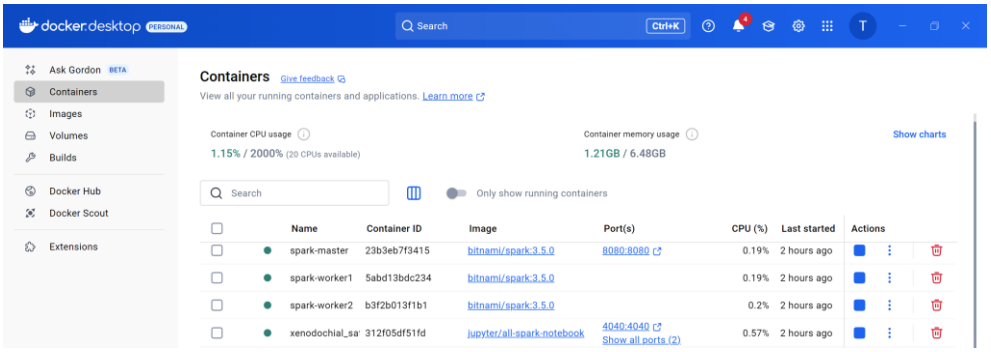
Nama : Triyana Dewi Fatmawati
NIM : 2241720206
Kelas : TI – 3D
Nomor : 21
Mata Kuliah : Big Data

# Tugas 10 – Data Cleaning dan Transformasi Menggunakan Apache Spark

## Persiapan Praktikum

- Menjalanlan Cluster

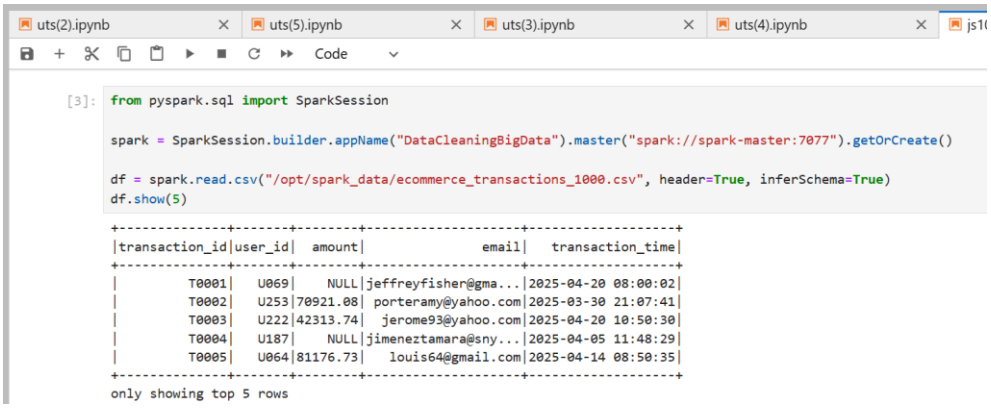



- Mount data

```
PS C:\Users\TRIYANA DF> docker exec -u root -it spark-master bash
root@23b3eb7f3415:/opt/bitnami/spark# mkdir -p /opt/spark_data
root@23b3eb7f3415:/opt/bitnami/spark# exit
exit
PS C:\Users\TRIYANA DF> docker cp ecommerce_transactions_1000.csv spark-master:/opt/spark_data/
Successfully copied 58.4kB to spark-master:/opt/spark_data/
PS C:\Users\TRIYANA DF> docker exec -u root -it spark-master bash
root@23b3eb7f3415:/opt/bitnami/spark# ls /opt/spark_data/
ecommerce_transactions_1000.csv
```

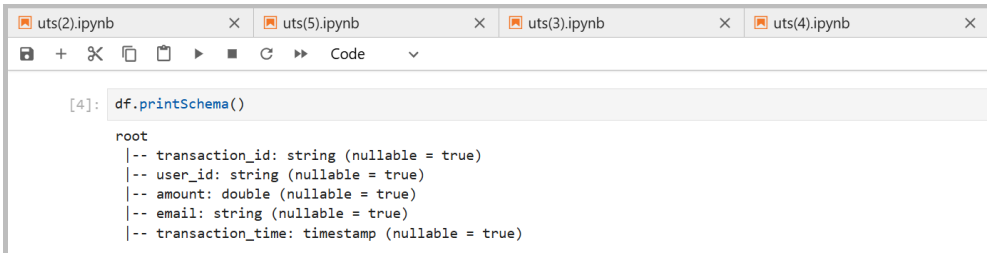
## Langkah Praktikum 1:

### 1. Load Data

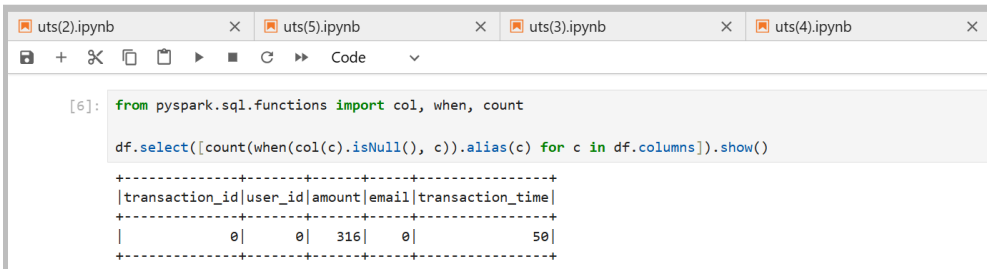


## 2. Inspeksi Data

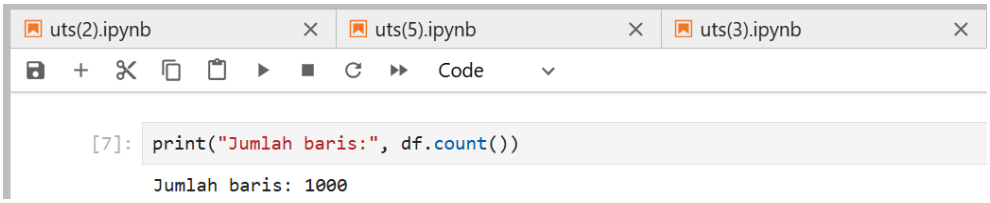
- Lihat struktur schema:




- Hitung missing values setiap kolom:




- Hitung jumlah total data:




## 3. Cleaning Data

a. Handling Missing Values

- Drop transaksi yang tidak memiliki transaction_time.
- Isi nilai kosong pada amount dengan 0 .

Hasil cleaning transaction_time dan amount.

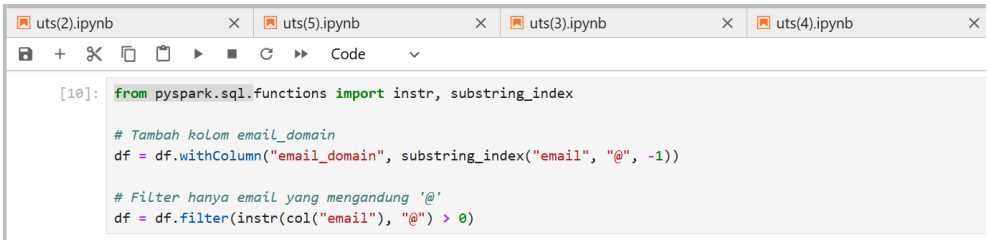
```
from pyspark.sql.functions import col, when, count

df.select([count(when(col(c).isNull(), c)).alias(c) for c in df.columns]).show()
```

```
+--------------+-------+------+-----+----------------+
|transaction_id|user_id|amount|email|transaction_time|
+--------------+-------+------+-----+----------------+
|             0|      0|     0|    0|               0|
+--------------+-------+------+-----+----------------+
```

b. Cleaning Format Email

- Buat kolom baru email_domain yang berisi domain email.
- Hapus transaksi yang email-nya tidak valid (tidak mengandung '@').




Sebelum dilakukan cleaning format email terdapat 83 transaksi yang format emailnya tidak valid.

```
df.select(count(when(~col("email").contains("@"), True)).alias("invalid_email_count")).show()
```

```
+-------------------+
|invalid_email_count|
+-------------------+
|                 83|
+-------------------+
```

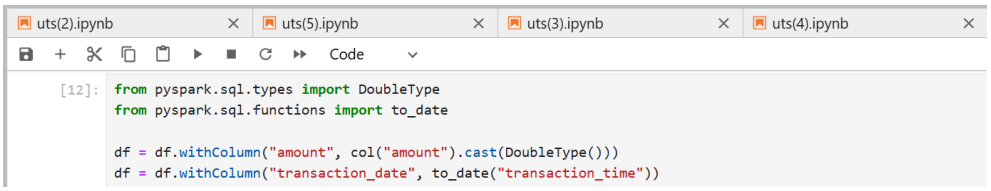
Setelah dilakukan cleaning data transaksi yang format emailnya tidak valid.

```
df.select(count(when(~col("email").contains("@"), True)).alias("invalid_email_count")).show()
```

```
+-------------------+
|invalid_email_count|
+-------------------+
|                  0|
+-------------------+
```

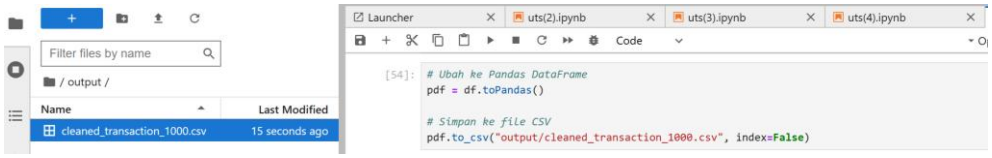
## 4. Transformasi Data

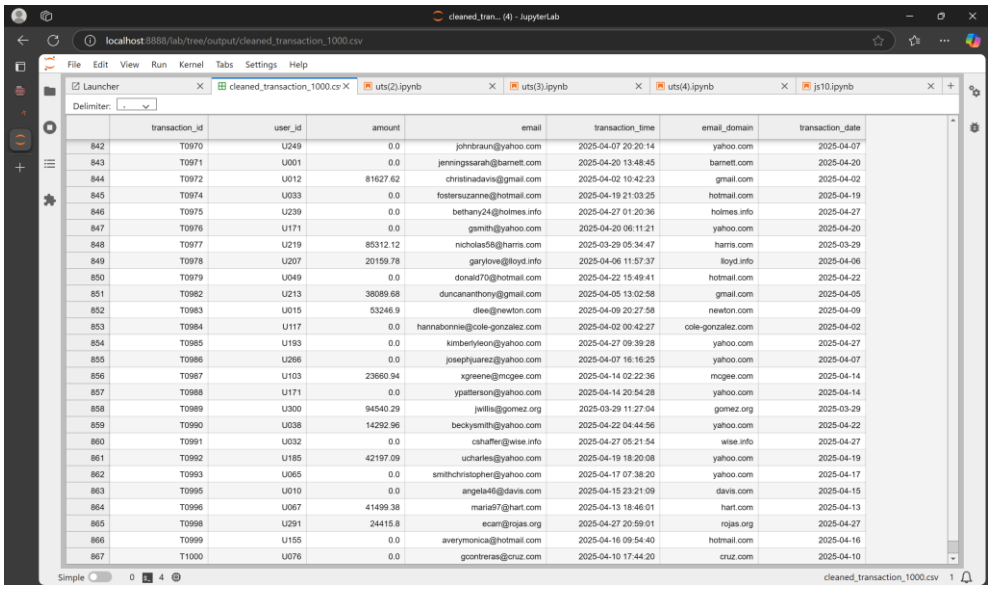
- Ubah kolom amount menjadi tipe DoubleType.
- Tambahkan kolom baru transaction_date dari transaction_time.



5. **Simpan Data Bersih**

- Simpan dataframe hasil cleaning ke file baru.




## Checklist Praktikum

- Dataset berhasil dibaca ✅
- Struktur schema dipahami ✅
- Missing value berhasil dihandle ✅
- Email valid berhasil diproses ✅
- Kolom baru email_domain dan transaction_date berhasil dibuat ✅
- Data bersih disimpan ulang ✅

**Pertanyaan dan Jawaban**

1. Berapa banyak data yang dibuang karena transaction_time kosong?

   **Jawaban:**

   Terdapat 50 data yang dibuang karena transaction_time kosong.

```
[6]: print("Jumlah baris:", df.count())
     Jumlah baris: 1000

[7]: df = df.dropna(subset=["transaction_time"])

[8]: print("Jumlah baris:", df.count())
     Jumlah baris: 950
```

```
+----------------+
|transaction_time|
+----------------+
|              50|
+----------------+
```

   Terlihat bahwa data yang awalnya 1000 setelah dilakukan drop transaksi yang tidak memiliki transaction_time menjadi 950, yang berarti data yang dibuang karena transaction_time kosong sebanyak 50 data.

2. Apakah semua data amount sudah bertipe numerik setelah cleaning?

   **Jawaban:**

   Iya, setelah cleaning semua data amount sudah bertipe numerik.

```
[11]: df.select(
          count(when(~col("amount").cast("double").isNotNull(), "amount")).alias("non_numeric_amount")
      ).show()
```

```
+------------------+
|non_numeric_amount|
+------------------+
|               299|
+------------------+
```

```
[12]: df = df.fillna({"amount": 0})
```

```
[13]: df.select(
          count(when(~col("amount").cast("double").isNotNull(), "amount")).alias("non_numeric_amount")
      ).show()
```

```
+------------------+
|non_numeric_amount|
+------------------+
|                 0|
+------------------+
```

3. Kenapa lebih baik memperbaiki email invalid sebelum menganalisis data transaksi?

   **Jawaban:**

   Memperbaiki email invalid sebelum menganalisis data transaksi akan memastikan hasil analisis data transaksi tersebut lebih akurat dan lengkap, karena analisis data transaksi sering melibatkan pengelompokan pelanggan berdasarkan email, misalnya frekuensi pembelian.

**Catatan Penting**

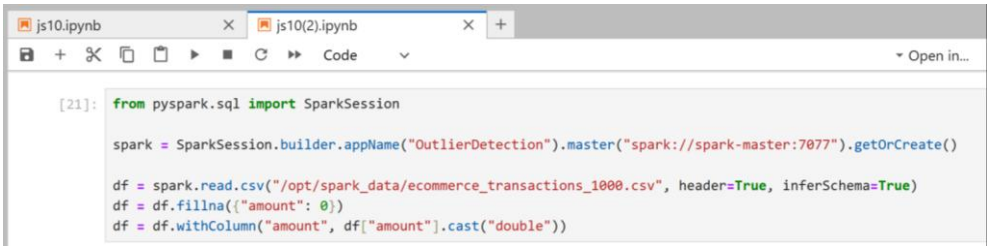
- **Handling data besar** : Proses cleaning jangan terlalu banyak chaining .filter() berurutan — lebih baik batch.
- **Ingat** : setiap operasi di Spark **lazy** — data baru di-load setelah .show() , .count() , atau .write()

## Praktikum 2: Deteksi Outlier Sederhana di Spark

### Kasus

Kita mau cek outlier pada kolom amount di data transaksi.

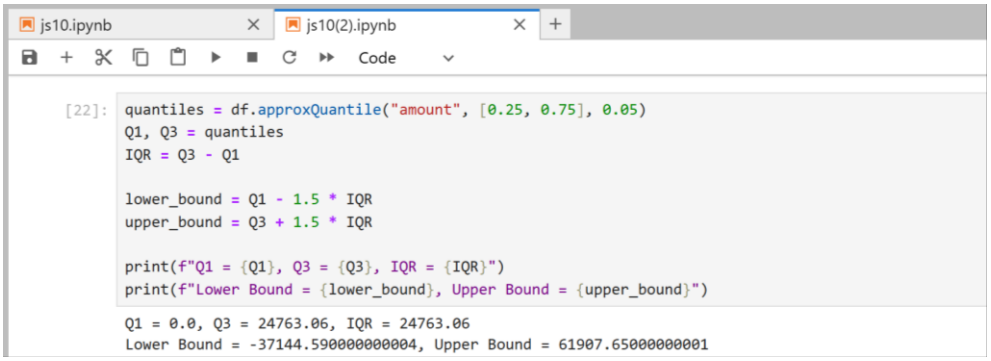
1. **Load Data**




2. **Hitung Statistik Dasar**

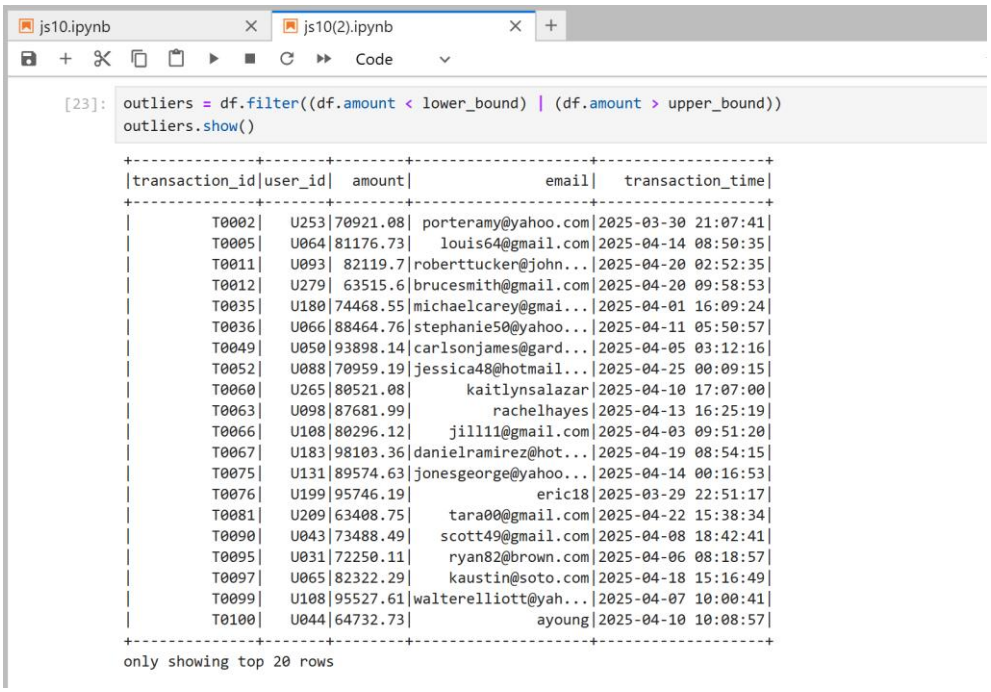
   Kita butuh:

   - Q1 (25th percentile)
   - Q3 (75th percentile)
   - IQR (Interquartile Range)



3. **Deteksi Outliers**

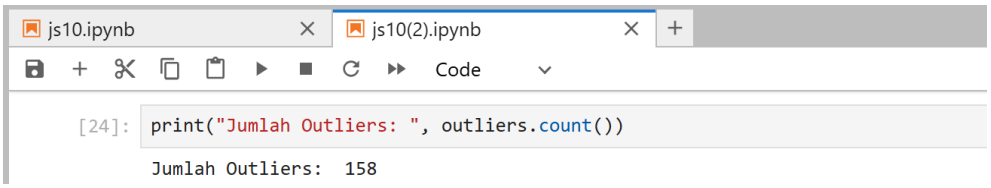
   Cari data amount yang **lebih kecil dari lower bound** atau **lebih besar dari upper bound.**



```
[23]: outliers = df.filter((df.amount < lower_bound) | (df.amount > upper_bound))
      outliers.show()
```

```
+--------------+-------+--------+--------------------+-------------------+
|transaction_id|user_id|  amount|               email|   transaction_time|
+--------------+-------+--------+--------------------+-------------------+
|         T0002|   U253|70921.08| porteramy@yahoo.com|2025-03-30 21:07:41|
|         T0005|   U064|81176.73|   louis64@gmail.com|2025-04-14 08:50:35|
|         T0011|   U093| 82119.7|roberttucker@john...|2025-04-20 02:52:35|
|         T0012|   U279| 63515.6|brucesmith@gmail.com|2025-04-20 09:58:53|
|         T0035|   U180|74468.55|michaelcarey@gmai...|2025-04-01 16:09:24|
|         T0036|   U066|88464.76|stephanie50@yahoo...|2025-04-11 05:50:57|
|         T0049|   U050|93898.14|carlsonjames@gard...|2025-04-05 03:12:16|
|         T0052|   U088|70959.19|jessica48@hotmail...|2025-04-25 00:09:15|
|         T0060|   U265|80521.08|      kaitlynsalazar|2025-04-10 17:07:00|
|         T0063|   U098|87681.99|         rachelhayes|2025-04-13 16:25:19|
|         T0066|   U108|80296.12|    jill11@gmail.com|2025-04-03 09:51:20|
|         T0067|   U183|98103.36|danielramirez@hot...|2025-04-19 08:54:15|
|         T0075|   U131|89574.63|jonesgeorge@yahoo...|2025-04-14 00:16:53|
|         T0076|   U199|95746.19|              eric18|2025-03-29 22:51:17|
|         T0081|   U209|63408.75|    tara00@gmail.com|2025-04-22 15:38:34|
|         T0090|   U043|73488.49|   scott49@gmail.com|2025-04-08 18:42:41|
|         T0095|   U031|72250.11|    ryan82@brown.com|2025-04-06 08:18:57|
|         T0097|   U065|82322.29|    kaustin@soto.com|2025-04-18 15:16:49|
|         T0099|   U108|95527.61|walterelliott@yah...|2025-04-07 10:00:41|
|         T0100|   U044|64732.73|              ayoung|2025-04-10 10:08:57|
+--------------+-------+--------+--------------------+-------------------+
only showing top 20 rows
```

4. **Hitung Berapa Banyak Outliers**



```
[24]: print("Jumlah Outliers: ", outliers.count())
```

```
Jumlah Outliers:  158
```

**Penjelasan Ringkas**

- **IQR (Interquartile Range)** = Jarak antara Q3 dan Q1 → rentang normal data.
- **Outlier** = Data di luar batas normal.
- Spark pakai .approxQuantile() karena menghitung quantile di dataset besar lebih cepat.

**Catatan Penting**

- Untuk dataset sangat besar, .approxQuantile() lebih cepat daripada .quantile() biasa.
- Outlier **tidak selalu error** — tergantung konteks (misal pembelian Rp 1.000.000 memang mungkin saja di e-commerce).
- Bisa lanjut dengan perintah: **drop** atau **flag** pada data outlier sesuai kebutuhan analisis.

**Tugas Praktikum**

1. Tampilkan top 5 transaksi dengan amount terbesar ?
2. Hitung **jumlah total transaksi** ?
3. Hitung **jumlah outlier** ?
4. Hitung **persentase outlier** terhadap seluruh transaksi ?

**Pengerjaan Tugas Praktikum**

**Pengerjaan menggunakan file ecommerce_transactions_1000.csv (Belum di cleaning)**

1. Top 5 transaksi dengan amount terbesar

```
# 1. Tampilkan top 5 transaksi dengan amount terbesar
top_5 = df.orderBy(col("amount").desc()).limit(5)
top_5.show()
```

```
+--------------+-------+--------+--------------------+-------------------+
|transaction_id|user_id|  amount|               email|   transaction_time|
+--------------+-------+--------+--------------------+-------------------+
|         T0437|   U233|99830.84|franklincraig@gma...|2025-03-31 01:07:47|
|         T0175|   U224|99410.65|natalie63@hotmail...|2025-04-10 14:15:20|
|         T0320|   U046|99399.22|bonniemack@yahoo.com|2025-04-05 21:15:08|
|         T0115|   U148|98589.66|          hillsophia|2025-03-29 20:30:24|
|         T0451|   U293|98343.68|  sean46@walters.com|2025-04-17 14:27:35|
+--------------+-------+--------+--------------------+-------------------+
```

2. Jumlah total transaksi

```
# 2. Hitung jumlah total transaksi
total_transaksi = df.count()
print(f"Jumlah total transaksi: {total_transaksi}")
```

```
Jumlah total transaksi: 1000
```

```
from pyspark.sql.functions import sum

# Hitung jumlah total nilai amount dari seluruh transaksi
total_amount = df.agg(sum("amount").alias("total_amount")).collect()[0]["total_amount"]
print(f"Jumlah total amount dari seluruh transaksi: {total_amount}")
```

```
Jumlah total amount dari seluruh transaksi: 19644994.95
```

3. Jumlah outlier

```
# 3. Hitung jumlah outlier
print("Jumlah Outliers: ", outliers.count())
```

```
Jumlah Outliers:  158
```

4. Persentase outlier terhadap seluruh transaksi

```
# 4. Hitung persentase outlier terhadap seluruh transaksi
jumlah_outlier = outliers.count()
persentase_outlier = (jumlah_outlier / total_transaksi) * 100
print(f"Persentase outlier: {persentase_outlier:.2f}%")
```

```
Persentase outlier: 15.80%
```

**Pengerjaan menggunakan file cleaned_transaction_1000.csv (Sudah di cleaning)**

1. Top 5 transaksi dengan amount terbesar

```
# 1. Tampilkan top 5 transaksi dengan amount terbesar
top_5 = df.orderBy(col("amount").desc()).limit(5)
top_5.show()
```

```
+--------------+-------+--------+--------------------+-------------------+------------+----------------+
|transaction_id|user_id|  amount|               email|   transaction_time|email_domain|transaction_date|
+--------------+-------+--------+--------------------+-------------------+------------+----------------+
|         T0437|   U233|99830.84|franklincraig@gma...|2025-03-31 01:07:47|   gmail.com|      2025-03-31|
|         T0175|   U224|99410.65|natalie63@hotmail...|2025-04-10 14:15:20| hotmail.com|      2025-04-10|
|         T0320|   U046|99399.22|bonniemack@yahoo.com|2025-04-05 21:15:08|   yahoo.com|      2025-04-05|
|         T0451|   U293|98343.68|  sean46@walters.com|2025-04-17 14:27:35| walters.com|      2025-04-17|
|         T0067|   U183|98103.36|danielramirez@hot...|2025-04-19 08:54:15| hotmail.com|      2025-04-19|
+--------------+-------+--------+--------------------+-------------------+------------+----------------+
```

2. Jumlah total transaksi

```
# 2. Hitung jumlah total transaksi
total_transaksi = df.count()
print(f"Jumlah total transaksi: {total_transaksi}")
```

```
Jumlah total transaksi: 867
```

```
from pyspark.sql.functions import sum

# Hitung jumlah total nilai amount dari seluruh transaksi
total_amount = df.agg(sum("amount").alias("total_amount")).collect()[0]["total_amount"]
print(f"Jumlah total amount dari seluruh transaksi: {total_amount}")
```

```
Jumlah total amount dari seluruh transaksi: 16922579.86999999
```

3. Jumlah outlier

```
# 3. Hitung jumlah outlier
quantiles = df.approxQuantile("amount", [0.25, 0.75], 0.05)
Q1, Q3 = quantiles
IQR = Q3 - Q1

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

print(f"Q1 = {Q1}, Q3 = {Q3}, IQR = {IQR}")
print(f"Lower Bound = {lower_bound}, Upper Bound = {upper_bound}")

outliers = df.filter((df.amount < lower_bound) | (df.amount > upper_bound))
print("Jumlah Outliers: ", outliers.count())
```

```
Q1 = 0.0, Q3 = 30029.83, IQR = 30029.83
Lower Bound = -45044.745, Upper Bound = 75074.57500000001
Jumlah Outliers:  86
```

4. Persentase outlier terhadap seluruh transaksi

```
# 4. Hitung persentase outlier terhadap seluruh transaksi
jumlah_outlier = outliers.count()
persentase_outlier = (jumlah_outlier / total_transaksi) * 100
print(f"Persentase outlier: {persentase_outlier:.2f}%")
```

```
Persentase outlier: 9.92%
```