OPTIMIZING HIGH-PERFORMANCE DATA PROCESSING FOR LARGE-SCALE WEB CRAWLERS

PG Mall

Group D

- WAN NUR SOFEA BINTI MOHD HASBULLAH (A22EC0115)
- LOW YING XI (A22EC0187)
- MUHAMMAD ARIFF DANISH BIN HASHNAN (A22EC0204)
- MUHAMMAD IMAN FIRDAUS BIN BAHARUDDIN (A22EC0216)

List of Contents

- Introduction
- Targeted website & Data Field
- System Architecture
- Tools & Frameworks
- Data Collection
- Data Processing
- Optimization Techniques
- Performance Evaluation
- Challenges & Limitations
- Conclusion

Introduction

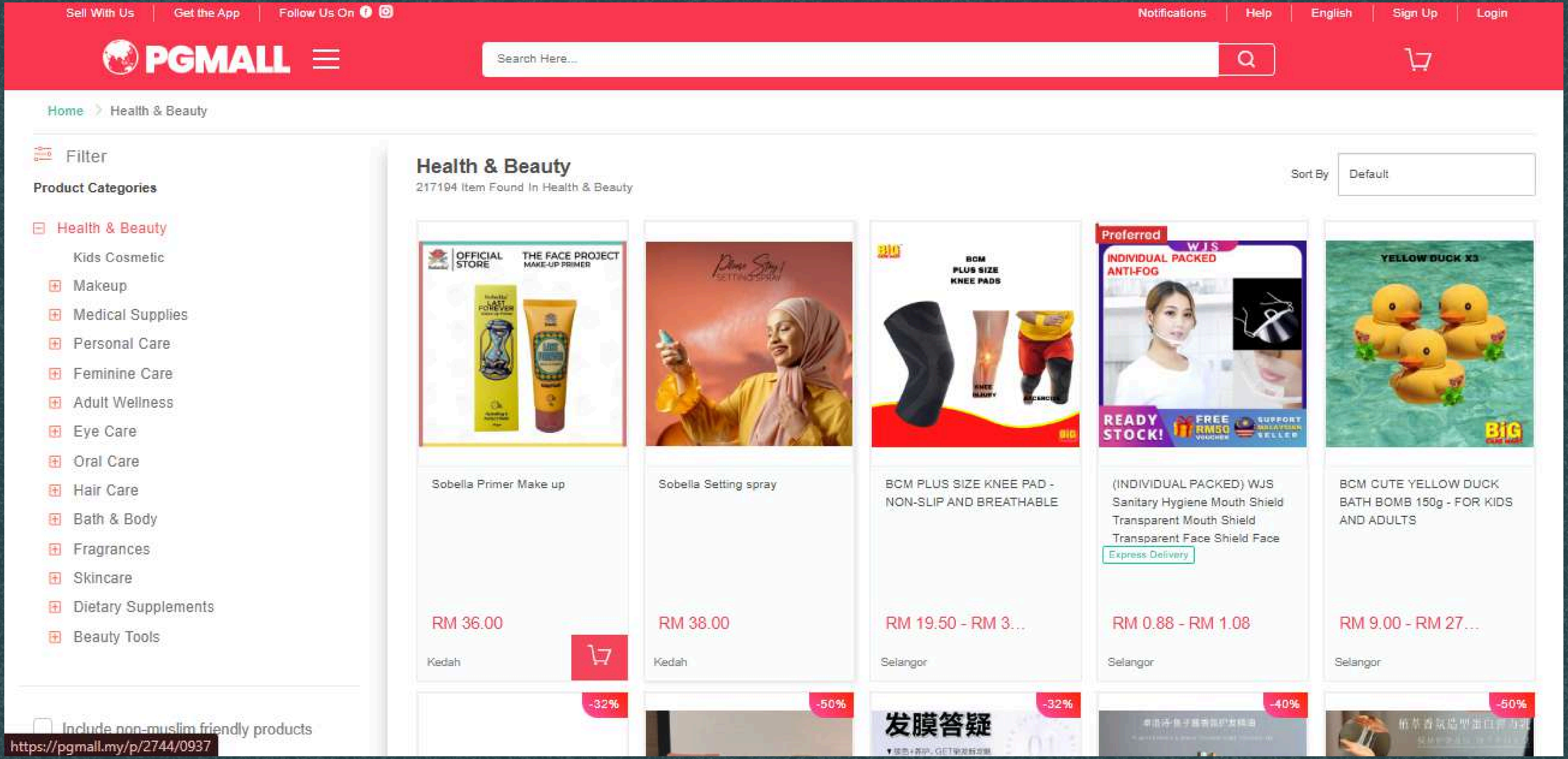
This project develops a scalable solution for extracting, cleaning, and analyzing real-world e-commerce data, focusing on web crawling, data preprocessing, and evaluating text-processing libraries under performance constraints.

Objectives:

- Build a web crawler that extracts >=100,000 data from the PG Mall
- Perform text processing and data cleaning on the scraped data using text normalisation and noise removal techniques.
- Compare text processing libraries based on key performance indicators
- Apply high-performance computing techniques to optimise the text processing pipeline.
- Evaluate the system’s performance, identifying bottlenecks and discussing improvements made.

Targeted Website & Data Field

PGMALL



Sell With Us | Get the App | Follow Us On

Notifications | Help | English | Sign Up | Login

PGMALL

Search Here...

Home > Health & Beauty

Filter

Product Categories

- Health & Beauty
  - Kids Cosmetic
  - Makeup
  - Medical Supplies
  - Personal Care
  - Feminine Care
  - Adult Wellness
  - Eye Care
  - Oral Care
  - Hair Care
  - Bath & Body
  - Fragrances
  - Skincare
  - Dietary Supplements
  - Beauty Tools

□ Include non-muslim friendly products

https://pgmall.my/p/2744/0937

Health & Beauty

217194 Item Found In Health & Beauty

Sort By: Default

Sobella Primer Make up
RM 36.00
Kedah

Sobella Setting spray
RM 38.00
Kedah

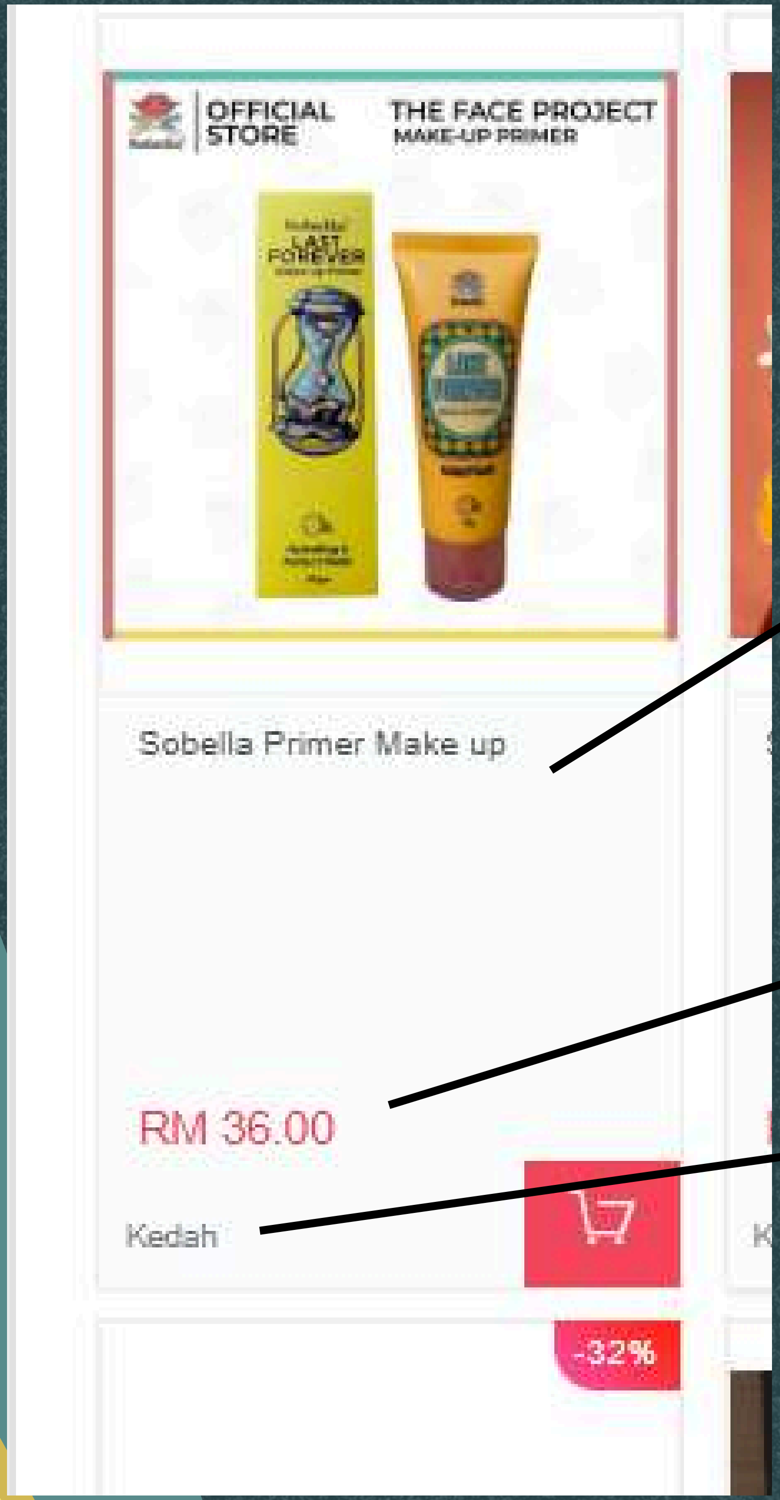
BCM PLUS SIZE KNEE PAD - NON-SLIP AND BREATHABLE
RM 19.50 - RM 3...
Selangor

Preferred
(INDIVIDUAL PACKED) WJS Sanitary Hygiene Mouth Shield Transparent Mouth Shield Transparent Face Shield Face
Express Delivery
RM 0.88 - RM 1.08
Selangor

BCM CUTE YELLOW DUCK BATH BOMB 150g - FOR KIDS AND ADULTS
RM 9.00 - RM 27...
Selangor

-32% -50% -32% -40% -50%

Targeted Website & Data Field



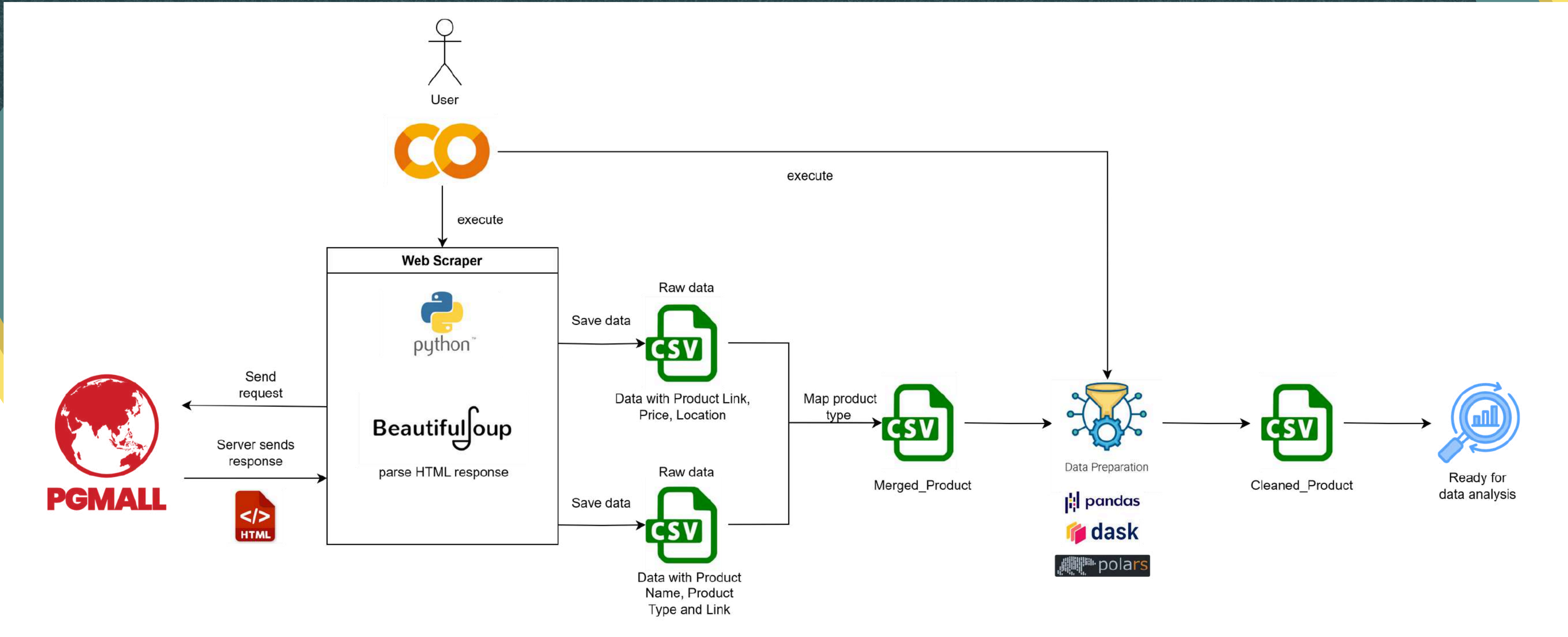
Sobella Primer Make up

RM 36.00

Kedah

- **Product Name**
- **Link** *extracted from HTML
- **Price**
- **Location**
- **Product Type** *added during mapping phase

System Architecture



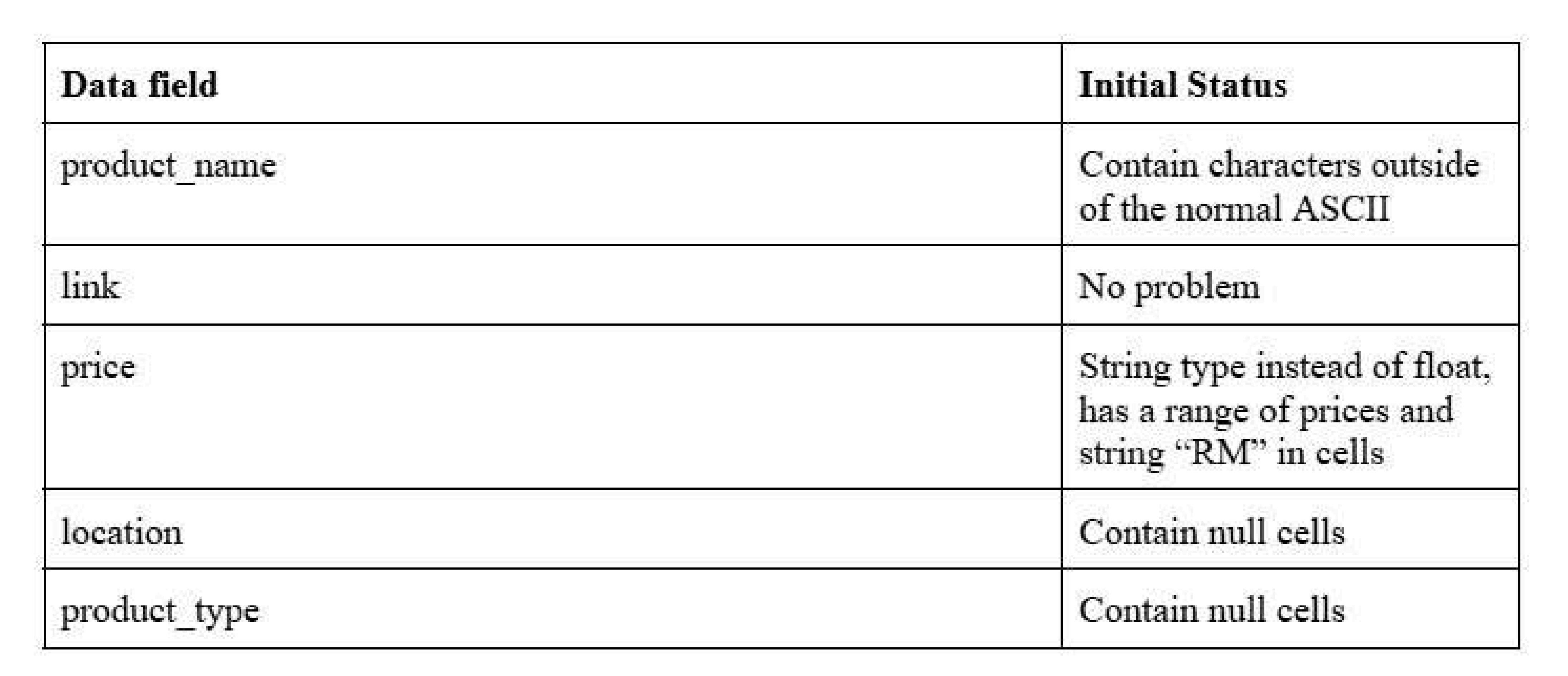
Tools & Frameworks

BeautifulSoup

Requests
http for humans

Data Collection

Data Field and Method

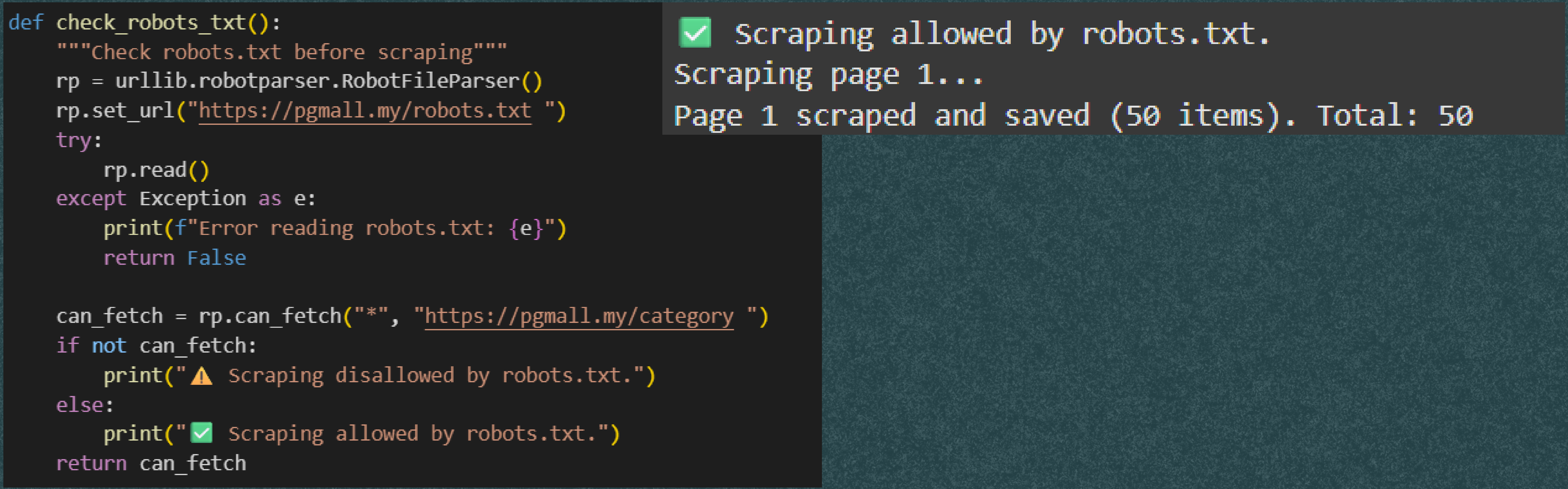


Data field	Initial Status
product_name	Contain characters outside of the normal ASCII
link	No problem
price	String type instead of float, has a range of prices and string “RM” in cells
location	Contain null cells
product_type	Contain null cells

- Sequential Pagination via While Loop
- Retry-Logic and Concurrency Control (max_workers = 5)
- Multithreaded Implementation (concurrent.futures.ThreadPoolExecutor)

Ethical Consideration

- Respects Retry-After headers during rate limiting
- Uses a generic User-Agent string to mimic a browser
- Manually checked robot.txt and confirmed using code

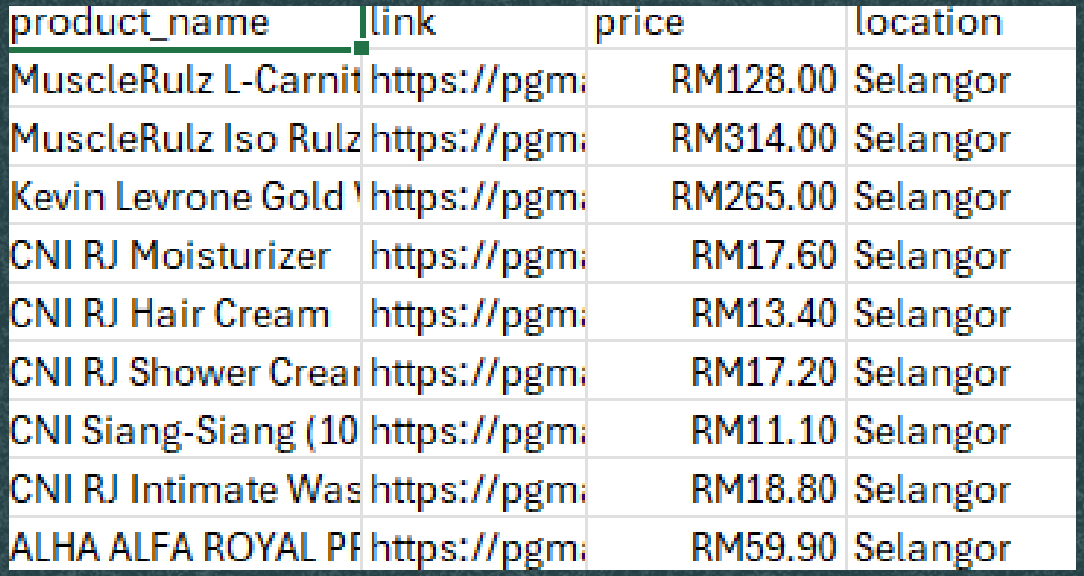


```
def check_robots_txt():
    """Check robots.txt before scraping"""
    rp = urllib.robotparser.RobotFileParser()
    rp.set_url("https://pgmall.my/robots.txt ")
    try:
        rp.read()
    except Exception as e:
        print(f"Error reading robots.txt: {e}")
        return False

    can_fetch = rp.can_fetch("*", "https://pgmall.my/category ")
    if not can_fetch:
        print("⚠️ Scraping disallowed by robots.txt.")
    else:
        print("✅ Scraping allowed by robots.txt.")
    return can_fetch
```

```
✅ Scraping allowed by robots.txt.
Scraping page 1...
Page 1 scraped and saved (50 items). Total: 50
```

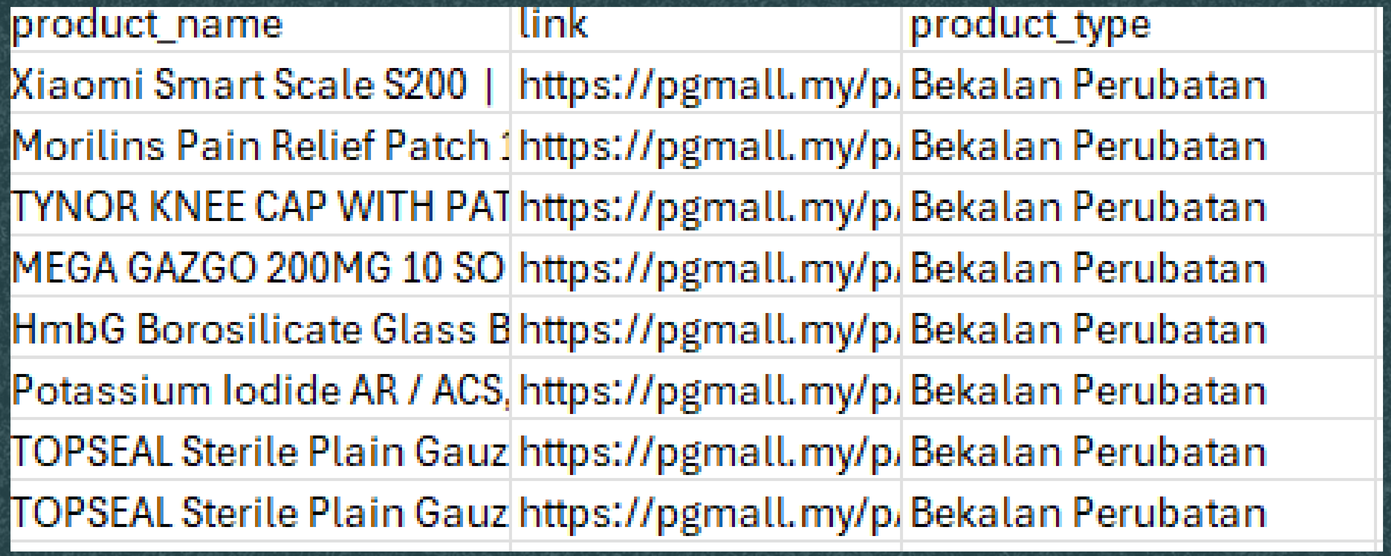
Data Processing

MAPPING PHASE



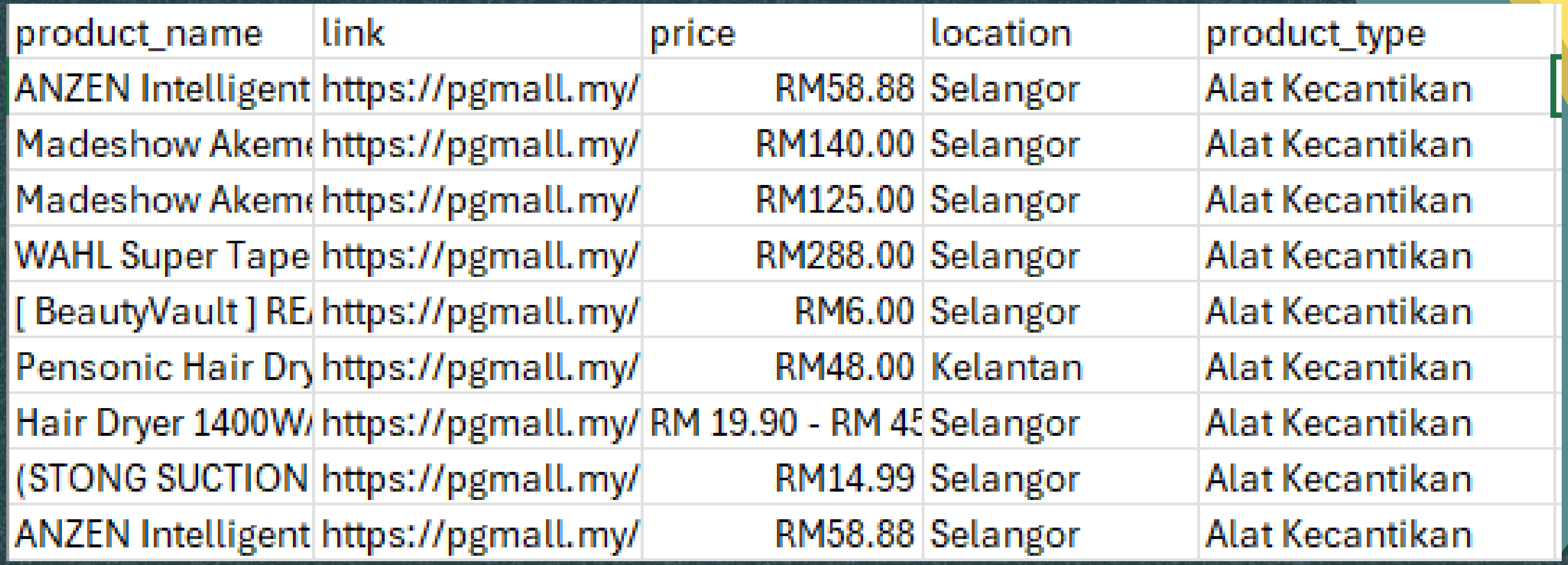
product_name	link	price	location
MuscleRulz L-Carnit	https://pgma	RM128.00	Selangor
MuscleRulz Iso Rulz	https://pgma	RM314.00	Selangor
Kevin Levrone Gold \	https://pgma	RM265.00	Selangor
CNI RJ Moisturizer	https://pgma	RM17.60	Selangor
CNI RJ Hair Cream	https://pgma	RM13.40	Selangor
CNI RJ Shower Crear	https://pgma	RM17.20	Selangor
CNI Siang-Siang (10	https://pgma	RM11.10	Selangor
CNI RJ Intimate Was	https://pgma	RM18.80	Selangor
ALHA ALFA ROYAL PI	https://pgma	RM59.90	Selangor

Item_list.csv



product_name	link	product_type
Xiaomi Smart Scale S200 |	https://pgmall.my/p/	Bekalan Perubatan
Morilins Pain Relief Patch 1	https://pgmall.my/p/	Bekalan Perubatan
TYNOR KNEE CAP WITH PAT	https://pgmall.my/p/	Bekalan Perubatan
MEGA GAZGO 200MG 10 SO	https://pgmall.my/p/	Bekalan Perubatan
HmbG Borosilicate Glass B	https://pgmall.my/p/	Bekalan Perubatan
Potassium Iodide AR / ACS,	https://pgmall.my/p/	Bekalan Perubatan
TOPSEAL Sterile Plain Gauz	https://pgmall.my/p/	Bekalan Perubatan
TOPSEAL Sterile Plain Gauz	https://pgmall.my/p/	Bekalan Perubatan

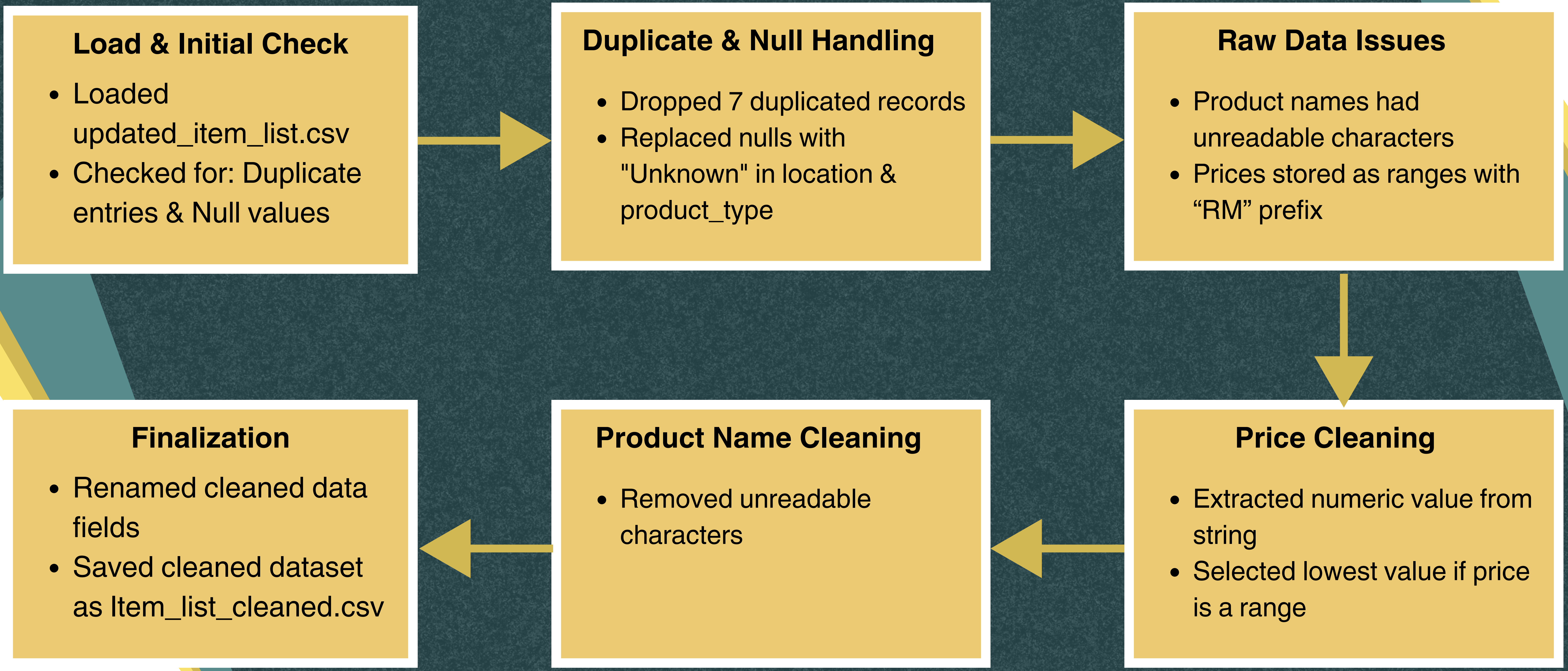
merged_product_list.csv



product_name	link	price	location	product_type
ANZEN Intelligent	https://pgmall.my/	RM58.88	Selangor	Alat Kecantikan
Madeshow Akeme	https://pgmall.my/	RM140.00	Selangor	Alat Kecantikan
Madeshow Akeme	https://pgmall.my/	RM125.00	Selangor	Alat Kecantikan
WAHL Super Tape	https://pgmall.my/	RM288.00	Selangor	Alat Kecantikan
[BeautyVault] RE	https://pgmall.my/	RM6.00	Selangor	Alat Kecantikan
Pensonic Hair Dry	https://pgmall.my/	RM48.00	Kelantan	Alat Kecantikan
Hair Dryer 1400W/	https://pgmall.my/	RM 19.90 - RM 45	Selangor	Alat Kecantikan
(STONG SUCTION	https://pgmall.my/	RM14.99	Selangor	Alat Kecantikan
ANZEN Intelligent	https://pgmall.my/	RM58.88	Selangor	Alat Kecantikan

updated_item_list.csv

DATA CLEANING



Load & Initial Check

- Loaded updated_item_list.csv
- Checked for: Duplicate entries & Null values

Duplicate & Null Handling

- Dropped 7 duplicated records
- Replaced nulls with "Unknown" in location & product_type

Raw Data Issues

- Product names had unreadable characters
- Prices stored as ranges with “RM” prefix

Price Cleaning

- Extracted numeric value from string
- Selected lowest value if price is a range

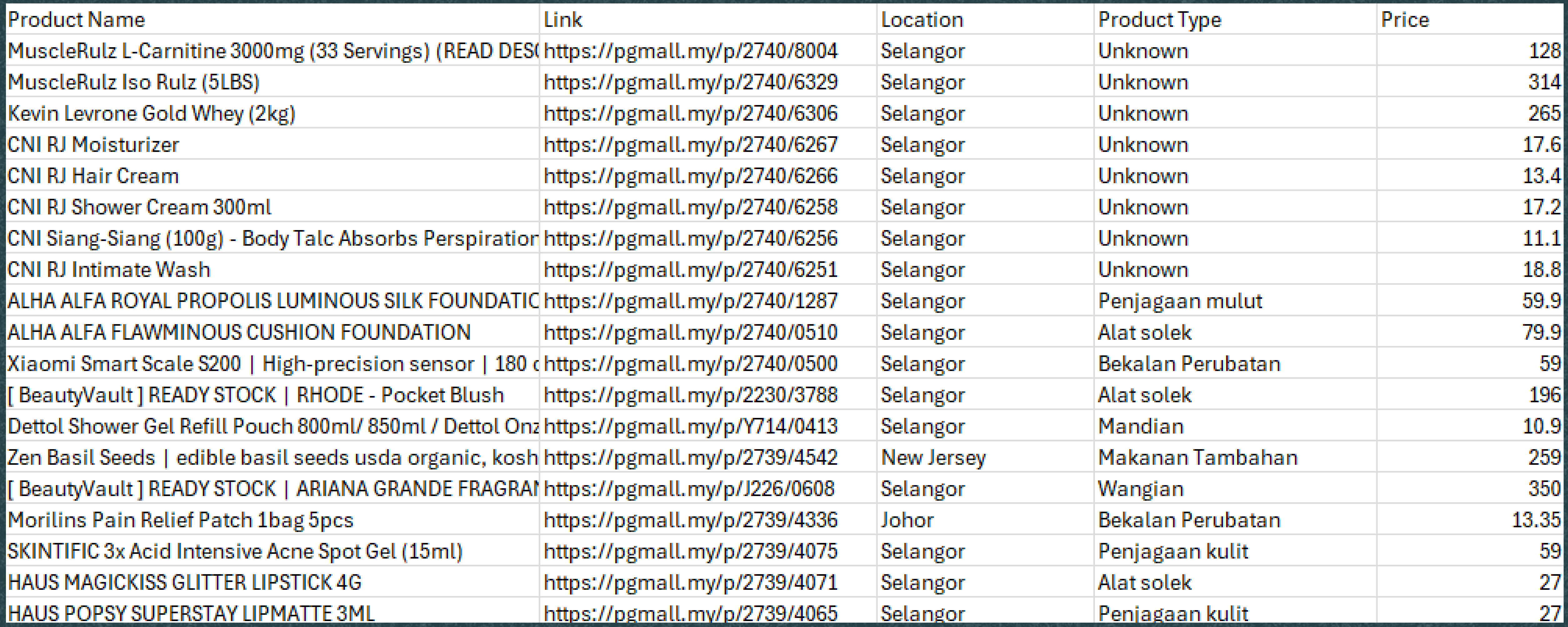
Product Name Cleaning

- Removed unreadable characters

Finalization

- Renamed cleaned data fields
- Saved cleaned dataset as Item_list_cleaned.csv

DATA STRUCTURE



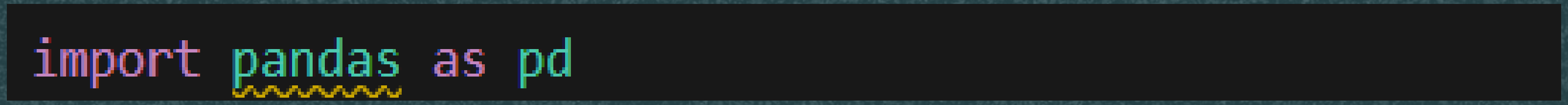
Product Name	Link	Location	Product Type	Price
MuscleRulz L-Carnitine 3000mg (33 Servings) (READ DES(	https://pgmall.my/p/2740/8004	Selangor	Unknown	128
MuscleRulz Iso Rulz (5LBS)	https://pgmall.my/p/2740/6329	Selangor	Unknown	314
Kevin Levrone Gold Whey (2kg)	https://pgmall.my/p/2740/6306	Selangor	Unknown	265
CNI RJ Moisturizer	https://pgmall.my/p/2740/6267	Selangor	Unknown	17.6
CNI RJ Hair Cream	https://pgmall.my/p/2740/6266	Selangor	Unknown	13.4
CNI RJ Shower Cream 300ml	https://pgmall.my/p/2740/6258	Selangor	Unknown	17.2
CNI Siang-Siang (100g) - Body Talc Absorbs Perspiration	https://pgmall.my/p/2740/6256	Selangor	Unknown	11.1
CNI RJ Intimate Wash	https://pgmall.my/p/2740/6251	Selangor	Unknown	18.8
ALHA ALFA ROYAL PROPOLIS LUMINOUS SILK FOUNDATIC	https://pgmall.my/p/2740/1287	Selangor	Penjagaan mulut	59.9
ALHA ALFA FLAWMINOUS CUSHION FOUNDATION	https://pgmall.my/p/2740/0510	Selangor	Alat solek	79.9
Xiaomi Smart Scale S200 | High-precision sensor | 180 c	https://pgmall.my/p/2740/0500	Selangor	Bekalan Perubatan	59
[BeautyVault] READY STOCK | RHODE - Pocket Blush	https://pgmall.my/p/2230/3788	Selangor	Alat solek	196
Dettol Shower Gel Refill Pouch 800ml/ 850ml / Dettol Onz	https://pgmall.my/p/Y714/0413	Selangor	Mandian	10.9
Zen Basil Seeds | edible basil seeds usda organic, kosh	https://pgmall.my/p/2739/4542	New Jersey	Makanan Tambahan	259
[BeautyVault] READY STOCK | ARIANA GRANDE FRAGRAI	https://pgmall.my/p/J226/0608	Selangor	Wangian	350
Morilins Pain Relief Patch 1bag 5pcs	https://pgmall.my/p/2739/4336	Johor	Bekalan Perubatan	13.35
SKINTIFIC 3x Acid Intensive Acne Spot Gel (15ml)	https://pgmall.my/p/2739/4075	Selangor	Penjagaan kulit	59
HAUS MAGICKISS GLITTER LIPSTICK 4G	https://pgmall.my/p/2739/4071	Selangor	Alat solek	27
HAUS POPSY SUPERSTAY LIPMATTE 3ML	https://pgmall.my/p/2739/4065	Selangor	Penjagaan kulit	27

Cleaned dataset with product name, link location, price and product type

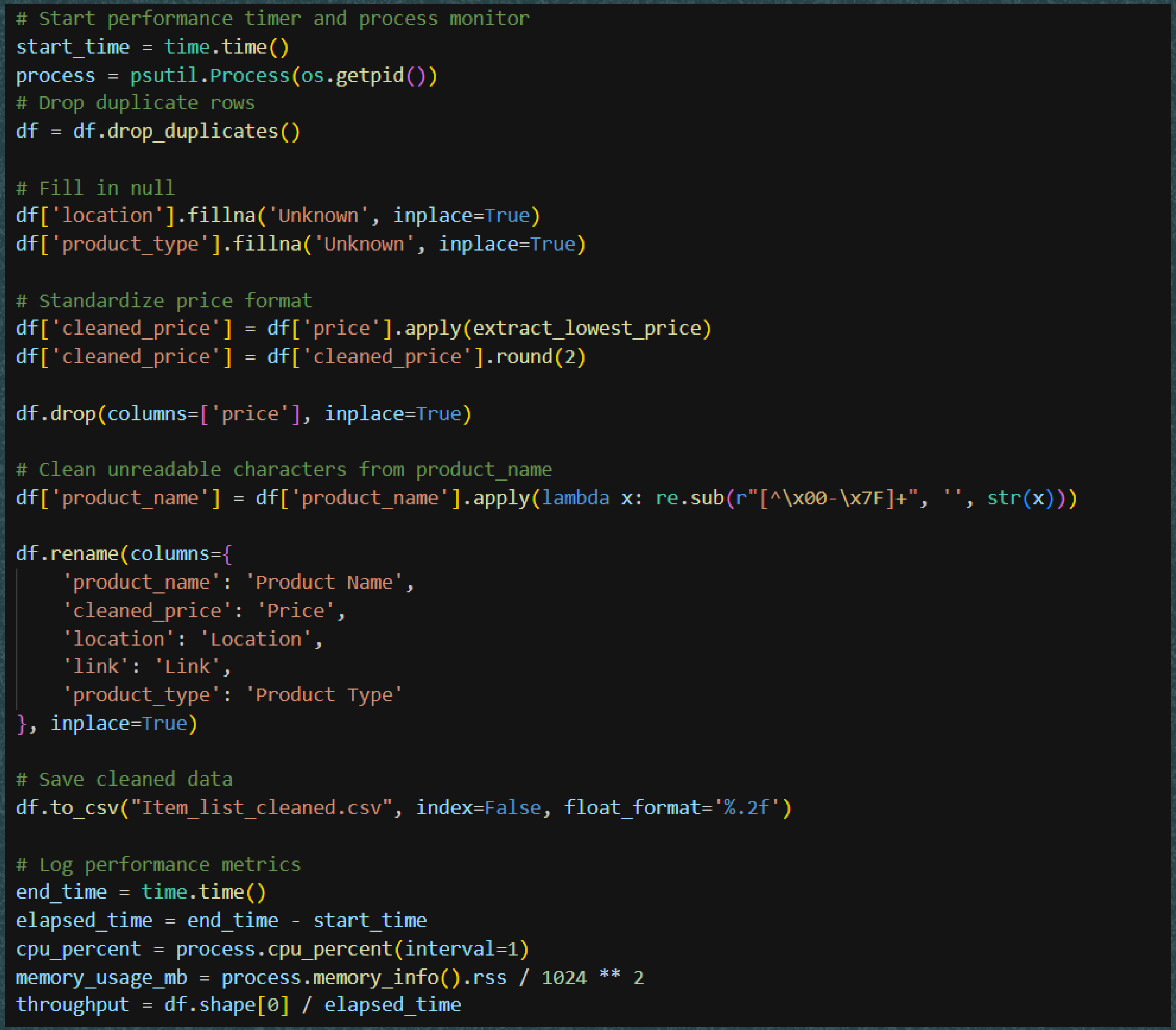
Optimization Techniques

Pandas

- Uses standard Pandas operations
- Processes data in-memory as a single chunk
- Applies functions row-by-row using .apply()
- Shows typical performance characteristics of unoptimized Pandas code



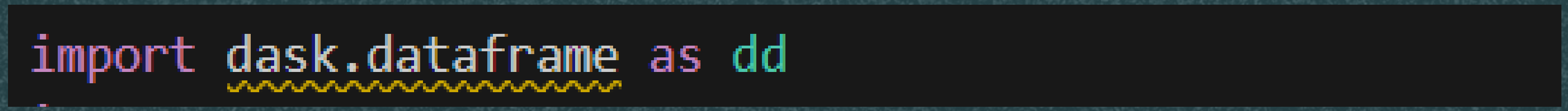
```python
import pandas as pd
```



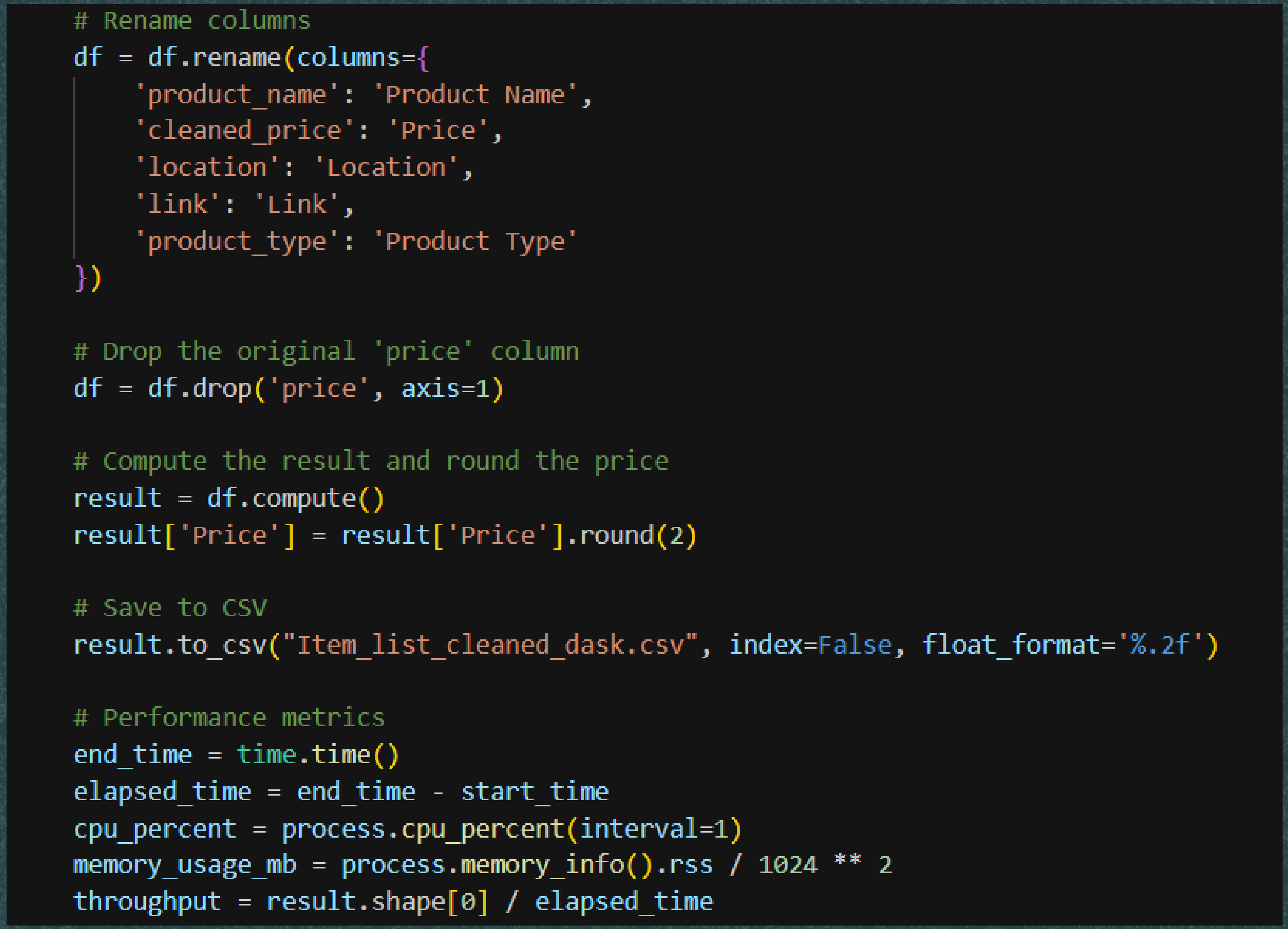
```python
# Start performance timer and process monitor
start_time = time.time()
process = psutil.Process(os.getpid())
# Drop duplicate rows
df = df.drop_duplicates()

# Fill in null
df['location'].fillna('Unknown', inplace=True)
df['product_type'].fillna('Unknown', inplace=True)

# Standardize price format
df['cleaned_price'] = df['price'].apply(extract_lowest_price)
df['cleaned_price'] = df['cleaned_price'].round(2)

df.drop(columns=['price'], inplace=True)

# Clean unreadable characters from product_name
df['product_name'] = df['product_name'].apply(lambda x: re.sub(r"[^\x00-\x7F]+", '', str(x)))

df.rename(columns={
    'product_name': 'Product Name',
    'cleaned_price': 'Price',
    'location': 'Location',
    'link': 'Link',
    'product_type': 'Product Type'
}, inplace=True)

# Save cleaned data
df.to_csv("Item_list_cleaned.csv", index=False, float_format='%.2f')

# Log performance metrics
end_time = time.time()
elapsed_time = end_time - start_time
cpu_percent = process.cpu_percent(interval=1)
memory_usage_mb = process.memory_info().rss / 1024 ** 2
throughput = df.shape[0] / elapsed_time
```

Dask

- Leverages Dask's parallel processing capabilities
- Uses lazy evaluation with explicit computation triggers
- Processes data in partitions for memory efficiency
- Demonstrates distributed computing benefits



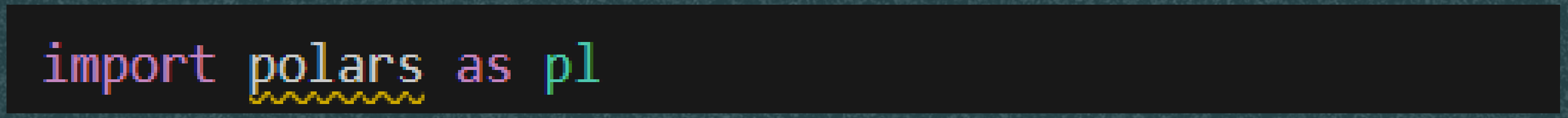
```
import dask.dataframe as dd
```



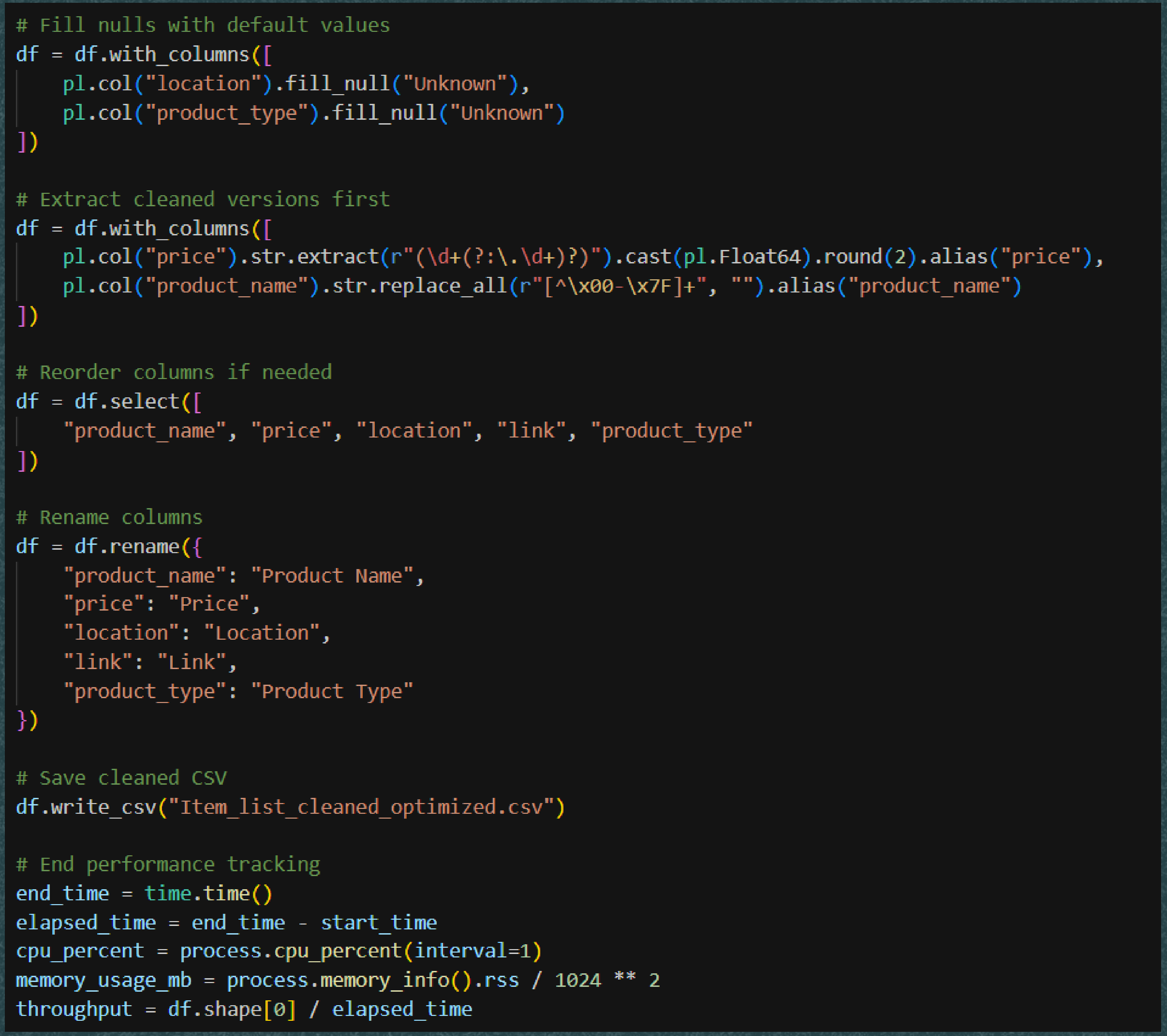
```
    # Rename columns
    df = df.rename(columns={
        'product_name': 'Product Name',
        'cleaned_price': 'Price',
        'location': 'Location',
        'link': 'Link',
        'product_type': 'Product Type'
    })

    # Drop the original 'price' column
    df = df.drop('price', axis=1)

    # Compute the result and round the price
    result = df.compute()
    result['Price'] = result['Price'].round(2)

    # Save to CSV
    result.to_csv("Item_list_cleaned_dask.csv", index=False, float_format='%.2f')

    # Performance metrics
    end_time = time.time()
    elapsed_time = end_time - start_time
    cpu_percent = process.cpu_percent(interval=1)
    memory_usage_mb = process.memory_info().rss / 1024 ** 2
    throughput = result.shape[0] / elapsed_time
```

Polars

- Utilizes Polars' Rust-based, columnar data processing
- Employs vectorized operations and expression-based transformations
- Shows native string operations and regex optimizations
- Demonstrates high-performance data frame operations



```python
import polars as pl
```



```python
# Fill nulls with default values
df = df.with_columns([
    pl.col("location").fill_null("Unknown"),
    pl.col("product_type").fill_null("Unknown")
])

# Extract cleaned versions first
df = df.with_columns([
    pl.col("price").str.extract(r"(\d+(?:\.\d+)?)").cast(pl.Float64).round(2).alias("price"),
    pl.col("product_name").str.replace_all(r"[^\x00-\x7F]+", "").alias("product_name")
])

# Reorder columns if needed
df = df.select([
    "product_name", "price", "location", "link", "product_type"
])

# Rename columns
df = df.rename({
    "product_name": "Product Name",
    "price": "Price",
    "location": "Location",
    "link": "Link",
    "product_type": "Product Type"
})

# Save cleaned CSV
df.write_csv("Item_list_cleaned_optimized.csv")

# End performance tracking
end_time = time.time()
elapsed_time = end_time - start_time
cpu_percent = process.cpu_percent(interval=1)
memory_usage_mb = process.memory_info().rss / 1024 ** 2
throughput = df.shape[0] / elapsed_time
```

Performance Evaluation

Web Scraping

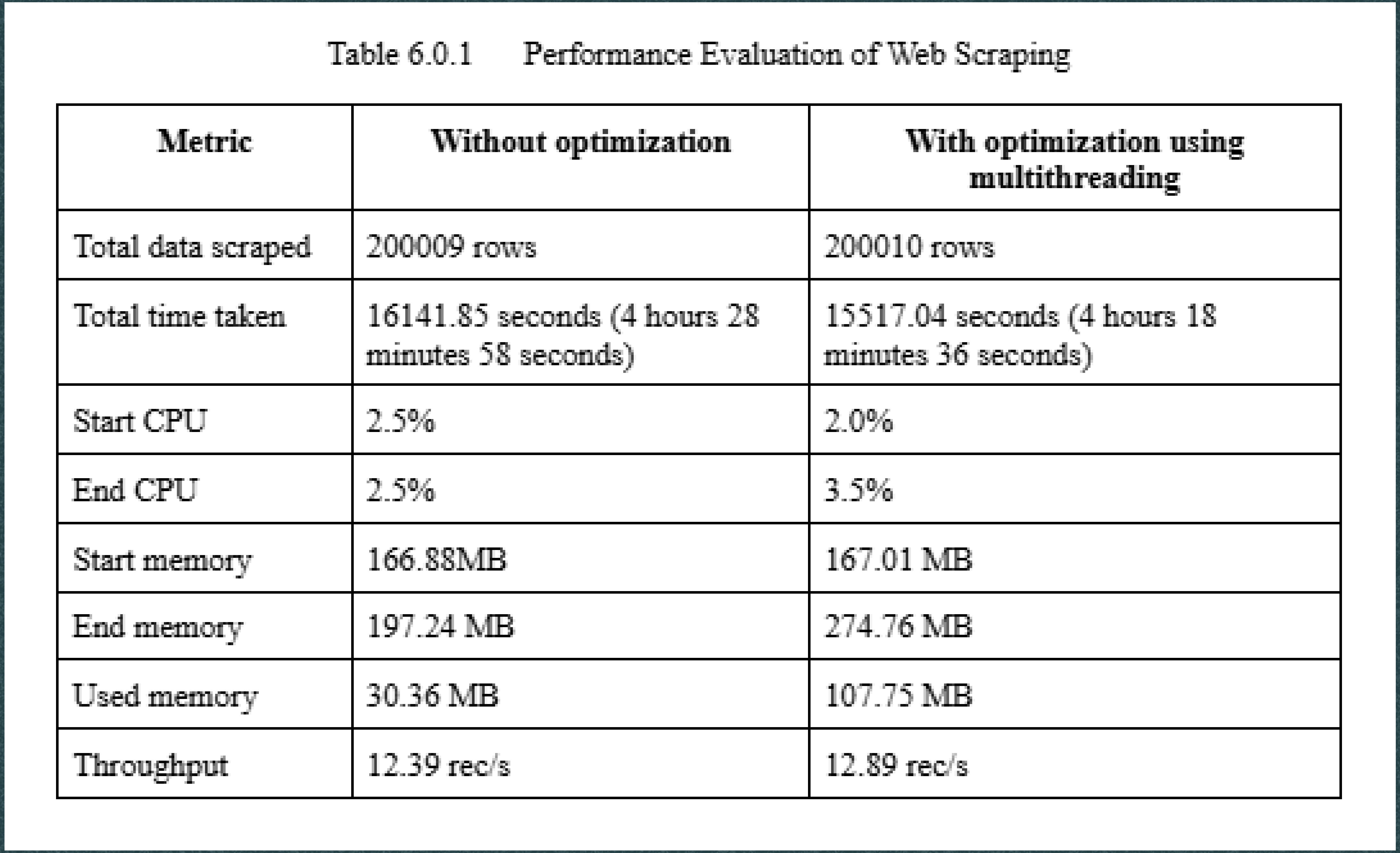
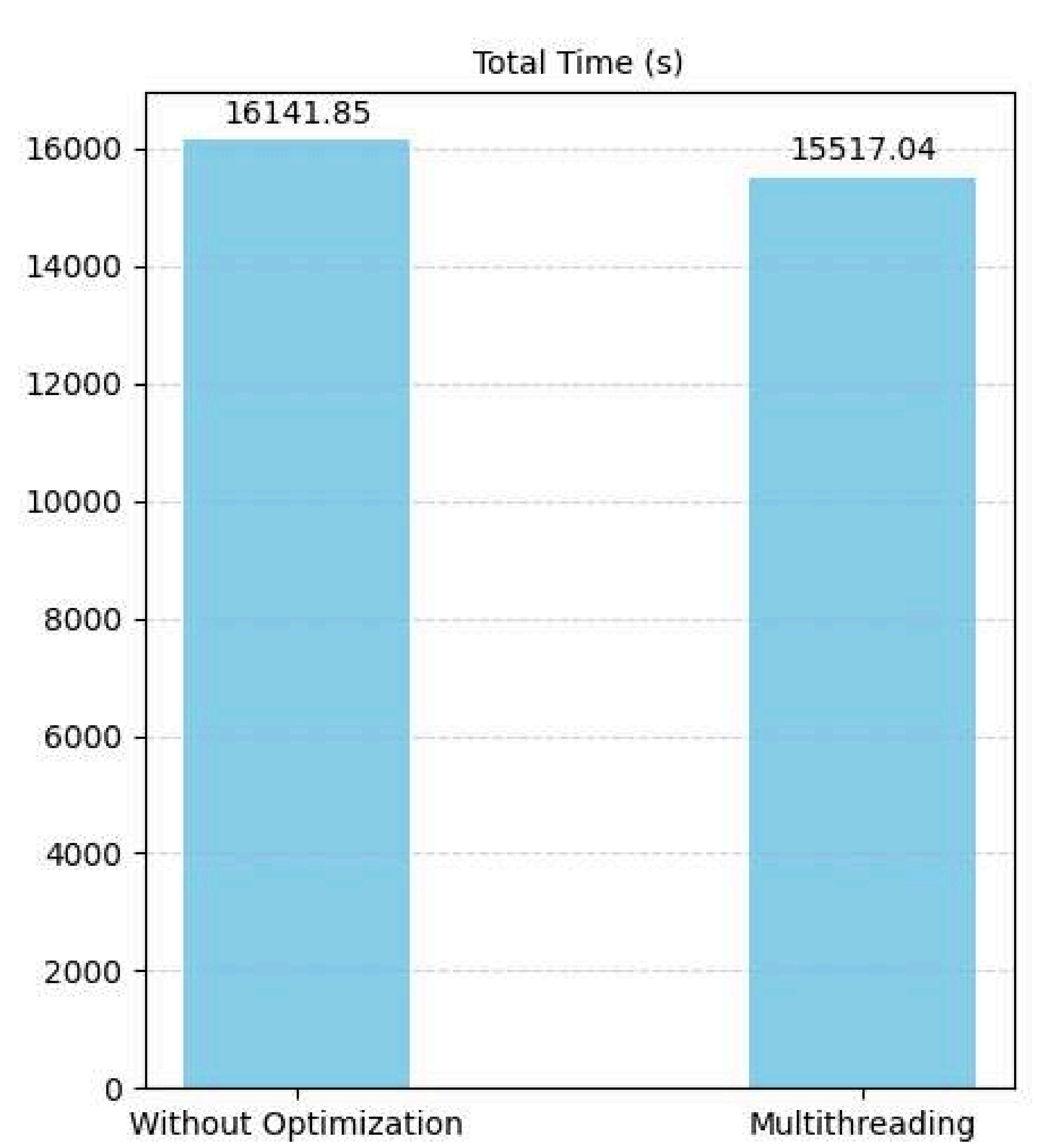


Table 6.0.1 Performance Evaluation of Web Scraping

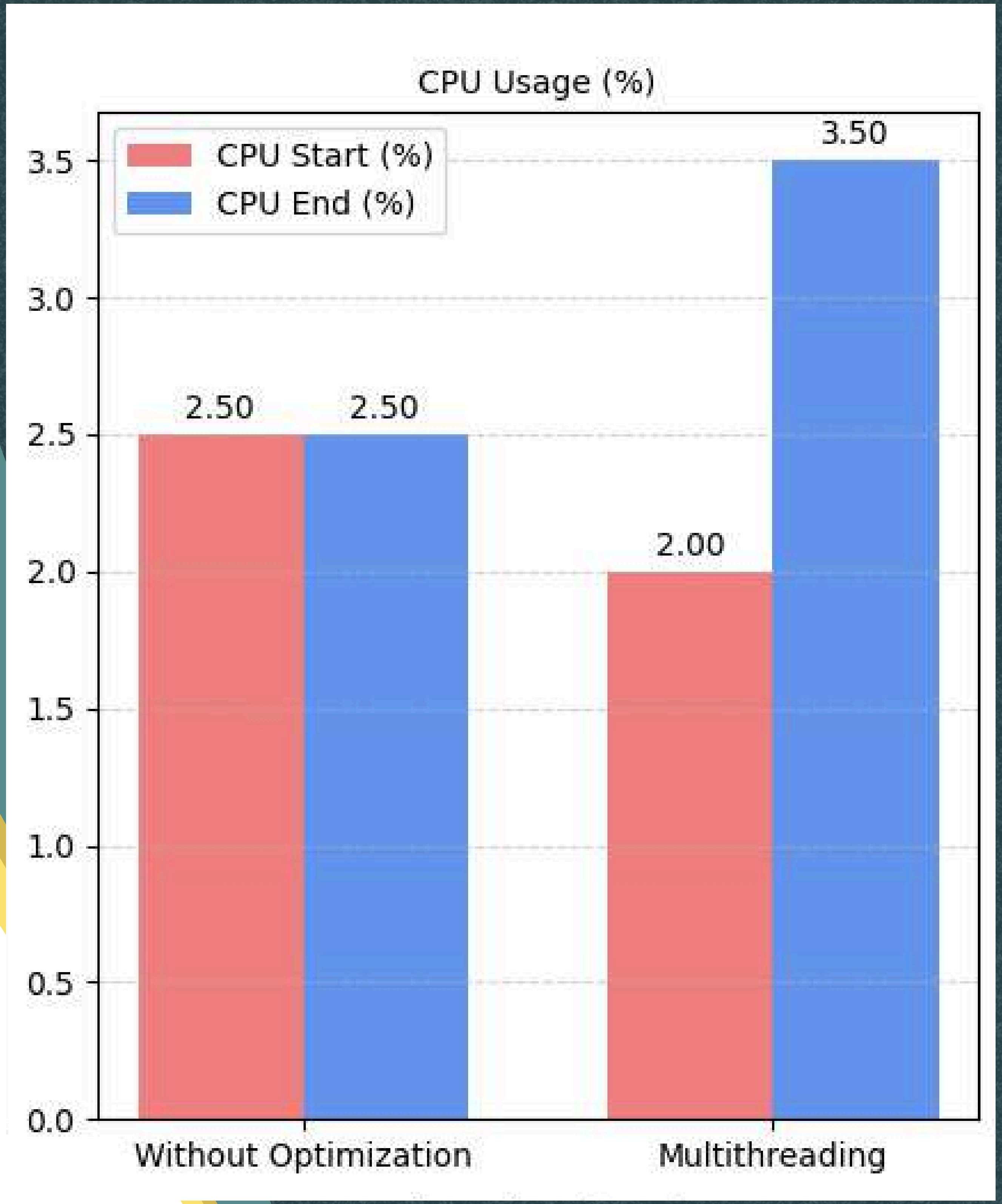
Metric	Without optimization	With optimization using multithreading
Total data scraped	200009 rows	200010 rows
Total time taken	16141.85 seconds (4 hours 28 minutes 58 seconds)	15517.04 seconds (4 hours 18 minutes 36 seconds)
Start CPU	2.5%	2.0%
End CPU	2.5%	3.5%
Start memory	166.88MB	167.01 MB
End memory	197.24 MB	274.76 MB
Used memory	30.36 MB	107.75 MB
Throughput	12.39 rec/s	12.89 rec/s

Web Scraping – Total Processing Time



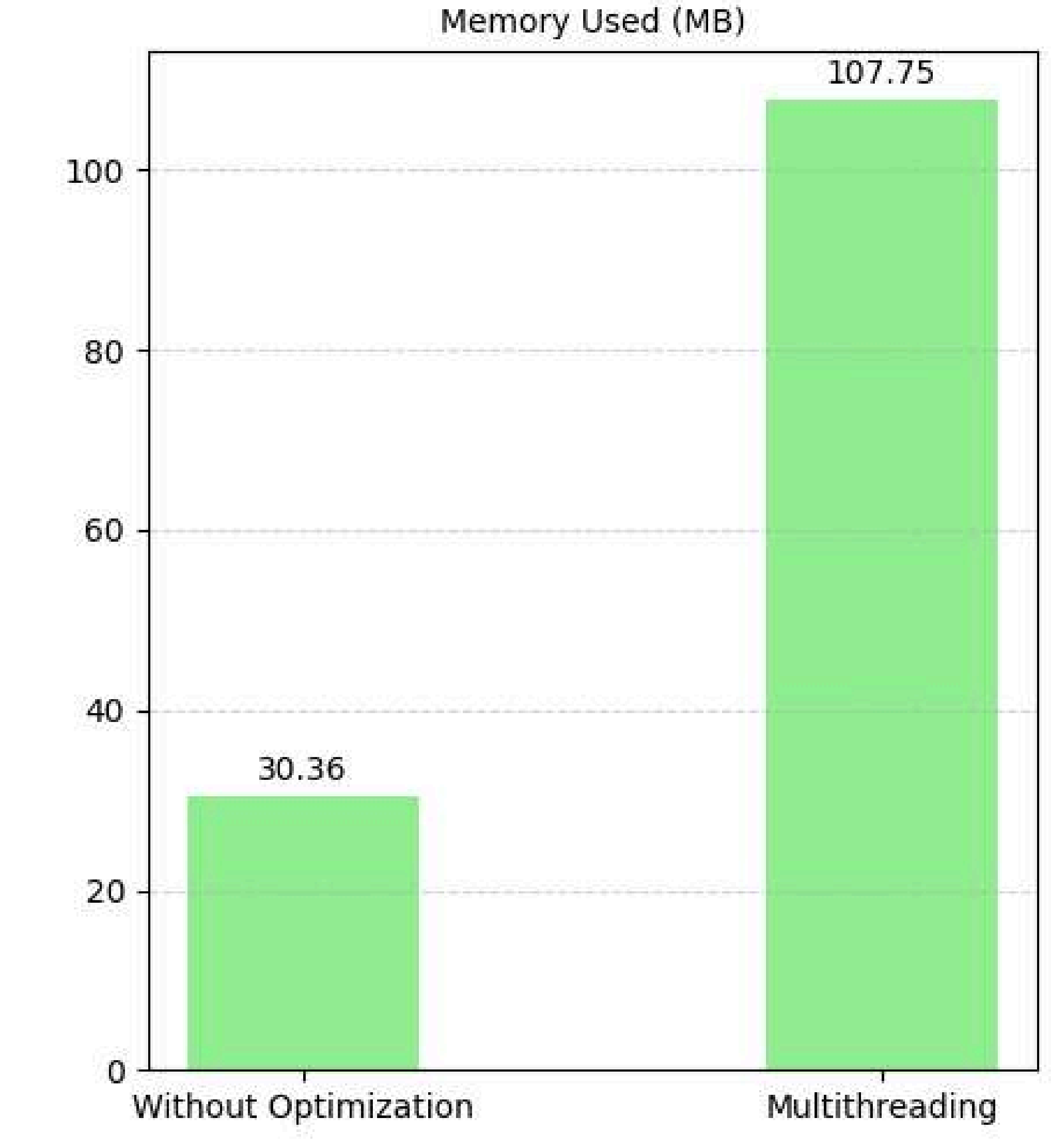
- Basic scraping: 16141.85 seconds
- Optimized scraping: 15517.04 seconds
- 4% faster with multithreading

Web Scraping – CPU Usage



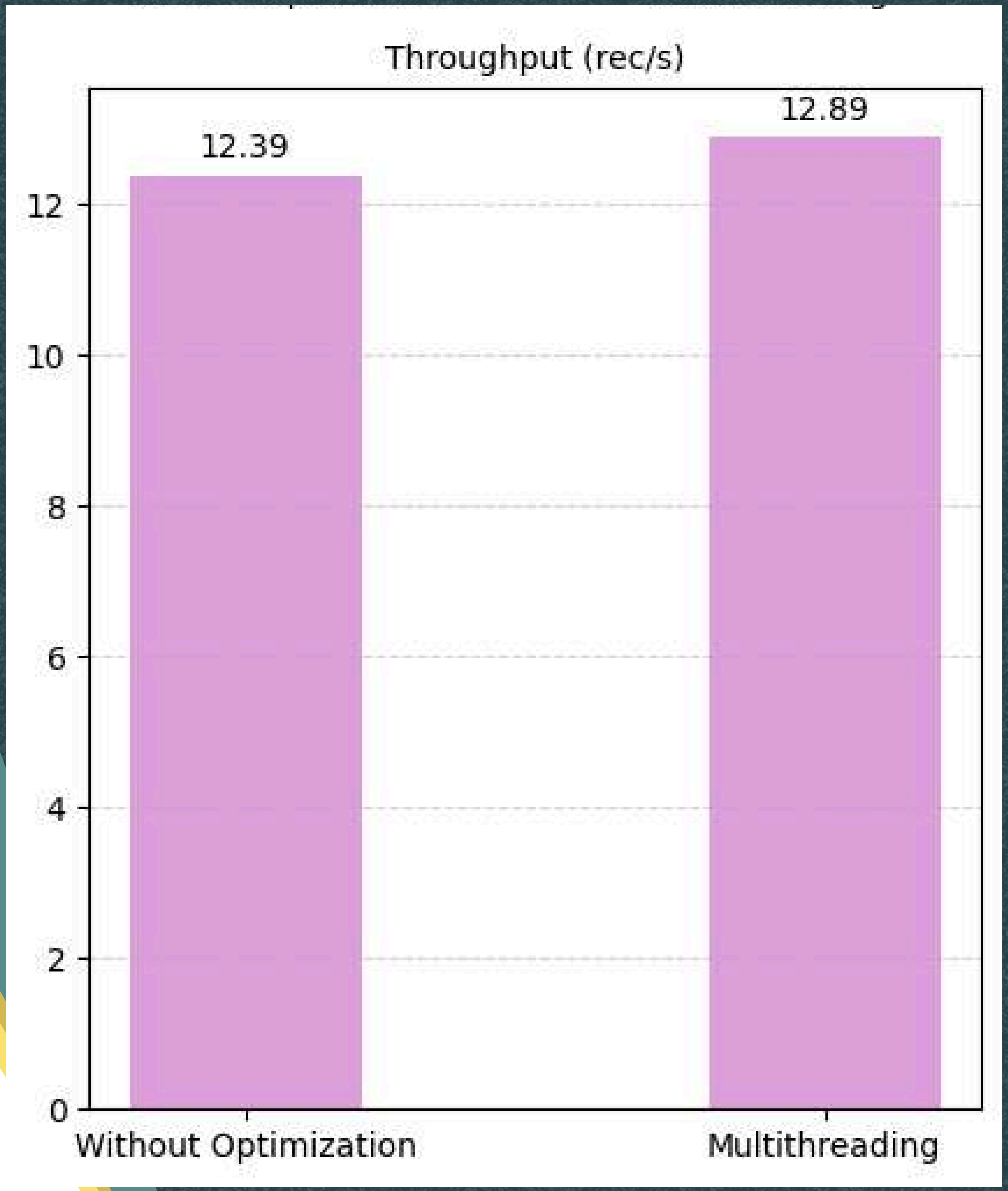
- Without optimization: Stable at 2.50%
- With optimization: Increased from 2.0% to 3.5%
- Multithreading uses more CPU for concurrent tasks

Web Scraping – Memory Usage



- Without optimization: 30.36 MB
- With optimization: 107.75 MB
- Multithreading increases memory usage due to parallel threads

Web Scraping – Throughput



- Without optimization: 12.39 rec/s
- With optimization: 12.89 rec/s
- 4% higher throughput with optimization

Data Cleaning

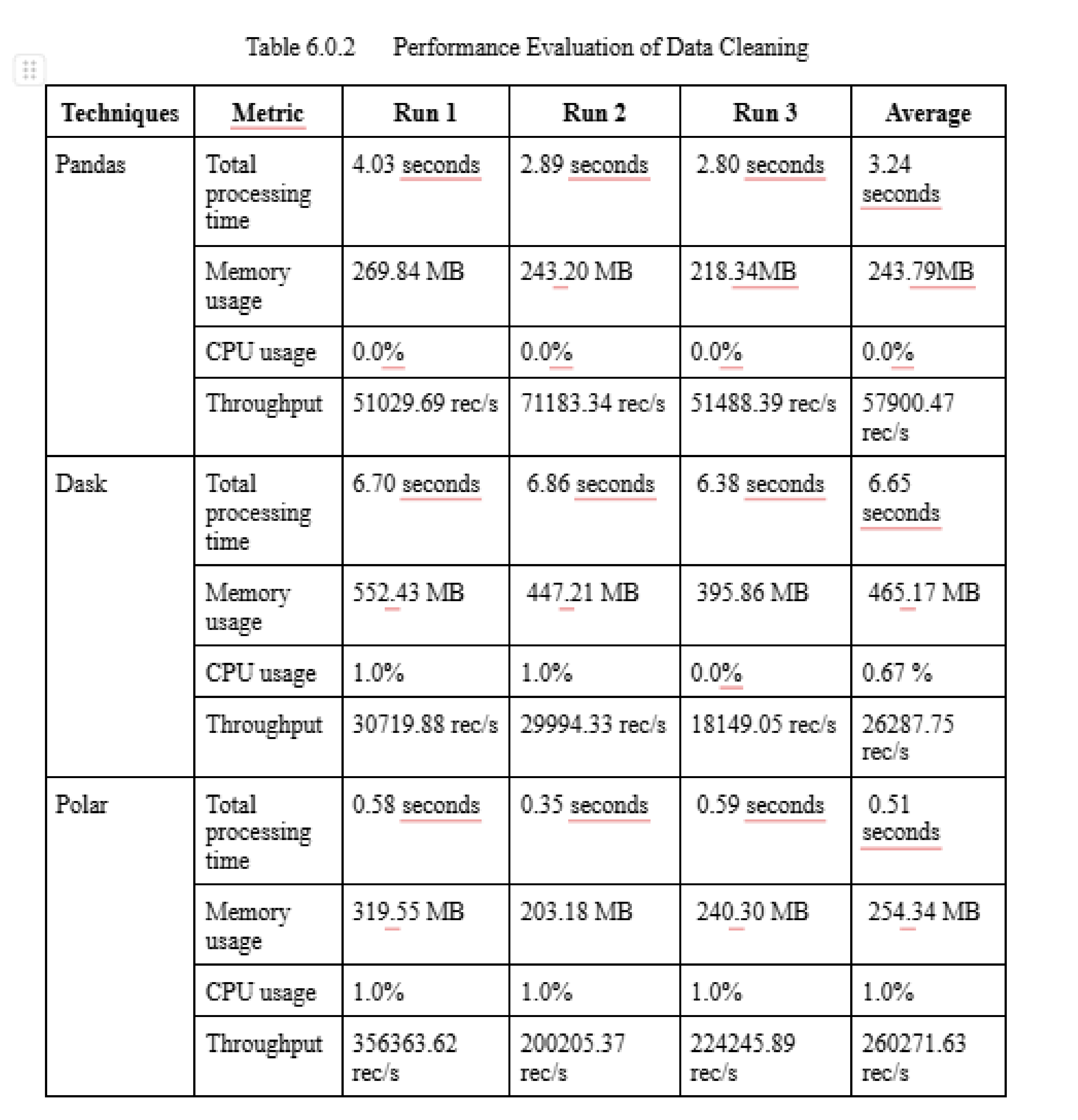
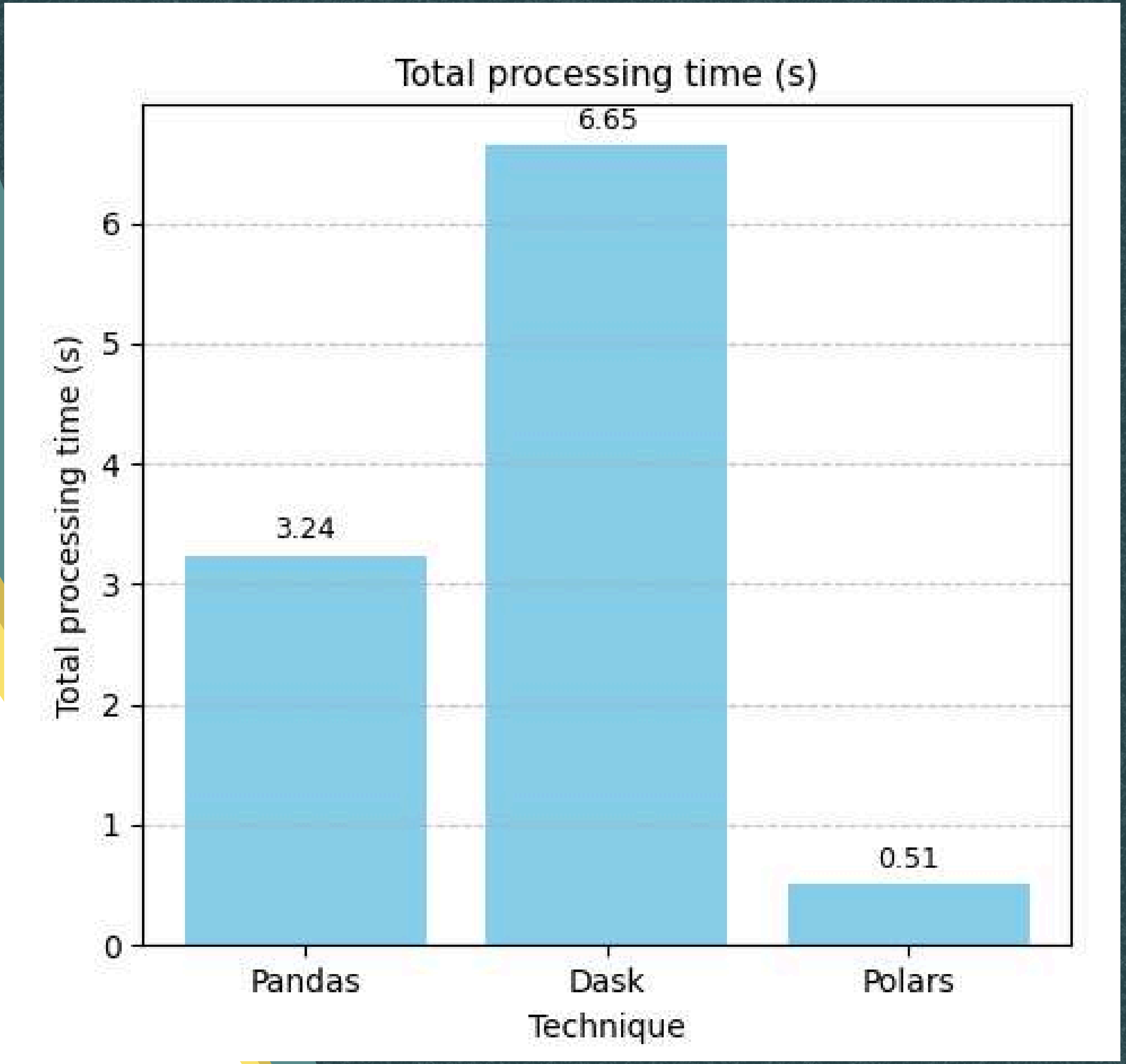


Table 6.0.2 Performance Evaluation of Data Cleaning

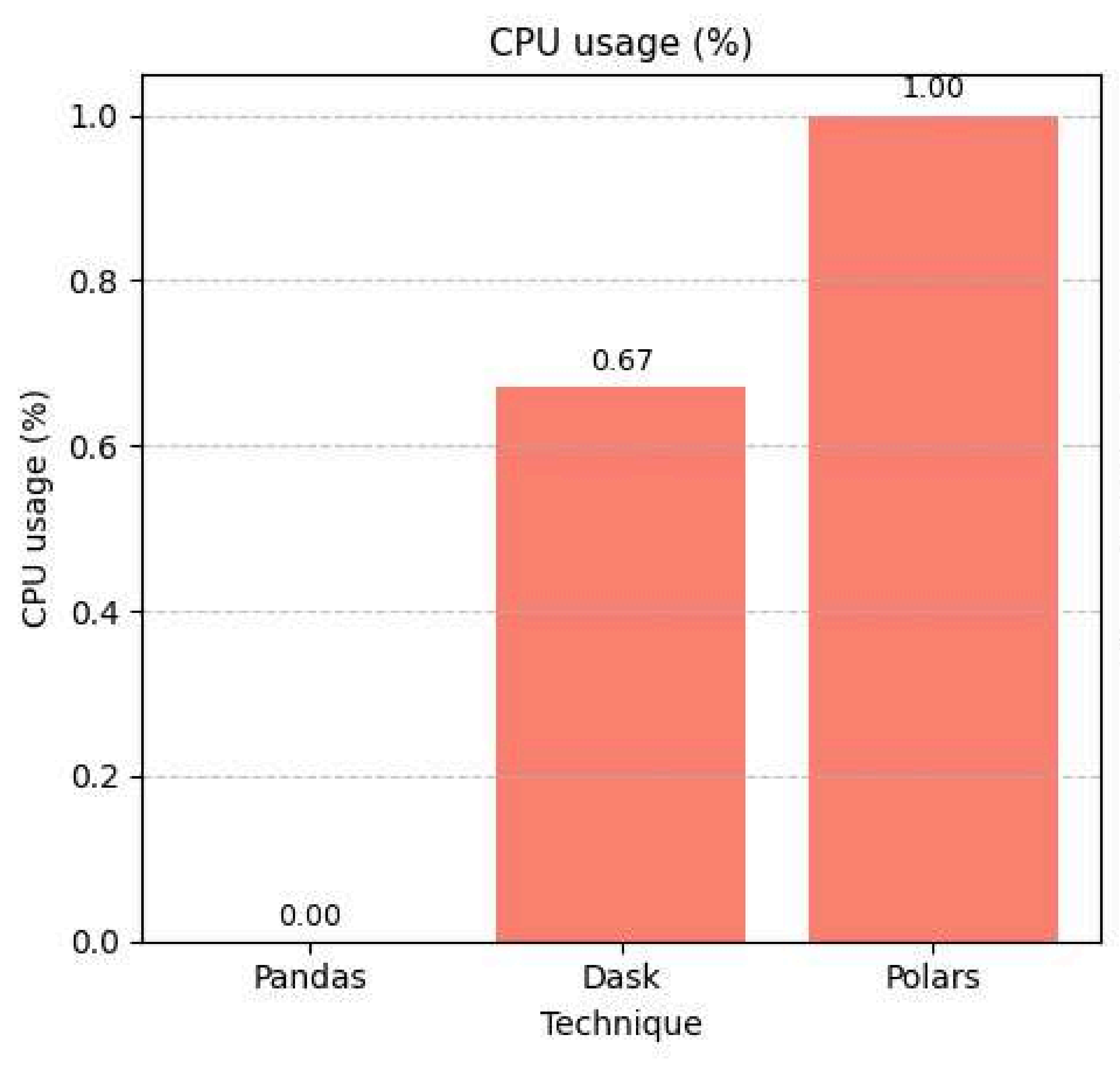
Techniques	Metric	Run 1	Run 2	Run 3	Average
Pandas	Total processing time	4.03 seconds	2.89 seconds	2.80 seconds	3.24 seconds
	Memory usage	269.84 MB	243.20 MB	218.34MB	243.79MB
	CPU usage	0.0%	0.0%	0.0%	0.0%
	Throughput	51029.69 rec/s	71183.34 rec/s	51488.39 rec/s	57900.47 rec/s
Dask	Total processing time	6.70 seconds	6.86 seconds	6.38 seconds	6.65 seconds
	Memory usage	552.43 MB	447.21 MB	395.86 MB	465.17 MB
	CPU usage	1.0%	1.0%	0.0%	0.67 %
	Throughput	30719.88 rec/s	29994.33 rec/s	18149.05 rec/s	26287.75 rec/s
Polar	Total processing time	0.58 seconds	0.35 seconds	0.59 seconds	0.51 seconds
	Memory usage	319.55 MB	203.18 MB	240.30 MB	254.34 MB
	CPU usage	1.0%	1.0%	1.0%	1.0%
	Throughput	356363.62 rec/s	200205.37 rec/s	224245.89 rec/s	260271.63 rec/s

Data Cleaning – Total Processing Time



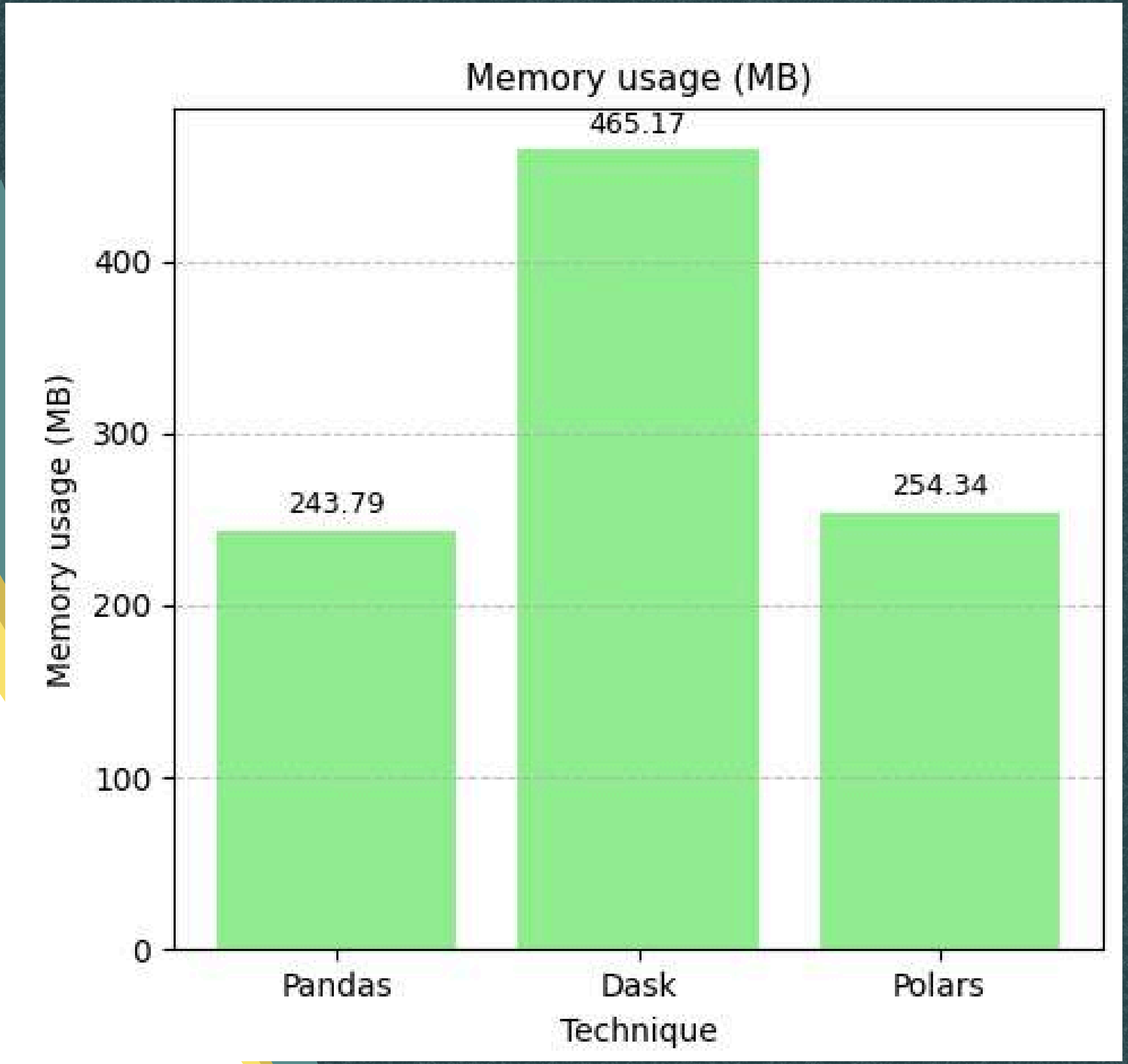
- **Polars: Fastest at 0.51 seconds**
- **Dask: Slowest at 6.65 seconds**
- **Pandas: Moderate speed at 3.24 seconds**
- **Polars wins due to multi-threaded, columnar processing**

Data Cleaning – CPU Usage



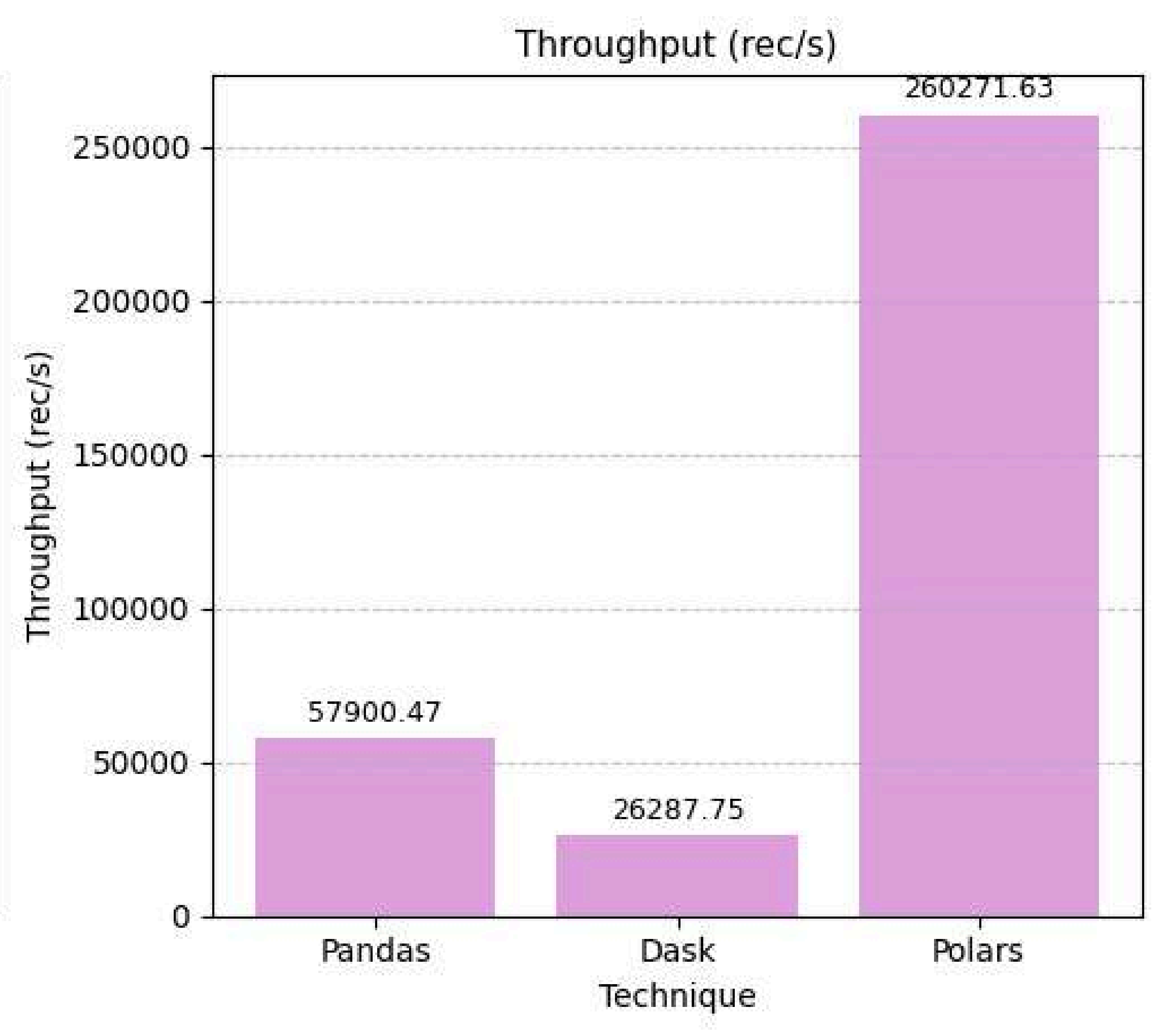
- **Pandas: 0.0%**
- **Dask: 0.67%**
- **Polars: 1.0% (highest – fully utilizes CPU cores)**

Data Cleaning – Memory Usage



- Pandas: 243.79 MB – least memory used
- Polars: 254.34
- Dask: 465.17 MB
- Dask uses more memory due to parallel partition handling

Data Cleaning – Throughput

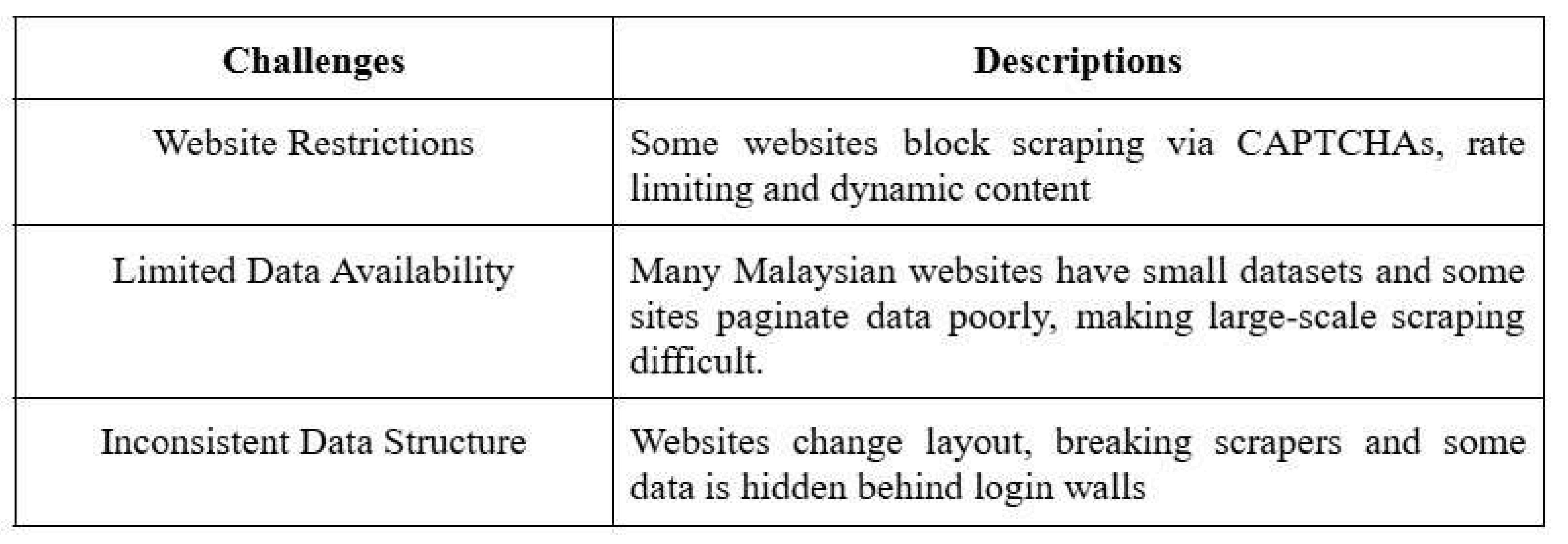


- Polars: 260271.63 rec/sec - Highest throughput
- Pandas: 57900.47rec/s
- Dask: 26287.75 rec/sec
- Polars processed data most efficiently

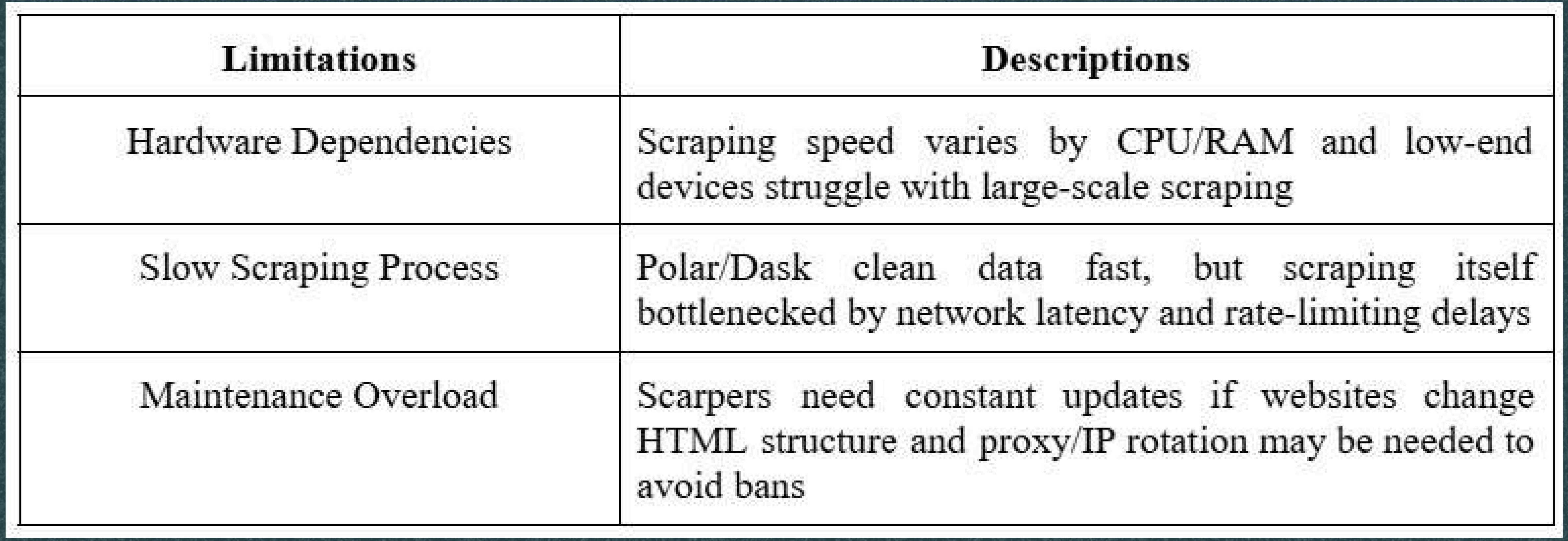
Challenges

&

Limitations



Challenges	Descriptions
Website Restrictions	Some websites block scraping via CAPTCHAs, rate limiting and dynamic content
Limited Data Availability	Many Malaysian websites have small datasets and some sites paginate data poorly, making large-scale scraping difficult.
Inconsistent Data Structure	Websites change layout, breaking scrapers and some data is hidden behind login walls



Limitations	Descriptions
Hardware Dependencies	Scraping speed varies by CPU/RAM and low-end devices struggle with large-scale scraping
Slow Scraping Process	Polar/Dask clean data fast, but scraping itself bottlenecked by network latency and rate-limiting delays
Maintenance Overload	Scarpers need constant updates if websites change HTML structure and proxy/IP rotation may be needed to avoid bans

Conclusion

Data Collection

- Over 200,000 cleaned records collected from PG Mall’s “Health & Beauty” category.

Performance Comparison

- Multithreaded scraping vs normal scraping.
- Multithreading offers faster data collection and higher throughput.
- Trade-off: higher CPU and memory usage, but acceptable for large data volumes.

Data Cleaning Frameworks

- Polars: Most efficient and scalable; best for high-performance needs.
- Pandas: Suitable for simplicity and low-memory environments.
- Dask: Best for very large or distributed datasets.

Areas for Improvement

- Use proxy rotation and headless browsers to bypass CAPTCHA and rate limits.
- Expand to more categories or websites for broader data coverage.
- Use asynchronous requests to reduce latency.
- Deploy scraping on cloud platforms (e.g., GCP, AWS) for better performance.
- Improve scraper adaptability using modular code or public APIs.

THANK YOU