Project Analysis

1. Sales Performance (SQL)
2. MUDAH Condominium Rental KL (Python)

Presentation

Project Analysis

Sales Performance (SQL)

Given tables

- **Table 1**: Information from Supply Chain
- **Table 2**: Information from Sales

Normalize Tables

- **Step 1**: Normalize *Supply Chain* products
- **Step 2**: Normalize *Sales* descriptions
- **Step 3**: Fill missing values

Join Tables & Aggregation

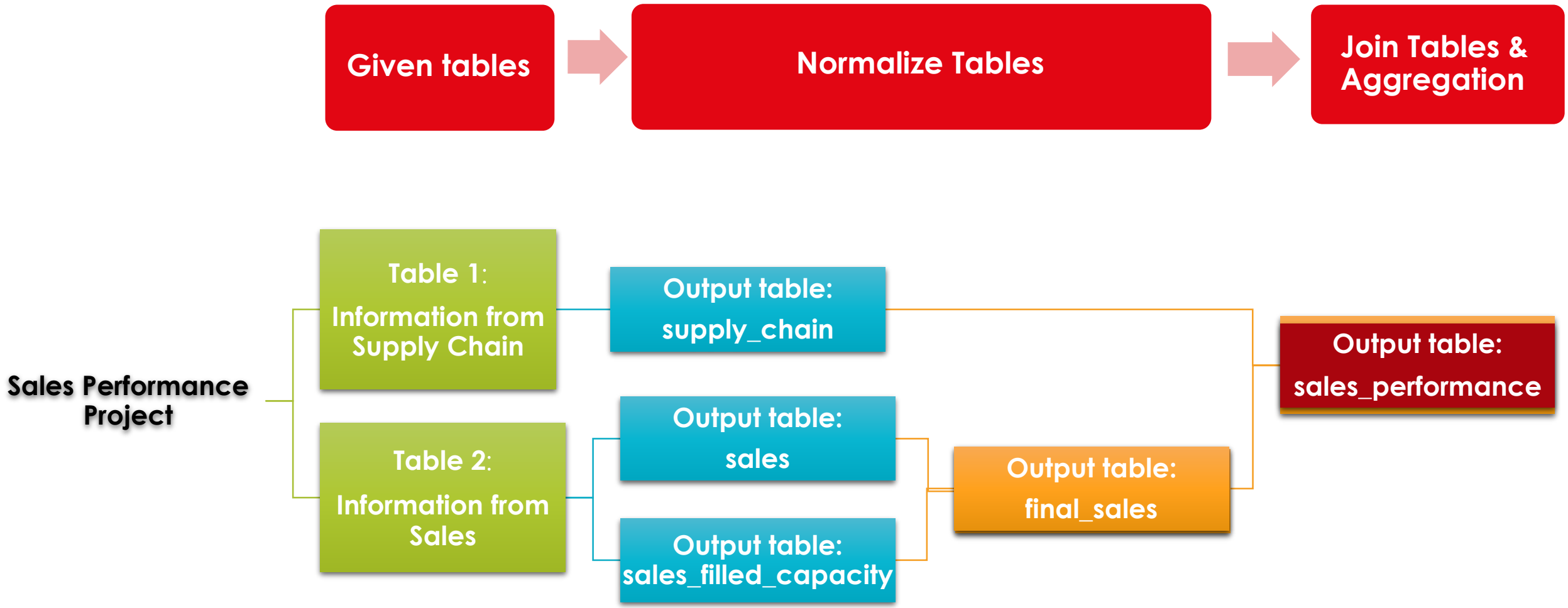
- **Step 4**: Join tables on Capacity *(64 GB/128 GB/256 GB)* & Aggregated to get *Sales Performance* table
  - Avg.Price
  - Sales Quantity

Sales Performance Report

- Report 'Sales Performance Summary by Product Capacity' to manager

Project Analysis

Sales Performance (SQL)



Project Analysis

Sales Performance (SQL)

Step 1:

Normalize *Supply Chain* products

Extract **Brand/Model/Capacity** from Table 1

```sql
-- Step 1: Normalize Supply Chain Table
With supply_chain AS (
  SELECT `Product ID`,
         'Apple' AS Brand,
         'iPhone 15' AS Model,
          TRIM(SUBSTRING_INDEX(`Product Name`,'/',-1)) AS Capacity
  FROM Information_from_Supply_Chain
  ),
```

Table 1: Information_from_Supply_Chain

Product ID	Product Name
1510	Apple iPhone 15 6 GB / 64 GB
1520	Apple iPhone 15 6 GB / 128 GB
1530	Apple iPhone 15 6 GB / 256 GB

Product ID	Brand	Model	Capacity
1510	Apple	iPhone 15	64 GB
1520	Apple	iPhone 15	128 GB
1530	Apple	iPhone 15	256 GB

Output Table: supply_chain

Project Analysis

Sales Performance (SQL)

Step 2:
Normalize *Sales* descriptions

→ Extract **Price/Model/Capacity** from Table 2

```
-
10 -- Step 2: Normalize Sales Table
11 sales AS (
12    SELECT Description,
13           `Price (MYR)`,
14           CASE WHEN Description LIKE '%iPhone 15%' THEN 'iPhone 15' ELSE NULL END AS Model,
15           REPLACE(REPLACE(REGEXP_SUBSTR(Description,'[0-9]{2,3} ?GB'),' ',''),'GB',' GB') AS Capacity
16    FROM Information_from_Sales
17    ),
18
```

Table 2: Information_from_Sales

Description	Price (MYR)
Apple iPhone 15 6 GB / 64 GB	1,250
Apple iPhone 15 256GB UCM11	999
Apple iPhone 15 64GB Grade A	1,249
Apple iPhone 15 256GB (UK Spec)	700
DD0026 Apple iPhone 15 128GB White	900
Apple iPhone 15 128 GB	1,299
Apple iPhone 15	1,250
Apple iPhone 15 6GB/128GB	999
iPhone 15 128GB	1,299
iPhone 15 6GB/128GB	1,199

→ Missing Capacity

Description	Price (MYR)	Model	Capacity
Apple iPhone 15 6 GB / 64 GB	1,250	iPhone 15	64 GB
Apple iPhone 15 256GB UCM11	999	iPhone 15	256 GB
Apple iPhone 15 64GB Grade A	1,249	iPhone 15	64 GB
Apple iPhone 15 256GB (UK Spec)	700	iPhone 15	256 GB
DD0026 Apple iPhone 15 128GB White	900	iPhone 15	128 GB
Apple iPhone 15 128 GB	1,299	iPhone 15	128 GB
Apple iPhone 15	1,250	iPhone 15	null
Apple iPhone 15 6GB/128GB	999	iPhone 15	128 GB
iPhone 15 128GB	1,299	iPhone 15	128 GB
iPhone 15 6GB/128GB	1,199	iPhone 15	128 GB

→ Missing Value

Output Table: sales

Project Analysis

Sales Performance (SQL)

Step 3:
Fill missing values

→ Imputed With Nearest price's Capacity from Table 2

```sql
-- Step 3: Imputed NULL values with Nearest Price's Capacity
sales_filled_capacity AS (
  SELECT S1.*,
        (SELECT S2.Capacity
         FROM sales S2
         WHERE S2.Capacity IS NOT NULL
         ORDER BY ABS(S2.`Price (MYR)`-S1.`Price (MYR)`) ASC
         LIMIT 1) AS Imputed_Capacity
  FROM sales S1
  WHERE S1.Capacity IS NULL
  ),

final_sales AS (
  SELECT S3.Description, S3.`Price (MYR)`, S3.Model,
            COALESCE(S3.Capacity,S4.Imputed_Capacity) AS Capacity
  FROM sales S3
  LEFT JOIN sales_filled_capacity S4
  ON S3.Description=S4.Description
  AND S3.`Price (MYR)`=S4.`Price (MYR)`
  )
```

Table 2: Information_from_Sales

Description	Price (MYR)
Apple iPhone 15 6 GB / 64 GB	1,250
Apple iPhone 15 256GB UCM11	999
Apple iPhone 15 64GB Grade A	1,249
Apple iPhone 15 256GB (UK Spec)	700
DD0026 Apple iPhone 15 128GB White	900
Apple iPhone 15 128 GB	1,299
Apple iPhone 15	1,250
Apple iPhone 15 6GB/128GB	999
iPhone 15 128GB	1,299
iPhone 15 6GB/128GB	1,199

Nearest Price

Description	Price (MYR)	Model	Capacity
Apple iPhone 15 6 GB / 64 GB	1,250	iPhone 15	64 GB
Apple iPhone 15 256GB UCM11	999	iPhone 15	256 GB
Apple iPhone 15 64GB Grade A	1,249	iPhone 15	64 GB
Apple iPhone 15 256GB (UK Spec)	700	iPhone 15	256 GB
DD0026 Apple iPhone 15 128GB White	900	iPhone 15	128 GB
Apple iPhone 15 128 GB	1,299	iPhone 15	128 GB
Apple iPhone 15	1,250	iPhone 15	64 GB
Apple iPhone 15 6GB/128GB	999	iPhone 15	128 GB
iPhone 15 128GB	1,299	iPhone 15	128 GB
iPhone 15 6GB/128GB	1,199	iPhone 15	128 GB

Output Table: final_sales

Project Analysis

Sales Performance (SQL)

Step 4:
Join tables on Capacity

→

Aggregated
Avg Price & Sales Qty

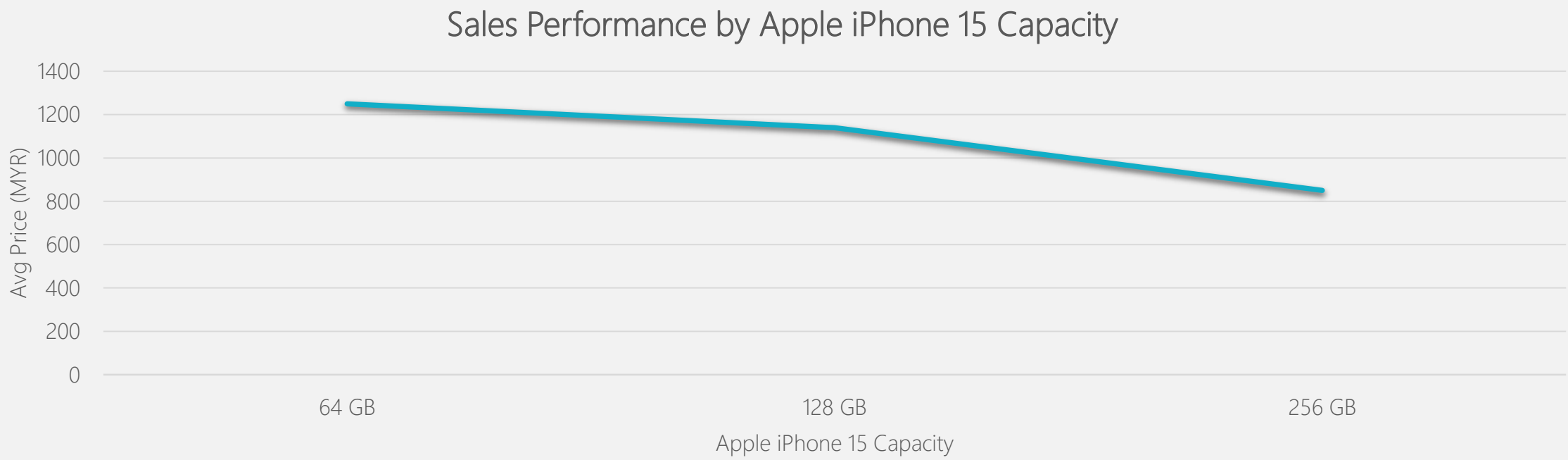
```sql
-- Step 4: Joined tables & Aggregated to get final report
SELECT C.`Product ID`,C.Brand,C.Model,FS.Capacity,
       FORMAT(AVG(FS.`Price (MYR)`),0) AS `Average Price (MYR)`,
       COUNT(*) AS `Sales Quantity`
FROM final_sales FS
LEFT JOIN supply_chain C
ON FS.Capacity=C.Capacity
GROUP BY C.`Product ID`,C.Brand,C.Model,FS.Capacity
ORDER BY C.`Product ID`
```

Product ID	Brand	Model	Capacity	Average Price (MYR)	Sales Quantity
1510	Apple	iPhone 15	64 GB	1,250	3
1520	Apple	iPhone 15	128 GB	1,139	5
1530	Apple	iPhone 15	256 GB	850	2

Output Table: sales_performance

Project Analysis

Sales Performance (SQL)



64 GB

MYR 1,250

Sales Quantity : 3

128 GB

MYR 1,139

Sales Quantity : 5

256 GB

MYR 850

Sales Quantity : 2

Project Analysis

MUDAH Web Scrape (Python)

WEB SCRAPING FOR APARTMENTS AND CONDOMINIUMS IN KUALA LUMPUR FROM MUDAH

Scrapes all listing pages	• Install packages (*requests* + *beautifulsoup*) • Extracts Property Name, Area, Size, Rental
Normalization	• Cleans text with regex • Handles missing/null fields
Aggregation	• Grouping Area and Property Name • Aggregate average rental and size
Exports	• Exports to CSV • Create dashboards

Project Analysis

MUDAH Web Scrape (Python)

Scrapes all listing pages

- Install packages (*requests* + *beautifulsoup*)
- Extracts Property Name, Area, Size, Rental

Packages Install

```
!pip install requests
!pip install beautifulsoup4
!pip install pandas
!pip install tqdm
```

```
import requests
from bs4 import BeautifulSoup
import json
import re
import pandas as pd
from tqdm import tqdm
```

Project Analysis

MUDAH Web Scrape (Python)

Scrapes all listing pages

- Install packages (*requests* + *beautifulsoup*)
- Extracts Property Name, Area, Size, Rental

Scrapes all listing pages

```
data=[]
page = 1

# tqdm with manual update
pbar = tqdm(desc="Scraping pages", unit="page")

while True:
  url=f"https://www.mudah.my/kuala-lumpur/apartment-condominium-for-rent?o={page}"
  response=requests.get(url)
  soup=BeautifulSoup(response.text,"html.parser")

  listings=soup.find_all('div',class_="w_100% p_12px_16px d_flex flex-d_column jc_space-between ai_stretch"

  # Stop if no more listings
  if not listings:
      break

  for listing in listings:
```

Website url

soup.prettify()

```
<div class="w_100% p_12px_16px d_flex flex-d_column jc_space-between ai_stretch">
          <a class="w_full d_flex flex-d_column gap_1 mdDown:gap_0" data-adid="1303794
           <div class="fs_12px font-style_normal fw_400 lh_16px c_var(--mudah-colors-t
            <p>
             Apartment
```

Project Analysis

MUDAH Web Scrape (Python)

Scrapes all listing pages

- Install packages (*requests* + *beautifulsoup*)
- Extracts Property Name, Area, Size, Rental

```python
# Extract property name (Remove text in parentheses
property_name = listing.find('h3', class_='c_black').get_text(strip=True)
property_name=re.sub(r"\s*.*?", "", property_name)
property_name=property_name.split(',')[0]
if '@' in property_name:
    property_name=property_name.split('@')[0]
else:
    property_name=property_name

# Extract area(Remove text in parentheses)
area=listing.find('h3', class_='c_black').get_text(strip=True)
area=re.sub(r"\s*.*?", "", area)
area=area.split(',')[1] if len(area) > 1 else None

# Extract size number only if followed by 'sq.ft'
size_number = listing.find('span', class_='fs_sm lh_1.25rem font-style_normal fw_bold c_var(--mudah-colors-text-hi-emp)')
size_unit = listing.find('span', class_='c_var(--mudah-colors-text-hi-emp) fs_sm lh_1.25rem font-style_normal fw_normal')
if size_number and size_unit:
    number_text = size_number.get_text(strip=True)
    unit_text = size_unit.get_text(strip=True)
    if 'sq.ft' in unit_text:
      size = int(re.sub(r'[^\d]', '', number_text))
    else:
      size = None
else:
  size = None

# Extract Rental (Remove everything except digits)
price_rm=listing.find('span', class_="currPrice")
if price_rm:
    price_text = price_rm.get_text(strip=True)
    price = int(re.sub(r'[^\d]', '', price_text))
else:
    price = None
```

Extract Property Name & Area

Extract Size (sqft)

Extract Rental

soup.prettify()

```
<div class="ov_hidden c_#000 tov_ellipsis ff_OpenSa
 <h3 class="c_black">
  Residensi Brickfields, Brickfields
 </h3>
</div>
<div class="d_flex cg_24px rg_4px pt_12px pb_8px">
 <div class="d_flex ai_center ac_center gap_1">
  <span class="d_flex ai_center ac_center c_var(--m
   <img alt="proprty-spce-icon" src="https://mcdn.m
  </span>
  <span class="fs_sm lh_1.25rem font-style_normal f
   800
  </span>
  <span class="c_var(--mudah-colors-text-hi-emp) fs
   sq.ft
  </span>
```

soup.prettify()

```
 <span class="currPrice">
  RM 2,500
  <!-- -->
  <span class="perMonthLabel">
   per month
  </span>
```

Project Analysis

MUDAH Web Scrape (Python)

Normalization

- Cleans text with regex
- Handles missing/null fields

```python
# Extract property name (Remove text in parentheses
property_name = listing.find('h3', class_='c_black').get_text(strip=True)
property_name=re.sub(r"\s*.*?", "", property_name)
property_name=property_name.split(',')[0]
if '@' in property_name:
    property_name=property_name.split('@')[0]
else:
    property_name=property_name

# Extract area(Remove text in parentheses)
area=listing.find('h3', class_='c_black').get_text(strip=True)
area=re.sub(r"\s*.*?", "", area)
area=area.split(',')[1] if len(area) > 1 else None

# Extract size number only if followed by 'sq.ft'
size_number = listing.find('span', class_='fs_sm lh_1.25rem font-style_normal fw_bold c_var(--mudah-colors-text-hi-emp)')
size_unit = listing.find('span', class_='c_var(--mudah-colors-text-hi-emp) fs_sm lh_1.25rem font-style_normal fw_normal')
if size_number and size_unit:
    number_text = size_number.get_text(strip=True)
    unit_text = size_unit.get_text(strip=True)
    if 'sq.ft' in unit_text:
      size = int(re.sub(r'[^\d]', '', number_text))
    else:
      size = None
else:
  size = None

# Extract Rental (Remove everything except digits)
price_rm=listing.find('span', class_="currPrice")
if price_rm:
    price_text = price_rm.get_text(strip=True)
    price = int(re.sub(r'[^\d]', '', price_text))
else:
    price = None
```

Normalization for

- Property name - str
- Area - str
- Size (sqft) - int
- Rental (MYR) - int

Project Analysis

MUDAH Web Scrape (Python)

Normalization

- Cleans text with regex
- Handles missing/null fields

```
  page += 1
  pbar.update(1)

pbar.close()
df=pd.DataFrame(data)

# replace any missing with 'None'
df.fillna("None", inplace=True)
df = df.dropna(subset=["Rental (MYR)","Size (Squared Feet)"])
```

Handle Missing Value

- Replace any missing vale with ‘None’
- Drop all rows contain ‘None’ especially in Rental & Size columns

Project Analysis

MUDAH Web Scrape (Python)

Aggregation

- Grouping Area and Property Name
- Aggregate average rental and size

Groupby

Aggregation

Find Average Price & Average Size

```
#Group & Summarize
summary=df.groupby(["Area","Property Name"]).agg({"Rental (MYR)":"mean","Size (Squared Feet)":"mean"}).reset_index()
summary.rename(columns={"Size (Squared Feet)":"Average Size (Squared Feet)","Rental (MYR)":"Average Rental (MYR)"},inplace=True)
summary["Average Size (Squared Feet)"]=summary["Average Size (Squared Feet)"].round(0).astype(int)
summary["Average Rental (MYR)"]=summary["Average Rental (MYR)"].round(0).astype(int)
sorted_summary = summary.sort_values(by=["Average Rental (MYR)","Average Size (Squared Feet)"], ascending=[False,False])

sorted_summary
```

Project Analysis

MUDAH Web Scrape (Python)

Aggregation

- Grouping Area and Property Name
- Aggregate average rental and size

sorted_summary

Scraping pages: 25page [01:15, 3.03s/page]

	Area	Property Name	Average Rental (MYR)	Average Size (Squared Feet)
245	Mont Kiara	10 Mont Kiara	14000	3720
160	KL City	One KL	14000	3285
252	Mont Kiara	Serene Mont Kiara	14000	2918
251	Mont Kiara	Pavilion Hilltop	13000	2767
20	Bangsar	The Loft	12888	3800
...	...	...	...	...
208	Kepong	Taman Bukit Desa	750	688
269	Old Klang Road	Sri Lempah	750	600
137	Desa Petaling	Desa Sri Puteri B Apartments	700	650
135	Desa Petaling	Desa Petaling Flat	700	600
75	Cheras	Cheras Ria	600	570

433 rows × 4 columns

Project Analysis

MUDAH Web Scrape (Python)

Exports

- Exports to CSV
- Create dashboards

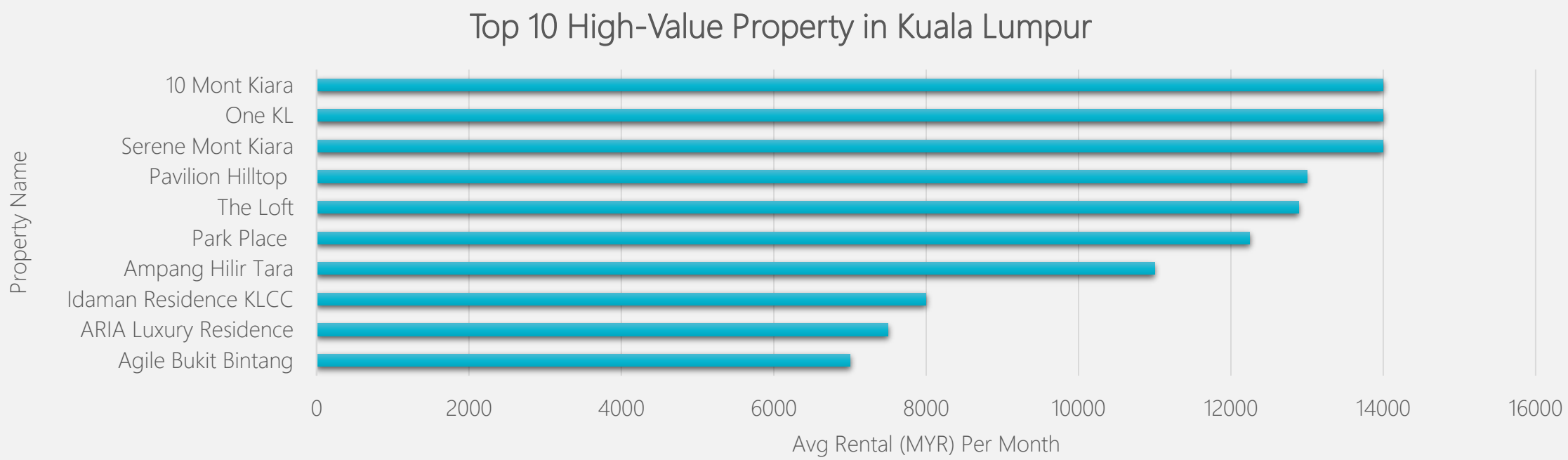
csv.file

```
sorted_summary.to_csv('mudah_rental_data.csv', index=False)
```

Area	Property Name	Average Rental (MYR)	Average Size (Squared Feet)
Mont Kiara	10 Mont Kiara	14000	3720
KL City	One KL	14000	3285
Mont Kiara	Serene Mont Kiara	14000	2918
Mont Kiara	Pavilion Hilltop	13000	2767
Bangsar	The Loft	12888	3800
Desa ParkCity	Park Place	12250	1780
Ampang Hilir	Ampang Hilir Tara	11000	2600
KLCC	Idaman Residence KLCC	7999	1700
KLCC	ARIA Luxury Residence	7500	1000
Bukit Bintang	Agile Bukit Bintang	7000	950

Project Analysis

MUDAH Web Scrape (Python)



TOP 1

10 Mont Kiara

Area: Mont Kiara
Average Size(Sq.ft): 3,720
Average Rental (MYR) : 14,000

TOP 2

One KL

Area: KL City
Average Size(Sq.ft): 3,285
Average Rental (MYR) : 14,000

TOP 3

Serene Mont Kiara

Area: Mont Kiara
Average Size(Sq.ft): 2,918
Average Rental (MYR) : 14,000

Project Analysis

MUDAH Web Scrape (Python)

Python Code:

```
!pip install requests
!pip install beautifulsoup4
!pip install pandas
!pip install tqdm

import requests
from bs4 import BeautifulSoup
import json
import re
import pandas as pd
from tqdm import tqdm

data=[]
page = 1

# tqdm with manual update
pbar = tqdm(desc="Scraping pages", unit="page")

while True:
  url=f"https://www.mudah.my/kuala-lumpur/apartment-condominium-for-rent?o={page}"
  response=requests.get(url)
  soup=BeautifulSoup(response.text,"html.parser")
  #soup.prettify()

  listings=soup.find_all('div',class_="w_100% p_12px_16px d_flex flex-d_column jc_space-between ai_stretch")

  # Stop if no more listings
  if not listings:
      break

  for listing in listings:

      # Extract property name (Remove text in parentheses
      property_name = listing.find('h3', class_='c_black').get_text(strip=True)
      property_name=re.sub(r"\s*\(.*?\)", "", property_name)
      property_name=property_name.split(',')[0]
      if '@' in property_name:
          property_name=property_name.split('@')[0]
      else:
          property_name=property_name

      # Extract area(Remove text in parentheses)
      area=listing.find('h3', class_='c_black').get_text(strip=True)
      area=re.sub(r"\s*\(.*?\)", "", area)
      area=area.split(',')[1] if len(area) > 1 else None

      # Extract size number only if followed by 'sq.ft'
      size_number = listing.find('span', class_='fs_sm lh_1.25rem font-style_normal fw_bold c_var(--mudah-colors-text-hi-emp)')
      size_unit = listing.find('span', class_='c_var(--mudah-colors-text-hi-emp) fs_sm lh_1.25rem font-style_normal fw_normal')
      if size_number and size_unit:
          number_text = size_number.get_text(strip=True)
          unit_text = size_unit.get_text(strip=True)
          if 'sq.ft' in unit_text:
            size = int(re.sub(r'[^\d]', '', number_text))
          else:
            size = None
      else:
        size = None

      # Extract Rental (Remove everything except digits)
      price_rm=listing.find('span', class_="currPrice")
      if price_rm:
          price_text = price_rm.get_text(strip=True)
          price = int(re.sub(r'[^\d]', '', price_text))
      else:
          price = None


      data.append({'Area': area,
                   'Property Name':property_name,
                   'Rental (MYR)':price,
                   'Size (Squared Feet)': size
                  })

  page += 1
  pbar.update(1)

pbar.close()
df=pd.DataFrame(data)

# replace any missing with 'None'
df.fillna("None", inplace=True)
df = df.dropna(subset=["Rental (MYR)","Size (Squared Feet)"])



#Group & Summarize
summary=df.groupby(["Area","Property Name"]).agg({"Rental (MYR)":"mean","Size (Squared Feet)":"mean"}).reset_index()
summary.rename(columns={"Size (Squared Feet)":"Average Size (Squared Feet)","Rental (MYR)":"Average Rental (MYR)"},inplace=True)
summary["Average Size (Squared Feet)"]=summary["Average Size (Squared Feet)"].round(0).astype(int)
summary["Average Rental (MYR)"]=summary["Average Rental (MYR)"].round(0).astype(int)
sorted_summary = summary.sort_values(by=["Average Rental (MYR)","Average Size (Squared Feet)"], ascending=[False,False])

sorted_summary


sorted_summary.to_csv('mudah_rental_data.csv', index=False)
```

Project Analysis

Sales Performance (SQL)

SQL Code:

```
-- Step 1: Normalize Supply Chain Table
With supply_chain AS (
  SELECT `Product ID`,
             'Apple' AS Brand,
             'iPhone 15' AS Model,
              TRIM(SUBSTRING_INDEX(`Product Name`,'/',-1)) AS Capacity
  FROM Information_from_Supply_Chain
  ),

-- Step 2: Normalize Sales Table
sales AS (
   SELECT Description,
              `Price (MYR)`,
               CASE WHEN Description LIKE '%iPhone 15%' THEN 'iPhone 15' ELSE NULL END AS Model,
               REPLACE(REPLACE(REGEXP_SUBSTR(Description,'[0-9]{2,3} ?GB'),' ',''),'GB',' GB') AS Capacity

   FROM Information_from_Sales
   ),

-- Step 3: Imputed NULL values with Nearest Price's Capacity
sales_filled_capacity AS (
  SELECT S1.*,
        (SELECT S2.Capacity
         FROM sales S2
         WHERE S2.Capacity IS NOT NULL
         ORDER BY ABS(S2.`Price (MYR)`-S1.`Price (MYR)`) ASC
         LIMIT 1) AS Imputed_Capacity
  FROM sales S1
  WHERE S1.Capacity IS NULL
  ),

final_sales AS (
  SELECT S3.Description, S3.`Price (MYR)`, S3.Model,
          COALESCE(S3.Capacity,S4.Imputed_Capacity) AS Capacity
  FROM sales S3
  LEFT JOIN sales_filled_capacity S4
  ON S3.Description=S4.Description
  AND S3.`Price (MYR)`=S4.`Price (MYR)`
  )


-- Step 4: Joined tables & Aggregated to get final report
SELECT C.`Product ID`,C.Brand,C.Model,FS.Capacity,
       FORMAT(AVG(FS.`Price (MYR)`),0) AS `Average Price (MYR)`,
       COUNT(*) AS `Sales Quantity`
FROM final_sales FS
LEFT JOIN supply_chain C
ON FS.Capacity=C.Capacity
GROUP BY C.`Product ID`,C.Brand,C.Model,FS.Capacity
ORDER BY C.`Product ID`
```

Thank You