STAGE 2  
DATAPREPROCESSING: AMAZON SALES DATASET

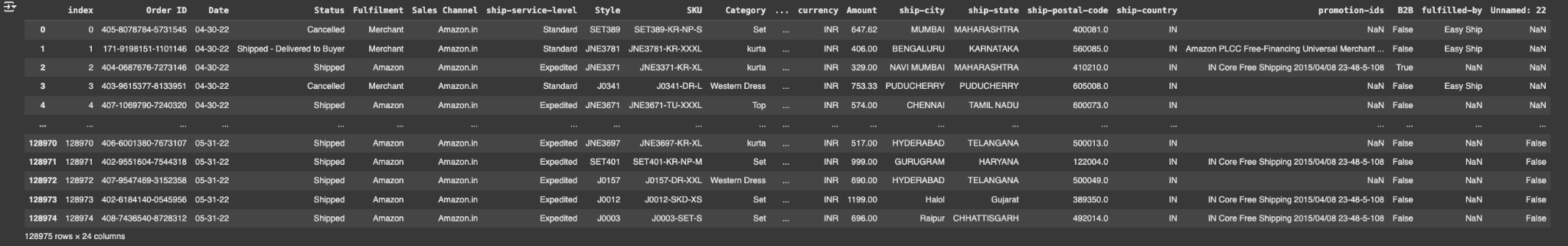
**\_\_\_**

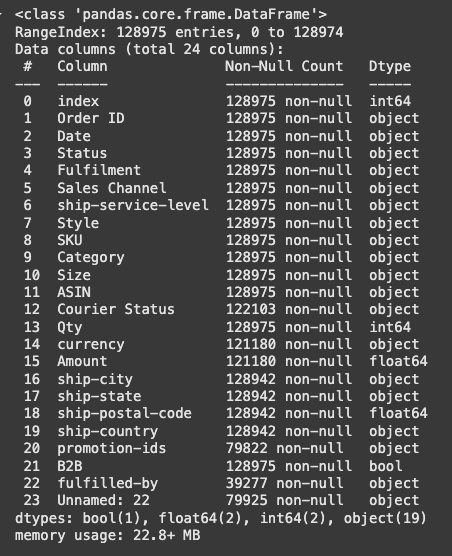


Oleh Kelompok 1- Last Voyage:

1. Mardio Edana Putra
2. Mochamad Rizqi
3. Muhammad Gusti Maulana Alfin
4. Nena Herdiani

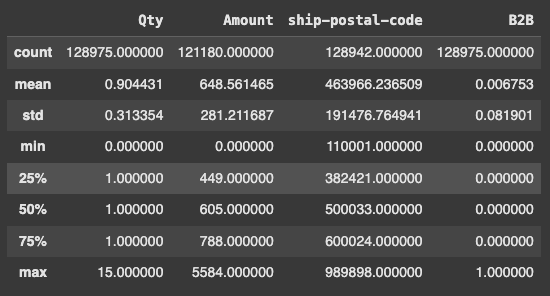
# MEMBACA FILE

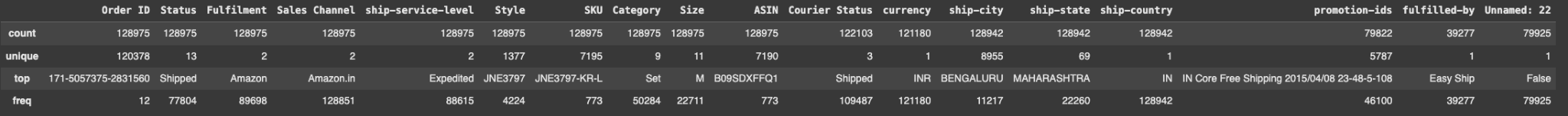




Terdapat beberapa kolom yang memiliki null-values yaitu: **Courier Status, currency, Amount, ship-city, ship-state, ship-postal-code, ship-country, promotion-ids, fulfilled-by, dan** **Unnamed:22**. Kemudian kolom date memiliki tipe data object akan diubah menjadi date, dan kolom B2B dari bool menjadi integer (0 atau 1).

# DESCRIPTIVE STATISTICS



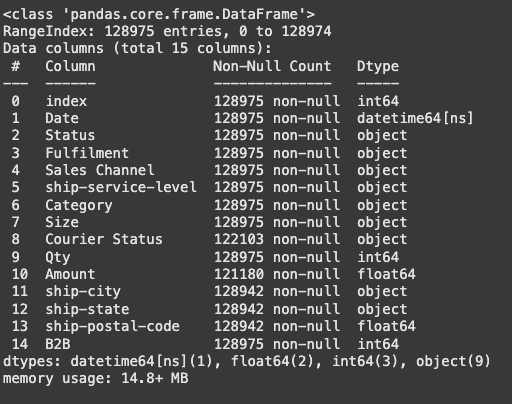


Kolom kategorikal dengan nilai tunggal akan dihapus karena tidak memberikan informasi yang berguna untuk analisis atau pemodelan dan tidak memiliki hubungan statistik dengan variabel target karena tidak ada perbedaan nilai untuk dipelajari model. Kolom tersebut adalah **currency, ship-country, fulfilled-by, dan Unnamed: 22.** Beberapa kolom lainnya yang dirasa tidak berpengaruh adalah

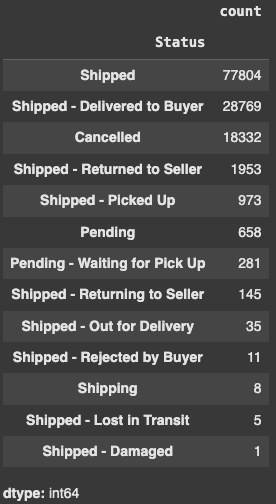
1. Order ID : ID adalah transaksi unik yang hanya berfungsi sebagai identifikasi, bisa dihapus karena tidak memberikan informasi yang berguna untuk prediksi.

2. Style, SKU, ASIN bisa dipertimbangkan untuk dihapus karena dapat diwakilkan dengan category dan size.

**Filter data selain kolom tertentu**



**Melihat distribusi pada kolom target status**



**Relevansi Bisnis:**

1. Dalam konteks bisnis e-commerce atau retail, fokus utama adalah pada pengalaman konsumen dan efisiensi proses logistik.

2. Kategori seperti **Delivered to Buyer**, **Returned to Seller**, dan **Rejected by Buyer** mencerminkan status akhir transaksi yang sangat relevan untuk mengevaluasi kepuasan konsumen dan efektivitas pengiriman.

3. Status lain mungkin lebih terkait dengan proses internal (seperti packing, shipped, dll.) dan kurang memberikan insight dari sisi pengalaman konsumen.

**Simplifikasi Model:**

1. Mengurangi jumlah kelas target dari 13 menjadi 3 akan membuat proses klasifikasi lebih sederhana dan meningkatkan performa model.

2. Dengan jumlah kelas yang terlalu banyak, model bisa menjadi terlalu kompleks (high variance) dan lebih sulit diinterpretasi.

3. Fokus pada 3 kelas utama juga membantu mencegah masalah ketidakseimbangan kelas yang berlebihan.

**Actionable Insights:**

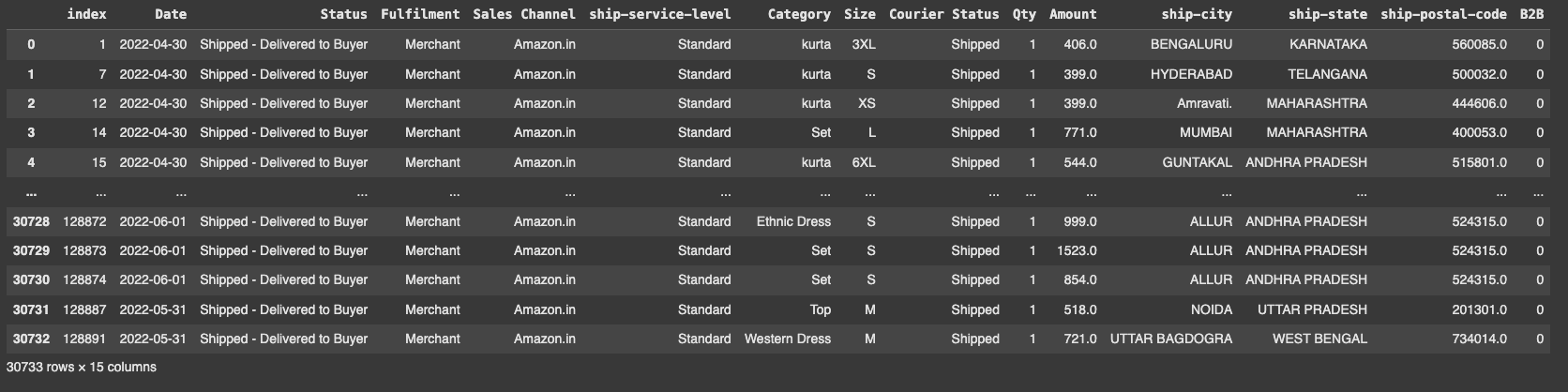
Ketiga status yang dipilih memungkinkan tim bisnis untuk mengambil tindakan yang jelas dan dapat ditindaklanjuti:

\* **Returned to Seller**: Mengidentifikasi masalah kualitas produk atau proses pengiriman yang buruk.

\* **Rejected by Buyer**: Menganalisis alasan penolakan, seperti ketidakpuasan terhadap deskripsi produk atau masalah pengiriman.

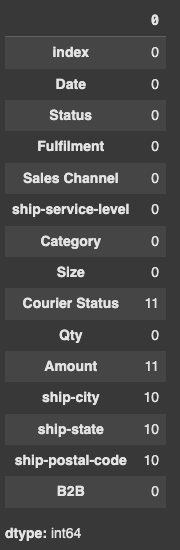
\* **Delivered to Buye**r: Mengukur tingkat keberhasilan pengiriman dan kepuasan pelanggan.

**Melakukan Filtering Status**



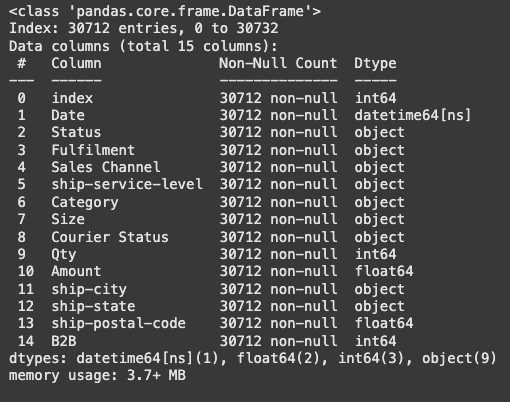
# DATA CLEANSING

1. **Handling Missing Value**

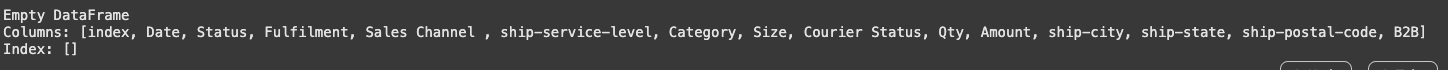


Terdapat 5 kolom yang memiliki missing values, solusi yang akan diberikan adalah langsung menghapus kolom ini karena jumlahnya hanya sedikit sehingga dirasa tidak terlalu mempengaruhi analisis dan pemodelan kedepannya dan menghapus missing value dirasa lebih praktis.

Setelah dihapus dan dicek semua data sudah tidak memiliki missing values dan berjumlah 30712 baris.

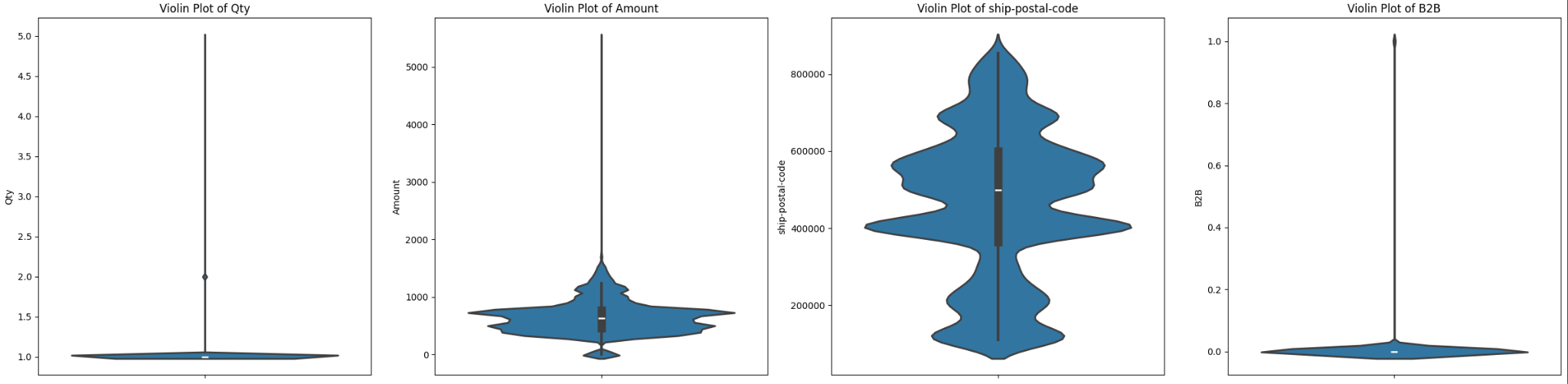


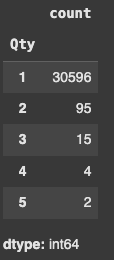
1. **Handling Duplicated Data**



Setelah dilakukan pengecekan, tidak terdapat data yang duplikat atau memiliki nilai yang sama pada lebih dari satu baris, oleh sebab itu dataset dianggap sudah bersih dari missing values dan juga adanya duplikasi data dan dapat dilanjutkan ke tahapan pengecekan Outlier.

1. **Handle Outliers**





Setelah dilakukan pengecekan outlier dengan menggambarkan ViolinPlot pada kolon numerikal Qty dan Amount, didapatkan bahwa fitur tersebut memiliki outlier yang sangat tinggi dengan grafik skewed ke kanan. Untuk kolom Qty hampir seluruh baris memiliki jumlah 1 sedangkan kolom Amount, distribusi data yang terbentuk skewed ke kanan dengan nilai mean yang lebih tinggi daripada median.

Outlier pada kolom Qty dan Amount akan dipertahankan di dataset ini dan tidak dihapus karena:

1. **Mencerminkan Realitas Bisnis**: Transaksi besar atau harga tinggi adalah bagian alami dari e-commerce

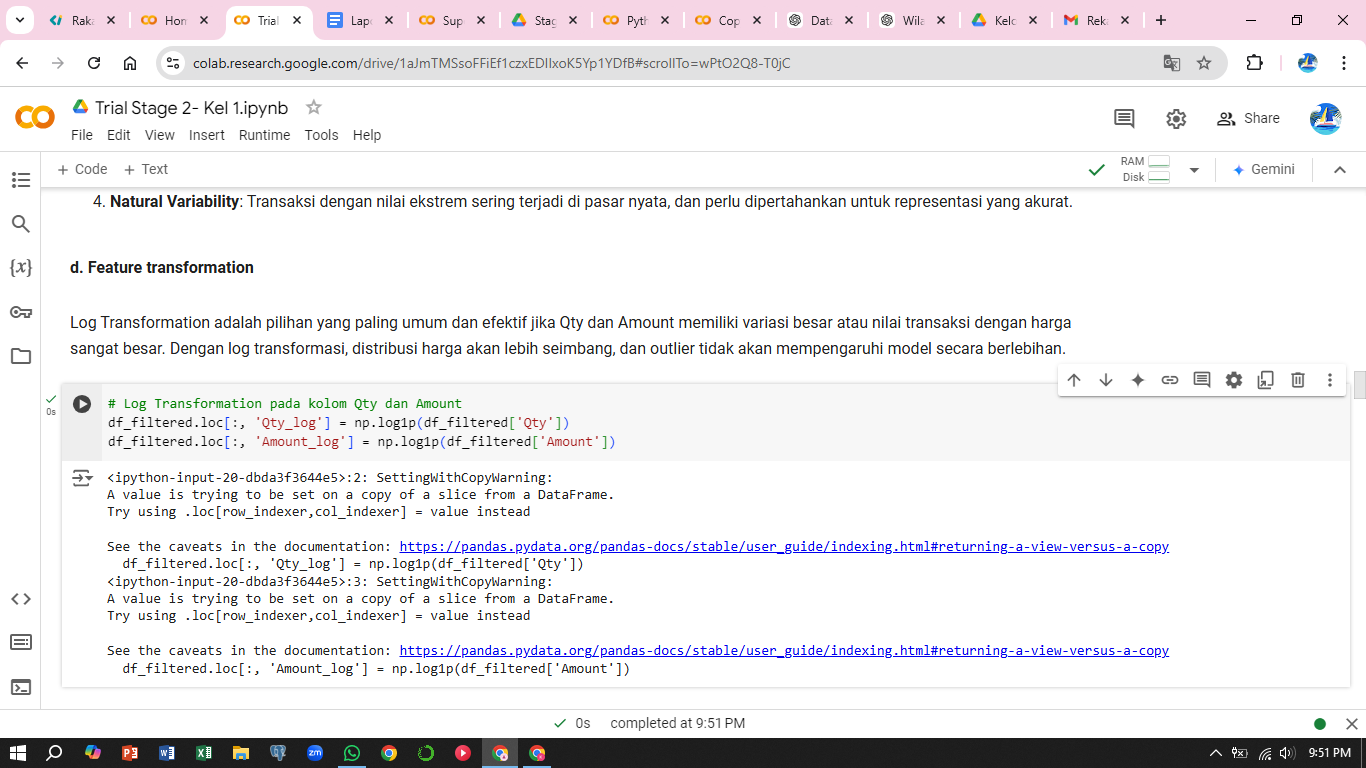
2. **Menambah Wawasan:** Outlier bisa memberikan informasi penting tentang pola pembelian.

3. **Pengaruh pada Pemodelan**: Menghapus outlier dapat menghilangkan variasi penting yang mempengaruhi model dan keputusan bisnis.

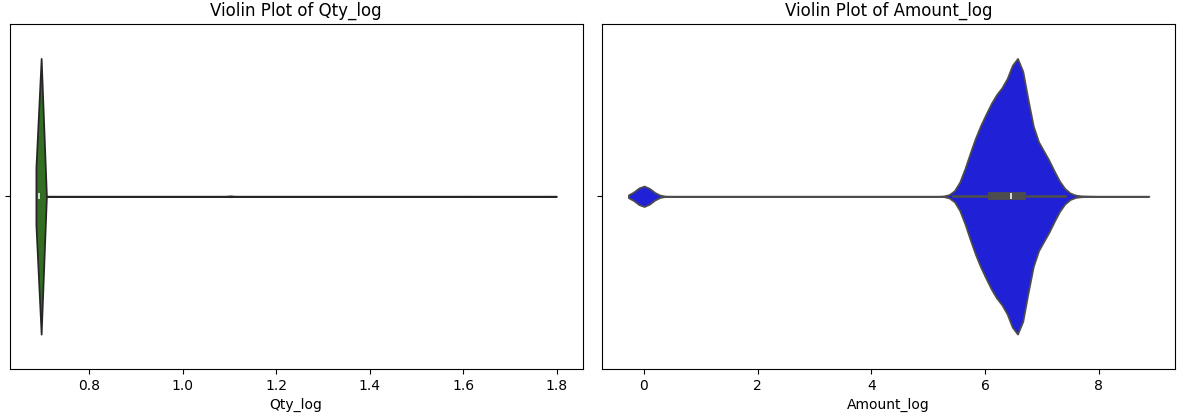
4. **Natural Variability**: Transaksi dengan nilai ekstrem sering terjadi di pasar nyata, dan perlu dipertahankan untuk representasi yang akurat.

1. **Feature Transformation**

Log Transformation adalah pilihan yang paling umum dan efektif jika Qty dan Amount memiliki variasi besar atau nilai transaksi dengan harga sangat besar. Dengan log transformasi, distribusi harga akan lebih seimbang, dan outlier tidak akan mempengaruhi model secara berlebihan. Berikut syntax untuk melakukan transformation fitur pada Qty dan Amount.



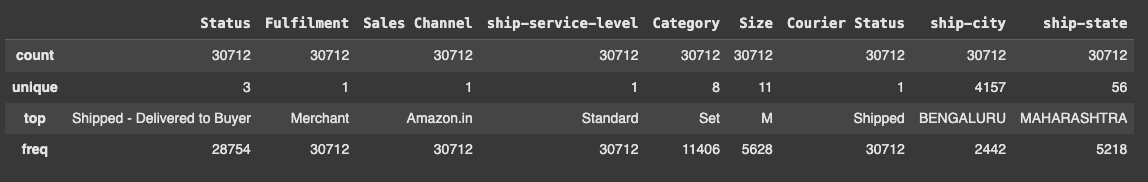
Akan terbentuk kolom baru bernama Qty\_log dan Amount\_log. Selanjutnya, dilakukan pembuatan ViolinPlot untuk mengecek grafik setelah transformasi.



Terlihat bahwa untuk fitur Amount sudah cenderung mendekati distribusi normal walaupun masih terlihat skewed, namun pada fitur Qty masih terlihat skewed. Namun, hal ini tidak menjadi masalah karena kedepannya akan digunakan algoritma atau model yang robust terhadap outlier-outlier yang ada.

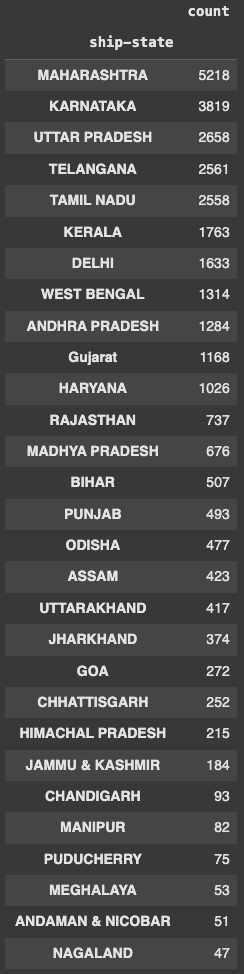
1. **Feature Encoding**

Pertama-tama dilakukan pengecekan terhadap kolom kategorikal yang ada seperti di bawah ini.



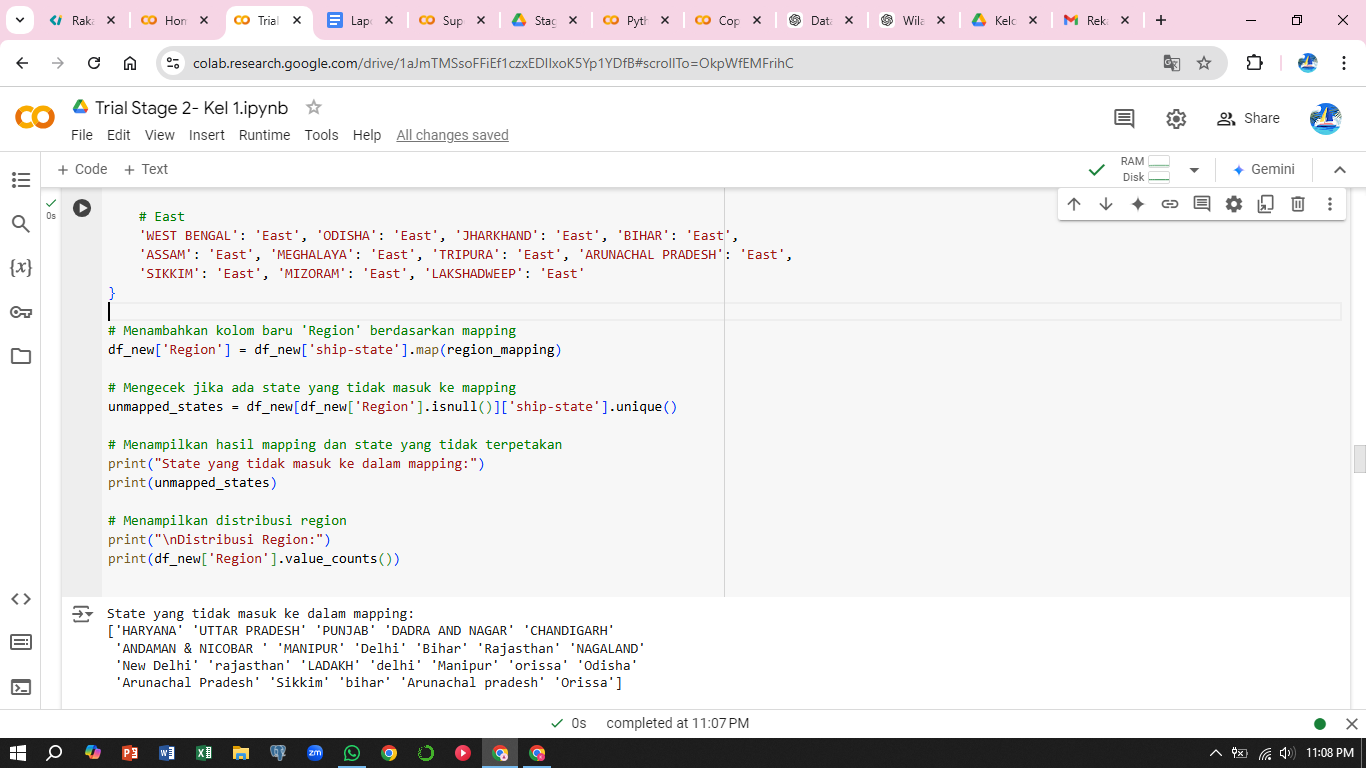
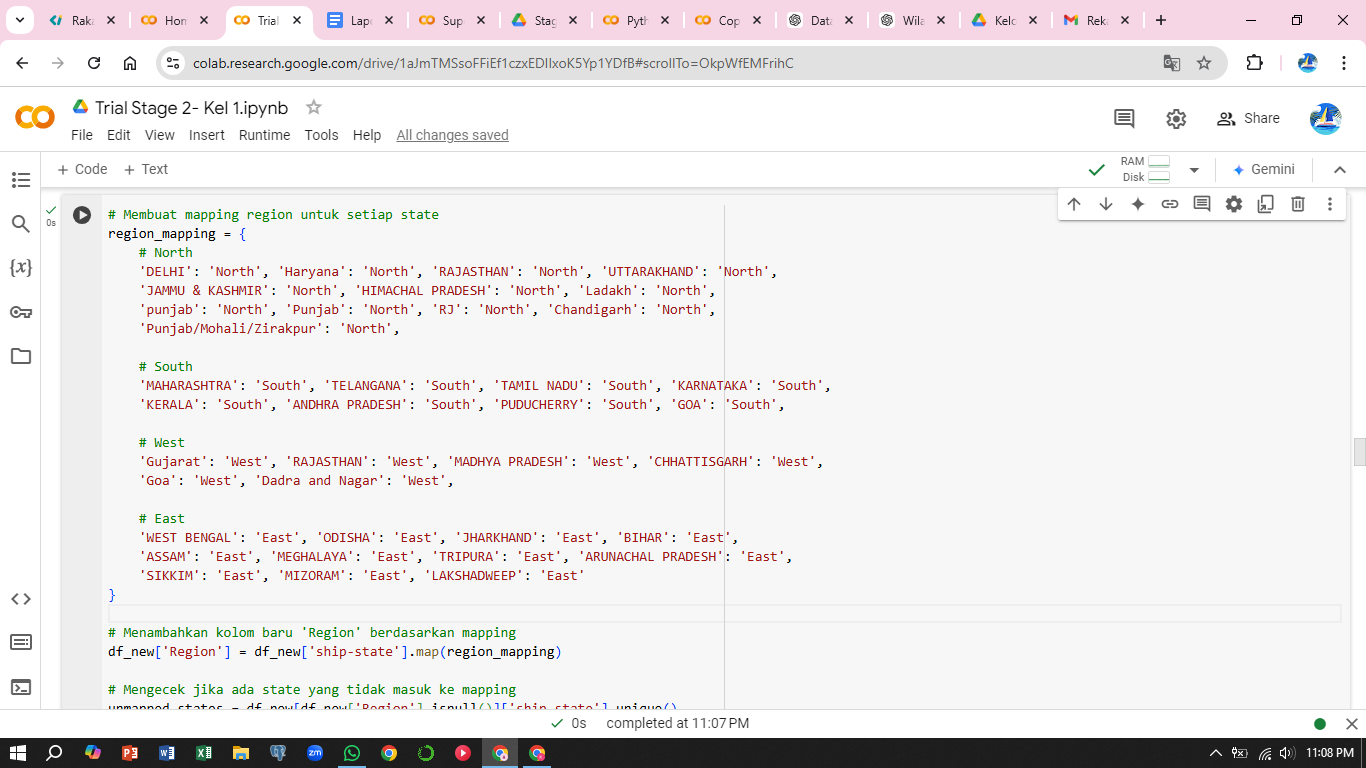
Setelah dilakukan filtering terhadap tiga kolom status yang diinginkan, terlihat banyak kolom kategorikal seperti **Fulfilment, Sales Channel, ship-service-level, dan Courier Status** nilainya hanya berisikan 1 kategori saja sehingga tidak akan berpengaruh kedepannya dalam pemodelan dan diputuskan dihapus. Untuk kolom **ship-city** dikarenakan dapat diperkecil menjadi ship-state akan dipilih salah satu saja, yaitu **ship-state.**

**Melihat distribusi data pada kolom ship-state**



Berdasarkan gambar diatas terlihat bahwa terdapat sekitar 56 state yang merupakan kota-kota yang ada di India, apabila dilakukan one-hot-encoding dirasa terlalu banyak sehingga diusulkan alternatif untuk memetakan setiap kota berdasarkan kondisi geografisnya seperti North, South, East, West, dan kota yang tidak terdeteksi menjadi Others seperti dibawah ini.

**Membuat Mapping Region untuk setiap State**

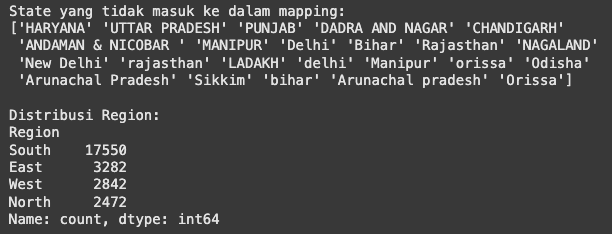
****

**Berikut adalah penjelasan kode di atas:**

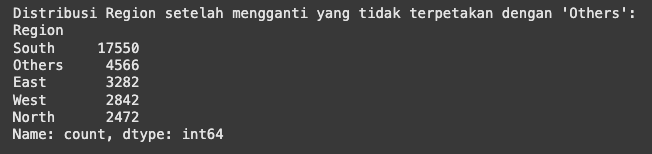
1. Mapping Region
   * region\_mapping: Kamus (dictionary) yang memetakan nama-nama state di India ke region geografis mereka (North, South, West, East).
   * Ini digunakan untuk mengelompokkan data ke dalam region berdasarkan kolom ship-state.
2. Menambahkan Kolom Region
   * df\_new['Region'] = df\_new['ship-state'].map(region\_mapping): Menambahkan kolom baru Region ke DataFrame df\_new.
   * Fungsi map() memetakan setiap nilai pada kolom ship-state menggunakan region\_mapping.
3. Mengecek State yang Tidak Masuk ke Mapping
   * unmapped\_states: Menyimpan daftar state di kolom ship-state yang tidak berhasil dipetakan karena tidak ada di region\_mapping.
   * State ini ditandai dengan nilai NaN pada kolom Region.
4. Distribusi Region
   * df\_new['Region'].value\_counts(): Menampilkan jumlah data untuk setiap region (North, South, dll.) setelah pemetaan.

### Tujuan Kode:

* Mengelompokkan data berdasarkan region geografis untuk analisis lebih lanjut.
* Memastikan semua state memiliki region yang valid, dan mengidentifikasi state yang belum terpetakan untuk diperbaiki.

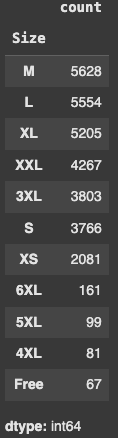


Untuk kota yang tidak terdeteksi lokasi geografisnya, diusulkan untuk diubah menjadi Others.



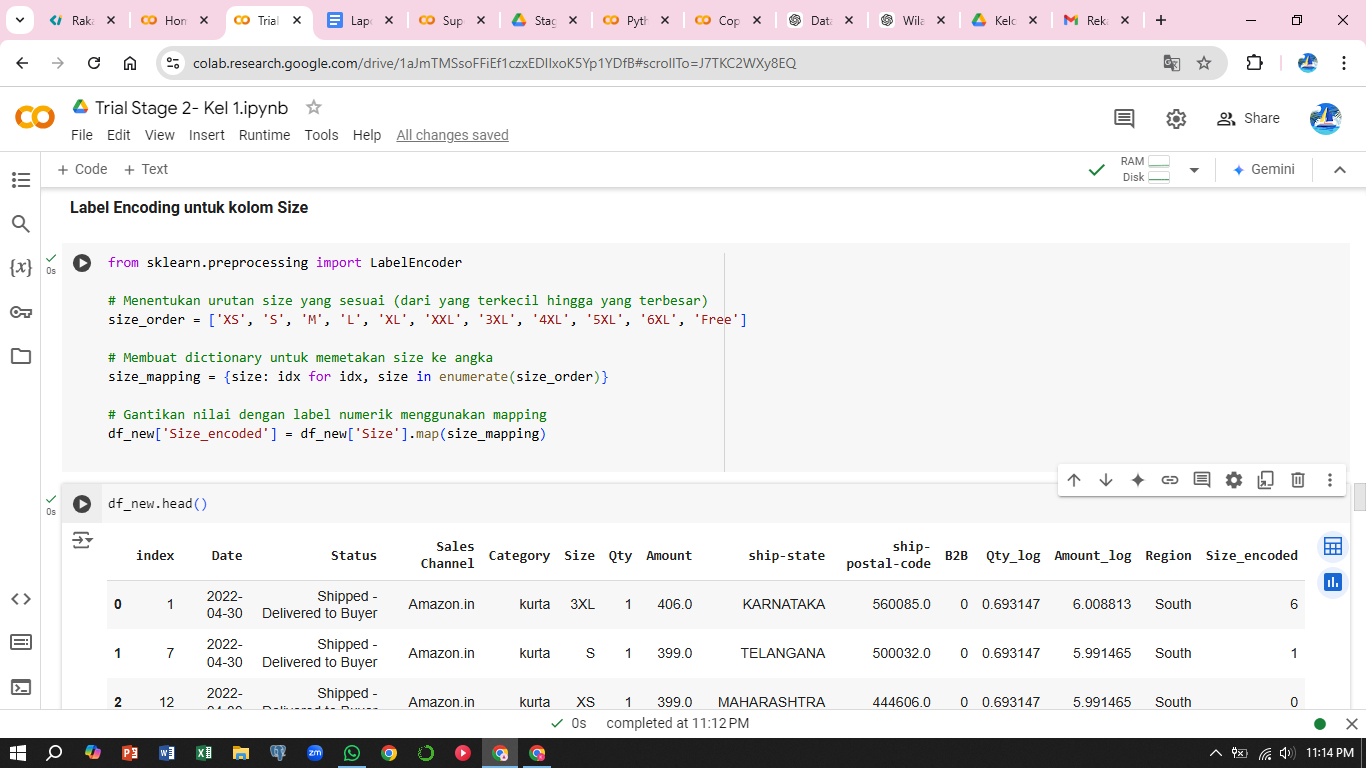
Gambar diatas menunjukkan distribusi hasil akhir dari mapping region.

**Melihat distribusi data pada kolom Size**



Berdasarkan gambar diatas terlihat bahwa fitur Size berisikan ukuran dari produk yang dapat diurutkan tingkatannya mulai dari yang paling kecil XS hingga 6XL sehingga dapat dilakukan Label Encoding untuk mengubahnya ke tipe data numerikal. Tujuannya adalah:

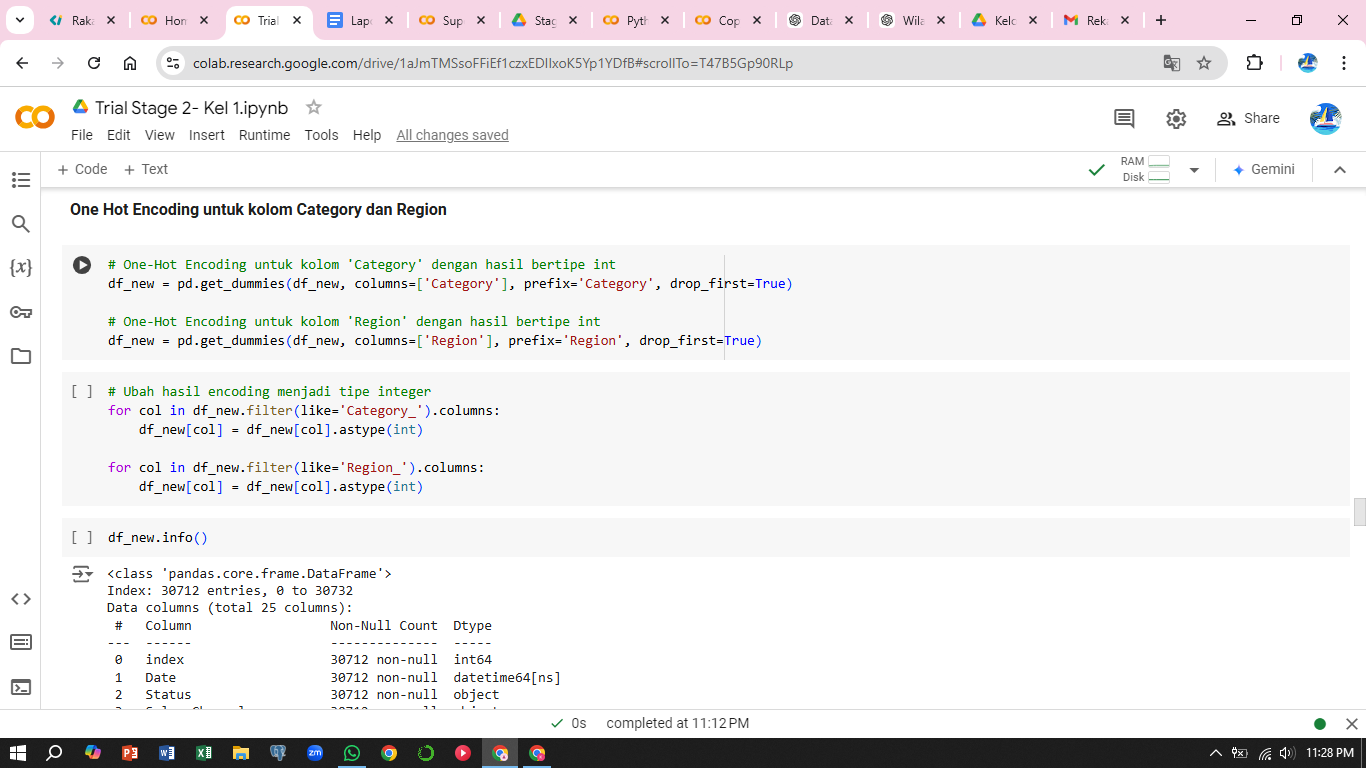
* Mengubah data kategorikal (size dalam bentuk teks) menjadi data numerik agar bisa digunakan dalam analisis atau model machine learning.
* Encoding ini mempertahankan informasi urutan (ordinal) ukuran sehingga model memahami hubungan antar ukuran (misalnya, L lebih besar dari M).



**df\_new['Size\_encoded'] = df\_new['Size'].map(size\_mapping)**:

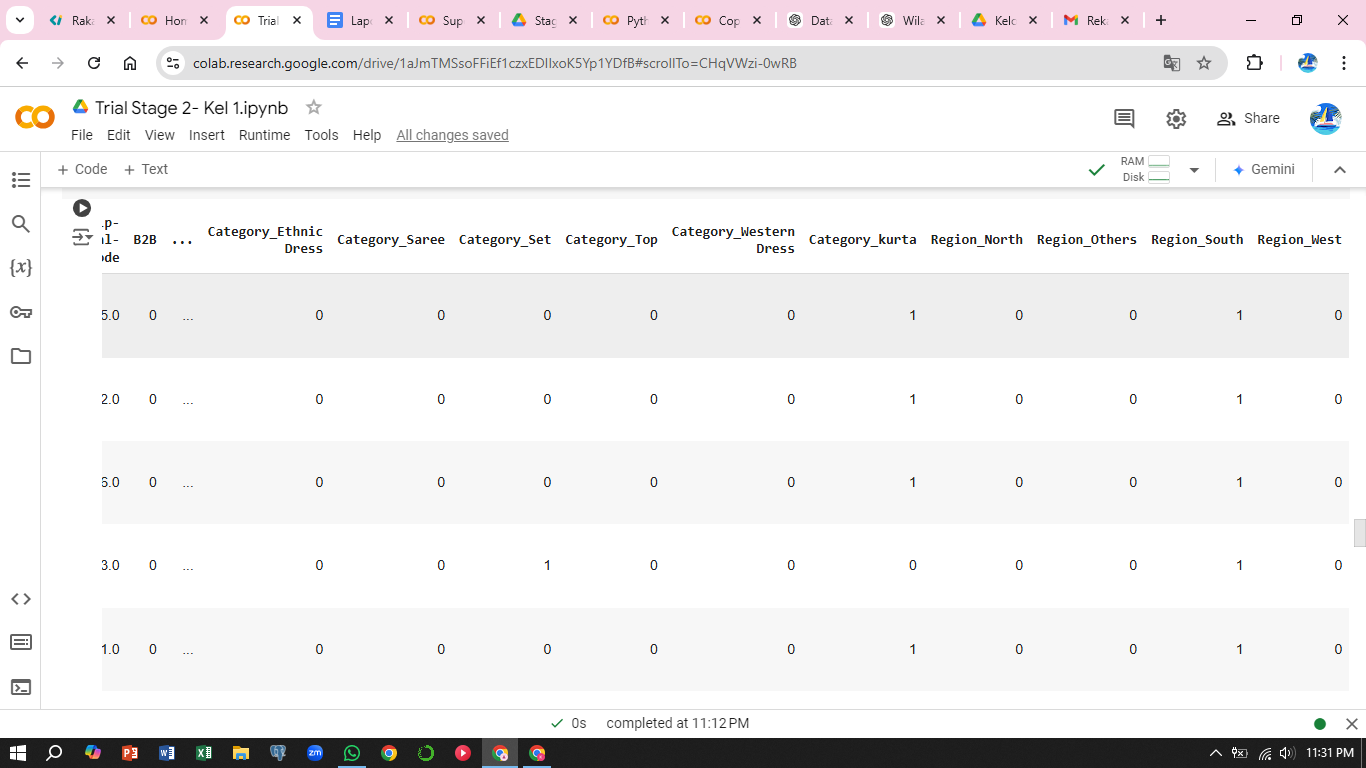
* Menggunakan fungsi **map()** untuk mengganti setiap nilai pada kolom **Size** dengan nilai numerik dari **size\_mapping**.
* Kolom baru bernama **Size\_encoded** dibuat untuk menyimpan nilai yang telah dikonversi.

**One Hot Encoding untuk kolom Category dan Region**

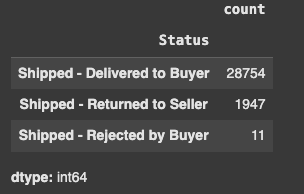


Kode di atas menggunakan **One-Hot Encoding** untuk mengonversi kolom **Category** dan **Region** yang berisi data kategorikal menjadi representasi numerik dalam bentuk kolom biner (0 atau 1). Dengan **pd.get\_dummies**, setiap kategori dalam kolom tersebut diubah menjadi kolom baru dengan nama diawali oleh prefix (Category\_ atau Region\_), dan nilai 1 menunjukkan keberadaan kategori tersebut. Parameter **drop\_first=True** digunakan untuk menghindari redundansi (dummy variable trap) dengan menghapus satu kategori dari encoding. Hasilnya, data menjadi lebih kompatibel untuk analisis numerik atau algoritma machine learning.

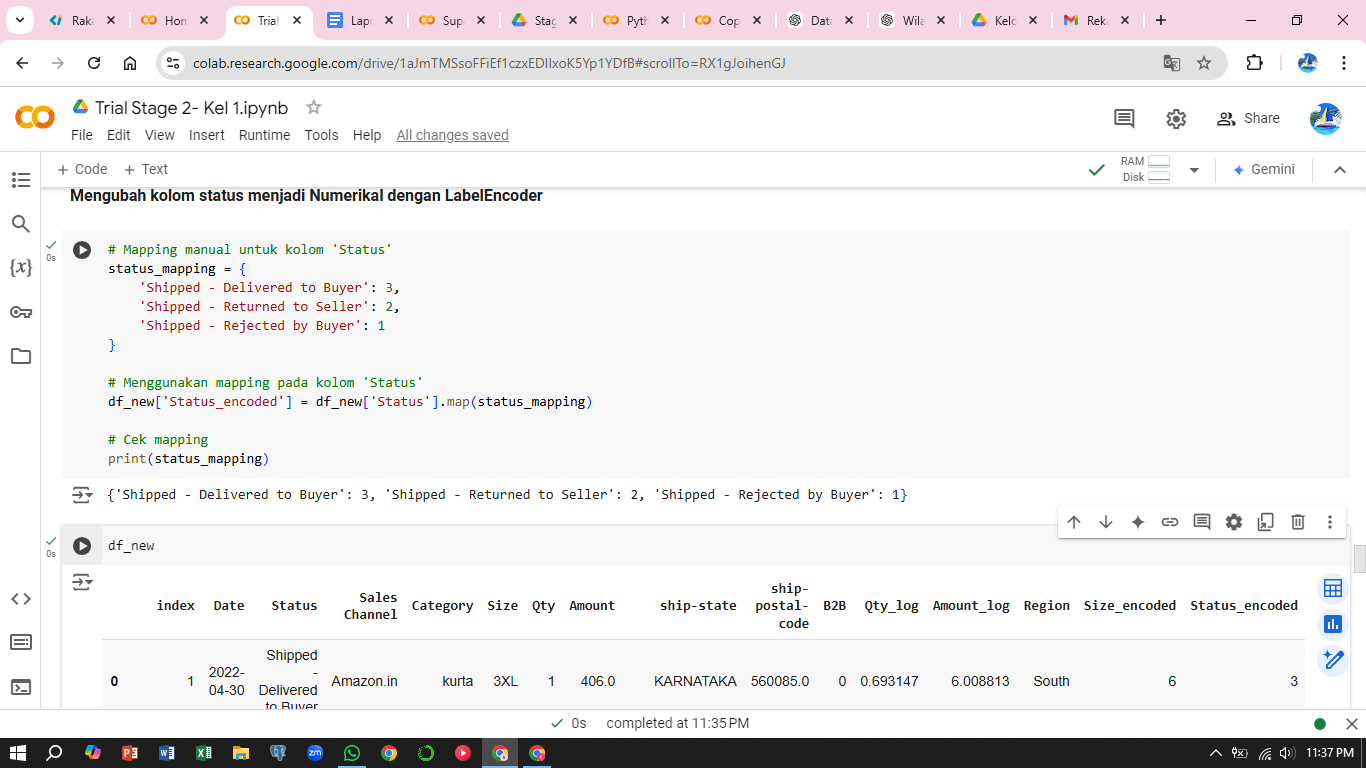
Berikut gambaran dataframe setelah dilakukan one hot encoding.



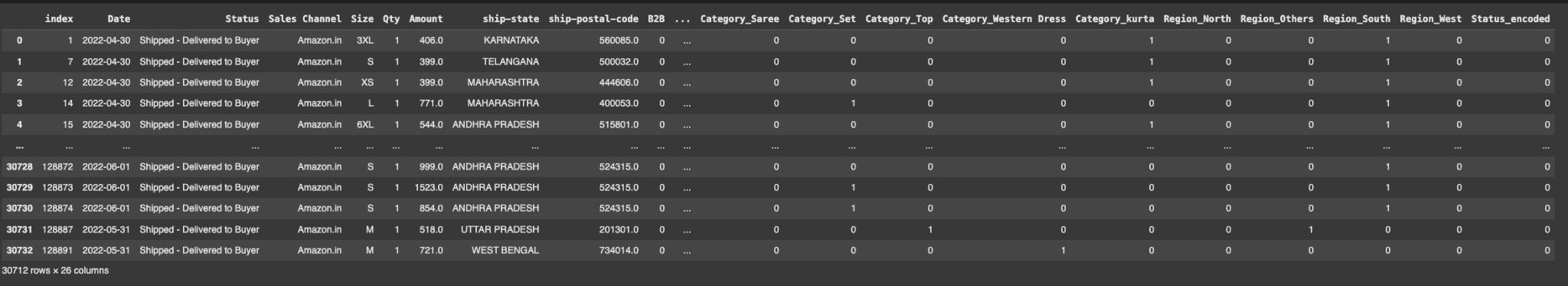
1. **Handled Class Imbalanced**



Dari gambar diatas terlihat untuk kolom target status memiliki jumlah kelas yang tidak seimbang, akan dilakukan penyeimbangan kelas dengan metode SMOTE. Sebelumnya, perlu dilakukan konversi menjadi numerikal untuk memudahkan pemodelan nantinya. Kolom status akan diubah mengunakan LabelEncoder dengan sintaks berikut.



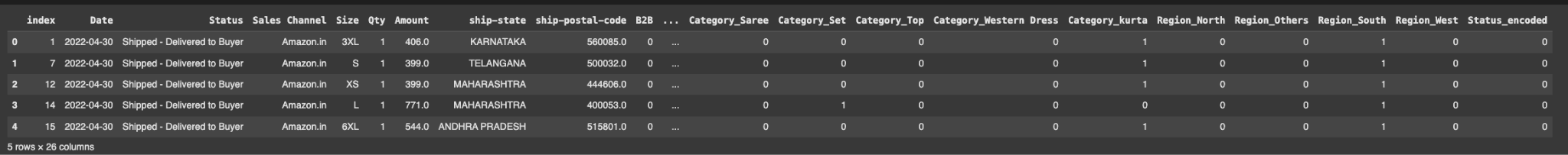
Kode tersebut membuat mapping manual untuk kolom **Status** menggunakan dictionary **status\_mapping**, di mana status yang lebih positif, seperti *"Shipped - Delivered to Buyer"*, diberi nilai tertinggi (3), sedangkan status yang lebih negatif, seperti *"Shipped - Rejected by Buyer"*, diberi nilai terendah (1). Fungsi **.map()** kemudian digunakan untuk mengganti nilai dalam kolom **Status** sesuai dengan mapping tersebut, sehingga menghasilkan kolom baru **Status\_encoded** yang berisi nilai numerik berdasarkan tingkatannya. Pendekatan ini memastikan data lebih sesuai dengan logika bisnis, di mana nilai numerik mencerminkan polaritas atau dampak dari status tersebut.



Class imbalanced diatasi menggunakan metode **SMOTE (Synthetic Minority Oversampling Technique)** untuk mengatasi masalah ketidakseimbangan kelas pada dataset. Pertama, kolom-kolom tertentu seperti **Date**, **Status**, dan beberapa kolom lainnya dikeluarkan dari dataset fitur **X**, sementara kolom **Status\_encoded** diambil sebagai target **y**. Dengan **smote.fit\_resample(X, y)**, SMOTE membuat sampel sintetis untuk kelas minoritas, sehingga jumlah data setiap kelas menjadi seimbang. Setelah itu, distribusi kelas hasil resampling diperiksa menggunakan **Counter**, yang menampilkan jumlah data per kelas dalam **y\_resampled**. Metode ini membantu model machine learning bekerja lebih baik dengan mengurangi bias terhadap kelas mayoritas. Hasilnya adalah sebagai berikut.



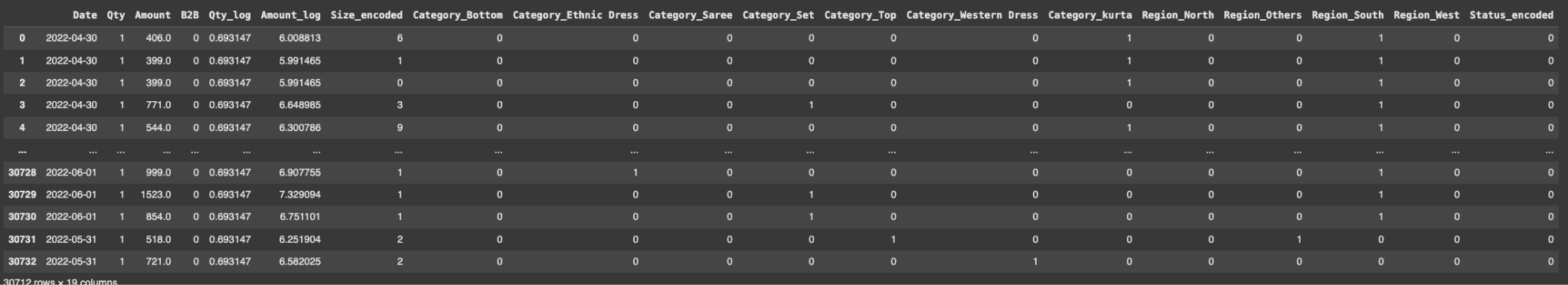
# FEATURE ENGINEERING



1. **Feature Selection**

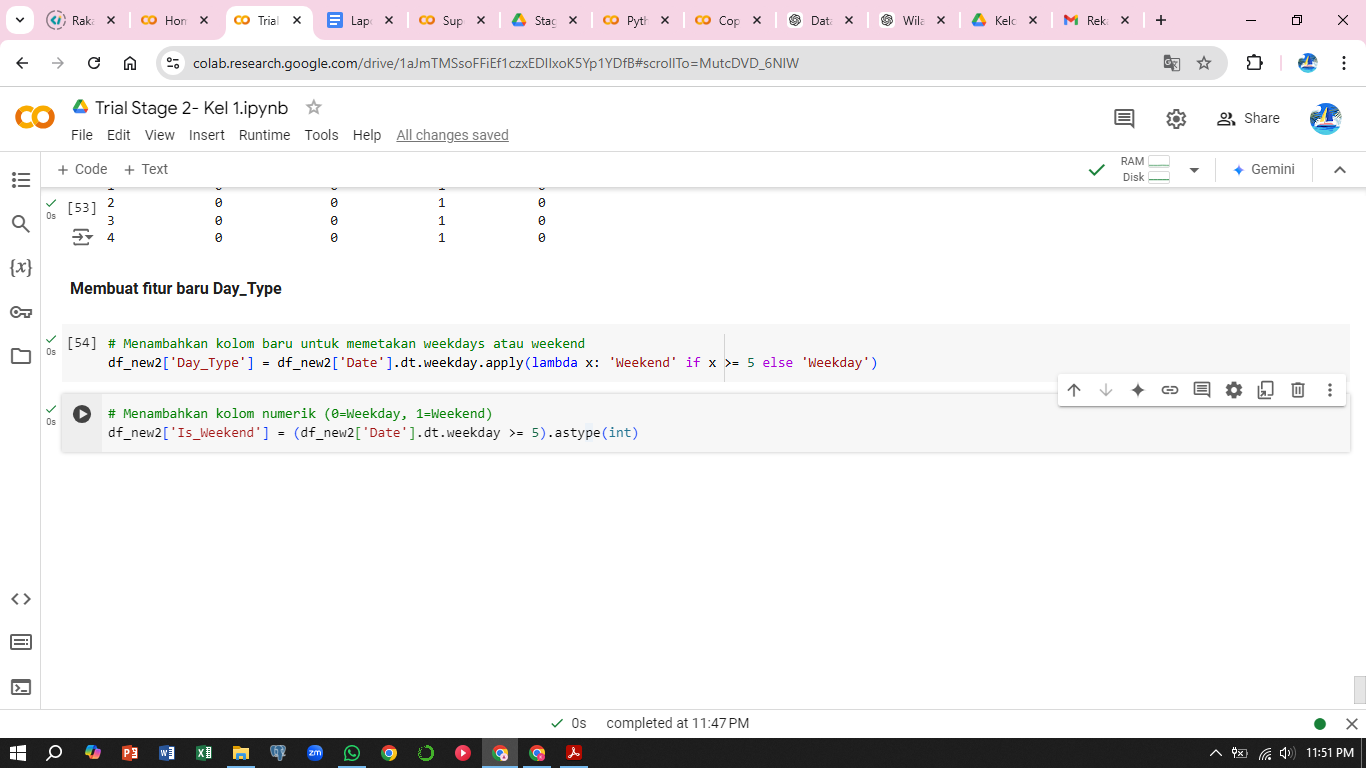
Beberapa fitur yang dapat dibuang adalah:

* index (Tidak relevan untuk analisis, hanya merupakan indeks dataset)
* ship-postal-code (sudah diwakilkan state)
* Sales Channel (sama semua, yaitu Amazon.in sehingga tidak berpengaruh ke analisis)
* Status (sudah di-encode, hapus saja)



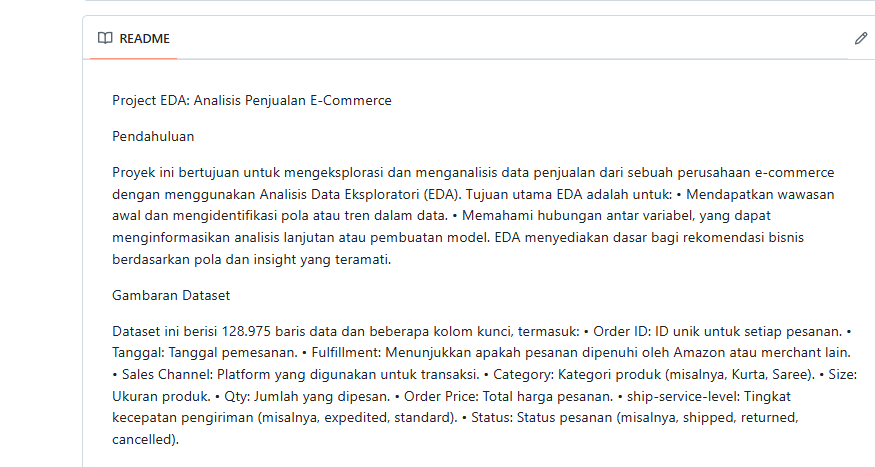
1. **Feature Extraction**

Fitur tambahan yang akan diekstrak adalah hanya hari dari kolom ‘Date’. Keputusan untuk tidak membuat banyak fitur tambahan didasarkan pada efisiensi dan relevansi dataset yang sudah memadai. Dataset ini telah memiliki fitur utama yang mencakup informasi penting seperti kategori produk, ukuran, wilayah, dan data kuantitatif, sehingga menambah banyak fitur hanya akan meningkatkan kompleksitas tanpa memberikan dampak signifikan pada performansi model. Selain itu, terlalu banyak fitur berisiko menyebabkan overfitting, di mana model belajar pola spesifik pada data latih tetapi gagal menggeneralisasi pada data baru. Fitur tambahan yang telah dibuat, seperti ekstraksi bulan dan tipe hari (weekday/weekend), sudah cukup untuk menganalisis tren musiman dan pola perilaku terkait waktu, yang relevan dengan tujuan analisis. Pendekatan ini memungkinkan penggunaan sumber daya secara efisien sekaligus memastikan kualitas dan fokus analisis tetap terjaga.



Kode tersebut digunakan untuk menambahkan kolom baru ke dalam dataset, yang memetakan hari pada kolom **Date** menjadi kategori **Weekday** (Senin-Jumat) atau **Weekend** (Sabtu-Minggu). Fungsi **.dt.weekday** mengambil indeks hari dalam bentuk numerik (0 untuk Senin hingga 6 untuk Minggu). Kemudian, dengan menggunakan **apply** dan fungsi lambda, setiap nilai hari dipetakan menjadi **'Weekend'** jika nilainya 5 atau 6 (Sabtu atau Minggu), dan **'Weekday'** untuk hari lainnya. Hasilnya disimpan dalam kolom baru bernama **Day\_Type**. Alternatifnya, jika ingin nilai numerik, ekspresi logika **df\_new2['Date'].dt.weekday >= 5** menghasilkan nilai True/False, yang dikonversi menjadi **0** (Weekday) dan **1** (Weekend) dengan **.astype(int)**, lalu disimpan dalam kolom **Is\_Weekend**.

# PEMBUATAN GIT



Link : https://github.com/Rizqii1/Finpro\_Kelompok1