, columns_normalized_transaction))

# Dataset Digital Payment Request
primary_key_request = 'dpt_id' # Ganti dengan nama Primary Key
sebenarnya
columns_normalized_request = ['total_fee_amount']
print("\nDigital Payment Request:")
print(tampilan_kolom_tertentu(dt_digital_payment_request,
primary_key_request, columns_normalized_request))

# Dataset Promotion
primary_key_promotion = 'dpt_promotion_id' # Ganti dengan nama
Primary Key sebenarnya
columns_normalized_promotion =
['transaction_promo_cashback_amount']
print("\nPromotion Data:")
print(tampilan_kolom_tertentu(dt_promotion,
primary_key_promotion, columns_normalized_promotion))
```

```
# Dataset Company
primary_key_company = 'company_id' # Ganti dengan nama Primary
Key sebenarnya
columns_normalized_company = ['company_registered_datetime']
print("\nCompany Data:")
print(tampilkan_kolom_tertentu(dt_company, primary_key_company,
columns_normalized_company))

Promotion Data:
dpt_promotion_id \
78      d6d58a75dd5d629a6db1cb9f383f468502bfe507d1352a...
119     492c51e6c6a4392b2758d93961bee0a778b42151644ebc...
127     657348cf65794dac63887d289733c45a553049b7664074...
149     35f36305f75d6878d2faef1d65872b3110d2cc67308839...
184     35f36305f75d6878d2faef1d65872b3110d2cc67308839...
...
49760   492c51e6c6a4392b2758d93961bee0a778b42151644ebc...
49811   b92d45bb90f211e9c80eda0ae701c69242c9858724d7d5...
49854   b92d45bb90f211e9c80eda0ae701c69242c9858724d7d5...
49930   b8cff4ea4b0826287e934ea6569821233b650ba101564e...
49970   81aceb1882fdedde3ce44f9a5aed7578493264c7d00b86...

transaction_promo_cashback_amount
78          150000.00
119         16344.00
127         45000.00
149         14375.00
184         75816.25
...
49760       117600.00
49811       27000.00
49854       9000.00
49930       5000.00
49970       245000.00

[937 rows x 2 columns]

Company Data:
company_id \
0      bbce610a3267808752a7ec263a7ecfbe76a4987d529bcb...
1      09eb3b80abae1238ef39d50b66215e02e1ac9891ad6e8f...
2      25d0774533d69564d0deca724a55a76c693ed5f7ffa12a...
3      5b846313375cb4f4d065e50a05833dc3ac20ba3f532bbe...
4      5c19a13a9b229340b584f621b648f4dec7491e12368392...
...
49995   5edae69c82a6ef7c97288ea207d4cf9193997657c93d22...
49996   f83e98fe775227a355ded830adb7c59914f2bcfd32965c...
49997   02611e2fdd7d730bddbd654baf24f03a739704bcb34c01...
49998   a55674077f7db3b4ecb2ba1805e58c966469e00a532dfb...
```

```

49999  39144a9f257a04b32f8e8dbbf00eaf37002eee6684373b...
   company_registered_datetime
0      2023-02-01 17:18:59
1      2023-06-08 12:53:10
2      2023-03-03 00:46:29
3      2022-12-20 15:06:00
4      2021-12-07 22:39:58
...
49995      2021-12-10 13:27:41
49996      2023-05-30 16:28:48
49997      2023-01-15 21:38:22
49998      2021-10-30 17:50:55
49999      2023-06-03 12:36:15

[50000 rows x 2 columns]

from sklearn.preprocessing import MinMaxScaler

# Normalisasi dataset `dt_digital_payment_transaction`
def normalize_digital_payment_transaction(df):
    df['transaction_created_datetime'] =
    pd.to_datetime(df['transaction_created_datetime'])
    df['transaction_created_datetime'] =
    df['transaction_created_datetime'].astype(int) // 10**9 # Convert
    to timestamp

    # Kolom yang akan dinormalisasi
    columns_to_normalize = ['transaction_amount',
    'transaction_created_datetime']
    scaler = MinMaxScaler()
    df[columns_to_normalize] =
    scaler.fit_transform(df[columns_to_normalize])
    return df

# Normalisasi dataset `dt_digital_payment_request`
def normalize_digital_payment_request(df):
    # Kolom yang akan dinormalisasi
    columns_to_normalize = ['total_fee_amount']
    scaler = MinMaxScaler()
    df[columns_to_normalize] =
    scaler.fit_transform(df[columns_to_normalize])

```

```

    return df

# Normalisasi dataset `dt_promotion`
def normalize_promotion(df):
    # Kolom yang akan dinormalisasi
    columns_to_normalize = ['transaction_promo_cashback_amount']
    scaler = MinMaxScaler()
    df[columns_to_normalize] =
    scaler.fit_transform(df[columns_to_normalize])
    return df

# Normalisasi dataset `dt_company`
def normalize_company(df):
    df['company_registered_datetime'] =
    pd.to_datetime(df['company_registered_datetime'])
    df['company_registered_datetime'] =
    df['company_registered_datetime'].astype(int) // 10**9 # Convert
    to timestamp

    # Kolom yang akan dinormalisasi
    columns_to_normalize = ['company_registered_datetime']
    scaler = MinMaxScaler()
    df[columns_to_normalize] =
    scaler.fit_transform(df[columns_to_normalize])
    return df

# Apply normalisasi ke setiap dataset
dt_digital_payment_transaction =
normalize_digital_payment_transaction(dt_digital_payment_transacti
on)
dt_digital_payment_request =
normalize_digital_payment_request(dt_digital_payment_request)
dt_promotion = normalize_promotion(dt_promotion)
dt_company = normalize_company(dt_company)

# Output hasil
# Dataset Digital Payment Transaction
primary_key_transaction = 'dpt_id' # Ganti dengan nama Primary

```

```

Key sebenarnya
columns_normalized_transaction = ['transaction_amount',
'transaction_created_datetime']
print("Digital Payment Transaction:")
print(tampilkan_kolom_tertentu(dt_digital_payment_transaction,
primary_key_transaction, columns_normalized_transaction))

# Dataset Digital Payment Request
primary_key_request = 'dpt_id' # Ganti dengan nama Primary Key
sebenarnya
columns_normalized_request = ['total_fee_amount']
print("\nDigital Payment Request:")
print(tampilkan_kolom_tertentu(dt_digital_payment_request,
primary_key_request, columns_normalized_request))

# Dataset Promotion
primary_key_promotion = 'dpt_promotion_id' # Ganti dengan nama
Primary Key sebenarnya
columns_normalized_promotion =
['transaction_promo_cashback_amount']
print("\nPromotion Data:")
print(tampilkan_kolom_tertentu(dt_promotion,
primary_key_promotion, columns_normalized_promotion))

# Dataset Company
primary_key_company = 'company_id' # Ganti dengan nama Primary
Key sebenarnya
columns_normalized_company = ['company_registered_datetime']
print("\nCompany Data:")
print(tampilkan_kolom_tertentu(dt_company, primary_key_company,
columns_normalized_company))

Digital Payment Transaction:
                                         dpt_id
transaction_amount \
22038  befdc59cf11ec39defab73c7baa0bd1c16ea7e003b59af...
0.000005
20987  2bc105494f6030d5194fc6fdc50d9acede7d122dd80b3a...
0.000007
30963  d8c359cc9de4f730939cfa6c5b50bac2215cf994a30816...
0.009930

```

```

9971    769c7fdcc0e36175c9363c6d822ebb5b9c8748acb8c12e...
0.014896
21264    d808c4f7daf19127c77a79dffadf947d9a5d4fa4a9ecb7...
0.001007
...
...
9049    3220d105503b34c8a43da87b7ac5365923742d74e699ae...
0.000504
25856    0785c548ad9260f4c65b37836520e3781c9fd1cd441431...
0.000140
17692    7862efbf8e68bd315332118e2c1dfb7e825ceb06819b0...
0.001241
24571    af0c97c859e45bcc17205bc081a7cfaa73c19a31369e22...
0.002514
38829    f502986b22a0a1e19aace5bf687694ccc083cd18d81849...
0.000003

transaction_created_datetime
22038          0.994749
20987          0.996363
30963          0.797673
9971           0.998276
21264          0.652530
...
...
9049           0.951517
25856          0.956143
17692          0.711952
24571          0.755803
38829          0.999071

[50000 rows x 3 columns]

```

Digital Payment Request:

	dpt_id
total_fee_amount	
0	69e9566b3f4d6cb7db2216121b0cd0308c516e680e9c78...
0.000027	
1	961d6f7efde3622da8f35e76e2b53acd84c647a661de86...
0.000019	
2	6441defc089b4ae947cce529904a5c7db2326ede52bdaa...
0.001343	
3	64152dd86b2c5d1af6aaa911e7a229dc539273e0a7b8be...
0.000031	
4	ae4ddde99c8fe7f73fb3c2ee0e157e886b52417ece17bd...
0.000070	
...	...
...	
49995	4da123d84a819edf166f67a9cc0c197f81967aa9717ed5...
0.000070	
49996	0b358a09e15d062e1d9e0fafb8a986d802c2044b150a6e...
0.051740	
49997	ba67db08614a504cc5c9042cc61c914fa82748d526f0f3...

```
0.002204
49998 351da251d731cba588418e4a175460cc822ecbeb751751...
0.008711
49999 c676fb603d97aefc62bff989158356e210f89d2cf5cb4a...
0.031065
```

[50000 rows x 2 columns]

Promotion Data:

	dpt_promotion_id \
78	d6d58a75dd5d629a6db1cb9f383f468502bfe507d1352a...
119	492c51e6c6a4392b2758d93961bee0a778b42151644ebc...
127	657348cf65794dac63887d289733c45a553049b7664074...
149	35f36305f75d6878d2faef1d65872b3110d2cc67308839...
184	35f36305f75d6878d2faef1d65872b3110d2cc67308839...
...	...
49760	492c51e6c6a4392b2758d93961bee0a778b42151644ebc...
49811	b92d45bb90f211e9c80eda0ae701c69242c9858724d7d5...
49854	b92d45bb90f211e9c80eda0ae701c69242c9858724d7d5...
49930	b8cff4ea4b0826287e934ea6569821233b650ba101564e...
49970	81aceb1882fdedde3ce44f9a5aed7578493264c7d00b86...

	transaction_promo_cashback_amount
78	0.059829
119	0.006357
127	0.017821
149	0.005569
184	0.030150
...	...
49760	0.046867
49811	0.010620
49854	0.003419
49930	0.001819
49970	0.097836

[937 rows x 2 columns]

Company Data:

	company_id \
0	bbce610a3267808752a7ec263a7ecfbe76a4987d529bcb...
1	09eb3b80abae1238ef39d50b66215e02e1ac9891ad6e8f...
2	25d0774533d69564d0deca724a55a76c693ed5f7ffa12a...
3	5b846313375cb4f4d065e50a05833dc3ac20ba3f532bbe...
4	5c19a13a9b229340b584f621b648f4dec7491e12368392...
...	...
49995	5edae69c82a6ef7c97288ea207d4cf9193997657c93d22...
49996	f83e98fe775227a355ded830adb7c59914f2bcfd32965c...
49997	02611e2fdd7d730bddbd654baf24f03a739704bcb34c01...
49998	a55674077f7db3b4ecb2ba1805e58c966469e00a532dfb...
49999	39144a9f257a04b32f8e8dbbf00eaf37002eee6684373b...

company_registered_datetim

```

0           0.855770
1           0.902243
2           0.866511
3           0.839978
4           0.701569
...
49995       ...
49996       0.899000
49997       0.849606
49998       0.687570
49999       0.900407

[50000 rows x 2 columns]

# Simpan file CSV langsung ke Google Drive
dt_company.to_csv('/content/drive/MyDrive/Colab
Dataset/comp-v2.csv', index=False)
dt_promotion.to_csv('/content/drive/MyDrive/Colab
Dataset/promotion-v2.csv', index=False)
dt_digital_payment_request.to_csv('/content/drive/MyDrive/Colab
Dataset/request-v2.csv', index=False)
dt_digital_payment_transaction.to_csv('/content/drive/MyDrive/Cola
b Dataset/transaction-v2.csv', index=False)

from google.colab import drive
drive.mount('/content/drive')

```

Output dari normalisasi data yang dilakukan menunjukkan bahwa kolom-kolom yang berhubungan dengan jumlah transaksi dan data berbasis waktu telah berhasil dinormalisasi menggunakan metode Min-Max Scaling. Pada dataset “dt_digital_payment_transaction”, kolom “transaction_amount” dan “transaction_created_datetime” telah diubah menjadi nilai yang terstandarisasi dalam rentang 0 hingga 1, sehingga analisis lebih lanjut dapat dilakukan tanpa terganggu oleh skala yang berbeda antara nilai transaksi dan waktu transaksi. Demikian juga, kolom “total_fee_amount” pada dataset “dt_digital_payment_request” dan kolom “transaction_promo_cashback_amount” pada dataset “dt_promotion” telah dinormalisasi dengan cara serupa. Pada dataset “dt_company”, kolom “company_registered_datetime” yang berbasis waktu juga telah dinormalisasi, sehingga perbandingan antara perusahaan berdasarkan waktu pendaftarannya menjadi lebih mudah. Normalisasi data ini penting

untuk menghilangkan bias akibat perbedaan skala antar kolom yang dapat memengaruhi hasil analisis lebih lanjut.

- **CSV Hasil Modif Phyton**

Kode Program 3.15. Proses penarikan csv hasil modif pada Python.

```
import pandas as pd

dt_company = pd.read_csv('/content/drive/MyDrive/Colab
Dataset/comp-v2.csv')
dt_promotion = pd.read_csv('/content/drive/MyDrive/Colab
Dataset/promotion-v2.csv')
dt_digital_payment_request =
pd.read_csv('/content/drive/MyDrive/Colab Dataset/request-v2.csv')
dt_digital_payment_transaction =
pd.read_csv('/content/drive/MyDrive/Colab
Dataset/transaction-v2.csv')

# Menampilkan informasi masing-masing dataset
print("== dt_company.info() ==")
dt_company.info()

print("\n== dt_promotion.info() ==")
dt_promotion.info()

print("\n== dt_digital_payment_request.info() ==")
dt_digital_payment_request.info()

print("\n== dt_digital_payment_transaction.info() ==")
dt_digital_payment_transaction.info()

==== dt_company.info() ===
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   company_id      50000 non-null   object  
 1   company_kyc_status_name  50000 non-null   object  
 2   company_kyb_status_name  50000 non-null   object  
 3   company_type_group    49999 non-null   object
```

```

4   company_phone_verified_flag    50000 non-null    float64
5   company_email_verified_flag   50000 non-null    float64
6   user_fraud_flag              50000 non-null    float64
7   testing_account_flag         50000 non-null    float64
8   blacklist_account_flag       50000 non-null    float64
9   package_active_name          50000 non-null    object
10  company_registered_datetime  50000 non-null    object
11  registered_date              50000 non-null    object
12  date_only                   50000 non-null    object
dtypes: float64(5), object(8)
memory usage: 5.0+ MB

==== dt_promotion.info() ====
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 937 entries, 0 to 936
Data columns (total 4 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
0   dpt_promotion_id      937 non-null   object 
1   promotion_code         645 non-null   object 
2   promotion_name         645 non-null   object 
3   transaction_promo_cashback_amount 937 non-null   float64
dtypes: float64(1), object(3)
memory usage: 29.4+ KB

==== dt_digital_payment_request.info() ====
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
0   dpt_id             50000 non-null   object 
1   total_fee_amount    50000 non-null   float64
2   document_type_name 50000 non-null   object 
dtypes: float64(1), object(2)
memory usage: 1.1+ MB

==== dt_digital_payment_transaction.info() ====
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
0   dpt_id             50000 non-null   object 
1   dpt_promotion_id   937 non-null   object 
2   buyer_id            50000 non-null   object 
3   seller_id           50000 non-null   object 
4   transaction_amount  50000 non-null   float64
5   payment_method_name 50000 non-null   object 
6   payment_provider_name 50000 non-null   object 
7   transaction_created_datetime 50000 non-null   object 
8   transaction_updated_datetime 50000 non-null   object

```

```

 9   transaction_count           50000 non-null  int64
10  relationship_score          50000 non-null  float64
11  suspicious                  50000 non-null  bool
12  time_diff_buyer            40473 non-null  float64
13  time_diff_seller           46081 non-null  float64
14  buyer_burst_flag           50000 non-null  bool
15  seller_burst_flag          50000 non-null  bool
dtypes: bool(3), float64(4), int64(1), object(8)
memory usage: 5.1+ MB

```

Output dari ‘info()’ menunjukkan rincian tentang setiap dataset:

- dt_company: 50.000 baris dan 13 kolom, sebagian besar kolom bertipe data string dan numerik (float). Satu kolom memiliki nilai kosong (missing).
- dt_promotion: 937 baris dan 4 kolom, dengan beberapa nilai kosong di kolom promotion_code dan promotion_name.
- dt_digital_payment_request: 50.000 baris dan 3 kolom, semua kolom lengkap tanpa nilai kosong.
- dt_digital_payment_transaction: 50.000 baris dan 16 kolom, beberapa kolom memiliki nilai kosong, terutama yang berkaitan dengan waktu transaksi.

3. Exploratory Data Analysis (EDA)

- **Exploratory Data Analysis**

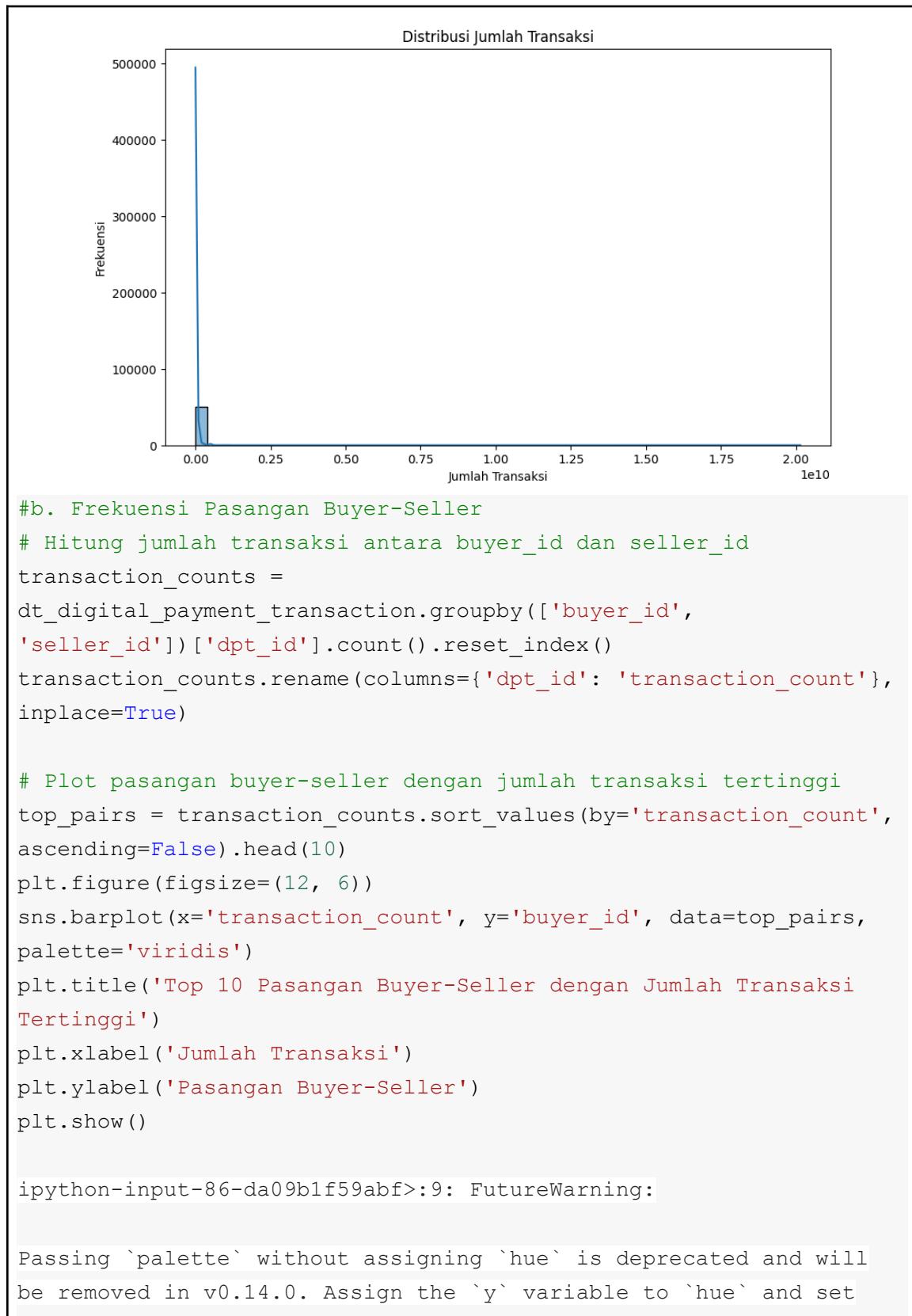
Kode Program 3.16. *Exploratory Data Analysis pada Python.*

```

import matplotlib.pyplot as plt
import seaborn as sns

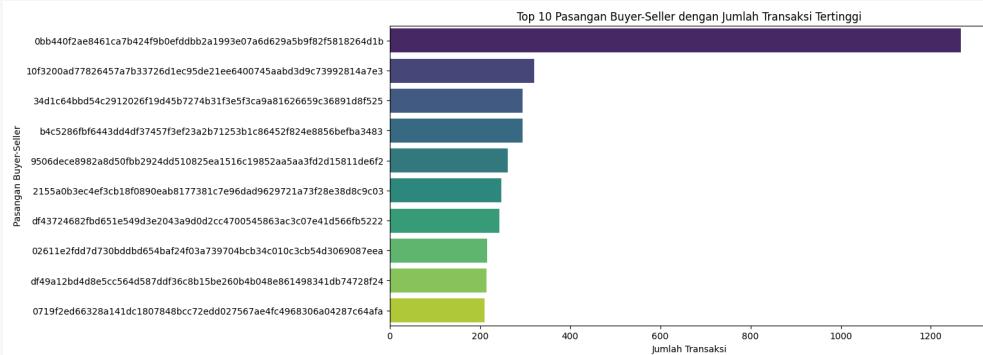
#a. Analisis Distribusi Jumlah Transaksi
# Plot distribusi jumlah transaksi
plt.figure(figsize=(10, 6))
sns.histplot(dt_digital_payment_transaction['transaction_amount'],
kde=True, bins=50)
plt.title('Distribusi Jumlah Transaksi')
plt.xlabel('Jumlah Transaksi')
plt.ylabel('Frekuensi')
plt.show()

```



```
`legend=False` for the same effect.
```

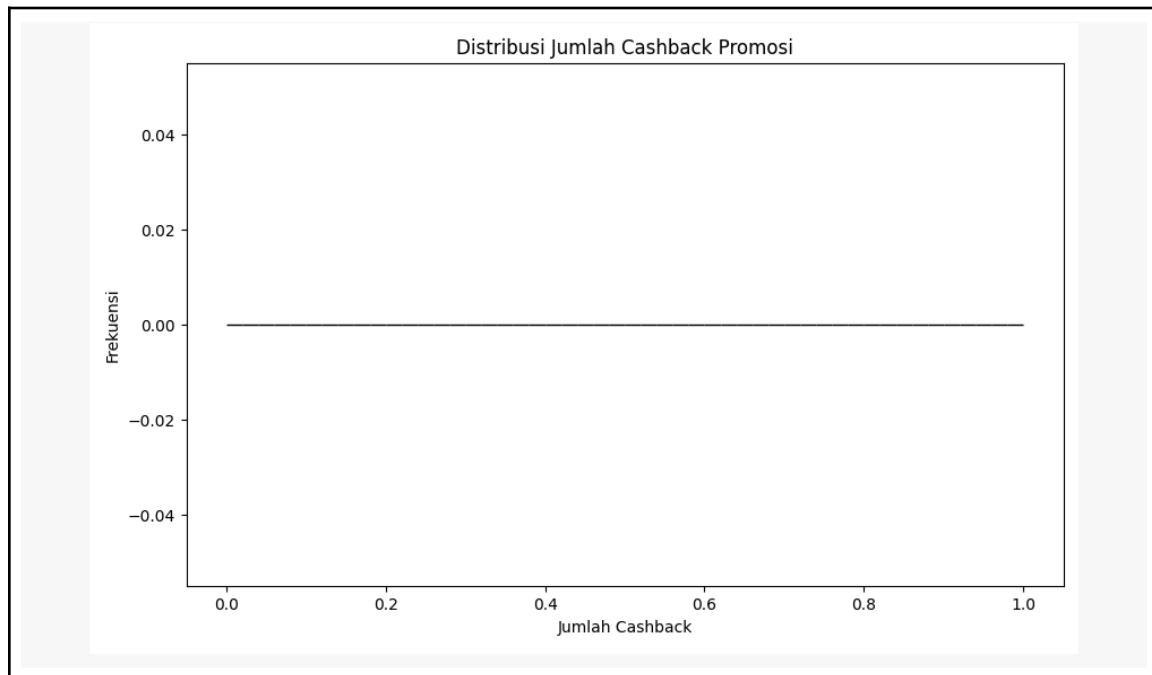
```
sns.barplot(x='transaction_count', y='buyer_id', data=top_pairs,
palette='viridis')
```



#c. Penggunaan Promosi

```
# Gabungkan data transaksi dengan data promosi
transaction_promos =
dt_digital_payment_transaction.merge(dt_promotion, how='left',
on='dpt_promotion_id')

# Plot distribusi jumlah cashback promosi
plt.figure(figsize=(10, 6))
sns.histplot(transaction_promos['transaction_promo_cashback_amount'],
kde=True, bins=50)
plt.title('Distribusi Jumlah Cashback Promosi')
plt.xlabel('Jumlah Cashback')
plt.ylabel('Frekuensi')
plt.show()
```

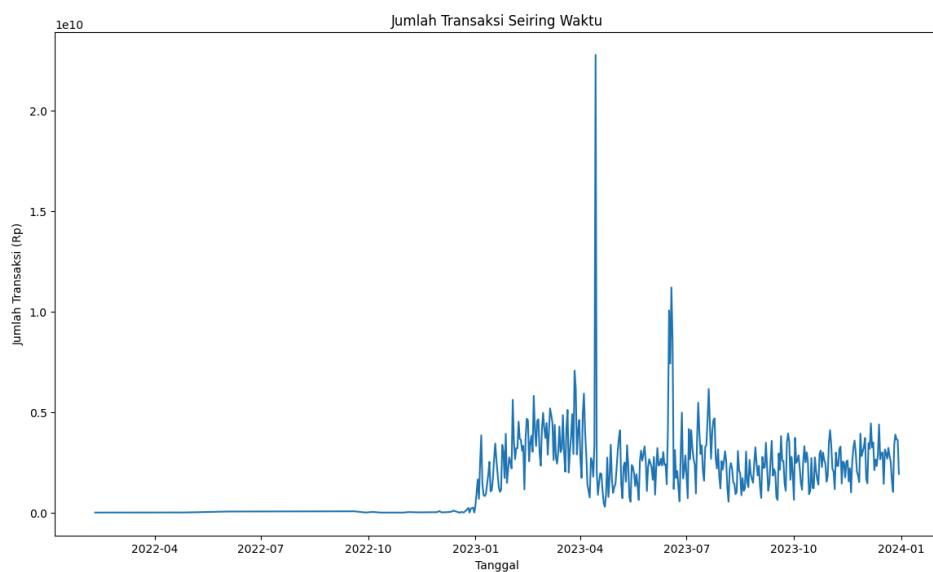


- **Visualization**

Kode Program 3.17. Visualisasi pada Python.

```
#VISUALISASI
#a. Plot Jumlah Transaksi Seiring Waktu
# Mengkonversi waktu transaksi menjadi format datetime
# dt_digital_payment_transaction['transaction_created_datetime'] =
# pd.to_datetime(dt_digital_payment_transaction['transaction_created_
# _datetime'], unit='s') # This line is causing the error. Removed
dt_digital_payment_transaction['transaction_created_datetime'] =
pd.to_datetime(dt_digital_payment_transaction['transaction_created_
_datetime']) # Changed to this

# Plot transaksi jumlah transaksi berdasarkan waktu
plt.figure(figsize=(14, 8))
dt_digital_payment_transaction.groupby(dt_digital_payment_transact
ion['transaction_created_datetime'].dt.date) ['transaction_amount']
.sum().plot()
plt.title('Jumlah Transaksi Seiring Waktu')
plt.xlabel('Tanggal')
plt.ylabel('Jumlah Transaksi (Rp)')
plt.show()
```



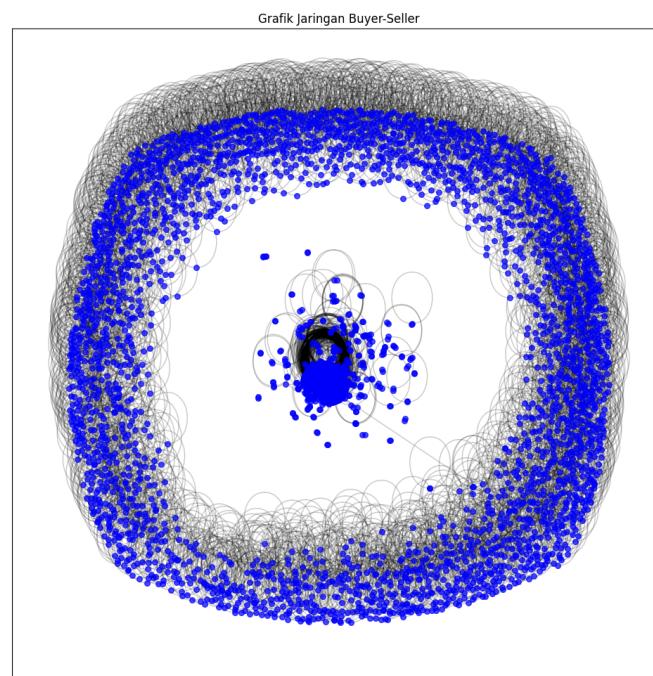
#b. Grafik Jaringan Buyer-Seller

```
import networkx as nx

# Membuat grafik hubungan buyer-seller
G = nx.Graph()

# Menambahkan node dan edge berdasarkan transaksi antara buyer dan seller
for _, row in dt_digital_payment_transaction.iterrows():
    G.add_edge(row['buyer_id'], row['seller_id'],
               weight=row['transaction_amount'])

# Visualisasi jaringan dengan node yang lebih besar untuk hubungan yang lebih kuat
plt.figure(figsize=(12, 12))
pos = nx.spring_layout(G, k=0.15, iterations=20)
nx.draw_networkx_nodes(G, pos, node_size=30, node_color='blue',
alpha=0.7)
nx.draw_networkx_edges(G, pos, alpha=0.2)
plt.title('Grafik Jaringan Buyer-Seller')
plt.show()
```

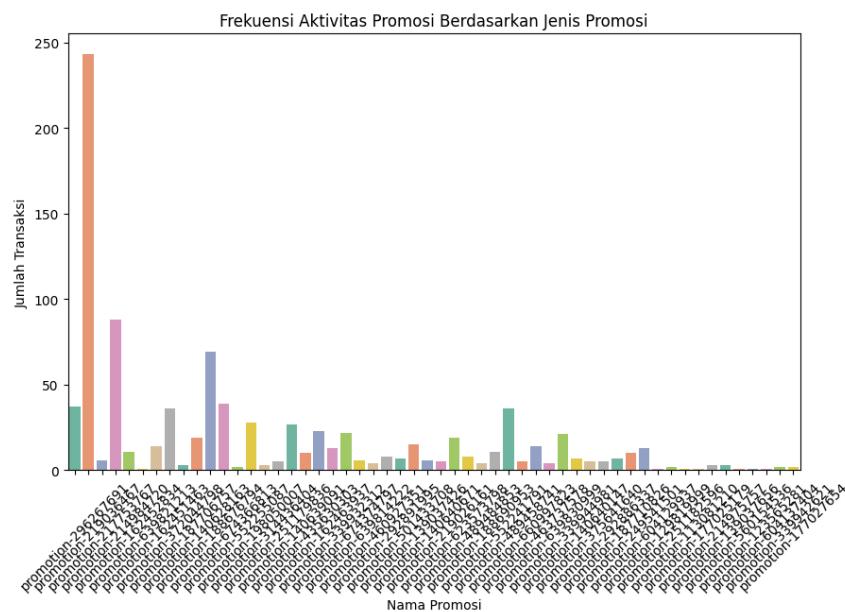


```
#c. Visualisasi Aktivitas Promosi
# Visualisasi distribusi frekuensi promosi berdasarkan jenis
promosi
plt.figure(figsize=(10, 6))
sns.countplot(x='dpt_promotion_id',
data=dt_digital_payment_transaction, palette='Set2')
plt.title('Frekuensi Aktivitas Promosi Berdasarkan Jenis Promosi')
plt.xlabel('Nama Promosi')
plt.ylabel('Jumlah Transaksi')
plt.xticks(rotation=45)
plt.show()
```

<ipython-input-94-74c93ef2b89f>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

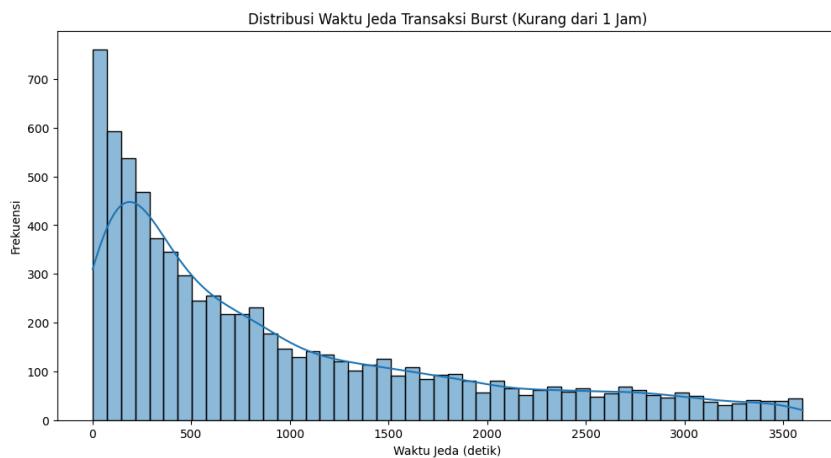
```
sns.countplot(x='dpt_promotion_id',
data=dt_digital_payment_transaction, palette='Set2')
```



```
#3. Investigasi Pola Transaksi Ulangan atau Nilai Transaksi yang Abnormal
#a. Mencari Pola Transaksi Ulangan
# Menambahkan kolom untuk menghitung selisih waktu antara transaksi berurutan
dt_digital_payment_transaction['time_diff'] =
dt_digital_payment_transaction.groupby(['buyer_id', 'seller_id'])['transaction_created_datetime'].diff()

# Filter transaksi dengan selisih waktu yang sangat kecil (burst)
burst_transactions =
dt_digital_payment_transaction[dt_digital_payment_transaction['time_diff'] < pd.Timedelta('1 hour')]

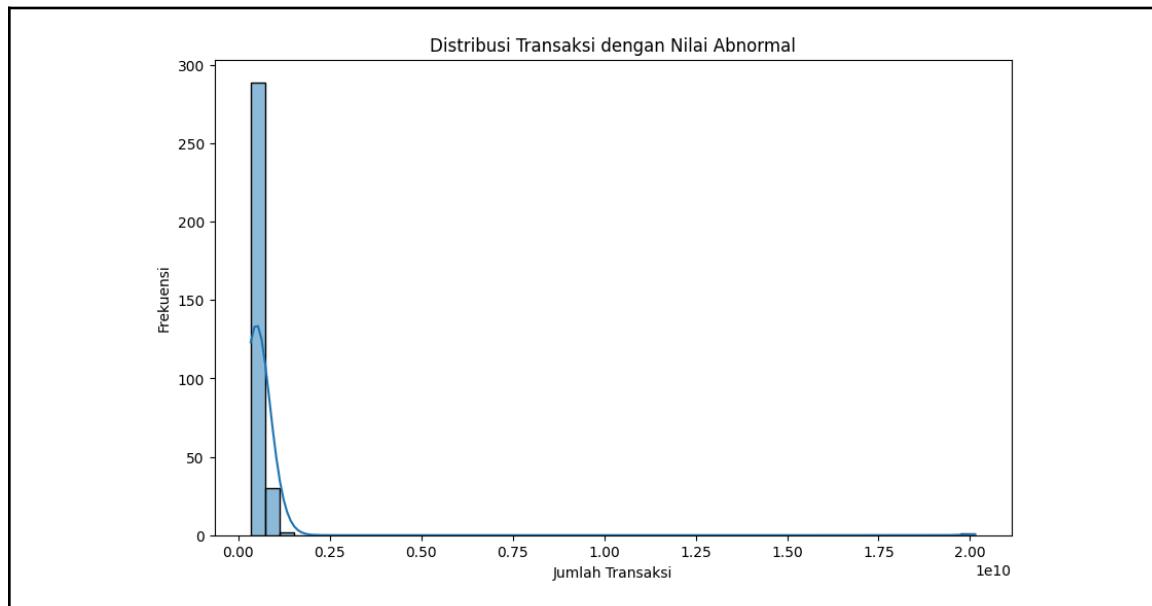
# Plot transaksi dengan selisih waktu cepat
plt.figure(figsize=(12, 6))
sns.histplot(burst_transactions['time_diff'].dt.total_seconds(), kde=True, bins=50)
plt.title('Distribusi Waktu Jeda Transaksi Burst (Kurang dari 1 Jam)')
plt.xlabel('Waktu Jeda (detik)')
plt.ylabel('Frekuensi')
plt.show()
```



```
#b. Transaksi dengan Nilai Abnormal
# Mendefinisikan batas atas untuk transaksi yang abnormal
# misalnya lebih dari 3 kali standar deviasi)
transaction_threshold =
dt_digital_payment_transaction['transaction_amount'].mean() + 3 *
dt_digital_payment_transaction['transaction_amount'].std()

# Menyaring transaksi yang lebih besar dari threshold
abnormal_transactions =
dt_digital_payment_transaction[dt_digital_payment_transaction['transaction_amount'] > transaction_threshold]

# Plot transaksi abnormal
plt.figure(figsize=(10, 6))
sns.histplot(abnormal_transactions['transaction_amount'],
kde=True, bins=50)
plt.title('Distribusi Transaksi dengan Nilai Abnormal')
plt.xlabel('Jumlah Transaksi')
plt.ylabel('Frekuensi')
plt.show()
```



4. Advanced SQL Queries for Fraud Detection

- Transaction Anomalies

```

SELECT
    fpt.dpt_id,
    fpt.buyer_id,
    fpt.seller_id,
    fpt.transaction_amount,
    ts.avg_amount,
    ts.stddev_amount,
    -- Menentukan apakah transaksi dianggap anomali berdasarkan batas 2x deviasi standar
    CASE
        WHEN fpt.transaction_amount > ts.avg_amount + 2 * ts.stddev_amount THEN 'Anomaly'
        WHEN fpt.transaction_amount < ts.avg_amount - 2 * ts.stddev_amount THEN 'Anomaly'
        ELSE 'Normal'
    END AS transaction_status
FROM transactionTable fpt
JOIN (
    -- Subquery untuk menghitung rata-rata dan deviasi standar
    SELECT
        buyer_id,
        seller_id,
        AVG(transaction_amount) AS avg_amount,
        -- Jika STDDEV tidak didukung, kita bisa tentukan deviasi standar manual atau gunakan
        nilai tetap
        0 AS stddev_amount -- Jika STDDEV tidak bisa digunakan, coba menggunakan nilai 0
        atau tentukan batas manual.
        FROM transactionTable
        GROUP BY buyer_id, seller_id
) ts
    ON fpt.buyer_id = ts.buyer_id

```

	A-Z dpt_id	A-Z buyer_id	A-Z seller_id	123 transaction_amount	123 avg_amount	123 stddev_amount	A-Z transaction_status
1	69e9566b3f4d6c	bbce610a32678087	5d2233f5a1a64358	20,380	20,380	0	Normal
2	961d6f77e362	09eb3b80a8ae1238	5d2233f5a1a64358	14,673.6	24,926.3076923077	0	Anomaly
3	6441defc089b4a	25d0774533d69564	5d2233f5a1a64358	1,012,500	308,382.7142857143	0	Anomaly
4	64152dd8692c5	5b846313375cb4f4	5b846313375cb4f4	30,000	29,828.7063181818	0	Anomaly
5	ae4ddde99e8f7	5c19a13a9b2293401	5c19a13a9b229340	1,000,008	671,431.2857142857	0	Anomaly
6	8a9e80525be12	e788560fa1c4b223	e788560fa1c4b223	832,000	22,367,250	0	Anomaly
7	78026fd3461b0	0bb440f2a8461ca	0bb440f2a8461ca	10,000	82,430.1287519747	0	Anomaly
8	e162954639d4f	b4c5286fb6443dd4	5d2233f5a1a64358	11,000	63,576.7176870748	0	Anomaly
9	43e746b799d37	1df4d59374cb4f4	5d2233f5a1a64358	608,700	994,655.1423076923	0	Anomaly
10	7faa2b826be117	83e0c08c99b411b5	83e0c08c99b411b5	14,514,500	26,124,668.51851852	0	Anomaly
11	cb333a854859	9aea2c435d5b0f38	9aea2c435d5b0f38	165,000	199,278.3333333333	0	Anomaly
12	83065c132f3442	46294b2d0672a2a5	46294b2d0672a2a5	8,000,000	19,543,625	0	Anomaly
13	3994ae53eabf3f	4e748c692460aa2d	5d2233f5a1a64358	99,999,385.7255	147,619,533.67154998	0	Anomaly
14	ba14f4e13c10b7	57c28178bddad367	57c28178bddad367	3,000,000	3,000,000	0	Normal
15	a6dcc43fe2845	ffcb909f0b66998e	5d2233f5a1a64358	847,097,323.9848001	847,097,323.9848001	0	Normal
16	f05cc8fb2af816	288cf55dd3ca1b29	288cf55dd3ca1b29	201,400	1,903,480	0	Anomaly
17	2d2378ef0fa078	ecad68cf0f865f78	5d2233f5a1a64358	15,000	25,900	0	Anomaly
18	5dace8532f7b6	58b3359073cf6007	5d2233f5a1a64358	101,550,000	63,456,250	0	Anomaly
19	531fe69a34ad0	eeb9db6e35c7813b	eeb9db6e35c7813b	4,687,400	2,043,831.25	0	Anomaly
20	476338be3ccf3e1	5eda77d9dd9c5a43	5eda77d9dd9c5a43	2,002,500	939,930	0	Anomaly
21	a201c49f0f03e22	2155a0b3e4cf3cb1	2155a0b3e4cf3cb1	10,145	12,734,5967741936	0	Anomaly
22	e59315be3c4316	95e6f56c94526951	95e6f56c94526951	812,000	947,440	0	Anomaly
23	890aa93fb0e0e0	b9580abcc3036f39	5d2233f5a1a64358	250,000,000	110,131,755.57311112	0	Anomaly
24	5dace8532f7b6	58b3359073cf6007	5d2233f5a1a64358	1,171,850	1,467,950	0	Anomaly
25	a51c996d47c289	9a2e0b0e8beb00ef	9a2e0b0e8beb00ef	33,985.75	35,537,935	0	Anomaly
26	47224d9454a983	0d7c84540ba7688	5d2233f5a1a64358	11,165,000	7,769,388	0	Anomaly
27	4de3eefed474f7	8035461b986119b1	8035461b986119b1	501,000	1,575,490	0	Anomaly
28	c46c78d22c05b4	92045433d4f39a7f9	92045433d4f39a7f9	20,310	41,693.91925	0	Anomaly

Refresh Save Cancel Export data [200] 200+ ... 200 row(s) fetched - 0.696s (0.005s fetch)

WITA en Writable Smart Insert 27 : 38 [966] Sel: 966 | 27 ...

• Buyer-Seller Relationship Analysis

```

SELECT
    t.buyer_id,
    t.seller_id,
    COUNT(t.dpt_id) AS transaction_count,
    SUM(t.transaction_amount) AS total_transaction_amount,
    AVG(t.relationship_score) AS avg_relationship_score
FROM
    transaction_v2 t
GROUP BY
    t.buyer_id, t.seller_id
HAVING
    COUNT(t.dpt_id) > (SELECT AVG(transaction_count) FROM (
        SELECT
            COUNT(dpt_id) AS transaction_count
        FROM
            transaction_v2
        GROUP BY
            buyer_id, seller_id))
    OR SUM(t.transaction_amount) > (SELECT AVG(transaction_amount) FROM (
        SELECT
            SUM(transaction_amount) AS transaction_amount
        FROM
            transaction_v2
        GROUP BY
            buyer_id, seller_id))
ORDER BY
    
```

			transaction_count DESC, total_transaction_amount DESC;			
O	A-Z buyer_id	A-Z seller_id	I23 transaction_count	I23 total_transaction_amount	I23 avg_relationship_score	
1	0bb440f2ae8451	0bb440f2ae8461	1,266	0.0051815297	1	
2	10f3200ad778264	10f3200ad778264	321	0.0007349421	0.2535545024	
3	b4c5286fbf6443d	5d2233f5a1a6435	294	0.0009280764	0.2322274882	
4	34d1c64bbd54c2	34d1c64bbd54c2	294	0.0008477807	0.2322274882	
5	9506decc8982a8c	9506decc8982a8c	261	0.0022314669	0.2061611374	
6	2155a0b3e4ef3c	2155a0b3e4ef3c	248	0.0001568105	0.195892575	
7	df43724682fb6d5	df43724682fb6d5	243	0.0021265875	0.191943128	
8	02611e2fd7d73	5d2233f5a1a6435	216	0.0004931077	0.1706161137	
9	df49a12bd4d8e5	5d2233f5a1a6435	215	0.142569682	0.1698262243	
10	0719f2ed6d328a1	5d2233f5a1a6435	210	0.0002996965	0.1658767773	
11	576ce938a1c057c	576ce938a1c057c	190	0.0002069685	0.1500798889	
12	e86bd6df55286c6	5d2233f5a1a6435	184	0.0002909699	0.1453396524	
13	12f7e7ab41e58c5	12f7e7ab41e58c5	175	0.0002209522	0.1382306477	
14	05db8bbeef844b	05db8bbeef844b	158	0.1672811748	0.1248025276	
15	a388da6750af2df	5d2233f5a1a6435	158	0.000185881	0.1248025276	
16	5bf35dd8aa92e1!	5bf35dd8aa92e1!	157	0.0373466977	0.1240126382	
17	1b8d66f4b6c47e2	1b8d66f4b6c47e2	153	0.0000770733	0.1208530806	
18	6328714f366f41ff	6328714f366f41ff	146	0.000220657	0.1153238547	
19	43497d14df19fed	43497d14df19fec	144	0.0013951717	0.1137440758	
20	558ee63bc4cb36f	558ee63bc4cb36f	142	0.1001603246	0.112164297	
21	60093b33509283	60093b33509283	141	0.000110771	0.1113744076	
22	b4c5286fbf6443d	b4c5286fbf6443d	140	0.0019141649	0.1105845182	
23	39f2037996fc6d1!	5d2233f5a1a6435	137	0.4322311094	0.1082148499	
24	10da3616325109!	10da3616325109!	134	0.0033161291	0.1058451817	
25	02611e2fd7d73	02611e2fd7d73	124	0.0099140182	0.0979462875	
26	316381f7db51ce2	316381f7db51ce2	123	0.0651068659	0.0971563981	
27	d76c958f4e0dd8c	d76c958f4e0dd8c	117	0.1198578763	0.0924170616	
28	67be2ad9dc1d49	5d2233f5a1a6435	117	0.0684192554	0.0924170616	

- Promotion Misuse Detection

```

WITH promotion_usage AS (
    SELECT
        t.buyer_id,
        t.dpt_promotion_id,
        t.transaction_created_datetime,
        LAG(t.transaction_created_datetime) OVER (PARTITION BY t.buyer_id,
        t.dpt_promotion_id ORDER BY t.transaction_created_datetime) AS previous_transaction_datetime
    FROM transaction_y2 t
    WHERE t.dpt_promotion_id IS NOT NULL
)
SELECT
    buyer_id,
    dpt_promotion_id,
    transaction_created_datetime,
    previous_transaction_datetime,
    julianday(transaction_created_datetime) - julianday(previous_transaction_datetime) AS time_diff
FROM promotion_usage
WHERE previous_transaction_datetime IS NOT NULL
AND (julianday(transaction_created_datetime) - julianday(previous_transaction_datetime)) < 1 --
Selisih waktu kurang dari 1 hari
ORDER BY buyer_id, dpt_promotion_id, transaction_created_datetime;

```

	A-Z buyer_id	A-Z dpt_promotion_id	123 transaction_created_datetime	123 previous_transaction_datetime	123 time_diff
1	00048ebf5503ef1x		0.9963629688	0.9947485809	0.0016143981
2	001046b5061e284		0.9982764597	0.7976726096	0.2006038542
3	0013cdaff46e6757		0.5370315304	0.507431756	0.0295997685
4	0013cdaff46e6757		0.5721624879	0.5370315304	0.0351309606
5	0013cdaff46e6757		0.5731193425	0.5721624879	0.0009568519
6	0013cdaff46e6757		0.5733722463	0.5731193425	0.0002529051
7	0013cdaff46e6757		0.5903288084	0.5733722463	0.0169565625
8	0013cdaff46e6757		0.5975904985	0.5903288084	0.0072616898
9	0013cdaff46e6757		0.6509885519	0.5975904985	0.0533980556
10	0013cdaff46e6757		0.7438773804	0.6509885519	0.092888831
11	0013cdaff46e6757		0.796177413	0.7438773804	0.0523000231
12	0013cdaff46e6757		0.8361361903	0.796177413	0.0399587847
13	0013cdaff46e6757		0.8490288651	0.8361361903	0.0128926736
14	0013cdaff46e6757		0.8737115774	0.8490288651	0.0246827083
15	002ead1395e772d		0.5352125532	0.5323292141	0.0028833449
16	002ead1395e772d		0.5355257242	0.5352125532	0.0003131713
17	002ead1395e772d		0.6772997704	0.5355257242	0.1417740394
18	002ead1395e772d		0.6977769759	0.6772997704	0.0204772106
19	002ead1395e772d		0.8604447928	0.6977769759	0.1626678125
20	002ead1395e772d		0.8775372834	0.8604447928	0.0170924884
21	002ead1395e772d		0.9123971393	0.8775372834	0.0348598611
22	002ead1395e772d		0.9137384951	0.9123971393	0.0013413542
23	002ead1395e772d		0.9327285789	0.9137384951	0.018990081
24	002ead1395e772d promotion-219036467		0.7528424657	0.7386367557	0.014205706
25	004a216de975ded		0.9664867208	0.8577725399	0.1087141898
26	0053525a6f5f50aa		0.60636011	0.6020729992	0.0042871181
27	0053525a6f5f50aa		0.6164628514	0.60636011	0.0101027315
28	0053525a6f5f50aa		0.6439201533	0.6164628514	0.0274573032

- Suspicious Timing

```

SELECT
    t1.buyer_id,
    t1.seller_id,
    t1.transaction_created_datetime,
    STRFTIME('%s', t1.transaction_created_datetime) - STRFTIME('%s',
t2.transaction_created_datetime) AS time_diff,
    strftime('%H', t1.transaction_created_datetime) AS transaction_hour
FROM
    transaction_v2 t1
JOIN
    transaction_v2 t2
    ON t1.buyer_id = t2.buyer_id
    AND t1.seller_id = t2.seller_id
    AND t1.transaction_created_datetime > t2.transaction_created_datetime
WHERE
    (STRFTIME('%s', t1.transaction_created_datetime) - STRFTIME('%s',
t2.transaction_created_datetime) < 300) -- 5 menit
    OR
    (strftime('%H', t1.transaction_created_datetime) < '06' OR strftime('%H',
t1.transaction_created_datetime) > '23') -- Jam malam
ORDER BY
    t1.buyer_id, t1.seller_id, t1.transaction_created_datetime;

```

	A-Z buyer_id	A-Z seller_id	123 transaction_created_datetime	123 time_diff	A-Z transaction_hour
1	00048ebf5503ef1	00048ebf5503ef1	0.9963629688	139	11
2	0013cdaff46e6757	5d2233f5a1a643	0.5370315304	2,557	00
3	0013cdaff46e6757	5d2233f5a1a643	0.5721624879	5,592	01
4	0013cdaff46e6757	5d2233f5a1a643	0.5721624879	3,035	01
5	0013cdaff46e6757	5d2233f5a1a643	0.5731193425	5,675	01
6	0013cdaff46e6757	5d2233f5a1a643	0.5731193425	3,118	01
7	0013cdaff46e6757	5d2233f5a1a643	0.5731193425	83	01
8	0013cdaff46e6757	5d2233f5a1a643	0.5733722463	5,697	01
9	0013cdaff46e6757	5d2233f5a1a643	0.5733722463	3,140	01
10	0013cdaff46e6757	5d2233f5a1a643	0.5733722463	105	01
11	0013cdaff46e6757	5d2233f5a1a643	0.5733722463	22	01
12	0013cdaff46e6757	5d2233f5a1a643	0.5903288084	7,162	02
13	0013cdaff46e6757	5d2233f5a1a643	0.5903288084	4,605	02
14	0013cdaff46e6757	5d2233f5a1a643	0.5903288084	1,570	02
15	0013cdaff46e6757	5d2233f5a1a643	0.5903288084	1,487	02
16	0013cdaff46e6757	5d2233f5a1a643	0.5903288084	1,465	02
17	0013cdaff46e6757	5d2233f5a1a643	0.5975904985	7,789	02
18	0013cdaff46e6757	5d2233f5a1a643	0.5975904985	5,232	02
19	0013cdaff46e6757	5d2233f5a1a643	0.5975904985	2,197	02
20	0013cdaff46e6757	5d2233f5a1a643	0.5975904985	2,114	02
21	0013cdaff46e6757	5d2233f5a1a643	0.5975904985	2,092	02
22	0013cdaff46e6757	5d2233f5a1a643	0.5975904985	627	02
23	0013cdaff46e6757	5d2233f5a1a643	0.6509885519	12,403	03
24	0013cdaff46e6757	5d2233f5a1a643	0.6509885519	9,846	03
25	0013cdaff46e6757	5d2233f5a1a643	0.6509885519	6,811	03
26	0013cdaff46e6757	5d2233f5a1a643	0.6509885519	6,728	03
27	0013cdaff46e6757	5d2233f5a1a643	0.6509885519	6,706	03
28	0013cdaff46e6757	5d2233f5a1a643	0.6509885519	5,241	03

- Flagged User Connections

```
WITH FraudOrBlacklistedUsers AS (
    -- Menyaring pengguna yang terflag fraud atau blacklist
    SELECT
        company_id,
        user_fraud_flag,
        blacklist_account_flag
    FROM
        comp_v2
    WHERE
        user_fraud_flag = 1 OR blacklist_account_flag = 1
)
-- Menyaring dan menghitung transaksi antara buyer dan seller yang terflag fraud atau blacklist
SELECT
    c.company_id, -- ID perusahaan yang terdeteksi fraud atau blacklist
    c.user_fraud_flag, -- Flag fraud untuk perusahaan
    c.blacklist_account_flag, -- Flag blacklist untuk perusahaan
    t.buyer_id, -- ID pembeli
    t.seller_id, -- ID penjual
    COUNT(t.dpt_id) AS transaction_count, -- Jumlah transaksi antara buyer dan seller ini
    SUM(t.transaction_amount) AS total_transaction_amount, -- Total transaksi antara buyer
    dan seller ini
```

```

t.suspicious, -- Kolom suspicious yang sudah ada
t.relationship_score -- Skor hubungan antara buyer dan seller
FROM
    transaction_v2 t
JOIN
    FraudOrBlacklistedUsers c ON (t.buyer_id = c.company_id OR t.seller_id = c.company_id)
-- Hanya transaksi dengan pengguna fraud atau blacklist
GROUP BY
    t.buyer_id, t.seller_id, c.company_id -- Mengelompokkan berdasarkan interaksi antara
buyer dan seller
ORDER BY
    transaction_count DESC, total_transaction_amount DESC; -- Urutkan berdasarkan
jumlah transaksi dan total nilai transaksi
  
```

company_id	user_fraud_flag	blacklist_account_flag	buyer_id	seller_id	transaction_count	total_transaction_amount	suspicious	relationship
0 e86bdddff55286dc8b120958807c7307bc5d2b28c5c227...	1.0	0.0	e86bdddff55286dc8b120958807c7307bc5d2b28c5c227...	5d2233f5a1e6435891142442fec0977890d01649507...	33856	5.3538845e-02	0	0.14
1 05dbbbee84409e5e35996d8caff234c778945e827d4...	1.0	0.0	05dbbbee84409e5e35996d8caff234c778945e827d4...	5d2233f5a1e6435891142442fec09778945e827d4...	24964	2.643049e+01	0	0.12
2 e383dded750a7f2ffcc1a491040e093956d0835ee20943...	1.0	0.0	e383dded750a7f2ffcc1a491040e093956d0835ee20943...	5d2233f5a1e6435891142442fec0977890d01649507...	24964	2.936919e-02	0	0.12
3 d3497d14cf19fed4b4ff6fbce21265ef7926185e0d...	1.0	0.0	d3497d14cf19fed4b4ff6fbce21265ef7926185e0d...	43497d14cf19fed4b4ff6fbce21265ef7926185e0d...	21024	2.036951e-01	0	0.11
4 d76c958f4e0dd86fb00166da4c5d1d132a72144212008...	1.0	0.0	d76c958f4e0dd86fb00166da4c5d1d132a72144212008...	d76c958f4e0dd86fb00166da4c5d1d132a72144212008...	13689	1.402337e-01	0	0.09
...
809 58547af6c50d50a6a1fc8a2a59d9e1dbd04f1f991dc...	1.0	0.0	58547af6c50d50a6a1fc8a2a59d9e1dbd04f1f991dc...	58547af6c50d50a6a1fc8a2a59d9e1dbd04f1f991dc...	1	4.965218e-05	0	0.00
810 450000050e402e2b350302e34376d3b2127e0a3b01e9b1...	1.0	0.0	450000050e402e2b350302e34376d3b2127e0a3b01e9b1...	5d2233f5a1e6435891142442fec0977890d01649507...	1	2.630970e-06	0	0.00
811 f88094debfec4634d90034352c52c71a7482e292e5d981...	1.0	0.0	f88094debfec4634d90034352c52c71a7482e292e5d981...	5d2233f5a1e6435891142442fec0977890d01649507...	1	1.740488e-05	0	0.00
812 1c9ea6457229e9ff17fb0b8a8d735a934308d8759e3...	1.0	0.0	1c9ea6457229e9ff17fb0b8a8d735a934308d8759e3...	1c9ea6457229e9ff17fb0b8a8d735a934308d8759e3...	1	1.117174e-05	0	0.00
813 9086363d5aa7e8ff4479bc30d732914ee5d50b85f4c4...	0.0	1.0	9086363d5aa7e8ff4479bc30d732914ee5d50b85f4c4...	5d2233f5a1e6435891142442fec0977890d01649507...	1	5.461740e-07	0	0.00

814 rows x 9 columns

● SQL Joins for User-Company Fraud Insights

```

SELECT
    t.dpt_id,
    t.buyer_id,
    t.seller_id,
    t.transaction_amount,
    t.transaction_created_datetime,
    t.relationship_score,
    t.suspicious AS transaction_suspicious,
    c.company_id,
    c.company_kyc_status_name,
    c.company_kyb_status_name,
    c.user_fraud_flag,
    c.package_active_name,
    c.company_phone_verified_flag,
    c.company_email_verified_flag
FROM
    transaction_v2 t
JOIN
    comp_v2 c ON t.seller_id = c.company_id OR t.buyer_id = c.company_id -- Memeriksa
buyer_id dan seller_id
WHERE
    t.suspicious = "True" -- Memfilter transaksi yang dicurigai
ORDER BY
    t.transaction_created_datetime DESC;
  
```

⑤	A-Z dpt_id	A-z buyer_id	A-Z seller_id	123 transac	123 transaction	123 ri	123 transat	A-Z company	A-Z company_	A-Z comp	123	A-Z packa	123 con	123
1	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	
2	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	
3	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	
4	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	
5	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	
6	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	
7	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	
8	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	
9	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	
10	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	
11	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	
12	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	
13	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	
14	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	
15	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	
16	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	
17	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	
18	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	
19	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	
20	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	
21	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	
22	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	
23	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	
24	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	
25	a5f4a5f491bac	0bb440f2ae8461ca	0bb440f2ae8461ci	0.0000004965	0.9968242224	1	True	0bb440f2ae8461c	VALIDASI_BERHASII	BELUM_VALIDI	0	PAPER+ MARK	1	

5. SQL Views

- Top Fraudulent Buyer-Seller Pairs

```
CREATE VIEW top_fraudulent_pairs AS
SELECT
    t.buyer_id,
    t.seller_id,
    COUNT(t.dpt_id) AS transaction_frequency,
    SUM(t.transaction_amount) AS total_transaction_amount
FROM
    transaction_v2 t
WHERE
    t.suspicious = "True"
GROUP BY
    t.buyer_id, t.seller_id
ORDER BY
```

- Flagged Users and Their Transactions

```
CREATE VIEW flagged_users_transactions AS
SELECT
    t.dpt_id AS transaction_id,
    t.buyer_id,
    t.seller_id,
    t.transaction_amount,
    t.payment_method_name,
    t.payment_provider_name,
    t.transaction_created_datetime,
    c.company_id AS user_id,
    c.company_kyc_status_name AS kyc_status,
    c.company_kyb_status_name AS kyb_status,
    c.blacklist_account_flag AS is_blacklisted,
```

```

    c.user_fraud_flag AS is_flagged
FROM
    transaction_v2 t
JOIN
    comp_v2 c
ON
    t.buyer_id = c.company_id OR t.seller_id = c.company_id
WHERE
    c.blacklist_account_flag = 1 OR c.user_fraud_flag = 1;

```

- **Stored Procedures**

Monthly Fraud Report Procedure

```

CREATE VIEW Monthly_Fraud_Report AS
SELECT
    -- Total Fraud Amount
    (SELECT SUM(transaction_amount)
     FROM transaction_v2
     WHERE suspicious = "True"
     AND strftime('%Y-%m', transaction_created_datetime) = strftime('%Y-%m',
CURRENT_DATE)) AS total_fraud_amount,

    -- Flagged Users
    (SELECT GROUP_CONCAT(DISTINCT buyer_id)
     FROM transaction_v2
     WHERE suspicious = "True"
     AND strftime('%Y-%m', transaction_created_datetime) = strftime('%Y-%m',
CURRENT_DATE)) AS flagged_buyers,

    (SELECT GROUP_CONCAT(DISTINCT seller_id)
     FROM transaction_v2
     WHERE suspicious = "True"
     AND strftime('%Y-%m', transaction_created_datetime) = strftime('%Y-%m',
CURRENT_DATE)) AS flagged_sellers,

    -- Suspicious Buyer-Seller Pairs
    buyer_id,
    seller_id,
    relationship_score
FROM transaction_v2
WHERE suspicious = "True"
AND strftime('%Y-%m', transaction_created_datetime) = strftime('%Y-%m',
CURRENT_DATE);

```

- Automated Promotion Misuse Detection

```

log_id INTEGER PRIMARY KEY AUTOINCREMENT,
promotion_code TEXT,
buyer_id TEXT,
seller_id TEXT,
transaction_amount REAL,
timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,
misuse_reason TEXT
);

CREATE TRIGGER detect_promotion_misuse
AFTER INSERT ON transaction_v2
FOR EACH ROW
BEGIN
    -- Misuse Check 1: Cashback amount mismatch
    INSERT INTO promotion_misuse_log (promotion_code, buyer_id, seller_id,
transaction_amount, misuse_reason)
    SELECT
        p.promotion_code,
        NEW.buyer_id,
        NEW.seller_id,
        NEW.transaction_amount,
        'Cashback amount mismatch'
    FROM promotion_v2 p
    WHERE NEW.dpt_promotion_id = p.dpt_promotion_id
    AND NEW.transaction_amount < p.transaction_promo_cashback_amount;

    -- Misuse Check 2: Excessive use of a promotion (example: more than 3 times per day)
    INSERT INTO promotion_misuse_log (promotion_code, buyer_id, seller_id,
transaction_amount, misuse_reason)
    SELECT
        p.promotion_code,
        NEW.buyer_id,
        NEW.seller_id,
        NEW.transaction_amount,
        'Excessive use of promotion'
    FROM promotion_v2 p
    WHERE NEW.dpt_promotion_id = p.dpt_promotion_id
    AND (SELECT COUNT(*)
    FROM transaction_v2 t
    WHERE t.dpt_promotion_id = NEW.dpt_promotion_id
    AND t.buyer_id = NEW.buyer_id
    AND DATE(t.transaction_created_datetime) =
    DATE(NEW.transaction_created_datetime)) > 3;

    -- Misuse Check 3: Invalid promotion code
    INSERT INTO promotion_misuse_log (promotion_code, buyer_id, seller_id,
transaction_amount, misuse_reason)
    SELECT
        'Invalid',
        NEW.buyer_id,
        NEW.seller_id,
        NEW.transaction_amount,
        'Invalid promotion code'
    FROM transaction_v2
    WHERE dpt_promotion_id NOT IN (SELECT dpt_promotion_id
    FROM promotion_v2);

```

'Invalid promotion code'
 WHERE NEW.dpt_promotion_id IS NULL OR NEW.dpt_promotion_id = ";

6. Social Network Analysis

- Identify clusters of buyers and sellers that frequently interact, especially those involved in flagged or blacklisted transactions

```

import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
import numpy as np

# Membaca data transaksi
df = dt_digital_payment_transaction

# Filter transaksi yang mencurigakan atau blacklist
suspicious_data = df[(df['suspicious'] == True) |
                      (df['buyer_id'].isin(df[df['buyer_burst_flag'] == True]['buyer_id'])) |
                      (df['seller_id'].isin(df[df['seller_burst_flag'] == True]['seller_id']))]

# Membuat Graph
G = nx.Graph()

# Menambahkan node dan edge (hubungan buyer-seller)
for idx, row in suspicious_data.iterrows():
    G.add_edge(row['buyer_id'], row['seller_id'])

# Ambil semua node yang terlibat dalam transaksi
nodes = list(G.nodes())

# Buat fitur untuk DBSCAN: menghitung interaksi antara node
interaction_matrix = np.zeros((len(nodes), len(nodes)))

```

```

# Menyusun matriks interaksi
node_indices = {node: idx for idx, node in enumerate(nodes)}

# Mengisi matriks interaksi
for edge in G.edges():
    i = node_indices[edge[0]]
    j = node_indices[edge[1]]
    interaction_matrix[i, j] = 1
    interaction_matrix[j, i] = 1

# Menggunakan DBSCAN untuk clustering
db = DBSCAN(eps=0.5, min_samples=5,
metric='euclidean').fit(interaction_matrix)

# Menambahkan informasi cluster ke node
node_clusters = pd.Series(db.labels_, index=nodes)

# Menghilangkan node dengan label -1 (noise) dan menyaring hasil
node_clusters_filtered = node_clusters[node_clusters != -1]

# Menambahkan label cluster ke DataFrame transaksi
suspicious_data['cluster'] =
suspicious_data['buyer_id'].map(node_clusters)

# Menampilkan hasil cluster yang berisi buyer_id, seller_id, dan
# cluster label
cluster_result = suspicious_data[['buyer_id', 'seller_id',
'cluster']]
print(cluster_result)

#### VISUALISASI

# Menyaring node dengan ranking tertinggi (misalnya berdasarkan
# relationship_score)
top_n = 20 # Ambil top 20 node dengan relationship_score
tertinggi
top_nodes =

```

```
suspicious_data.groupby('buyer_id')['relationship_score'].sum().nl
argest(top_n).index

# Membuat subgraph hanya untuk top nodes
G_top = G.subgraph(top_nodes)

# Visualisasi hasil clustering untuk top nodes
plt.figure(figsize=(12, 12))
pos = nx.spring_layout(G_top)
node_colors = [node_clusters.get(node, -1) for node in
G_top.nodes()]

# Menggambar nodes dengan warna berdasarkan cluster
nx.draw_networkx_nodes(G_top, pos, node_size=50,
node_color=node_colors, cmap=plt.cm.jet)

# Menggambar edges antara nodes
nx.draw_networkx_edges(G_top, pos, alpha=0.7)

# Menambahkan label untuk setiap node (buyer_id atau seller_id)
node_labels = {node: node for node in G_top.nodes()}
nx.draw_networkx_labels(G_top, pos, labels=node_labels,
font_size=8, font_color="black")

# Menambahkan judul ke visualisasi
plt.title(f"Top {top_n} Buyer-Seller Network - Clusters")
plt.axis('off') # Matikan axis
plt.show()

# Menampilkan hasil cluster yang telah disaring
print(node_clusters_filtered)
```

<ipython-input-30-f940c643b638>:48: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:

https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view

-versus-a-copy

```
suspicious_data['cluster'] = suspicious_data['buyer_id'].map(node_clusters)
                                buyer_id \
```

```
30963 001046b5061e28476b83fe2335b04d3210bed72a2fee17...
```

```
9971 001046b5061e28476b83fe2335b04d3210bed72a2fee17...
```

```
21264 00119737eef11ff1d30c2061dd1e19c06d963d5a125c92...
```

```
14723 0012614e5a1366f102a3497b67f8ec9a8009c802aa6959...
```

```
28195 0013cdaff46e67574660e0ddd214e5032e3ff5d94744e1...
```

```
... ...
```

```
9049 fff5f7cf3deff1c2acad4533c4b845f8f2c0c05168f668...
```

```
25856 fff77b856ac2478f8911b9cddf65980ac7fdc4d700e354...
```

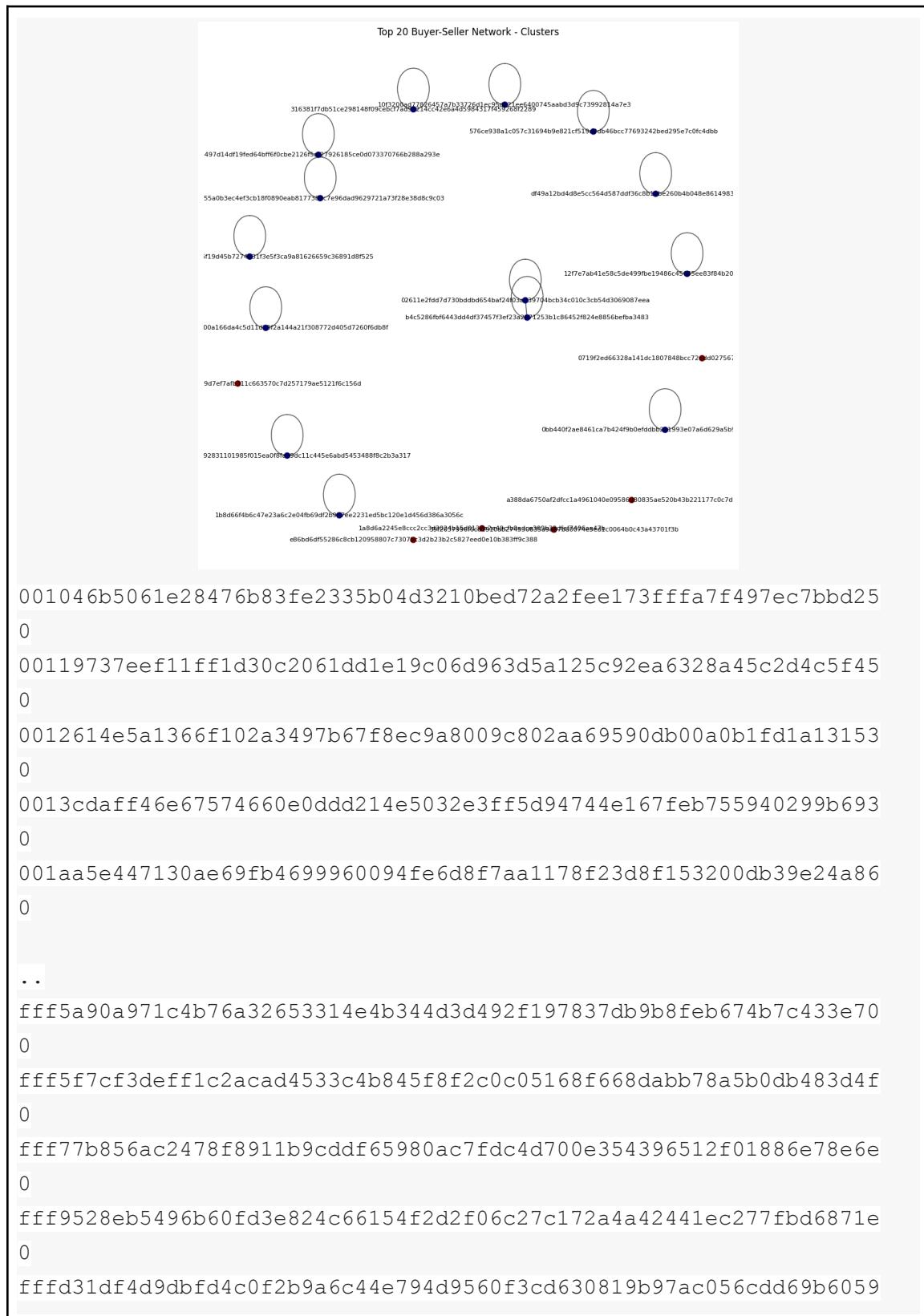
```
17692 fff9528eb5496b60fd3e824c66154f2d2f06c27c172a4a...
```

```
24571 fff9528eb5496b60fd3e824c66154f2d2f06c27c172a4a...
```

```
38829 fffd31df4d9dbfd4c0f2b9a6c44e794d9560f3cd630819...
```

seller_id	cluster	
30963	5d2233f5a1a6435891142442fac09a77809d0c16496f07...	0
9971	5d2233f5a1a6435891142442fac09a77809d0c16496f07...	0
21264	5d2233f5a1a6435891142442fac09a77809d0c16496f07...	0
14723	5d2233f5a1a6435891142442fac09a77809d0c16496f07...	0
28195	5d2233f5a1a6435891142442fac09a77809d0c16496f07...	0
...
9049	5d2233f5a1a6435891142442fac09a77809d0c16496f07...	0
25856	5d2233f5a1a6435891142442fac09a77809d0c16496f07...	0
17692	5d2233f5a1a6435891142442fac09a77809d0c16496f07...	0
24571	5d2233f5a1a6435891142442fac09a77809d0c16496f07...	0
38829	5d2233f5a1a6435891142442fac09a77809d0c16496f07...	0

[33670 rows x 3 columns]



```
0
Length: 5868, dtype: int64
```

- **Find central users (e.g., buyers or sellers) who are highly connected and may be involved in fraudulent networks.**

```
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt

# Membaca dataset transaksi (dt_digital_payment_transaction)
df = dt_digital_payment_transaction # Jika menggunakan variabel
dt_digital_payment_transaction yang sudah ada

# Membuat graf hubungan antara buyer dan seller
G = nx.Graph()

# Menambahkan edges untuk setiap transaksi antara buyer dan seller
for _, row in df.iterrows():
    buyer = row['buyer_id']
    seller = row['seller_id']

    # Menambahkan edge antara buyer dan seller, dengan bobot
    berdasarkan transaction_amount atau relationship_score
    G.add_edge(buyer, seller, weight=row['relationship_score'])

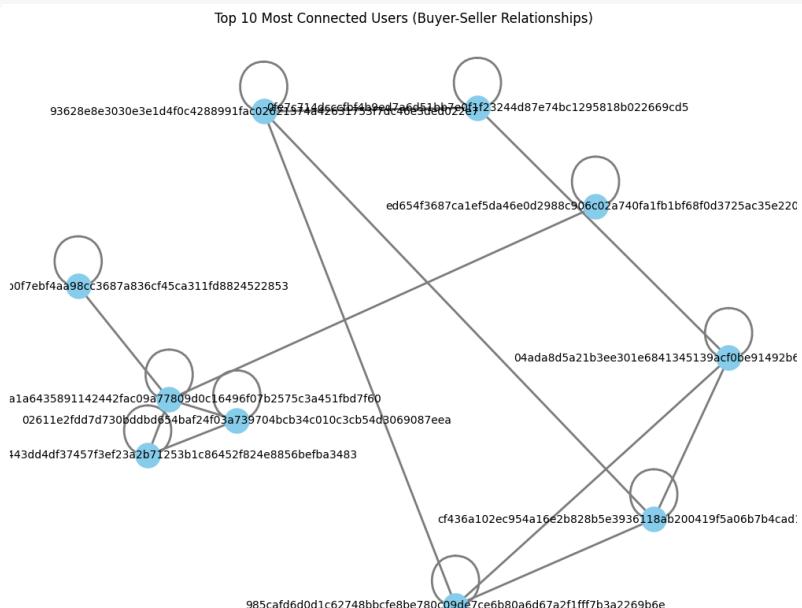
# Mencari central users dengan Degree Centrality
degree_centrality = nx.degree_centrality(G)

# Mengurutkan dan mengambil 10 pengguna dengan degree centrality tertinggi
top_users = sorted(degree_centrality.items(), key=lambda x: x[1],
reverse=True)[:10]
top_users_ids = [user[0] for user in top_users] # Hanya mengambil
ID pengguna dengan centrality tertinggi

# Membuat subgraf yang hanya mencakup pengguna top 10
```

```
subgraph = G.subgraph(top_users_ids)

# Visualisasi subgraf (hanya top 10 pengguna)
plt.figure(figsize=(10, 8))
pos = nx.spring_layout(subgraph) # Layout untuk menggambar
jaringan
nx.draw(subgraph, pos, with_labels=True, node_size=500,
font_size=10, node_color="skyblue", edge_color="gray", width=2)
plt.title("Top 10 Most Connected Users (Buyer-Seller
Relationships)")
plt.show()
```



- **Visualizing User Networks:**

```
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt

# Load data transaksi (pastikan DataFrame
'dt_digital_payment_transaction' sudah ada)
# dt_digital_payment_transaction =
```

```

pd.read_csv('path_to_transaction_data.csv')

# -----
# Membuat Graph Network Buyer-Seller
# -----


G = nx.Graph()

# Menambahkan edges berdasarkan data transaksi
for index, row in dt_digital_payment_transaction.iterrows():
    buyer = row['buyer_id']
    seller = row['seller_id']
    suspicious = row['suspicious']

    # Menambahkan edge antara buyer dan seller
    G.add_edge(buyer, seller, suspicious=suspicious)

# -----
# Menyaring Node dengan Transaksi Terbanyak atau Paling Sedikit
# -----


# Hitung jumlah transaksi antara setiap buyer-seller
transaction_counts = pd.Series([f'{u}-{v}' for u, v in G.edges()])
transaction_count = transaction_counts.value_counts()

# Ambil 10 hubungan terbanyak
top_10_edges = transaction_count.head(10).index.tolist()

# Ambil 10 hubungan paling sedikit (jika ada lebih dari 10 edge)
bottom_10_edges = transaction_count.tail(10).index.tolist()

# -----
# Menyaring pasangan buyer-seller dari string ke tuple
# -----


# Mengubah 'buyer-seller' menjadi tuple (buyer, seller)
top_10_edges_tuples = [(edge.split('-')[0], edge.split('-')[1]) for edge in top_10_edges]

```

```

bottom_10_edges_tuples = [(edge.split('-')[0], edge.split('-')[1])
for edge in bottom_10_edges]

# -----
# Visualisasi Jaringan Top 10 Terbanyak
# -----

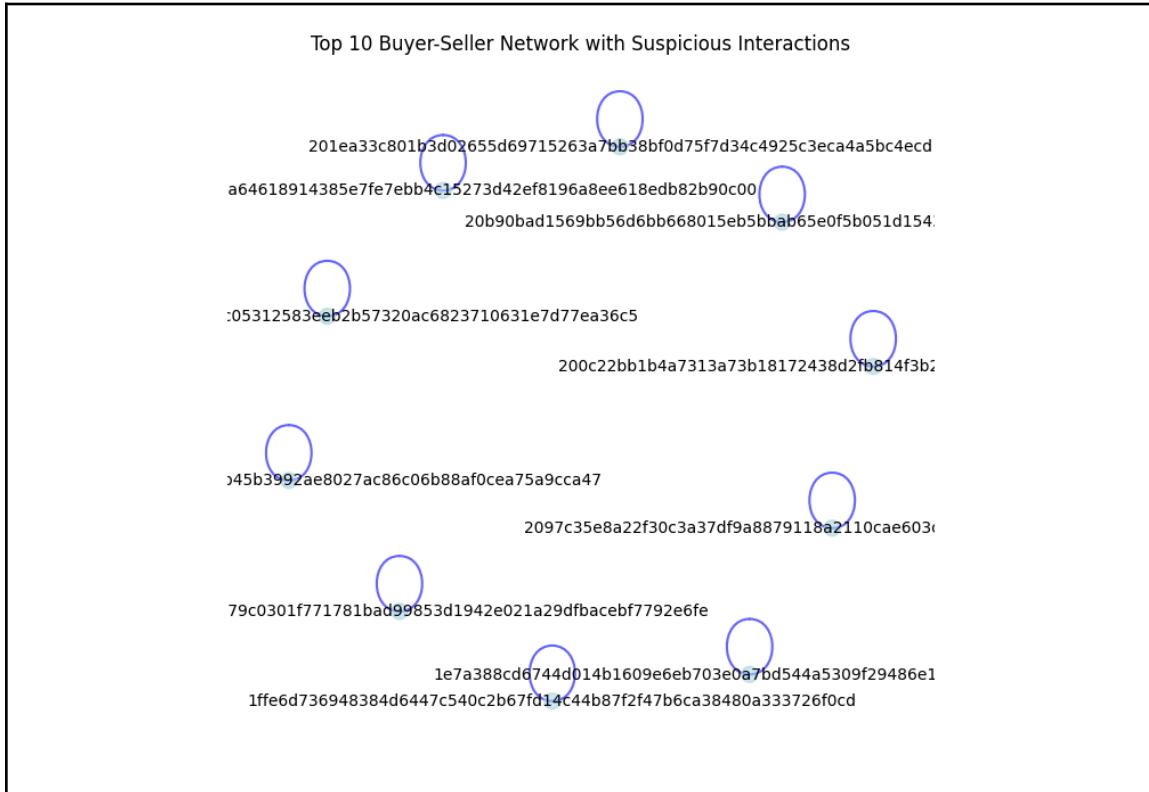

# Filter hanya edge top 10 terbanyak
top_10_graph = G.edge_subgraph(bottom_10_edges_tuples).copy()

# Visualisasi jaringan top 10 terbanyak
plt.figure(figsize=(8, 8))
pos = nx.spring_layout(top_10_graph) # Penataan posisi node menggunakan spring layout
edge_colors = ['red' if top_10_graph[u][v]['suspicious'] else
'blue' for u, v in top_10_graph.edges()]
node_sizes = [len(list(top_10_graph.neighbors(node))) * 100 for
node in top_10_graph.nodes()]

nx.draw_networkx_nodes(top_10_graph, pos, node_size=node_sizes,
node_color='lightblue', alpha=0.6)
nx.draw_networkx_edges(top_10_graph, pos, edge_color=edge_colors,
width=1.5, alpha=0.6)
nx.draw_networkx_labels(top_10_graph, pos, font_size=10,
font_color='black')

plt.title('Top 10 Buyer-Seller Network with Suspicious
Interactions')
plt.axis('off')
plt.show()

```



- **Cohort Analysis**

Group buyers by their first transaction date and measure their continued activity over time

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

# 1. Load Data
# Gantilah dt_digital_payment_transaction dengan data yang ada di Colab
df_transaction = dt_digital_payment_transaction.copy()

# 2. Convert datetime columns to proper datetime format
df_transaction['transaction_created_datetime'] =
pd.to_datetime(df_transaction['transaction_created_datetime'],
```

```

unit='s')

# 3. Determine the first transaction date for each buyer
df_first_transaction =
df_transaction.groupby('buyer_id')['transaction_created_datetime']
.min().reset_index()
df_first_transaction.rename(columns={'transaction_created_datetime':
': 'first_transaction_date'}, inplace=True)

# 4. Merge first transaction date with the original dataset
df_transaction = pd.merge(df_transaction, df_first_transaction,
on='buyer_id', how='left')

# 5. Create a cohort group for each buyer based on the first
transaction month
df_transaction['cohort_month'] =
df_transaction['first_transaction_date'].dt.to_period('M')

# 6. Calculate the difference in months between each transaction
and the first transaction
df_transaction['transaction_month_diff'] =
((df_transaction['transaction_created_datetime'].dt.year -
df_transaction['first_transaction_date'].dt.year) * 12) +
(df_transaction['transaction_created_datetime'].dt.month -
df_transaction['first_transaction_date'].dt.month)

# 7. Calculate the number of transactions per cohort group and
month difference
cohort_data = df_transaction.groupby(['cohort_month',
'transaction_month_diff'])['buyer_id'].nunique().reset_index()

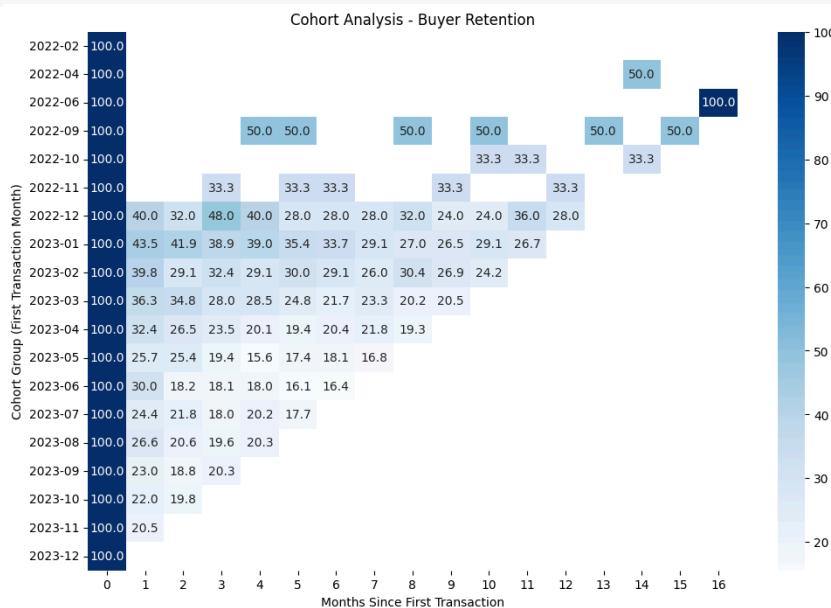
# 8. Pivot the data to create a cohort retention table
cohort_pivot = cohort_data.pivot_table(index='cohort_month',
columns='transaction_month_diff', values='buyer_id',
aggfunc='sum')

# 9. Calculate retention rates (the proportion of repeat
transactions over time)

```

```
cohort_retention = cohort_pivot.divide(cohort_pivot.iloc[:, 0], axis=0) * 100

# 10. Plot the cohort retention table as a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(cohort_retention, annot=True, fmt='.1f', cmap='Blues',
cbar=True)
plt.title('Cohort Analysis - Buyer Retention')
plt.xlabel('Months Since First Transaction')
plt.ylabel('Cohort Group (First Transaction Month)')
plt.show()
```



- Identify whether certain buyers engage in fraudulent behavior after a period of inactivity or repeatedly interact with the same sellers.

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

# Load dataset
```

```

dt_transaction = dt_digital_payment_transaction

# Pastikan kolom 'transaction_created_datetime' dalam format
# datetime
dt_transaction['transaction_created_datetime'] =
pd.to_datetime(dt_transaction['transaction_created_datetime'],
unit='s')

# Tambahkan kolom 'cohort_month' (bulan pertama transaksi) dan
# 'transaction_month' (bulan transaksi)
dt_transaction['cohort_month'] =
dt_transaction.groupby('buyer_id')['transaction_created_datetime']
.transform('min').dt.to_period('M')
dt_transaction['transaction_month'] =
dt_transaction['transaction_created_datetime'].dt.to_period('M')

# Menghitung jumlah transaksi per bulan dan cohort
cohort_data = dt_transaction.groupby(['cohort_month',
'transaction_month']).agg(n_customers=('buyer_id',
'nunique')).reset_index()

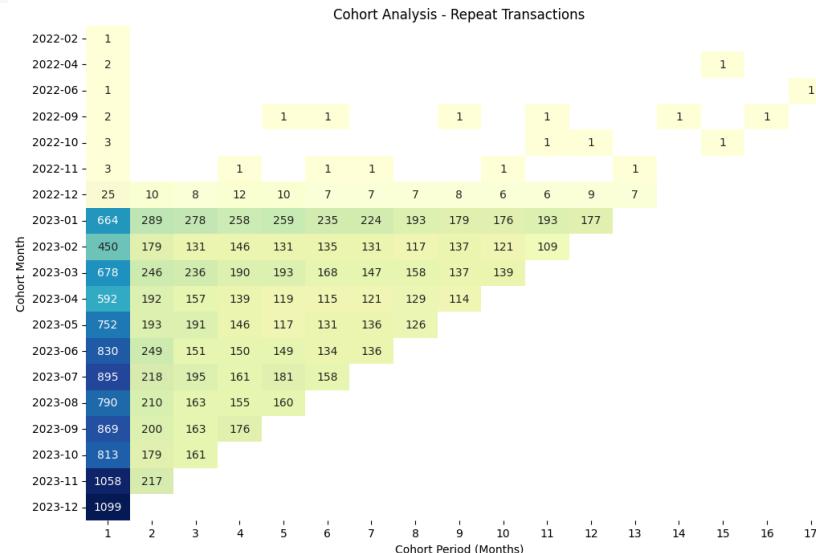
# Hitung selisih bulan antara 'cohort_month' dan
# 'transaction_month'
cohort_data['cohort_period'] =
(cohort_data['transaction_month'].dt.year -
cohort_data['cohort_month'].dt.year) * 12 +
(cohort_data['transaction_month'].dt.month -
cohort_data['cohort_month'].dt.month) + 1

# Pivot data untuk memudahkan analisis cohort
cohort_pivot = cohort_data.pivot_table(index='cohort_month',
columns='cohort_period', values='n_customers')

# Visualisasi cohort analysis
plt.figure(figsize=(12, 8))
sns.heatmap(cohort_pivot, annot=True, fmt=".0f", cmap="YlGnBu",
cbar=False) # Ubah format menjadi ".0f" untuk format angka bulat
plt.title('Cohort Analysis - Repeat Transactions')

```

```
plt.xlabel('Cohort Period (Months)')
plt.ylabel('Cohort Month')
plt.show()
```



- **Insight Generation**

Analyze cohort behavior to detect recurring fraud patterns. For example, are there spikes in fraud after promotional campaigns

```
import pandas as pd
import matplotlib.pyplot as plt

# Asumsi kamu sudah memuat data dengan nama yang sesuai
# dt_transaction (dataset transaksi), dt_promotion (dataset promosi)

# Mengonversi kolom 'transaction_created_datetime' menjadi
# datetime
dt_transaction['transaction_created_datetime'] =
pd.to_datetime(dt_transaction['transaction_created_datetime'],
unit='s')

# Menambahkan kolom 'year_month' untuk analisis berdasarkan bulan
dt_transaction['year_month'] =
```

```

dt_transaction['transaction_created_datetime'].dt.to_period('M')

# Menyaring transaksi yang mencurigakan
dt_suspicious = dt_transaction[dt_transaction['suspicious'] == True]

# Menghitung persentase transaksi mencurigakan per bulan
suspicious_monthly = dt_suspicious.groupby('year_month').size()
all_monthly = dt_transaction.groupby('year_month').size()
fraud_percentage = (suspicious_monthly / all_monthly) * 100

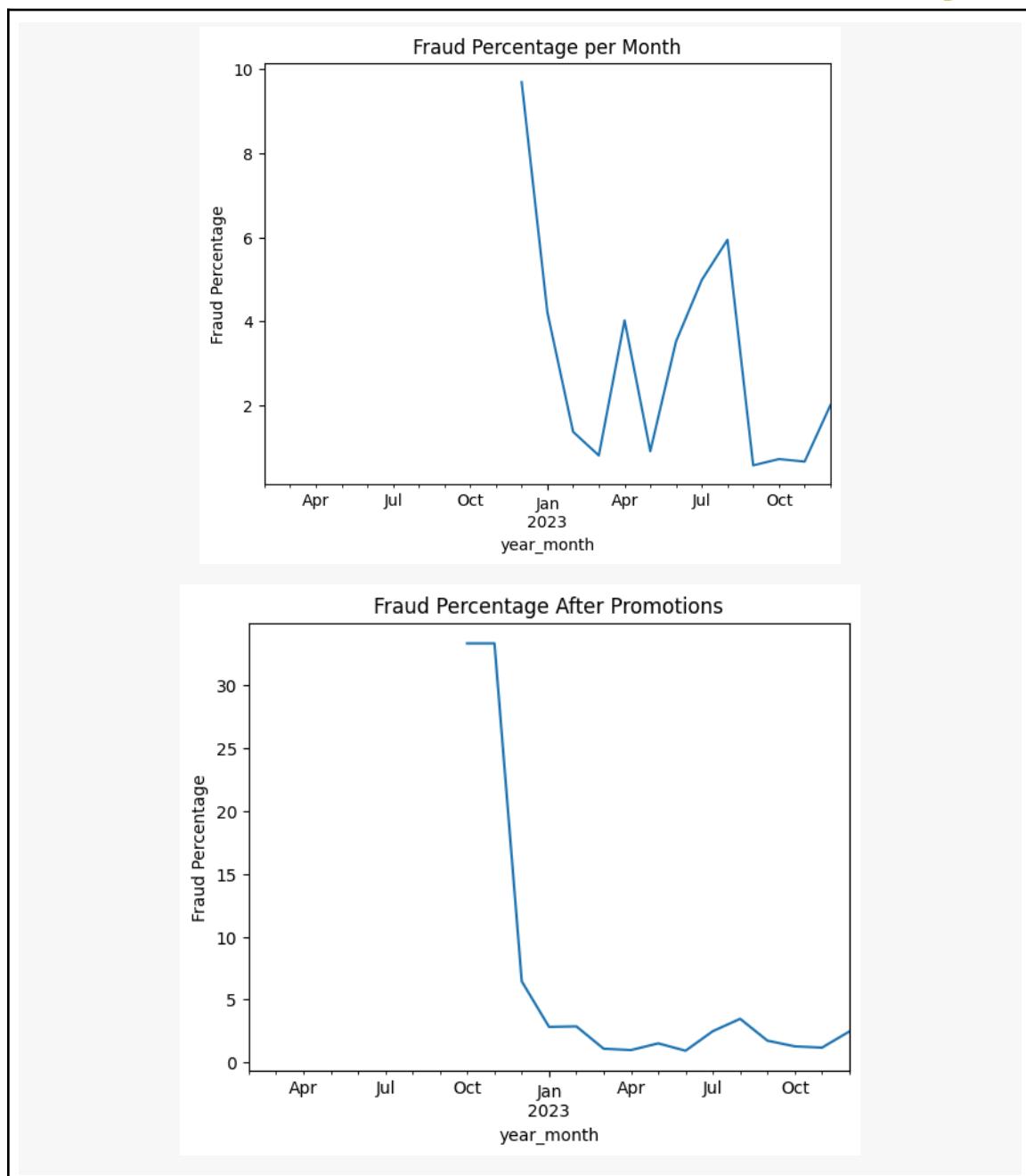
# Plot persentase kecurangan per bulan
fraud_percentage.plot(kind='line', title='Fraud Percentage per Month')
plt.ylabel('Fraud Percentage')
plt.show()

# Menyaring transaksi yang terkait dengan promosi berdasarkan dpt_promotion_id
dt_promotion_fraud =
dt_transaction[dt_transaction['dpt_promotion_id'].notna()]

# Menghitung persentase kecurangan yang terkait dengan promosi per bulan
promotion_fraud_monthly =
dt_promotion_fraud.groupby('year_month').size()
promotion_fraud_percentage = (promotion_fraud_monthly / all_monthly) * 100

# Plot persentase kecurangan setelah promosi
promotion_fraud_percentage.plot(kind='line', title='Fraud Percentage After Promotions')
plt.ylabel('Fraud Percentage')
plt.show()

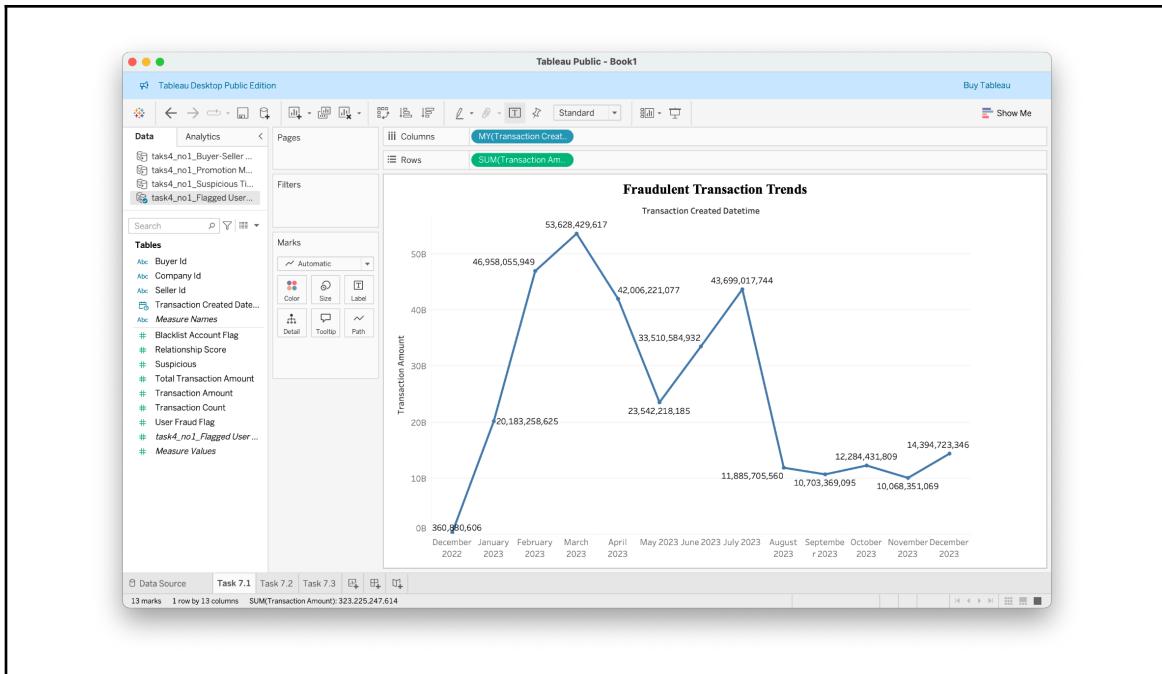
```



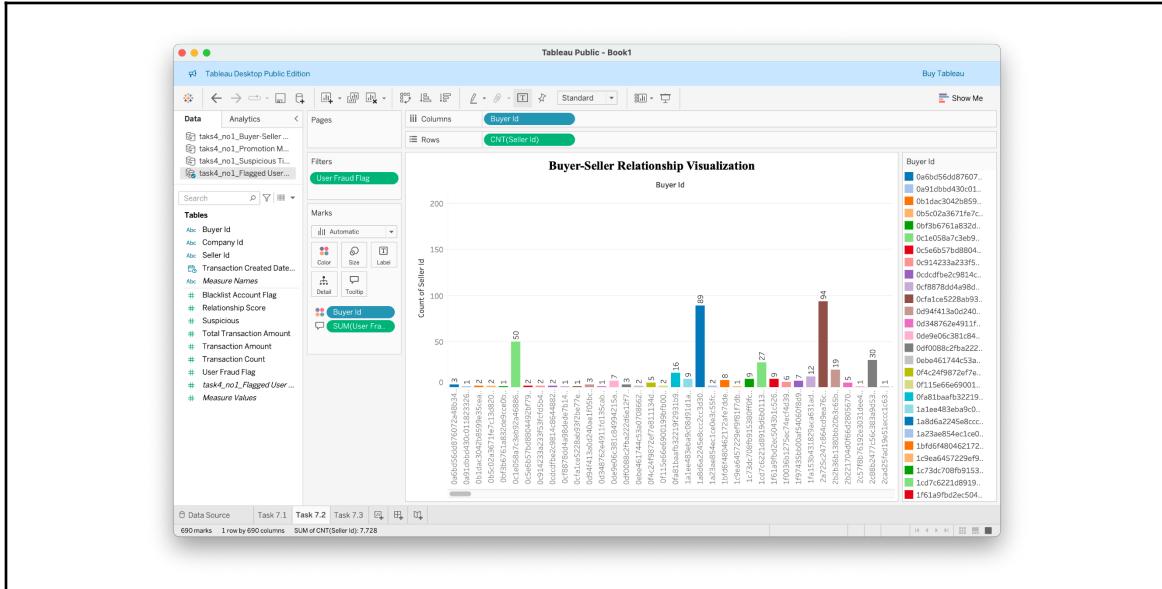
C. Pembahasan

1. Tableau Dashboards

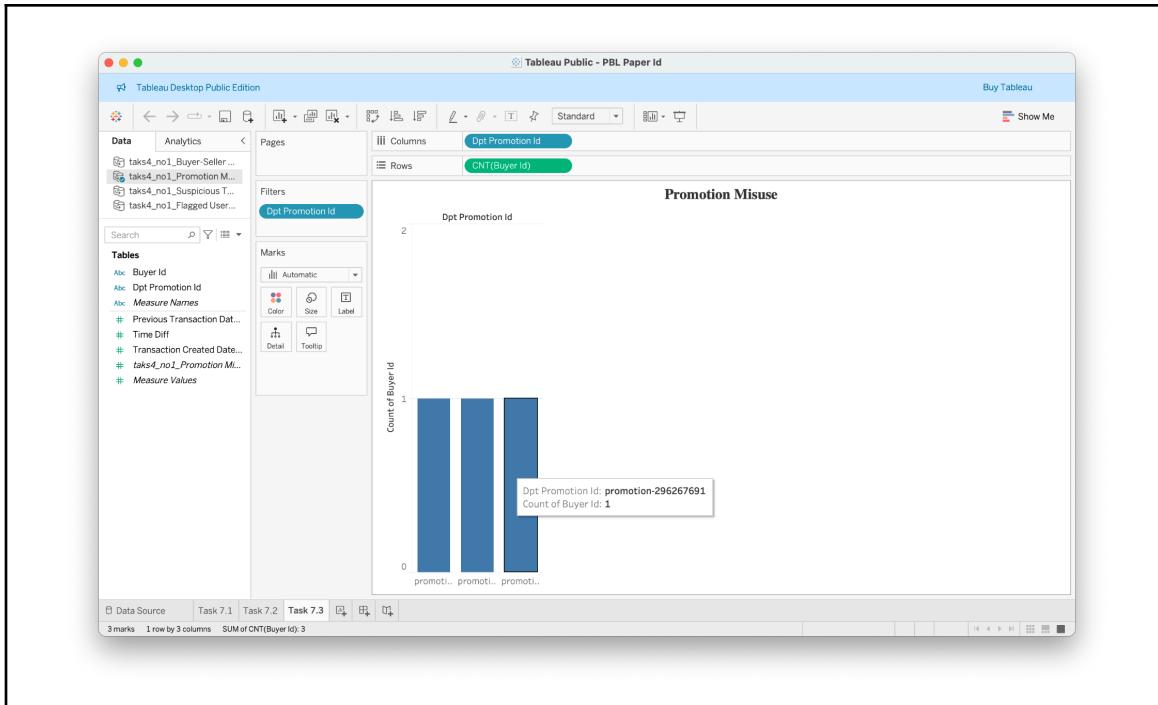
• **Fraudulent Transaction Trends**



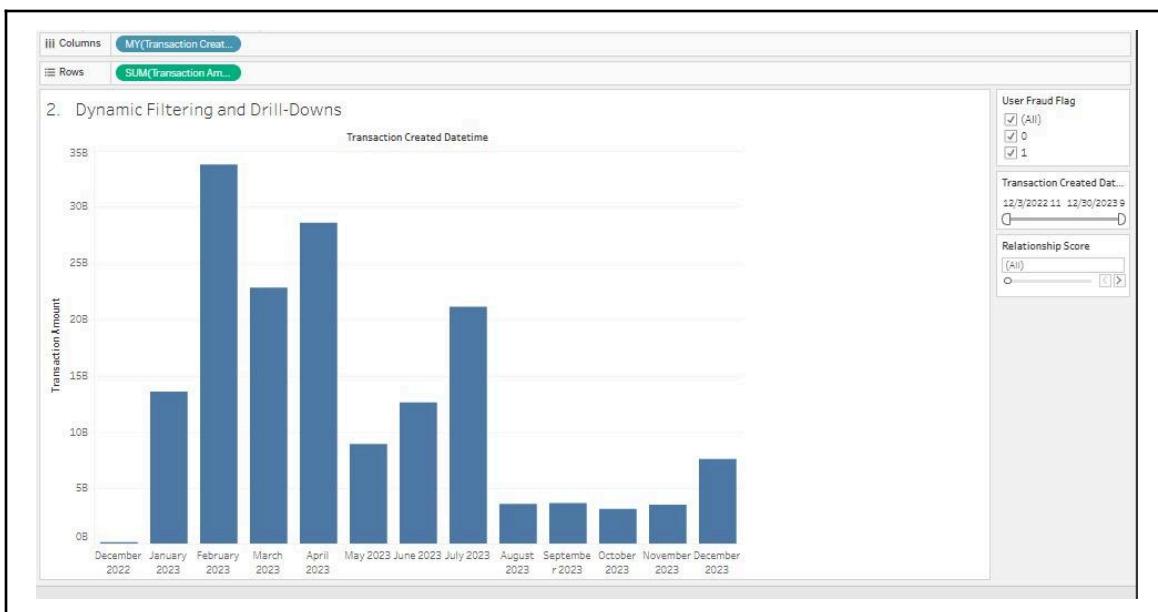
- **Buyer-Seller Relationship Visualization**



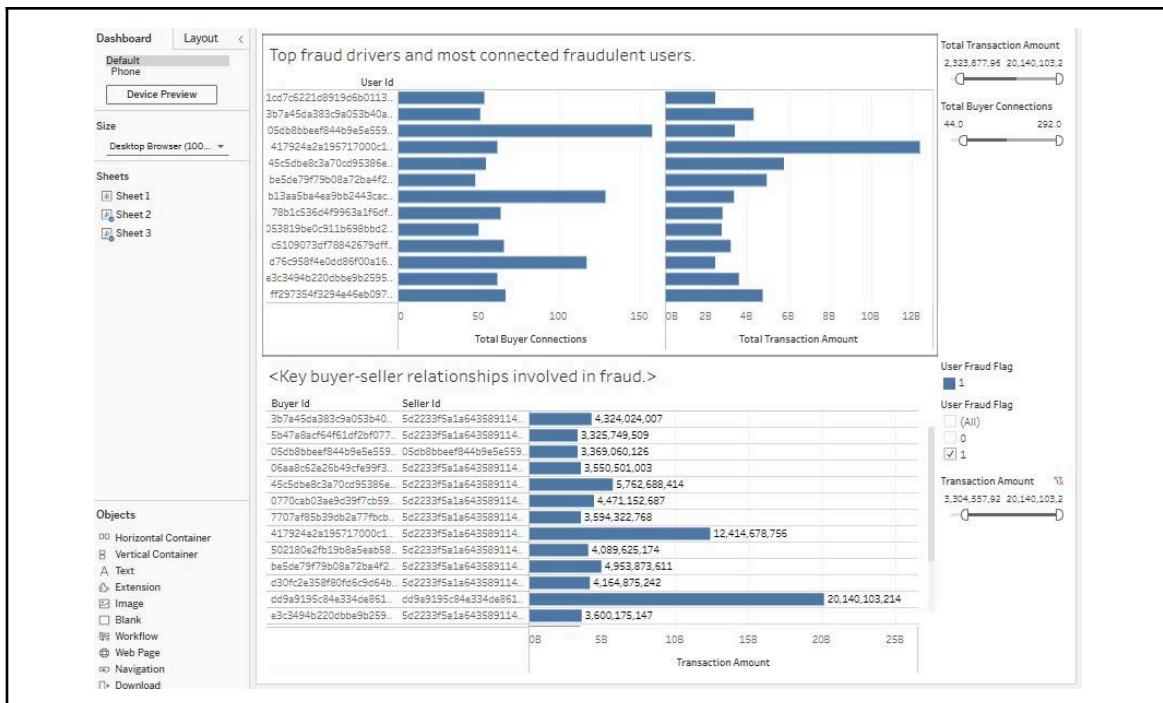
- **Promotion Misuse**



- **Dynamic Filtering and Drill-Downs:**



2. Key Fraud Insights



• Action Plans

1. Implementasikan Proses Verifikasi Pengguna yang Lebih Ketat:

Verifikasi yang lebih ketat dalam pendaftaran akun dengan meminta dokumen tambahan sangat penting untuk memastikan bahwa pengguna tidak mendaftar dengan tujuan menipu perusahaan, mengurangi risiko penipuan, meningkatkan kepercayaan pengguna, menjaga integritas data, mencegah penyalahgunaan akun, dan memastikan kepatuhan terhadap regulasi yang berlaku.

2. Memantau dan Menandai Aktivitas Mencurigakan antara Pembeli dan Penjual Secara Real-Time:

Salah satu cara untuk mendeteksi fraud adalah dengan memantau aktivitas antara pembeli dan penjual secara real-time. Hal ini mencakup pemantauan transaksi yang dilakukan oleh pengguna yang sering bertransaksi dalam waktu singkat, transaksi dengan nilai yang

tidak wajar, atau hubungan yang mencurigakan antara pembeli dan penjual. Sistem otomatis yang memantau transaksi secara langsung dapat memberi peringatan jika ada pola yang tidak biasa. Misalnya, jika seorang pembeli selalu bertransaksi dengan penjual yang sama atau jika transaksi dilakukan dalam jumlah besar tanpa alasan yang jelas, maka sistem bisa langsung mendekati aktivitas tersebut sebagai mencurigakan dan memblokir sementara akun yang terlibat untuk verifikasi lebih lanjut.

3. Menetapkan Kebijakan Promosi yang Lebih Ketat untuk Mencegah Penyalahgunaan:

Penyalahgunaan program promosi sering kali digunakan oleh pelaku fraud untuk mendapatkan keuntungan yang tidak sah, misalnya dengan membuat banyak akun untuk memanfaatkan diskon atau promo yang sama berulang kali. Untuk mencegah hal ini, perusahaan perlu menetapkan kebijakan yang lebih ketat terkait penggunaan promosi, seperti membatasi jumlah penggunaan kode promo per akun atau transaksi. Selain itu, penting untuk memonitor secara aktif penggunaan promo untuk mendekripsi pola yang tidak wajar, seperti transaksi yang dilakukan dalam waktu singkat dengan kode promo yang sama. Jika ditemukan adanya penyalahgunaan, perusahaan dapat menghentikan penggunaan promo tersebut dan menginvestigasi lebih lanjut akun yang terlibat.

BAB IV

Kesimpulan dan Saran

A. Simpulan

Penelitian ini menunjukkan adanya beberapa pola mencurigakan dalam transaksi yang terjadi di platform Paper.id, seperti transaksi dengan nilai atau frekuensi yang tidak wajar, kolusi antara pembeli dan penjual, serta penyalahgunaan program promosi. Analisis jaringan sosial mengidentifikasi kluster hubungan mencurigakan yang berpotensi membentuk jaringan fraud terorganisir. Selain itu, terdeteksi pula pola anomali dalam waktu transaksi, dengan lonjakan aktivitas yang tidak biasa pada jam-jam tertentu. Temuan ini mengindikasikan bahwa fraud tidak hanya merugikan secara finansial, tetapi juga dapat merusak reputasi dan kepercayaan pelanggan terhadap keamanan platform.

B. Saran

Untuk mencegah fraud, perusahaan perlu mengambil langkah-langkah strategis yang mencakup tiga area utama. Pertama, memperketat proses verifikasi pengguna dengan meminta dokumen tambahan saat pendaftaran akun untuk memastikan hanya pengguna yang sah yang dapat mengakses platform. Kedua, memantau dan menandai aktivitas mencurigakan antara pembeli dan penjual secara real-time, dengan menggunakan sistem otomatis untuk mendeteksi pola yang tidak biasa, seperti transaksi berulang atau dengan nilai yang tidak wajar. Ketiga, menetapkan kebijakan promosi yang lebih ketat, membatasi penggunaan kode promo per akun, serta memonitor secara aktif untuk mendeteksi penyalahgunaan yang mungkin terjadi. Langkah-langkah ini dapat membantu mengurangi risiko fraud, memperkuat integritas platform, dan meningkatkan kepercayaan pengguna.

Lampiran

A. Online Diagram

Silahkan lampirkan link Online Diagram Tools anda disini.

Link BPMN:

<https://drive.google.com/file/d/1KpuDOSu6KIVqhi6r-Lwdiyj-am8ELSmh/view?usp=sharing>

Link Flowchart:

https://drive.google.com/file/d/1a2aHmWToERMXpnOHE1_3chqcWHK4lhci/view?usp=sharing

B. Python Code

Silahkan lampirkan link Google Colab anda disini. **PASTIKAN LINK DAPAT DIAKSES SECARA PUBLIC.**

Task 2,3 dan 6:

<https://colab.research.google.com/drive/10xFiMExcUlyoWqxycXBKxT0yznBX7x6x?usp=sharing#scrollTo=3BbkVo3W-HL2>

C. Notulen

https://docs.google.com/document/d/1XUwTOekboe1eDSykYR16ACsHFCT_YwOj/edit?usp=sharing&ouid=101041898445335190620&rtpof=true&sd=true

D. Recording

<https://youtu.be/cjCLomk1WQo>

E. Tableau

<https://public.tableau.com/app/profile/naifa.syakira/viz/PBLPaperIdKelompok5CS09/Dashboards1?publish=yes>

https://public.tableau.com/views/pbldinamiicfilteringandrilldowns/Sheet1?:language=en-US&:sid=&:redirect=auth&:display_count=n&:origin=viz_share_link

F. PPT

<https://drive.google.com/file/d/1CBxSa6DnewXthh0i3kuNSWGBYOPch0eZ/view?usp=sharing>