



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

- Menjalankan Cluster

The screenshot shows the Docker Desktop interface. On the left, there's a sidebar with options like Ask Gordon, Containers (which is selected), Images, Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main area is titled 'Containers' with a 'Give feedback' link. It displays container usage statistics: Container CPU usage (1.15% / 2000%) and Container memory usage (1.21GB / 6.48GB). A 'Show charts' button is also present. Below these stats is a search bar and a checkbox for 'Only show running containers'. A table lists five containers:

	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
1	spark-master	23b3eb7f3415	bitnami/spark:3.5.0	8080:8080	0.19%	2 hours ago	[blue square, three dots, trash]
2	spark-worker1	5abd13bdc234	bitnami/spark:3.5.0		0.19%	2 hours ago	[blue square, three dots, trash]
3	spark-worker2	b3f2b013f1b1	bitnami/spark:3.5.0		0.2%	2 hours ago	[blue square, three dots, trash]
4	xenodochial_sa	312f05df51fd	jupyter/all-spark-notebook	4040:4040	0.57%	2 hours ago	[blue square, three dots, trash]

- 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

The screenshot shows a Jupyter Notebook with several tabs: uts(2).ipynb, uts(5).ipynb, uts(3).ipynb, uts(4).ipynb, and js1(. The active cell (cell 3) contains the following Python code using PySpark:

```
[3]: from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("DataCleaningBigData").master("spark://spark-master:7077").getOrCreate()
df = spark.read.csv("/opt/spark_data/ecommerce_transactions_1000.csv", header=True, inferSchema=True)
df.show(5)
```

Below the code, the output shows the first 5 rows of the DataFrame:

transaction_id	user_id	amount	email	transaction_time
T0001	U069	NULL	jeffreyfisher@gmail.com	2025-04-20 08:00:02
T0002	U253	70921.08	porteramy@yahoo.com	2025-03-30 21:07:41
T0003	U222	42313.74	jerome93@yahoo.com	2025-04-20 10:50:30
T0004	U187	NULL	jimeneztamarra@snyc.com	2025-04-05 11:48:29

only showing top 5 rows

2. Inspeksi Data

- Lihat struktur schema:

```
[4]: df.printSchema()
root
|-- transaction_id: string (nullable = true)
|-- user_id: string (nullable = true)
|-- amount: double (nullable = true)
|-- email: string (nullable = true)
|-- transaction_time: timestamp (nullable = true)
```

- Hitung missing values setiap kolom:

```
[6]: 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|   316|    0|           50|
+-----+-----+-----+
```

- Hitung jumlah total data:

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

3. Cleaning Data

a. Handling Missing Values

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

```
[9]: df = df.dropna(subset=["transaction_time"])
df = df.fillna({"amount": 0})
```

Hasil cleaning transaction_time dan amount.

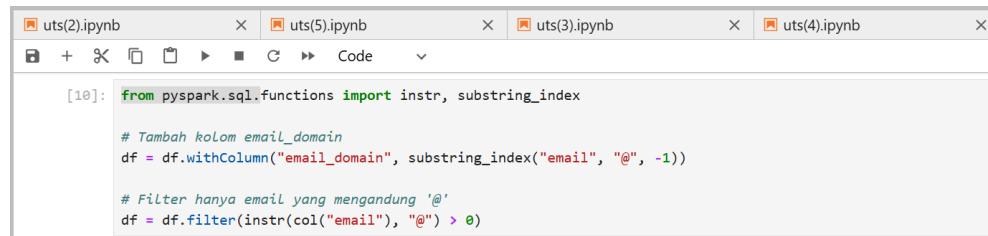
```
from pyspark.sql.functions import col, when, count

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 '@').



```
[10]: from pyspark.sql.functions import instr, substring_index

# Tambah kolom email_domain
df = df.withColumn("email_domain", substring_index("email", "@", -1))

# Filter hanya email yang mengandung @@
df = df.filter(instr(col("email"), "@") > 0)
```

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|
+-----+
```

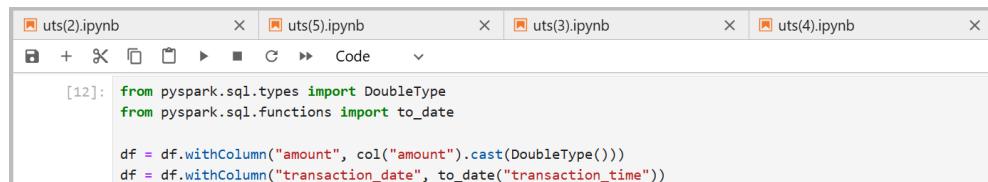
Setelah dilakukan cleaning data transaksi yang format emailnya tidak valid.

```
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.



```
[12]: from pyspark.sql.types import DoubleType
from pyspark.sql.functions import to_date

df = df.withColumn("amount", col("amount").cast(DoubleType()))
df = df.withColumn("transaction_date", to_date("transaction_time"))
```

5. Simpan Data Bersih

- Simpan dataframe hasil cleaning ke file baru.

The screenshot shows a JupyterLab interface with several tabs open. One tab contains Python code to convert a DataFrame to a PDF and then save it as a CSV:

```
[54]: # Ubah ke Pandas DataFrame
pdf = df.toPandas()

# Simpan ke file CSV
pdf.to_csv("output/cleaned_transaction_1000.csv", index=False)
```

The other tab shows a preview of the 'cleaned_transaction_1000.csv' file, which contains transaction data with columns: transaction_id, user_id, amount, email, transaction_time, email_domain, and transaction_date. The data is as follows:

transaction_id	user_id	amount	email	transaction_time	email_domain	transaction_date	
842	T0970	U249	0.0	johnbraun@yahoo.com	2025-04-07 20:14	yahoo.com	2025-04-07
843	T0971	U001	0.0	jemingsarah@bamett.com	2025-04-20 10:48:45	bamett.com	2025-04-20
844	T0972	U012	81627.62	christinadavis@gmail.com	2025-04-02 10:42:23	gmail.com	2025-04-02
845	T0974	U033	0.0	fostersuzanne@hotmail.com	2025-04-19 21:03:25	hotmail.com	2025-04-19
846	T0975	U239	0.0	bethany24@holmes.info	2025-04-27 01:20:36	holmes.info	2025-04-27
847	T0976	U171	0.0	gsmith@yahoo.com	2025-04-20 09:11:21	yahoo.com	2025-04-20
848	T0977	U219	85312.12	nicholas58@harris.com	2025-03-29 09:34:47	harris.com	2025-03-29
849	T0978	U207	20159.78	garylove@lloyd.info	2025-04-06 11:57:37	lloyd.info	2025-04-06
850	T0979	U049	0.0	donald70@hotmail.com	2025-04-22 15:49:41	hotmail.com	2025-04-22
851	T0982	U213	38089.68	duncananthony@gmail.com	2025-04-05 13:02:58	gmail.com	2025-04-05
852	T0983	U015	53246.9	dee@newton.com	2025-04-09 20:27:58	newton.com	2025-04-09
853	T0984	U117	0.0	hannabonnie@cole-gonzalez.com	2025-04-02 09:42:27	cole-gonzalez.com	2025-04-02
854	T0985	U193	0.0	kimberlyleon@yahoo.com	2025-04-27 09:39:28	yahoo.com	2025-04-27
855	T0986	U266	0.0	josephjuarez@yahoo.com	2025-04-07 16:16:25	yahoo.com	2025-04-07
856	T0987	U103	23660.94	xgrehne@gomez.com	2025-04-14 02:22:36	gomez.com	2025-04-14
857	T0988	U171	0.0	ypatterson@yahoo.com	2025-04-14 20:54:28	yahoo.com	2025-04-14
858	T0989	U300	94540.29	jwillis@gomez.org	2025-03-29 11:27:04	gomez.org	2025-03-29
859	T0990	U038	14292.96	beckysmith@yahoo.com	2025-04-22 09:44:56	yahoo.com	2025-04-22
860	T0991	U032	0.0	cahaffier@wise.info	2025-04-27 09:21:54	wise.info	2025-04-27
861	T0992	U185	42197.09	ucharles@yahoo.com	2025-04-19 18:20:08	yahoo.com	2025-04-19
862	T0993	U065	0.0	smithchristopher@yahoo.com	2025-04-17 07:38:20	yahoo.com	2025-04-17
863	T0995	U010	0.0	angelat4@davies.com	2025-04-15 23:21:09	davies.com	2025-04-15
864	T0996	U067	41499.38	maria97@hart.com	2025-04-13 18:46:01	hart.com	2025-04-13
865	T0998	U291	24415.8	ecam@rjqas.org	2025-04-27 20:59:01	rjas.org	2025-04-27
866	T0999	U155	0.0	averymonica@hotmail.com	2025-04-16 09:54:40	hotmail.com	2025-04-16
867	T1000	U076	0.0	gocontreras@cruz.com	2025-04-10 11:44:20	cruz.com	2025-04-10

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
+-----+
|transaction_time| [7]: df = df.dropna(subset=["transaction_time"])
+-----+
|      50| [8]: print("Jumlah baris:", df.count())
+-----+           Jumlah baris: 950
```

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

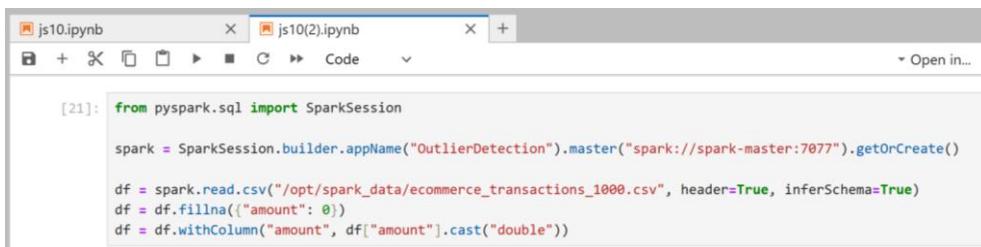
- **Handling data besar** : Proses cleaning jangan terlalu banyak chaining `.filter()` berurutan — lebih baik batch.
- **Ingat** : setiap operasi di Spark **lazy** — data baru di-load setelah `.show()` , `.count()` , atau `.write()`

Praktikum 2: Deteksi Outlier Sederhana di Spark

Kasus

Kita mau cek outlier pada kolom `amount` di data transaksi.

1. Load Data

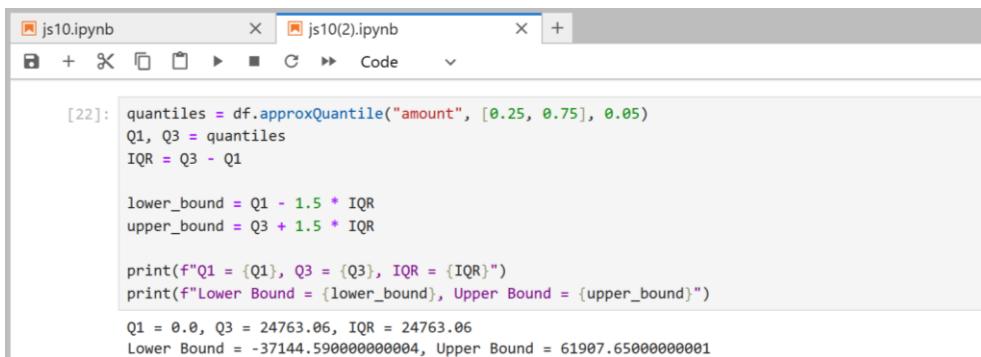


```
[21]: from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("OutlierDetection").master("spark://spark-master:7077").getOrCreate()
df = spark.read.csv("/opt/spark_data/ecommerce_transactions_1000.csv", header=True, inferSchema=True)
df = df.fillna({ "amount": 0 })
df = df.withColumn("amount", df["amount"].cast("double"))
```

2. Hitung Statistik Dasar

Kita butuh:

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



```
[22]: 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}")

Q1 = 0.0, Q3 = 24763.06, IQR = 24763.06
Lower Bound = -37144.59000000004, Upper Bound = 61907.65000000001
```

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	U189	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	88296.12	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|franklin craig@gmail.com|2025-03-31 01:07:47| gmail.com| 2025-03-31|
| T0175| U224|99410.65|natalie63@hotmail.com|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| sean4@walters.com|2025-04-17 14:27:35| walters.com| 2025-04-17|
| T0067| U183|98103.36|danielramirez@hotmail.com|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%
```