



UTM
UNIVERSITI TEKNOLOGI MALAYSIA

SECP3133-02 High Performance Data Processing

Project 1 Final Report

14/5/2025

FACULTY OF COMPUTING

Prepared By: Group 2 Hepatitis

LIM JING YONG A22EC0182

LEE SOON DER A22EC0065

JASLENE YU A22EC0171

NIK ZULAIKHAA A22EC0232

Lecturer:

DR.ARYATI BINTI BAKRI

Table of Content

1.0 Introduction	3
1.1 Project Background	3
1.2 Objectives	3
1.3 Target Website & Data to be Extracted	3
2.0 System Design and Architecture	6
2.1 Architecture	6
2.2 Tools and Framework used	7
2.3 Roles of team members	8
3.0 Data Collection	9
3.1 Crawling method	9
3.1.1 Code Overview	9
3.2 Number of records collected	14
3.3 Ethical considerations	15
4.0 Data Processing	16
4.1 Cleaning methods	16
4.2 Data Structure	16
4.3 Transformation and formatting	18
5.0 Optimization Techniques	20
5.1 Methods used	20
5.2 Code Overview	21
6.0 Performance Evaluation	23
6.1 Before optimization	23
6.2 Comparison to Optimized Solution	24
6.3 Graph Visualization	25
7.0 Challenges & Limitations	28
7.1 Challenges	28
7.2 Limitations	28
8.0 Conclusion	30
8.1 Our Findings	30
8.2 Improvement to be done	31
Appendix	32

1.0 Introduction

1.1 Project Background

As technology of modern days grows, speed and effectiveness of computer processes has become significant to those who require data to work, which practically includes almost every existing field in the market. High performance data processing, being one of the more recent technology innovations meant to accelerate compute rate, has become an important topic in data analytics. Hence, from this project, we would like to evaluate the practicality of said innovation:

1. Are high performance computing (HPC) solutions truly accelerating the computing processes, specifically the data cleaning process?
2. How does each python library (Modin, Dask, Joblib etc) differ in the ways they handle HPC?
3. Which library delivers the best performance when dealing with 100k+ datasets?

Through this project, we want to make a conclusion from the listed questions above, and potentially help answer questions by more IT users or business users, whom may be interested in discovering pros and cons of each optimization library.

1.2 Objectives

- To develop a web scraping system using several libraries, such as BeautifulSoup, Selenium and more to extract up to 100,000 real estate data from the [iProperty Malaysia website](#).
- To perform heavy loaded data cleaning on said dataset.
- To utilize four libraries with optimization techniques including multithreading, multiprocessing, Spark, etc. to enhance data processing efficiency
- To evaluate and compare performance before and after optimization based on:
 - Time
 - Memory usage
 - CPU usage
 - Throughput
- To visualize insights and performance metrics using chart and graphs for clear comparison

1.3 Target Website & Data to be Extracted

The target website for data scraping in this project is [iProperty Malaysia](#), a well-known online real estate platform that provides listings for various residential properties that are available for sale across Malaysia. The site offers detailed information on each property,

including its location, price, size, furnishing status, type and the contact agent responsible for the listing.

The data extracted from iProperty includes the following key attributes:

1. Property title (e.g., apartment, condominium, terrace house)
2. Location (state and city)
3. Price
4. Built-up size (in square feet)
5. Furnishing status (furnished, partially furnished or unfurnished)
6. Agent name

This information allows for valuable insights into the Malaysian property market and also provides a robust dataset to evaluate the performance of different high-performance data processing techniques in later stages of the project.

Note: There exist 3 different card styles for properties based on time of publish

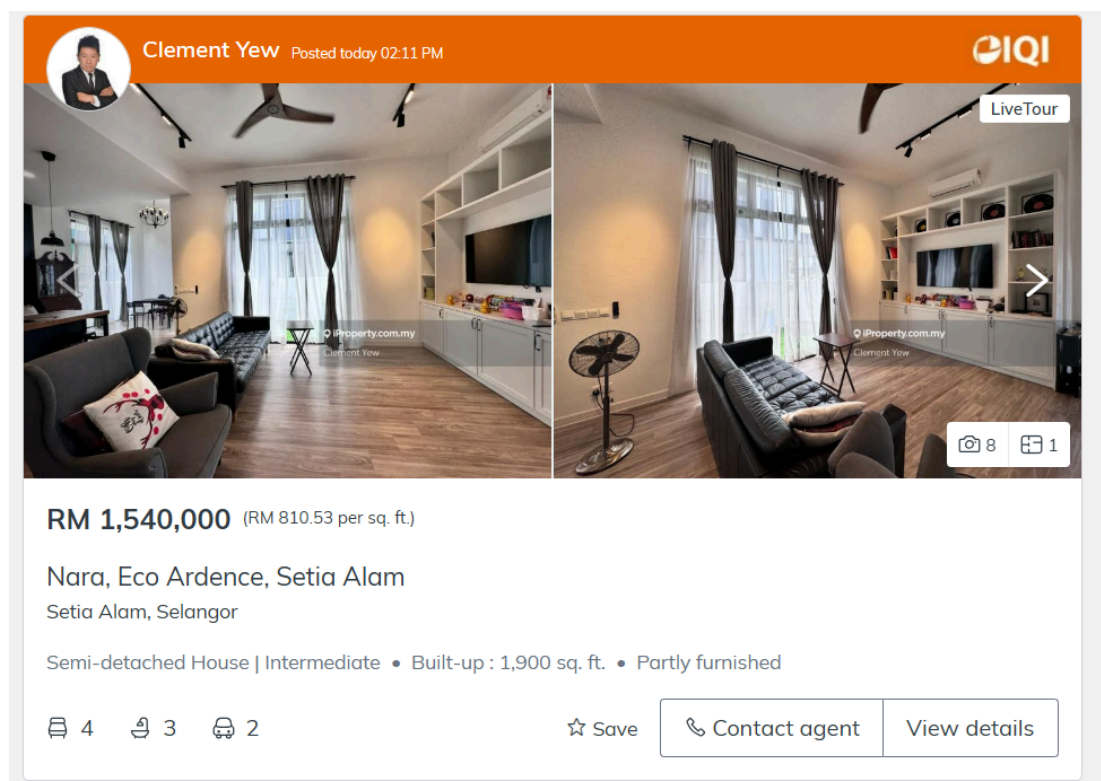




Figure 1.3.1 Example premium style containers in iProperty website



Steven Eng
Posted today 11:16 AM



LiveTour

11

RM 598,000 (RM 254.9 per sq. ft.)

Meru Valley, Ipoh

Ipoh, Perak

Townhouse | Intermediate • Built-up : 2,346 sq. ft. • Fully furnished

4
 4
 4

☆ Save
 [Contact agent](#)
[View details](#)

Figure 1.3.2 Example featured style containers in iProperty website



Sherry Lee
Posted on 04 Apr 2025 11:33 AM



RM 270,000 (RM 192.86 per sq. ft.)

Taman Ilmu, Sitiawan

Sitiawan, Perak

2-sty Terrace/Link House | Duplex • Built-up : 1,400 sq. ft. • Partly furnished

4
 3
 2

☆ Save
 [View details](#)

Figure 1.3.3 Example basic style containers in iProperty website

2.0 System Design and Architecture

2.1 Architecture

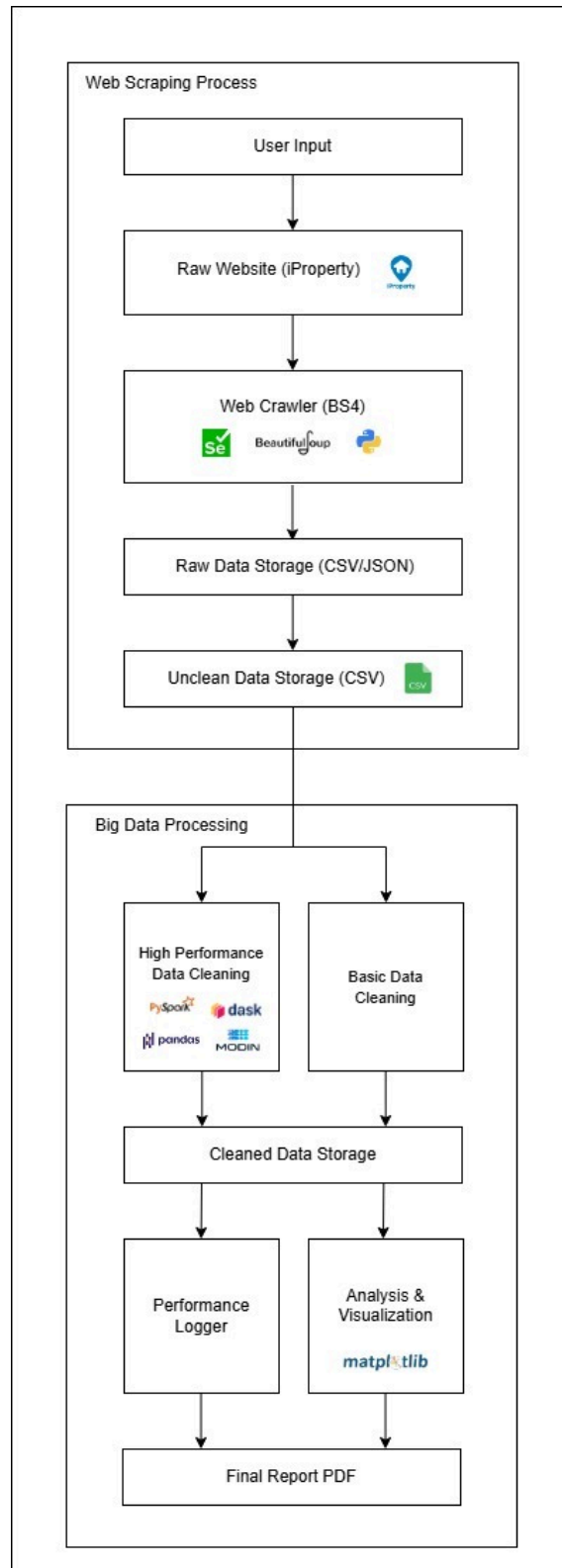


Figure 2.1 Architecture Design of our System

2.2 Tools and Framework used

Category	Software/Application/Library/Website
Documentation	Google Doc
Progression Monitoring	Github
Architecture Design	Draw.io
Web Scraping Library	BeautifulSoup, Selenium
Initial Dataset Form	Excel file (.csv)
IDE	Google Colab
Coding Language	Python
Data Visualization	Matplotlib Library (Python)
Basic Data Cleaning	Pandas Library (Python)
Optimization Libraries	Modin & Ray, Polars, Spark, Joblib, Multiprocessing
Database Management System	SQLite

Table 2.2 Tools used in this Project

2.3 Roles of team members

Member	Role	Responsibility
Lim Jing Yong	Lead Crawler Developer	<ul style="list-style-type: none">• Leading the implementation of the web crawler using BeautifulSoup and Selenium• Handle pagination, data structure identification, and crawl delay management• Ensure ethical scraping practices and data completeness• Optimization library used: Modin & Ray
Lee Soon Der	Data Cleaning and Storage Specialist	<ul style="list-style-type: none">• Leading the cleaning and preprocessing of the raw data• Standardize fields and manage datasets for further processing• Optimization library used: Polars
Jaslene Yu	Performance and Optimization Lead	<ul style="list-style-type: none">• Leading the process applying high performance computing techniques• Monitor and document CPU/memory usage and system throughput• Optimization library used: Multiprocessing & Spark
Nik Zulaikhaa	Performance Analyst and Documentation Lead	<ul style="list-style-type: none">• Leading the comparison of optimization result• Generate visualizations and compile the final technical report• Optimization library used: Joblib

Table 2.3 Member Roles

3.0 Data Collection

3.1 Crawling method

To collect the required property data from iProperty Malaysia, we implemented a **web crawling system** by using **Python** by combining the **Selenium** library with **BeautifulSoup** for HTML parsing. Selenium was used to simulate a real browser environment that handles dynamic content loading. BeautifulSoup was implemented to extract and parse the HTML structure of the web pages.

3.1.1 Code Overview

1. Using CLI commands to download selenium files locally.

```
# STEP 1: Install Selenium
!pip install -q selenium

# STEP 2: Download a working version of Chromium and ChromeDriver
!wget -q https://storage.googleapis.com/chromium-browser-snapshots/Linux_x64/1140136/chrome-linux.zip
!wget -q https://storage.googleapis.com/chromium-browser-snapshots/Linux_x64/1140136/chromedriver_linux64.zip

# STEP 3: Unzip and move
!unzip -q chrome-linux.zip
!unzip -q chromedriver_linux64.zip
!mv chrome-linux /usr/local/chrome
!mv chromedriver_linux64/chromedriver /usr/local/chromedriver
!chmod +x /usr/local/chromedriver
```

2. Import all necessary libraries.

```
import csv
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from bs4 import BeautifulSoup
import os
import time
import random
```

3. Configure Chrome Driver (used to translate Selenium coding for Chrome browser to understand)

```
# Setup Chrome driver
os.environ["PATH"] += ":/usr/local/chrome:/usr/local/chromedriver"
chrome_options = Options()
chrome_options.binary_location = "/usr/local/chrome/chrome"
chrome_options.add_argument("--headless")
chrome_options.add_argument("--no-sandbox")
chrome_options.add_argument("--disable-dev-shm-usage")
chrome_options.add_argument("--disable-gpu")
chrome_options.add_argument("--window-size=1920,1080")
chrome_options.add_argument("--disable-blink-features=AutomationControlled")
chrome_options.add_experimental_option("excludeSwitches", ["enable-automation"])
chrome_options.add_experimental_option("useAutomationExtension", False)
```

4. Set up different user agent (avoid detection/CAPTCHA)

```
# List of fake user-agent strings
fake_user_agents = [
    "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36",
    "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:40.0) Gecko/20100101 Firefox/40.0",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0.3202.94 Safari/537.36",
    "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36 Edge/17.17134",
    "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0",
    "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.67 Safari/537.36 Edge/18.18363"
]

# Randomly pick a fake user-agent
user_agent = random.choice(fake_user_agents)
chrome_options.add_argument(f"user-agent={user_agent}")

driver = webdriver.Chrome(service=Service("/usr/local/chromedriver"), options=chrome_options)
```

5. Define css selector used by BeautifulSoup to extract data.

```
# Wrapper selectors
card_wrappers = [
    "PremiumCardstyle__CardWrapper-sywKC",
    "FeaturedCardstyle__FeaturedCardWrapper-jxoTmP",
    "BasicCardstyle__BasicCardWrapper-iiXerd"
]

title_selector = [
    "h2[class^='PremiumCardstyle__TitleWrapper']",
    "h2[class^='FeaturedCardstyle__TitleWrapper']",
    "h2[class^='BasicCardstyle__TitleWrapper']"
]

price_selector = [
    "li[class^='ListingPricestyle__ItemWrapper']",
    "li[class^='ListingPricestyle__ItemWrapper']"
]

location_selector = [
    "div[class^='PremiumCardstyle__AddressWrapper']",
    "div[class^='FeaturedCardstyle__AddressWrapper']",
    "div[class^='BasicCardstyle__AddressWrapper']"
]
```

... truncated

6. Define function to timeout 60 seconds to wait for browser container to load.

```
# Wait for main container to load
def wait_for_any_class(driver, classes, timeout=60):
    WebDriverWait(driver, timeout).until(
        lambda d: any(d.find_elements(By.CLASS_NAME, cls) for cls in classes)
    )
```

7. Define function to extract data based on previously defined selectors.

```
[ ] # CSS Selector extracting data
def extract_from_selectors(card, selectors):
    for selector in selectors:
        element = card.select_one(selector)
        if element:
            return element.get_text(strip=True)
    return None
```

8. Define scraping function, define website URL, wait 10 seconds until “body” selector recognized by driver

```
# Defining scraping function
def scrape_iproperty_area(area_slug, pages_to_scrape=100):
    base_url = f"https://www.iproperty.com.my/sale/{area_slug}/all-residential/"
    driver.get(base_url)
    listings = []

    try:
        WebDriverWait(driver, 10).until(
            EC.presence_of_element_located((By.TAG_NAME, "body"))
        )
    except Exception as e:
        print(f"[{area_slug}] Failed to load page: {e}")
    return listings
```

9. In iteration, call waiting function, parse html, add every found card wrapper into property card list

```
for page in range(1, pages_to_scrape + 1):
    try:
        wait_for_any_class(driver, card_wrappers, timeout=90)

        soup = BeautifulSoup(driver.page_source, 'html.parser')

        property_cards = []
        for wrapper in card_wrappers:
            property_cards += soup.find_all('div', class_=wrapper)

        if not property_cards:
            print(f"[{area_slug}] Page {page}: No listings found.")
            break

        print(f"[{area_slug}] Page {page}: Found {len(property_cards)} listings")
```

10. Extract data using function and save into local variable

```
for card in property_cards:
    property_title = extract_from_selectors(card, title_selector)
    property_price = extract_from_selectors(card, price_selector)
    property_location = extract_from_selectors(card, location_selector)
    property_details = extract_from_selectors(card, details_selector)
    property_agent = extract_from_selectors(card, agent_selector)

    listings.append({
        'Area': area_slug,
        'Property Title': property_title,
        'Property Price': property_price,
        'Property Location': property_location,
        'Property Details': property_details,
        'Property Agent': property_agent,
        'Source URL': base_url
    })
```

11. Added logic to move onto next area if max page reached

```
if page < pages_to_scrape:
    try:
        next_btn = WebDriverWait(driver, 5).until(
            EC.presence_of_element_located((By.XPATH, "//a[@aria-label='Go to next page']"))
        )

        # Check if the button is actually clickable (enabled and visible)
        if next_btn.is_enabled() and next_btn.is_displayed():
            next_btn.click()
            time.sleep(random.uniform(3, 5))
        else:
            print(f"[{area_slug}] No next page, ending scrape early.")
            break

    except Exception:
        print(f"[{area_slug}] Next page not available or failed to load. Ending scrape early.")
        break
```

12. Define function to save data as a .csv file

```
# Method to save data to csv
def save_to_csv(data, filename="iproperty_listings.csv"):
    if not data:
        print("No data to save.")
        return

    keys = data[0].keys()
    with open(filename, "w", newline='', encoding='utf-8') as output_file:
        dict_writer = csv.DictWriter(output_file, fieldnames=keys)
        dict_writer.writeheader()
        dict_writer.writerows(data)
    print(f"\nSaved {len(data)} listings to '{filename}'")
```

13. List all area codes to iterate over when scraping data

```
area_slugs = [
    "perlis-zop7y",
    "kedah-x5i6n",
    "sarawak-dh4eg",
    "sabah-cc02j",
    "pahang-9ycjt",
    "tambun-z20xa",
    "menglembu-ypm1e",
    "bidor-x843z",
    "seri-iskandar-tydb0",
    "ayer-tawar-tgq9o",
    "kuala-kangsar-s96fp",
    "tanjung-malim-r82wm",
    "lahat-pvsye",
    "ipoh-nnv3d",
    "kuala-kurau-iqkyi",
    ... truncated (223 areas)
```

14. Execute scraping function and run function to save data into .csv file

```
# Main Method
if __name__ == "__main__":

    all_listings = []

    for slug in area_slugs:
        driver.delete_all_cookies()
        area_data = scrape_iproperty_area(slug, pages_to_scrape=100)
        all_listings.extend(area_data)

    save_to_csv(all_listings, filename="iproperty_all_areas.csv")

    time.sleep(random.uniform(5, 10))

    save_to_csv(all_listings, filename="iproperty_all_areas.csv")
    driver.quit()
```

3.2 Number of records collected

A total of **160,000+** records of properties are recorded.

1	Area	Property Title	Property Price	Property Location	Property Details	Property Agent	Source URL
2	perlis-zop7y	Semi D 2 Tingkat - Taman Jaya Diri - Seriab	RM 775,776	Kangar, Perlis	Semi-detached House Intermediate Â⚬Â Built-up: Haneef		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
3	perlis-zop7y	Semi D 1 Tingkat - Taman Nyu Indah 2 - Arat	RM 398,000	Arau, Perlis	Semi-detached House Â⚬Â Built-up : 1,021 sq. ft. / Haneef		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
4	perlis-zop7y	Taman Seri Manis Dua, Kangar	RM 306,000	Kangar, Perlis	Semi-detached House Â⚬Â Built-up : 1,938 sq. ft. Joyce Chan		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
5	perlis-zop7y	Teres 1 Tingkat - Bandar Baharu Putra Heigh	RM 185,000	Arau, Perlis	1-sty Terrace/Link House Intermediate Â⚬Â Built Haneef		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
6	perlis-zop7y	Teres 1 Tingkat - Bandar Baharu Putra Heigh	RM 210,000	Arau, Perlis	1-sty Terrace/Link House Intermediate Â⚬Â Built Haneef		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
7	perlis-zop7y	Taman Seri Bintong Maju, Kangar	RM 247,000	Kangar, Perlis	1-sty Terrace/Link House Corner lot Â⚬Â Built-up: Leslie Low		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
8	perlis-zop7y	Taman Tengku Budriah, Arau	RM 416,000	Arau, Perlis	Semi-detached House Â⚬Â Land area : 4,884 sq. ft Leslie Low		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
9	perlis-zop7y	Rumah Berkembar Setingkat di Jejawi, Perlis	RM 441,000	Kangar, Perlis	Semi-detached House Â⚬Â Built-up : 1,150 sq. ft. Lim		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
10	perlis-zop7y	Taman Kuala Perlis, Kuala Perlis	RM 220,000	Kuala Perlis, Perlis	2-sty Terrace/Link House Intermediate Â⚬Â Built Adam Wong		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
11	perlis-zop7y	Teres (1) Setingkat di Irama Residensi - Kang	RM 298,000	Kangar, Perlis	1-sty Terrace/Link House Â⚬Â Built-up : 990 sq. ft. Haneef		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
12	perlis-zop7y	Taman Sri Pauh, Arau	RM 171,000	Arau, Perlis	1-sty Terrace/Link House Â⚬Â Land area : 1,195 sq Joyce Chan		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
13	perlis-zop7y	Taman Empiang Ceria, Kangar	RM 409,920	Kangar, Perlis	Semi-detached House Intermediate Â⚬Â Built-up: Shafina		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
14	perlis-zop7y	Padang Blak, Arau	RM 3,147,210	Arau, Perlis	Residential Land Â⚬Â Land area : 2.89 acres Dickson Lum		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
15	perlis-zop7y	Teres 2 Tingkat - Taman Bunga Padi - Kanga	RM 180,000	Kangar, Perlis	2-sty Terrace/Link House Intermediate Â⚬Â Built Haneef		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
16	perlis-zop7y	Rumah Berkembar 1 Tingkat - Taman Sri Gui	RM 385,000	Kangar, Perlis	Semi-detached House Intermediate Â⚬Â Built-up: Haneef		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
17	perlis-zop7y	Teres 2 Tingkat - Taman Kemajuan - Kangar,	RM 250,000	Kangar, Perlis	2-sty Terrace/Link House End lot Â⚬Â Built-up : Haneef		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
18	perlis-zop7y	Taman Padang Behor, Kangar	RM 370,000	Kangar, Perlis	Semi-detached House Â⚬Â Built-up : 960 sq. ft. Nur Suhaini		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
19	perlis-zop7y	Kangar	RM 450,000	Kangar, Perlis	Semi-detached House Corner lot Â⚬Â Built-up : Nur Suhaini		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
20	perlis-zop7y	Kangar	RM 385,000	Kangar, Perlis	Semi-detached House Intermediate Â⚬Â Built-up: Nur Suhaini		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
21	perlis-zop7y	(BAHARU) Taman Enyum Indah - Arau - Perli	RM 355,000	Arau, Perlis	1-sty Terrace/Link House Â⚬Â Built-up : 880 sq. ft. Haneef		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
22	perlis-zop7y	Taman Utara Jaya 1., Kuala Perlis	RM 398,000	Kuala Perlis, Perlis	Semi-detached House Intermediate Â⚬Â Built-up: Elwin		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
23	perlis-zop7y	Pauh, Arau	RM 6,970,000	Arau, Perlis	Residential Land Â⚬Â Land area : 9.7 acres Sally Chin		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
24	perlis-zop7y	Kangar, Perlis., Kangar	RM 750,000	Kangar, Perlis	Semi-detached House Intermediate Â⚬Â Built-up: Michael Lee		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
25	perlis-zop7y	Jalan Padang Katong, Kangar	RM 750,000	Kangar, Perlis	Semi-detached House Intermediate Â⚬Â Built-up: Michael Lee		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
26	perlis-zop7y	Rumah Berkembar Setingkat di Jejawi Perlis	RM 415,000	Kangar, Perlis	Semi-detached House Intermediate Â⚬Â Built-up: Mk Sangaran		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
27	perlis-zop7y	Taman Hijrah, Kangar	RM 350,000	Kangar, Perlis	1-sty Terrace/Link House Corner lot Â⚬Â Built-up: Muaz Atizan		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/
28	perlis-zop7y	Taman Temak Jaya, Seriab	RM 395,000	Seriab, Perlis	2-sty Terrace/Link House Intermediate Â⚬Â Built Faizals		https://www.iproperty.com.my/sale/perlis-zop7y/all-residential/

Figure 3.2.1 First 27 rows of data in uncleaned dataset

163512	simpang-ampat-41; Bandar Tasek Mutiara, Simpang Ampat	RM 6,546,428	Simpang Ampat, Penang	Residential Land Â¬ Land area : 96,271 sq. ft.	Crystal Tee	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163513	simpang-ampat-41; Bandar Tasek Mutiara, Simpang Ampat	RM 9,980,292	Simpang Ampat, Penang	Residential Land Â¬ Land area : 146,769 sq. ft.	Crystal Tee	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163514	simpang-ampat-41; Bandar Tasek Mutiara, Simpang Ampat	RM 9,149,536	Simpang Ampat, Penang	Residential Land Â¬ Land area : 134,552 sq. ft.	Crystal Tee	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163515	simpang-ampat-41; Bandar Tasek Mutiara, Simpang Ampat	RM 12,579,252	Simpang Ampat, Penang	Residential Land Â¬ Land area : 184,989 sq. ft.	Crystal Tee	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163516	simpang-ampat-41; Royale Infinity, Simpang Ampat	RM 380,000	Simpang Ampat, Penang	Condominium Â¬ Built-up : 1,480 sq. ft.	Jass Kong	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163517	simpang-ampat-41; Simpang Ampat	RM 570,000	Simpang Ampat, Penang	Semi-detached House Â¬ Built-up : 1,585 sq. ft.	Jetson Yip	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163518	simpang-ampat-41; Hijauan Hill, Simpang Ampat	RM 650,000	Simpang Ampat, Penang	Semi-detached House Â¬ Built-up : 2,280 sq. ft.	Anders Ong	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163519	simpang-ampat-41; Batu Kawan, Simpang Ampat	RM 858,000	Simpang Ampat, Penang	2-sty Terrace/Link House Â¬ Built-up : 2,290 sq.	Chris Huah	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163520	simpang-ampat-41; Bandar Tasek Mutiara, Simpang Ampat	RM 261,000	Simpang Ampat, Penang	1-sty Terrace/Link House Â¬ Land area : 1,195 sq	Koay CK	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163521	simpang-ampat-41; Taman Lembah Indah, Simpang Ampat	RM 1,400,000	Simpang Ampat, Penang	Bungalow Â¬ Built-up : 4,500 sq. ft.	Steven Hng	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163522	simpang-ampat-41; Royale Infinity, Simpang Ampat	RM 680,000	Simpang Ampat, Penang	Condominium Intermediate Â¬ Built-up : 1,400	Kelly Yee	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163523	simpang-ampat-41; Taman Lembah Indah, Simpang Ampat	RM 660,000	Simpang Ampat, Penang	3-sty Terrace/Link House Â¬ Built-up : 2,524 sq.	Winson Chong	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163524	simpang-ampat-41; Taman residenti alma, Simpang Ampat	RM 700,000	Simpang Ampat, Penang	2-sty Terrace/Link House Â¬ Built-up : 1,640 sq.	Suzzane Lee	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163525	simpang-ampat-41; Taman Idaman, Simpang Ampat	RM 810,000	Simpang Ampat, Penang	2-sty Terrace/Link House Corner lot Â¬ Built-uj	CH Ong	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163526	simpang-ampat-41; Taman Pervira Indah, Simpang Ampat	RM 430,000	Simpang Ampat, Penang	2-sty Terrace/Link House Â¬ Built-up : 2,000 sq.	Fionne Seah	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163527	simpang-ampat-41; Taman Pervira, Simpang Ampat	RM 315,000	Simpang Ampat, Penang	1-sty Terrace/Link House Â¬ Built-up : 1,200 sq.	Louis Ng	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163528	simpang-ampat-41; Bandar Tasek Mutiara, Simpang Ampat	RM 550,000	Simpang Ampat, Penang	2-sty Terrace/Link House Intermediate Â¬ Built	Alex Ho	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163529	simpang-ampat-41; Taman Lembah Indah, Simpang Ampat	RM 866,000	Simpang Ampat, Penang	3-sty Terrace/Link House Corner lot Â¬ Built-uj	CH Ong	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163530	simpang-ampat-41; Raintree Park, Simpang Ampat	RM 480,000	Simpang Ampat, Penang	2-sty Terrace/Link House Intermediate Â¬ Built	Francis Neow	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163531	simpang-ampat-41; Taman Tambun Emas, Simpang Ampat	RM 699,000	Simpang Ampat, Penang	3-sty Terrace/Link House Corner lot Â¬ Built-uj	Nicol Tan	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163532	simpang-ampat-41; Taman Murai Jaya, Simpang Ampat	RM 680,000	Simpang Ampat, Penang	Semi-detached House Intermediate Â¬ Built-up	CH Ong	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163533	simpang-ampat-41; Bandar Tasek Mutiara, Simpang Ampat	RM 950,000	Simpang Ampat, Penang	Semi-detached House Corner lot Â¬ Built-up :	CH Ong	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163534	simpang-ampat-41; Taman Lembah Indah, Simpang Ampat	RM 608,000	Simpang Ampat, Penang	2-sty Terrace/Link House Intermediate Â¬ Built	Jetson Yip	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163535	simpang-ampat-41; Taman Eco Meadows, Simpang Ampat	RM 850,000	Simpang Ampat, Penang	2-sty Terrace/Link House Intermediate Â¬ Built	Jass Ooi	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163536	simpang-ampat-41; Taman Tambun Indah, Simpang Ampat	RM 1,600,000	Simpang Ampat, Penang	Bungalow Intermediate Â¬ Built-up : 3,016 sq.	Francis Neow	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163537	simpang-ampat-41; Bandar Tasek Mutiara, Simpang Ampat	RM 690,000	Simpang Ampat, Penang	Semi-detached House Intermediate Â¬ Built-uj	Nicol Tan	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163538	simpang-ampat-41; Eco Bloom, Simpang Ampat	RM 470,000	Simpang Ampat, Penang	Condominium Intermediate Â¬ Built-up : 901 sq	Andy Chuah	https://www.iproperty.com.my/sale/simpang-ampat-41pqqd/all
163539						

Figure 3.2.2 Last 27 rows of data in uncleaned dataset (until 163538th row)

The dataset has the following fields as below:

1. Area - The state or city where the property is located.
2. Property Title - The title or headline of the property listing.
3. Property Price - The listed selling price of the property.
4. Property Location - A more specific address of the property.
5. Property Details - Detailed information about the property include size, type and furnishing.
6. Property Agent - The name of the real estate agent or agency responsible for selling or listing the property.
7. Source URL - The website link to the original source page.

3.3 Ethical considerations

To ensure ethical web scraping, we followed these key practices:

- Polite Crawling: Crawl delays and request throttling were applied to avoid overloading the server.
- No Sensitive Data: Only publicly available property listing information was collected. No personal or confidential data was accessed.
- Academic Use Only: All data is used strictly for educational purposes within the course scope and will not be used commercially.
- Secure Data Handling: The dataset is securely stored and not publicly shared.

4.0 Data Processing

4.1 Cleaning methods

Several cleaning methods are utilized to transition our newly built dataset of 160k+ data into a more readable format to help ease the process of analysis. More details regarding data cleaning can be referred to in [section 4.3](#).

Method	Involved Field	Description
Data Splitting	Property Title	By comma
	Property Location	By comma
	Property Detail	By separator
Title Casing	Property Title	First character uppercase
	Property Agent	First character uppercase
Punctuation Removal	Property Price	“RM”, comma
Data Type Conversion	Property Price	string to int
Special character replacement	Property Details	Multiple characters to “ ”
Numeric Value Extraction	Property Details	Extract property size
Filling Missing Values	Property Details	As “None”
Row Filtering	Property Agent	Remove non-agent name data
Null Data Removal	Entire Dataset	Remove data based on condition

Table 4.1 Data Cleaning Methods and related Fields

4.2 Data Structure

The cleaned property dataset was stored in an **SQLite database** for efficient data management.

A copy of the original DataFrame was created to preserve the raw data. The cleaned data was then saved into the SQLite database using the **to_sql()** method, where it was stored in a table named **cleaned_data**. If the table already existed, it was replaced with the new data.

The database file (**iproperty.db**) was then exported and made available for download in Google Drive. This allows easy access to the cleaned data for further analysis in SQLite-compatible applications.

Database Structure		
Database Structure	Browse Data	Edit Pragma
Execute SQL	Create Table	Create Index
Modify Table	Delete Table	Print
Refresh		
Name	Type	Schema
Tables (2)		
cleaned_data	CREATE TABLE "cleaned_data" ("Area Code" TEXT, "Property Title" TEXT, "Property Location (City)" TEXT, "Property	
raw_data	CREATE TABLE "raw_data" ("Area" TEXT, "Property Title" TEXT, "Property Price" TEXT, "Property Location" TEXT, "P	
Indices (0)		
Views (0)		
Triggers (0)		

Figure 4.2.1 Database Structure in SQLite

Area	Property Title	Property Price	Property Location	Property Details	Property Agent	Source URL
Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	perlis-zop7y Semi D 2 Tingkat - Taman Jaya Diri ...	RM 775,776	Kangar, Perlis	Semi-detached House Intermediate ...	Haneef	https://www.iproperty.com.my/sale/...
2	perlis-zop7y Semi D 1 Tingkat - Taman Nyu Indah 2...	RM 398,000	Arau, Perlis	Semi-detached House • Built-up : ...	Haneef	https://www.iproperty.com.my/sale/...
3	perlis-zop7y Taman Seri Manis Dua, Kangar	RM 306,000	Kangar, Perlis	Semi-detached House • Built-up : ...	Joyce Chan	https://www.iproperty.com.my/sale/...
4	perlis-zop7y Teres 1 Tingkat - Bandar Baharu Putr...	RM 185,000	Arau, Perlis	1-sty Terrace/Link House ...	Haneef	https://www.iproperty.com.my/sale/...
5	perlis-zop7y Teres 1 Tingkat - Bandar Baharu Putr...	RM 210,000	Arau, Perlis	1-sty Terrace/Link House ...	Haneef	https://www.iproperty.com.my/sale/...
6	perlis-zop7y Taman Seri Bintong Maju, Kangar	RM 247,000	Kangar, Perlis	1-sty Terrace/Link House Corner lo...	Leslie Low	https://www.iproperty.com.my/sale/...
7	perlis-zop7y Taman Tengku Budriah, Arau	RM 416,000	Arau, Perlis	Semi-detached House • Land area : ...	Leslie Low	https://www.iproperty.com.my/sale/...
8	perlis-zop7y Rumah Berkembar Setingkat di Jejawi, ...	RM 441,000	Kangar, Perlis	Semi-detached House • Built-up : ...	Lim	https://www.iproperty.com.my/sale/...
9	perlis-zop7y Taman Kuala Perlis, Kuala Perlis	RM 220,000	Kuala Perlis, Perlis	2-sty Terrace/Link House ...	Adam Wong	https://www.iproperty.com.my/sale/...
10	perlis-zop7y Teres (1) Setingkat di Irama ...	RM 298,000	Kangar, Perlis	1-sty Terrace/Link House • Built-u...	Haneef	https://www.iproperty.com.my/sale/...
11	perlis-zop7y Taman Sri Pauh, Arau	RM 171,000	Arau, Perlis	1-sty Terrace/Link House • Land ...	Joyce Chan	https://www.iproperty.com.my/sale/...
12	perlis-zop7y Taman Empiang Ceria, Kangar	RM 409,920	Kangar, Perlis	Semi-detached House Intermediate ...	Shafina	https://www.iproperty.com.my/sale/...
13	perlis-zop7y Padang Blak, Arau	RM 3,147,210	Arau, Perlis	Residential Land • Land area : 2.6...	Dickson Lum	https://www.iproperty.com.my/sale/...
14	perlis-zop7y Teres 2 Tingkat - Taman Bunga Padi ...	RM 180,000	Kangar, Perlis	2-sty Terrace/Link House ...	Haneef	https://www.iproperty.com.my/sale/...
15	perlis-zop7y Rumah Berkembar 1 Tingkat - Taman Sr...	RM 385,000	Kangar, Perlis	Semi-detached House Intermediate ...	Haneef	https://www.iproperty.com.my/sale/...
16	perlis-zop7y Teres 2 Tingkat - Taman Kemajuan - ...	RM 250,000	Kangar, Perlis	2-sty Terrace/Link House End lot ...	Haneef	https://www.iproperty.com.my/sale/...
17	perlis-zop7y Taman Padang Behor, Kangar	RM 370,000	Kangar, Perlis	Semi-detached House • Built-up : ...	Nur Suhaini	https://www.iproperty.com.my/sale/...
18	perlis-zop7y Kangar	RM 450,000	Kangar, Perlis	Semi-detached House Corner lot • ...	Nur Suhaini	https://www.iproperty.com.my/sale/...
19	perlis-zop7y Kangar	RM 385,000	Kangar, Perlis	Semi-detached House Intermediate ...	Nur Suhaini	https://www.iproperty.com.my/sale/...
20	perlis-zop7y (BAHARU) Taman Enyum Indah - Arau - ...	RM 355,000	Arau, Perlis	1-sty Terrace/Link House • Built-u...	Haneef	https://www.iproperty.com.my/sale/...
21	perlis-zop7y Taman Utara Jaya 1,, Kuala Perlis	RM 398,000	Kuala Perlis, Perlis	Semi-detached House Intermediate ...	Elwin	https://www.iproperty.com.my/sale/...

Figure 4.2.2 Raw data table in SQLite

Database Structure		
Database Structure	Browse Data	Edit Pragma
Execute SQL	Filter in any column	
Area Code	Property Title	Property Location (City)
Filter	Filter	Filter
1	perlis-zop7y Semi D 2 Tingkat - Taman Jaya Diri ...	Kangar
2	perlis-zop7y Semi D 1 Tingkat - Taman Nyu Indah 2...	Arau
3	perlis-zop7y Taman Seri Manis Dua	Kangar
4	perlis-zop7y Teres 1 Tingkat - Bandar Baharu Putr...	Arau
5	perlis-zop7y Teres 1 Tingkat - Bandar Baharu Putr...	Arau
6	perlis-zop7y Taman Seri Bintong Maju	Kangar
7	perlis-zop7y Rumah Berkembar Setingkat Di Jejawi	Kangar
8	perlis-zop7y Taman Kuala Perlis	Kuala Perlis
9	perlis-zop7y Teres (1) Setingkat Di Irama ...	Kangar
10	perlis-zop7y Taman Empiang Ceria	Kangar
11	perlis-zop7y Teres 2 Tingkat - Taman Bunga Padi ...	Kangar
12	perlis-zop7y Rumah Berkembar 1 Tingkat - Taman Sr...	Kangar
13	perlis-zop7y Teres 2 Tingkat - Taman Kemajuan - ...	Kangar
14	perlis-zop7y Taman Padang Behor	Kangar
15	perlis-zop7y Kangar	Kangar
16	perlis-zop7y Kangar	Kangar
17	perlis-zop7y (Baharu) Taman Enyum Indah - Arau - ...	Arau
18	perlis-zop7y Taman Utara Jaya 1	Kuala Perlis
19	perlis-zop7y Kangar	Kangar
20	perlis-zop7y Jalan Padang Katong	Kangar
21	perlis-zop7y Rumah Berkembar Setingkat Di Jejawi ...	Kangar
22	perlis-zop7y Taman Hijjah	Kangar
23	perlis-zop7y Taman Temak Jaya	Seriab

Figure 4.2.3 Cleaned data table in SQLite

4.3 Transformation and formatting

Reformatting Property Titles:

```
df['Property Title'] = df['Property Title'].str.split(',').str[0]
df['Property Title'] = df['Property Title'].str.title()
```

- Capitalized the first letter of each word.
- Separated titles by commas for improved readability.

Dividing Property Locations:

```
split_cols = df['Property Location'].str.split(' ', expand=True)
split_cols.columns = ['Property Location (City)', 'Property Location (State)']
df.drop(columns=['Property Location'], inplace=True)
df.insert(2, 'Property Location (City)', split_cols['Property Location (City)'])
df.insert(3, 'Property Location (State)', split_cols['Property Location (State)'])
```

- Split into two components: city and state.

Handling Property Prices:

```
df['Property Price'] = df['Property Price'].str.replace('RM', '', regex=False)
df['Property Price'] = df['Property Price'].str.replace(',', '', regex=False)
df['Property Price'] = pd.to_numeric(df['Property Price'], errors='coerce')
df.rename(columns={'Property Price': 'Property Price (RM)'}, inplace=True)
```

- Removed entries with NaN values to ensure data integrity.
- Converted prices from string format to integers.
- Renamed the column to "rm" and separated data by commas.

Standardizing Property Agent Names:

```
df['Property Agent'] = df['Property Agent'].str.title()
df['Property Agent'] = df['Property Agent'].apply(
    lambda x: None if re.search(r'\bsd\.\s*bhd\.\b', str(x), re.IGNORECASE) else x
)
```

- Capitalized the first letter of each agent's name.
- Filtered out non-agent names (e.g., company names).

Splitting Property Details:

```
def parse_property_details(detail):
    if pd.isnull(detail):
        return pd.Series([None, None, None])

    # clean messy code (eg: Â â€¢Â )
    clean_detail = re.sub(r'^\x00-\x7F+', ' ', str(detail))

    # extract Type (all character before Built-up)
    type_match = re.search(r"^(.*?)Built-up", clean_detail, re.IGNORECASE)
    if type_match:
        raw_type = type_match.group(1).strip()
        property_type = re.split(r'\s*\|\s*', raw_type)[0]
    else:
        property_type = None

    # extract Area
    area_match = re.search(r'Built[-\s]*up[^\d-9]*([\d,]+\s*sq\.\s*ft', clean_detail, re.IGNORECASE)
    area = area_match.group(1).replace(',', '') if area_match else None

    # extract Furnishing status
    furnishing = "Unknown"
    if re.search(r'\bUnfurnished\b', clean_detail, re.IGNORECASE):
        furnishing = 'Unfurnished'
    elif re.search(r'\bPartially Furnished\b', clean_detail, re.IGNORECASE):
        furnishing = 'Partially Furnished'
    elif re.search(r'\bFully Furnished\b', clean_detail, re.IGNORECASE):
        furnishing = 'Fully Furnished'
    elif re.search(r'\bFurnished\b', clean_detail, re.IGNORECASE):
        furnishing = 'Furnished'

    return pd.Series([
        property_type.strip() if property_type else None,
        float(area) if area else None,
        furnishing
    ])
```

- Divided into three categories: type, size (in sqft), and furnishing status using the separator “|”.
- Converted size from string to float for quantitative analysis.
- Removed entries with sizes below 70 sqft to eliminate illogical data.
- Replaced NaN values in furnishing status with "Unknown."

5.0 Optimization Techniques

5.1 Methods used

Person In Charge	Library
Lee Soon Der	Polars: A high performance DataFrame library designed for faster data processing, especially for large volumes of dataset. It is built with rust as its core, offers multi-threaded executions and supports lazy evaluation, making it more memory-efficient than pandas.
Lim Jing Yong	Modin & Ray: Modin is a high-performance drop-in replacement for pandas that speeds up data processing by automatically distributing operations across all CPU cores. Ray, which is a general purpose distributed execution engine, is used by Modin to perform the parallel operation efficiently.
Jaslene Yu	Multiprocessing: A library in Python allows a program to utilize multiple CPU cores simultaneously, enabling parallel execution of tasks. This is particularly useful for CPU-bound data processing operations where processing can be split across processes to reduce runtime.
	Spark: Apache Spark is a distributed computing framework designed for large-scale data processing across clusters. It optimizes performance using in-memory computation, lazy evaluation, and task parallelism. Spark's DataFrame and SQL APIs allow developers to express complex data workflows that Spark automatically optimizes under the hood using the Catalyst optimizer and Tungsten engine.
Nik Zulaikhaa	Joblib: Joblib is a parallel processing library in Python that simplifies running tasks concurrently using multiple CPU cores. It is ideal for speeding up repetitive, CPU-bound operations like data parsing by distributing workloads efficiently with minimal code changes.

5.2 Code Overview

Person In Charge	Code
Lee Soon Der	<p>Use CLI command to download Polars</p> <pre>[] pip install polars</pre> <p>Load csv in lazy mode</p> <pre>df_lazy = pl.read_csv("new_iproperty_dataset.csv").lazy()</pre> <p>Execute lazy plan to collect all previous cleaning processes and execute everything in parallel</p> <pre>df = df_lazy.collect()</pre>
Lim Jing Yong	<p>Use CLI command to download Modin with Ray</p> <pre>!pip install -U modin[ray] ray</pre> <p>Implement Modin into Pandas</p> <pre>import pandas as pd</pre> <p>to</p> <pre>import modin.pandas as pd</pre> <p>Import Ray library, initiate the engine and set it to the OS's environment for Modin to use it</p> <pre>import ray ray.init(ignore_reinit_error=True) os.environ["MODIN_ENGINE"] = "ray"</pre> <p>Confirming which engine Modin is using (another option is Dask)</p> <pre>import modin.config as modin_cfg print(modin_cfg.Engine.get())</pre> <pre>Ray</pre>
Jaslene Yu	<p>Multiprocessing</p> <p>Import Multiprocessing library</p> <pre>import multiprocessing as mp</pre>

	<p>mp.Pool.map() divides the tasks into smaller chunks in order to processes them in parallel using all available CPU cores mp.cpu_count()</p> <pre>def parallel_parse(series, func, processes=None): with mp.Pool(processes or mp.cpu_count()) as pool: results = pool.map(func, series) return pd.DataFrame(results, columns=['Property Type', 'Property Size (sqft)', 'Property Furnishing Status'])</pre> <p>To avoid the infinite loop of processes by ensuring that the multiprocessing pool is only created when the script is executed directly</p> <pre>if __name__ == "__main__":</pre> <p>Spark</p> <p>Import SparkSession class from PySpark's sql module</p> <pre>from pyspark.sql import SparkSession</pre> <p>Using Spark SQL API for further operation</p> <pre>spark = SparkSession.builder \ .appName("iProperty Data Cleaning") \ .getOrCreate()</pre> <p>UDF is applied to the DataFrame in parallel across the partitions of the data and PySpark will distribute the UDF tasks across worker nodes</p> <pre>df = df.withColumn("parsed", parse_udf(col("Property Details")))</pre>
Nik Zulaikhaa	<p>Use CLI command to download JobLib</p> <pre>!pip install joblib</pre> <p>Imported Parallel and delayed from JobLib</p> <pre>from joblib import Parallel, delayed</pre> <p>Joblib is used to parallelize parsing tasks across CPU cores</p> <pre>results = Parallel(n_jobs=-1)(delayed(parse_property_details)(d) for d in df['Property Details'])</pre>

6.0 Performance Evaluation

6.1 Before optimization

Below shows several screenshots of the basic data cleaning process's performance without optimization techniques. Session is restarted after every run to ensure memory usage is started from the initial state.

Run 1:

```
Performance Summary
→ Total Records Processed: 144214
🕒 Total Processing Time: 113.38 seconds
⚙️ Memory Used (Before → After): 163.79 MB → 1241.14 MB
▲ Peak Memory Usage: 456.80 MB
📈 Throughput: 1271.91 records/second
```

Run 2:

```
Performance Summary
→ Total Records Processed: 144214
🕒 Total Processing Time: 107.69 seconds
⚙️ Memory Used (Before → After): 163.89 MB → 1242.30 MB
▲ Peak Memory Usage: 461.97 MB
📈 Throughput: 1339.11 records/second
```

Run 3:






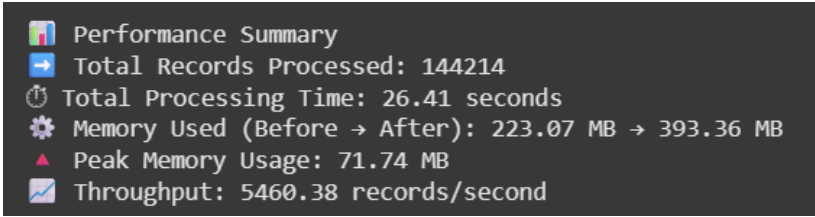
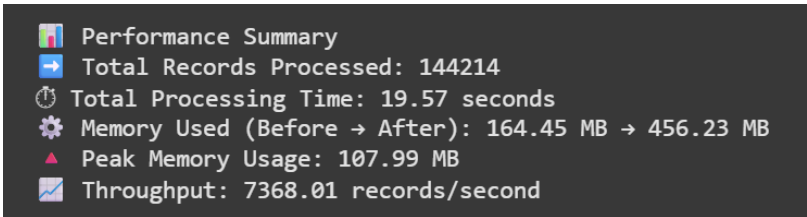
```
Performance Summary
→ Total Records Processed: 144214
🕒 Total Processing Time: 107.36 seconds
⚙️ Memory Used (Before → After): 164.23 MB → 1243.98 MB
▲ Peak Memory Usage: 461.96 MB
📈 Throughput: 1343.25 records/second
```













From the three runs:

Average Processing Time	109.48 seconds
Average Memory Used	1242.47 MB
Average Peak Memory Usage	460.24 MB
Average Throughput	1318.09 records/second

Table 6.1 Average Performance of 3 runs of unoptimized data cleaning

6.2 Comparison to Optimized Solution

Person In Charge	Performance
Lee Soon Der	 Elapsed Time: 5.15 sec  Memory Used (Start → End): 575.35 MB → 746.41 MB  Peak Memory (tracemalloc): 99.14 MB  Throughput: 27,990.23 records/sec  Total Records Cleaned: 144214 Processing Time Difference : 104.33 seconds Memory Usage Difference : 496.06 MB Peak Memory Usage Difference : 361.1 MB Throughput Difference : 26672.14 records/second
Lim Jing Yong	Modin & Ray  <p>Processing Time Difference : 83.07 seconds Memory Usage Difference : 849.11 MB Peak Memory Usage Difference : 388.5 MB Throughput Difference : 4142.29 records/second</p>
Jaslene Yu	Multiprocessing  <p>Processing Time Difference: 89.91 seconds Memory Usage Difference: 786.24 MB Peak Memory Usage Difference: 352.25 MB Throughput Difference: 6049.92 records/second</p>

	<p>Spark:</p> <div>  Performance Summary  Total Records Processed: 144211  Total Processing Time: 36.92 seconds  Memory Used (Before → After): 106.32 MB → 106.51 MB  Peak Memory Usage: 0.15 MB  Throughput: 3906.31 records/second </div> <p>Processing Time Difference: 72.56 seconds Memory Usage Difference: 1135.96 MB Peak Memory Usage Difference: 460.09 MB Throughput Difference: 2,588.22 records/second</p>
Nik Zulaikhaa	<p>Joblib:</p> <div>  Performance Summary  Total Records Processed: 144214  Total Processing Time: 20.19 seconds  Memory Used (Before → After): 182.28 MB → 567.96 MB  Peak Memory Usage: 385.68 MB  Throughput: 7141.52 records/second </div> <p>Processing Time Difference: 89.29 seconds Memory Usage Difference: 674.51 MB Peak Memory Usage Difference: 74.56 MB Throughput Difference: 5823.43 records/second</p>

6.3 Graph Visualization

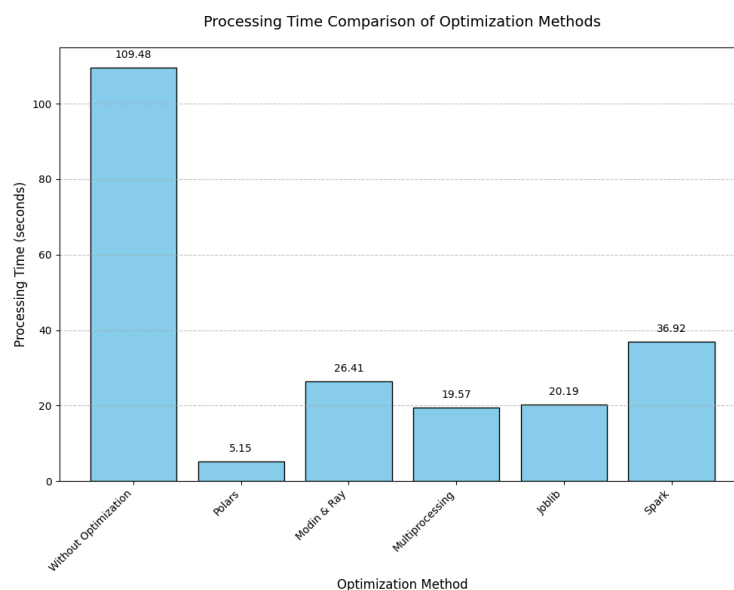


Figure 6.3.1 Graph of Comparison between Libraries (Processing Time)

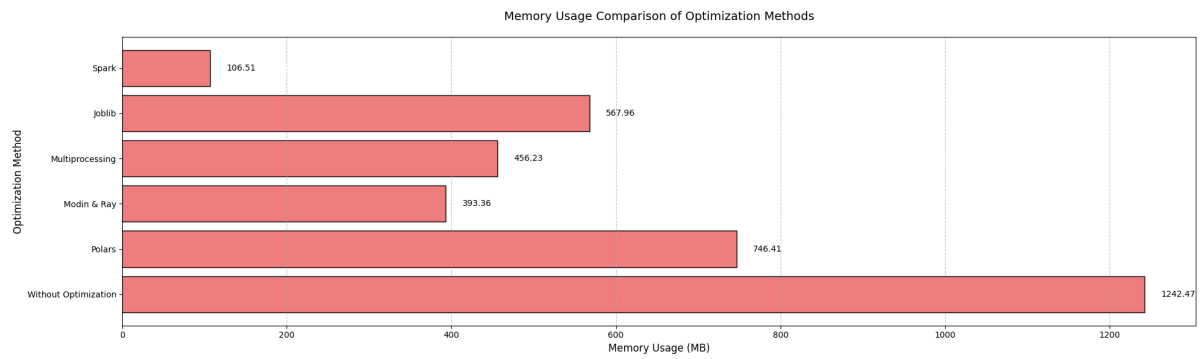


Figure 6.3.2 Graph of Comparison between Libraries (Memory Usage)

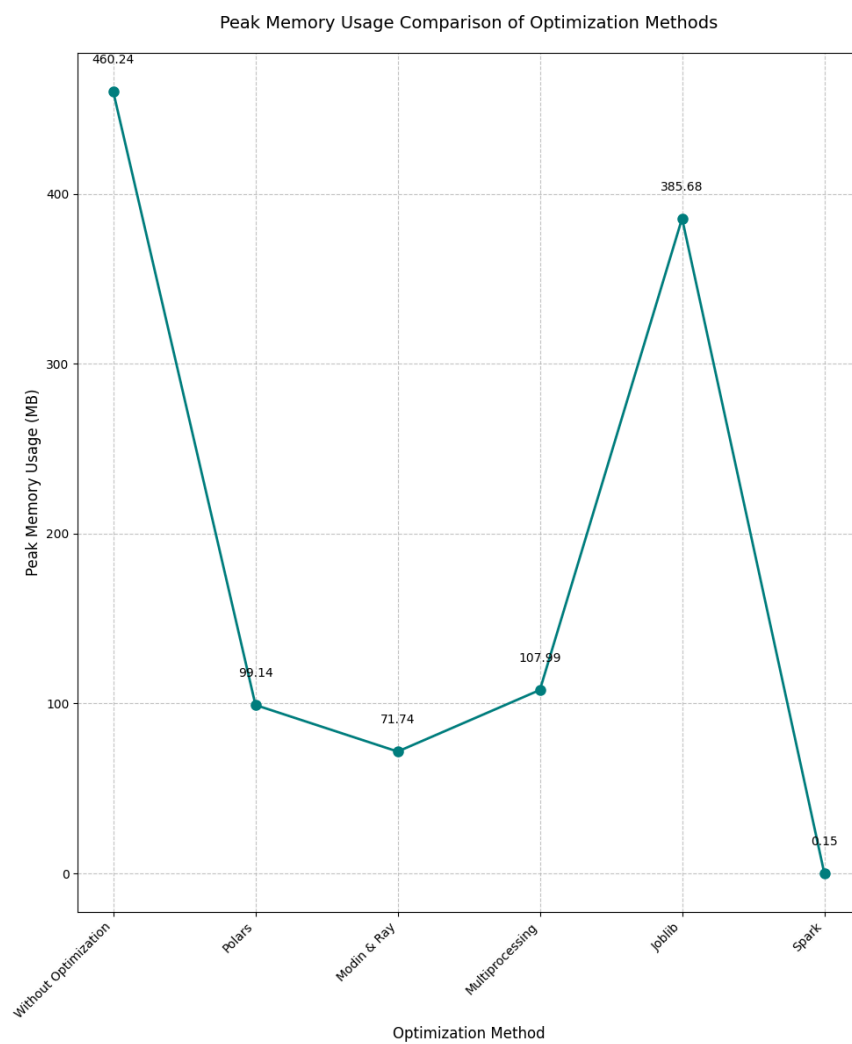


Figure 6.3.3 Graph of Comparison between Libraries (Peak Memory Usage)

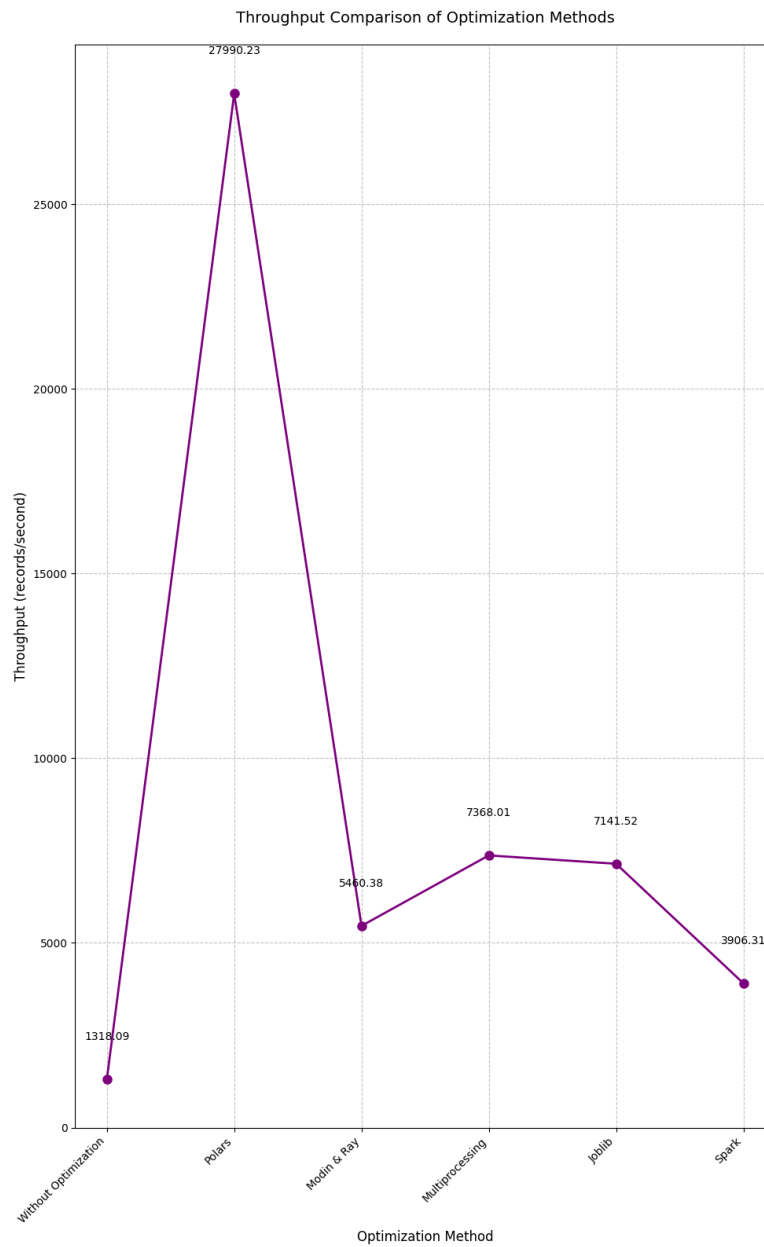


Figure 6.3.3 Graph of Comparison between Libraries (Throughput)

7.0 Challenges & Limitations

7.1 Challenges

- Iproperty website provides **multiple wrapper styles** (see [section 1.3](#)) which was unexpected initially. Due to BeautifulSoup's method of extracting data, additional modification must be done to our data scraping coding to scrap all wanted data.
- Iproperty website only has a maximum of 100 pages per filter, hence **area codes** must be listed individually in order to iterate between areas to scrap more than 100k data.
- During the data scraping process, Iproperty website's **CAPTCHA system** prevented us from extracting data. This issue was solved through usage of a user agent and an inconsistent (randomly timed) WebDriverWait() function.
- The property data in Iproperty website has **multiple blank columns and incorrect measurements** (exp. Property Size 1 sqft, NULL data for Property Furnishing Status, weird symbols in Property Name). This issue is addressed through criteria-based row dropping.
- Implementing Apache Spark for optimization required translating the existing data cleaning logic into Spark's **specific terminology and syntax**. For instance, operations like filtering and column manipulation, while conceptually similar to Pandas, necessitate the use of Spark's DataFrame API and functions (e.g., filter(), withColumn()), leading to a significant rewrite of the cleaning code.

7.2 Limitations

- An attempt was made to use Modin + Dask library for advanced optimization, however due to usage of **apply() function** in our data cleaning method, coding becomes too complicated and requires big modifications.
- Some high-performance libraries are **designed for processing massive volumes** of datasets with millions of rows such as Dask, Vaex and Apache Flink. When applied to smaller datasets like ours (~160k rows), the optimization performance will be lower than the lightweights tools like Pandas due to unnecessary overhead which results in slower execution and reduced efficiency.
- Although Joblib enabled parallel cleaning of the "Property Details" column, the task was lightweight, and multiprocessing overhead outweighed the benefits for small datasets (~240 rows), resulting in minimal performance gains.

- A key inconsistency with the Apache Spark implementation was a lower final row count compared to other methods, even with identical cleaning logic, likely due to differences in regex handling.

8.0 Conclusion

8.1 Our Findings

Throughout this project, we tested multiple Python libraries to perform high-performance data processing and compared their effectiveness in terms of processing time, memory usage, peak memory, and throughput. Based on our evaluations, we can rank the libraries from best to least efficient as follows:

1. Polars

- Best performance overall in terms of throughput (26,672.14 records/sec).
- Very fast due to Rust core, multi-threading, and lazy evaluation.
- Ideal for large-scale structured datasets.

2. Modin + Ray

- Great improvement over baseline, no need to change much code from pandas.
- Good CPU utilization with automatic parallelization.
- Slightly less efficient than Polars due to overhead from Ray.

3. Multiprocessing

- Effective CPU-bound task handling with custom control.
- Very flexible but requires careful management to avoid bugs.
- High throughput and memory savings.

4. Joblib

- Easy implementation for parallelizing loops and repetitive tasks.
- Performed well for parsing tasks, moderate improvements across metrics.

5. Apache Spark

- Strong for distributed processing, but showed relatively lower throughput (2,588.22 records/sec) in our use case.
- High memory overhead due to Spark's architecture.
- More suitable for truly massive datasets (millions of rows or distributed clusters).

In terms of the full process (scraping → cleaning → optimization), we observed:

- Web scraping was stable after implementing Selenium with anti-bot tactics.
- Cleaning pipeline was highly customizable; however, `apply()` usage limited compatibility with some optimization frameworks.
- Optimization techniques drastically reduced processing time from ~109s to as low as ~5s (Polars), proving the value of choosing the right tool for scale.

8.2 Improvement to be done

For future enhancements, the following improvements are recommended:

- Refactor cleaning code to avoid `apply()`: The current usage of `apply()` limits compatibility with high-performance libraries (e.g., Dask, Vaex). Refactoring using vectorized operations or built-in functions would enhance performance and reduce processing time.
- Mixing libraries for faster web scraping: Currently, Selenium and BeautifulSoup were used sequentially, which can be slow for large-scale scraping. Combining libraries like Scrapy (for high-speed crawling), Playwright (for better dynamic rendering), or aiohttp with asyncio (for concurrent HTTP requests) can significantly improve scraping speed and efficiency.
- Use Dask with proper refactoring: Though Dask was skipped due to `apply()` limitations, refactoring the code would allow its integration, offering strong parallelism and scalability for very large datasets.
- Benchmark with larger datasets: Testing on a dataset with millions of rows would provide more realistic insights into how libraries like Spark and Dask perform under true high-load conditions.
- Explore hybrid optimization pipelines: Combining strengths of multiple libraries—e.g., using Polars for initial fast processing and Spark for distributed querying—could balance speed and scalability.
- Automate and scale scraping pipelines: Tools like Airflow or Kafka could help automate and manage continuous scraping tasks for real-time or large-scale applications.

Implementing these improvements would enable the system to handle more data at higher speeds, with better compatibility and efficiency across different processing stages.

Appendix

Iproperty Website: <https://www.iproperty.com.my/sale/all-residential/>

Data Scraping Google Colab:

<https://colab.research.google.com/drive/13rqFmOfF7uuyUb1AsjxuVXrj2sjHaPgW?usp=sharing>

SQLite + Data Cleaning Google Colab:

https://colab.research.google.com/drive/1_jnPhfKX0OOMcsSbrDYIxRPJyHmyBOxP?usp=sharing

LogBook Github: <https://github.com/users/Jingyong14/projects/3>

Lim Jing Yong Optimization (Modin + Ray) Google Colab:

<https://colab.research.google.com/drive/1WthqdbUkXWVnRG9hnmNKKdJNTeTsuCX0?usp=sharing>

Lee Soon Der Optimization (Polars) Google Colab:

https://colab.research.google.com/drive/1j-R06GOCs3sAsXz6-0qcLT0PFf6Ee_jL?usp=sharing

Jaslene Yu Optimization (Multiprocessing + Spark) Google Colab:

<https://colab.research.google.com/drive/1a91UY1zZKkCBuA-a6MSroimJ8VQv5TTN?usp=sharing>

Nik Zulaikhaa Optimization (Joblib) Google Colab:

<https://colab.research.google.com/drive/1SvnCFwGiOnF4mSEyzNy58V2jCziaZILW?usp=sharing>

Visualization Google Colab:

https://colab.research.google.com/drive/1ETyMtF0aS4UgERg_PK4iGFIvApjYB_8o?usp=sharing