Hotel Price Prediction Example Project

王馬丁

25/07

Plan

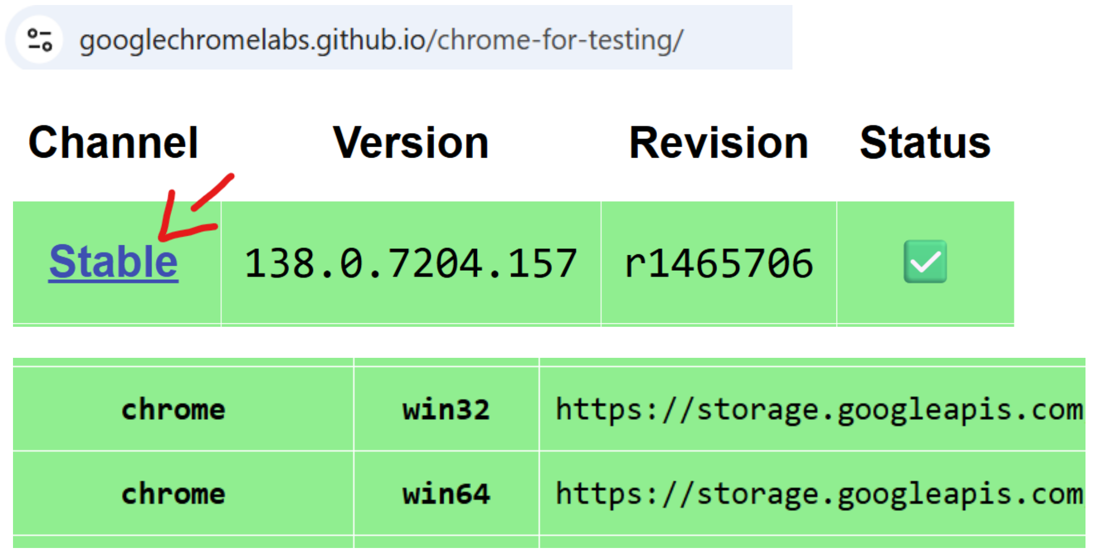
1. Scraping Hotel Price Data using **Python, BeautifulSoup, Selenium.**
2. Cleaning Data with **Power Query, Excel.**
3. Managing and Analyzing Data with **SQL, MySQL.**
4. Visualizing Data with **Looker Studio.**
5. **Machine Learning** Regression Model in **PyTorch**.

Concept

1. Comparing Hotel Prices in Taipei Metropolitain Area
2. Based on Month and Franchise Brand
3. Comparing chains of Marriott, Hilton & IHG
4. Analyze Trends and Extract a Lesson.
5. Build and Train Machine Learning Model to Predict Prices.

1A Download and Use Chromedriver

- Required for scraping through Chrome Browser
- Choosing location, as path is needed for further process.



googlechromelabs.github.io/chrome-for-testing/

Channel	Version	Revision	Status
Stable	138.0.7204.157	r1465706	✓

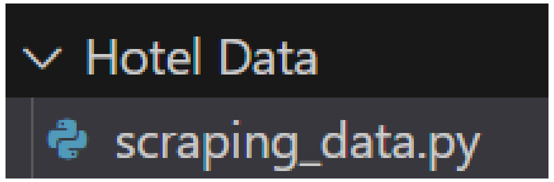
chrome	win32	https://storage.googleapis.com
chrome	win64	https://storage.googleapis.com

1B Choosing Booking.com for reference

- Easier to use than several separate hotel websites, because the scraping does not need to prepare for different clickables, cookie walls and protection systems.
- From a statistical best practice point of view, using the same reference base allows more justified comparison. Third-party sites may add some fees, therefore using one source for all the different Hotels and Dates, makes the comparison fairer.

1C Creating Base Folder in Visual Studio

- For Scraping Booking.com, we opt for using Python code, with Selenium & BeautifulSoup.
- We create base folder for the project, with scraping_data.py for command.



```
Hotel Data
    scraping_data.py
```

1D Booking URL Logic

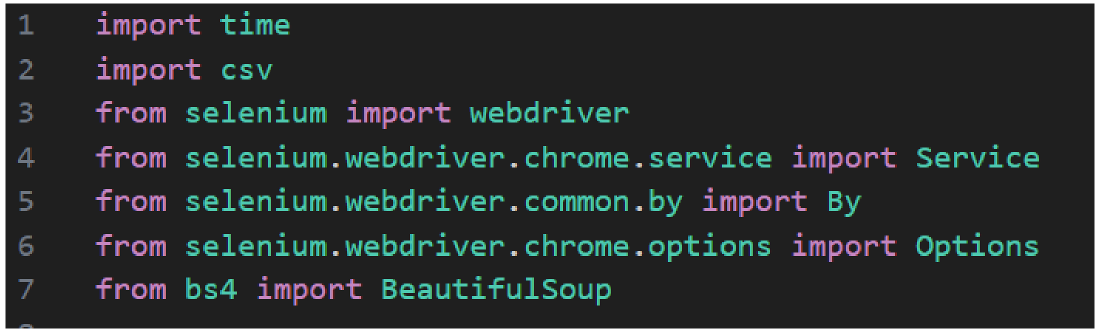
- After checking source code, the URL of Booking follows the below logic:

www.booking.com/**hotel**/**tw**/**hilton-taipei-sinban**.en-gb.html?checkin=**{checkin}**&checkout=**{checkout}**

- We select 6 hotels from Taipei Metropolitain Area, 2 each from the 3 franchise chains:
    - Marriott Taipei (Marriott)
    - Four Points by Sheraton Linkou (Marriott)
    - Hilton Taipei Sinban (Hilton)
    - Humble House Curio Collection (Hilton)
    - Holiday Inn Express Taoyuan (IHG)
    - Holiday Inn Express Taipei Main Station (IHG)
- We collect the right URL for each hotel, then add in {checkin} and {checkout}, as we plan to vary the dates.

1E Import scraping libraries

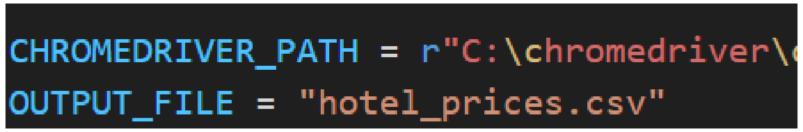
- We inquire Selenium, BeautifulSoup, and all scraping libraries, import them to scraping_data.py.



```python
import time
import csv
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
from bs4 import BeautifulSoup
```

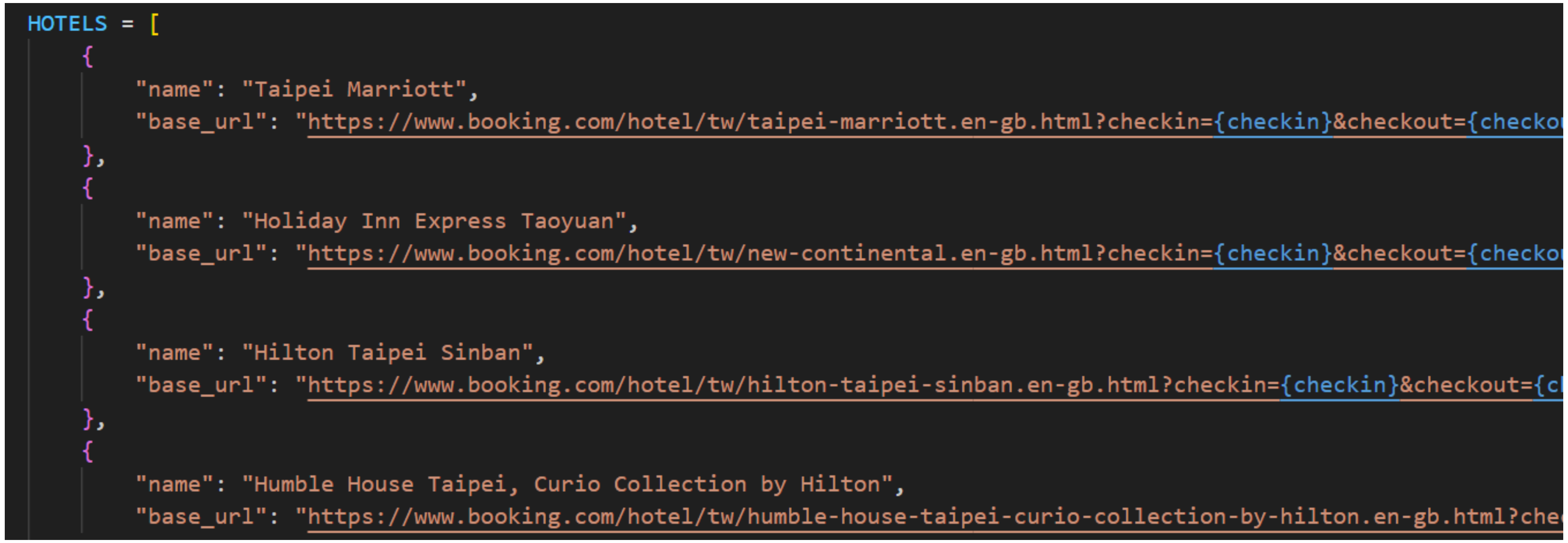
1F Complete Python Code for Scraping

- Use Chromedriver path and indicate output file as hotel_prices.csv



```
CHROMEDRIVER_PATH = r"C:\chromedriver\
OUTPUT_FILE = "hotel_prices.csv"
```

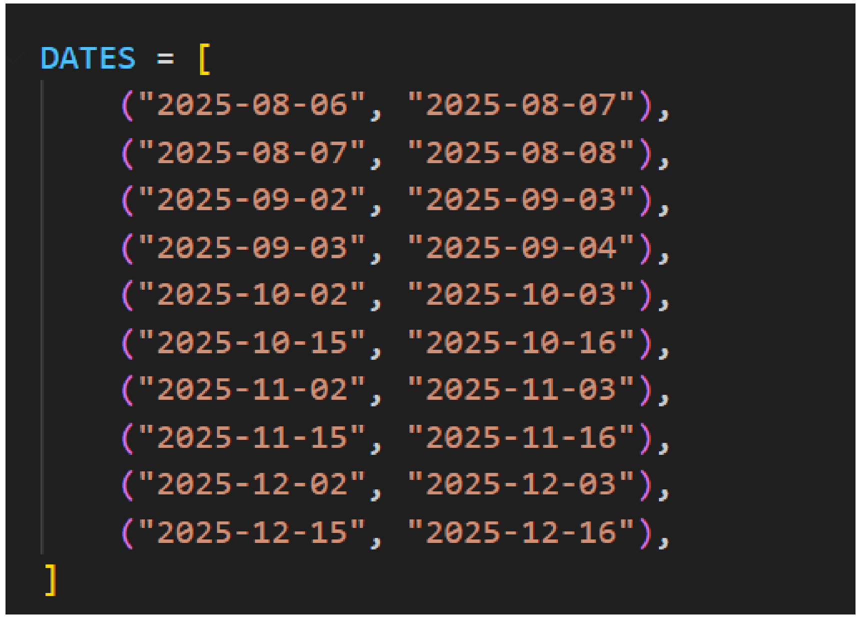
- Use previously collected hotel links.



```
HOTELS = [
    {
        "name": "Taipei Marriott",
        "base_url": "https://www.booking.com/hotel/tw/taipei-marriott.en-gb.html?checkin={checkin}&checkout={checko
    },
    {
        "name": "Holiday Inn Express Taoyuan",
        "base_url": "https://www.booking.com/hotel/tw/new-continental.en-gb.html?checkin={checkin}&checkout={checko
    },
    {
        "name": "Hilton Taipei Sinban",
        "base_url": "https://www.booking.com/hotel/tw/hilton-taipei-sinban.en-gb.html?checkin={checkin}&checkout={c
    },
    {
        "name": "Humble House Taipei, Curio Collection by Hilton",
        "base_url": "https://www.booking.com/hotel/tw/humble-house-taipei-curio-collection-by-hilton.en-gb.html?che
```

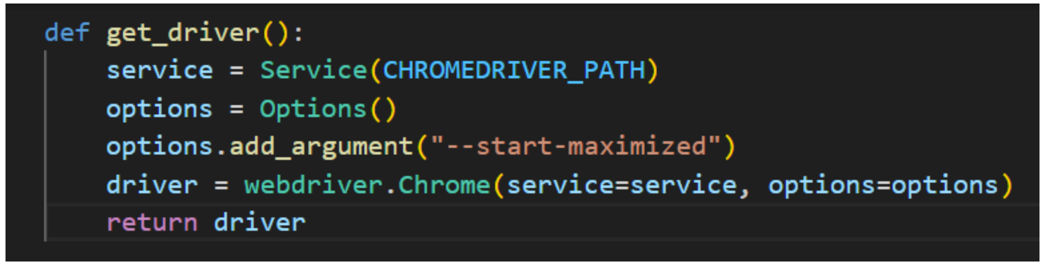
1G Complete Python Code for Scraping

- Precise the dates for which we want to collect data from each hotel.



```python
DATES = [
    ("2025-08-06", "2025-08-07"),
    ("2025-08-07", "2025-08-08"),
    ("2025-09-02", "2025-09-03"),
    ("2025-09-03", "2025-09-04"),
    ("2025-10-02", "2025-10-03"),
    ("2025-10-15", "2025-10-16"),
    ("2025-11-02", "2025-11-03"),
    ("2025-11-15", "2025-11-16"),
    ("2025-12-02", "2025-12-03"),
    ("2025-12-15", "2025-12-16"),
]
```

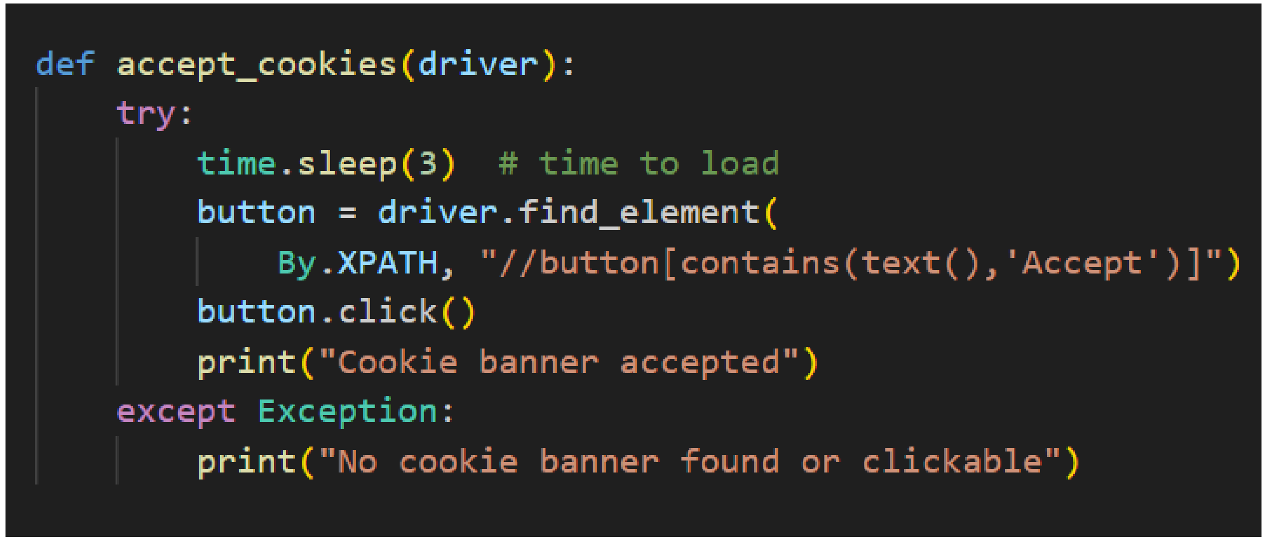
- Get driver.



```python
def get_driver():
    service = Service(CHROMEDRIVER_PATH)
    options = Options()
    options.add_argument("--start-maximized")
    driver = webdriver.Chrome(service=service, options=options)
    return driver
```

1H Avoid clickables and cookie windows

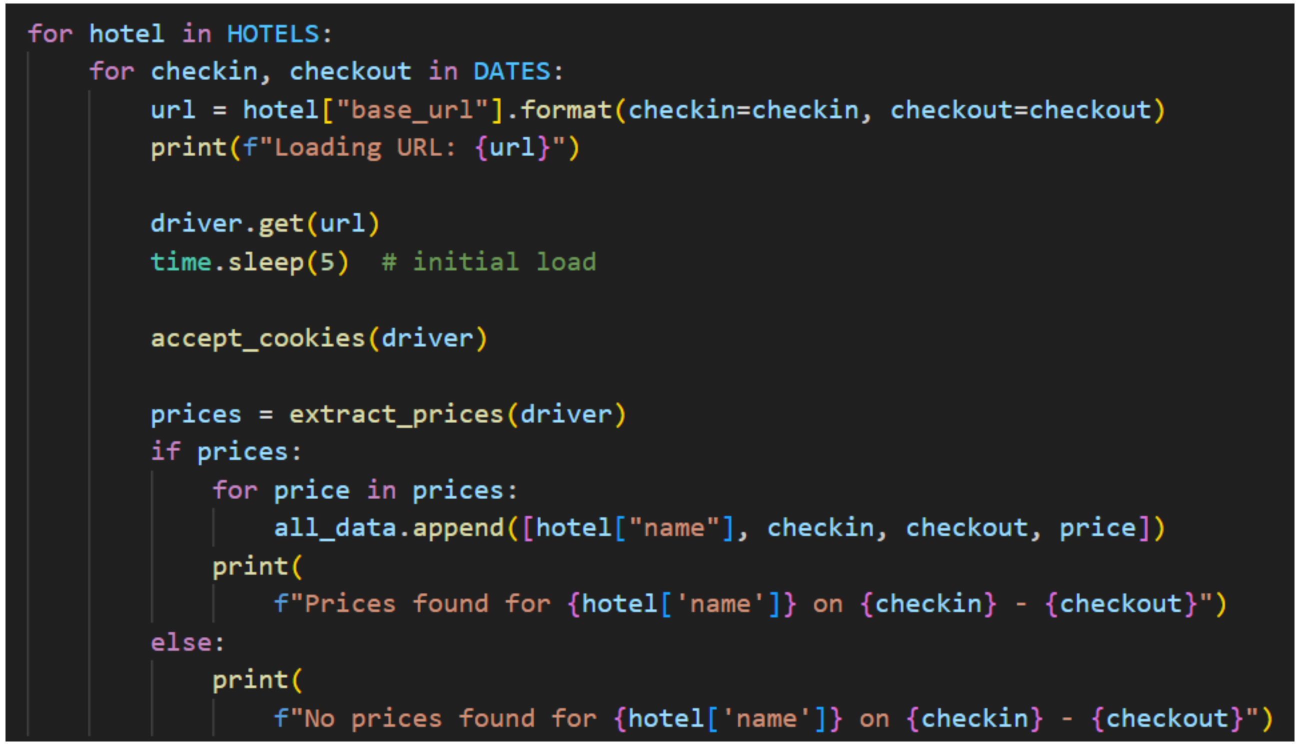
- By defining accept_cookies function, which takes one parameter, driver.
- Tries to find and click a cookie acceptance button using Selenium.



```
def accept_cookies(driver):
    try:
        time.sleep(3)  # time to load
        button = driver.find_element(
            By.XPATH, "//button[contains(text(),'Accept')]")
        button.click()
        print("Cookie banner accepted")
    except Exception:
        print("No cookie banner found or clickable")
```

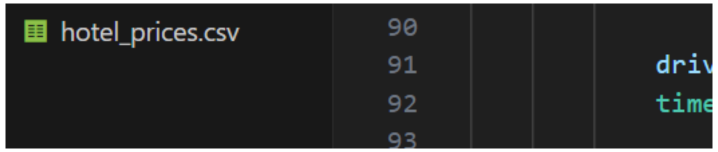
1| Double loop through hotels and dates

- With 5 seconds waiting time to give response time to booking.com

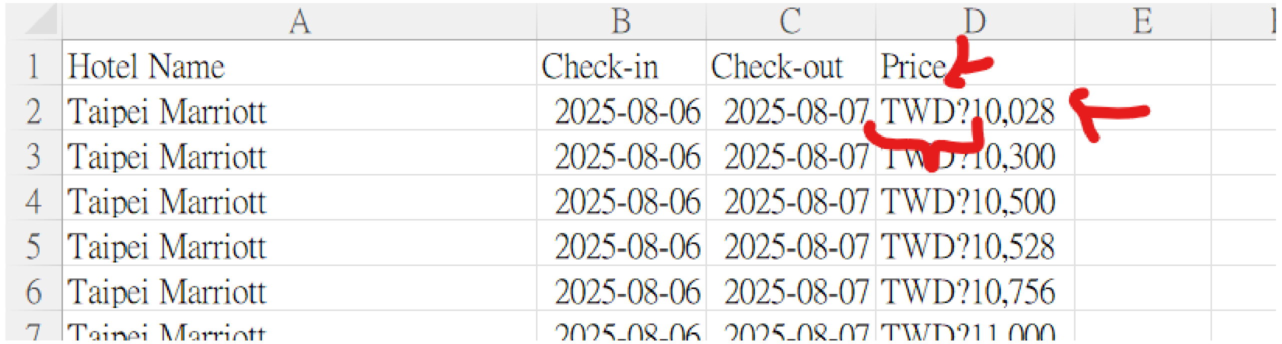


```python
for hotel in HOTELS:
    for checkin, checkout in DATES:
        url = hotel["base_url"].format(checkin=checkin, checkout=checkout)
        print(f"Loading URL: {url}")

        driver.get(url)
        time.sleep(5)  # initial load

        accept_cookies(driver)

        prices = extract_prices(driver)
        if prices:
            for price in prices:
                all_data.append([hotel["name"], checkin, checkout, price])
            print(
                f"Prices found for {hotel['name']} on {checkin} - {checkout}")
        else:
            print(
                f"No prices found for {hotel['name']} on {checkin} - {checkout}")
```

1J Save to CSV



```
hotel_prices.csv
                    driv
                    time
```

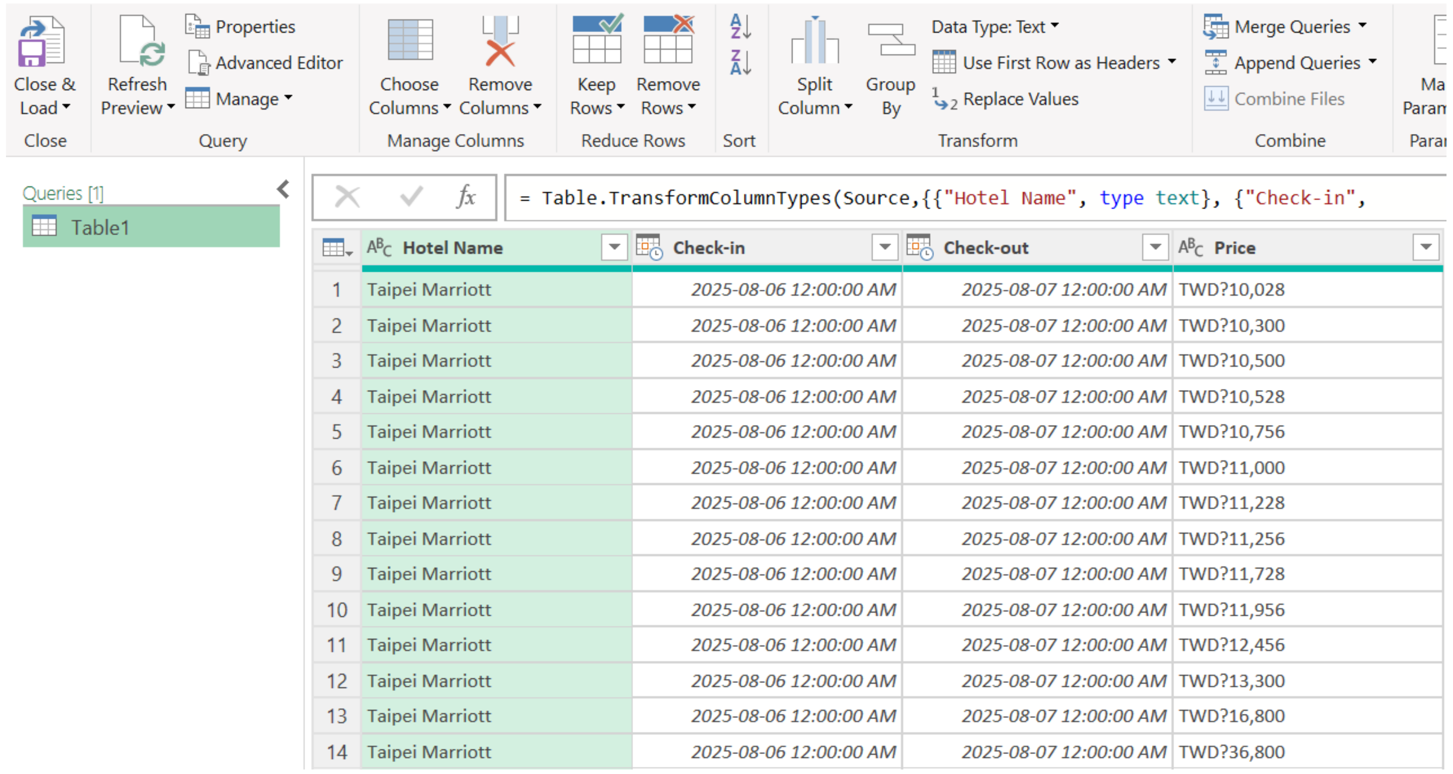


	A	B	C	D	E
1	Hotel Name	Check-in	Check-out	Price	
2	Taipei Marriott	2025-08-06	2025-08-07	TWD?10,028	
3	Taipei Marriott	2025-08-06	2025-08-07	TWD?10,300	
4	Taipei Marriott	2025-08-06	2025-08-07	TWD?10,500	
5	Taipei Marriott	2025-08-06	2025-08-07	TWD?10,528	
6	Taipei Marriott	2025-08-06	2025-08-07	TWD?10,756	
7	Taipei Marriott	2025-08-06	2025-08-07	TWD?11,000	

- In Excel, it is visible that Price is not in ideal format for analysis.
- We can use Power Query to clean the data.

2A Open Power Query Editor

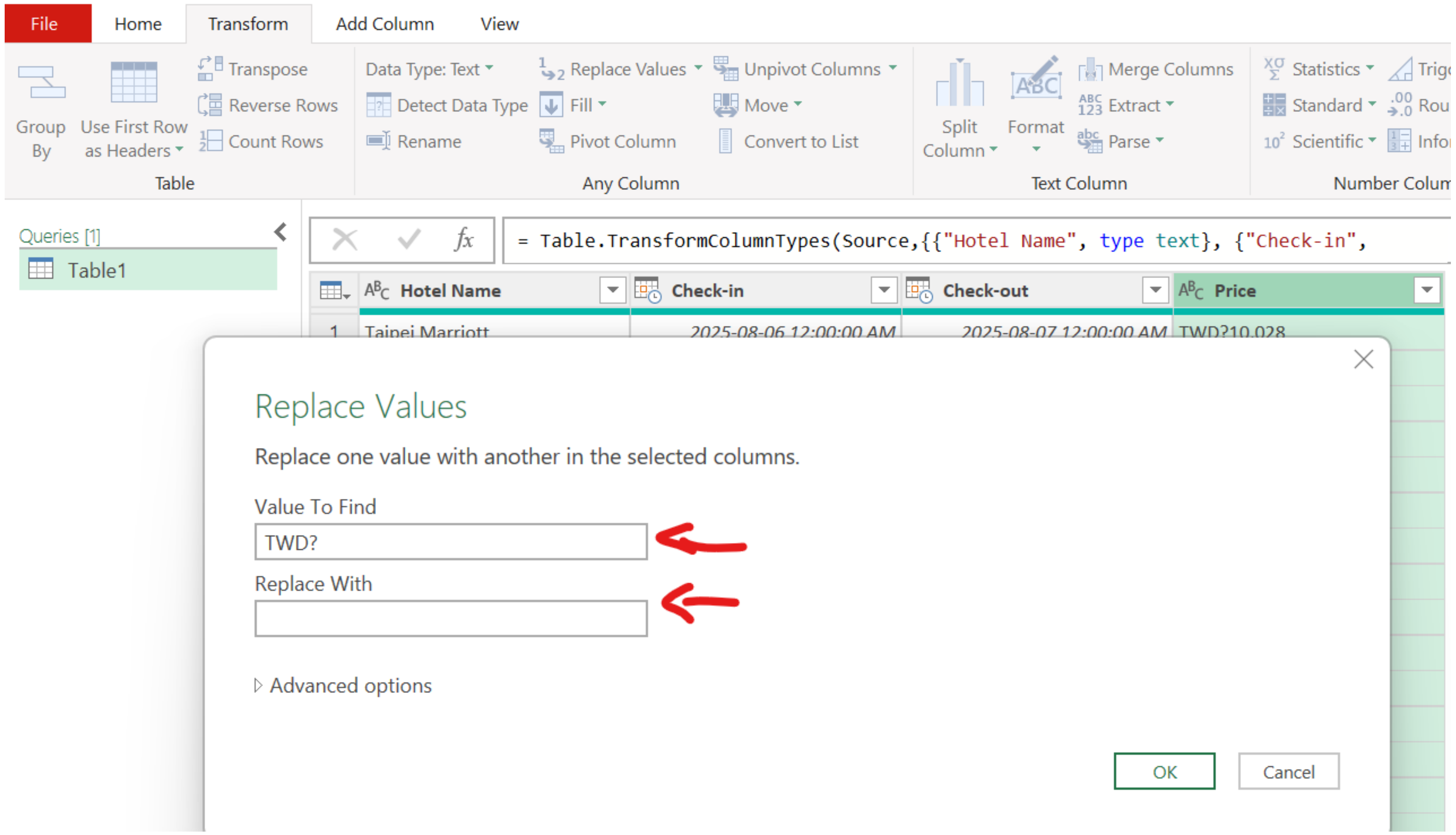
- Use get data from table / range.



```
= Table.TransformColumnTypes(Source,{{"Hotel Name", type text}, {"Check-in",
```

	Hotel Name	Check-in	Check-out	Price
1	Taipei Marriott	2025-08-06 12:00:00 AM	2025-08-07 12:00:00 AM	TWD?10,028
2	Taipei Marriott	2025-08-06 12:00:00 AM	2025-08-07 12:00:00 AM	TWD?10,300
3	Taipei Marriott	2025-08-06 12:00:00 AM	2025-08-07 12:00:00 AM	TWD?10,500
4	Taipei Marriott	2025-08-06 12:00:00 AM	2025-08-07 12:00:00 AM	TWD?10,528
5	Taipei Marriott	2025-08-06 12:00:00 AM	2025-08-07 12:00:00 AM	TWD?10,756
6	Taipei Marriott	2025-08-06 12:00:00 AM	2025-08-07 12:00:00 AM	TWD?11,000
7	Taipei Marriott	2025-08-06 12:00:00 AM	2025-08-07 12:00:00 AM	TWD?11,228
8	Taipei Marriott	2025-08-06 12:00:00 AM	2025-08-07 12:00:00 AM	TWD?11,256
9	Taipei Marriott	2025-08-06 12:00:00 AM	2025-08-07 12:00:00 AM	TWD?11,728
10	Taipei Marriott	2025-08-06 12:00:00 AM	2025-08-07 12:00:00 AM	TWD?11,956
11	Taipei Marriott	2025-08-06 12:00:00 AM	2025-08-07 12:00:00 AM	TWD?12,456
12	Taipei Marriott	2025-08-06 12:00:00 AM	2025-08-07 12:00:00 AM	TWD?13,300
13	Taipei Marriott	2025-08-06 12:00:00 AM	2025-08-07 12:00:00 AM	TWD?16,800
14	Taipei Marriott	2025-08-06 12:00:00 AM	2025-08-07 12:00:00 AM	TWD?36,800

2B Transform Price Column by Replace



```
= Table.TransformColumnTypes(Source,{{"Hotel Name", type text}, {"Check-in",
```

	Hotel Name	Check-in	Check-out	Price
1	Taipei Marriott	2025-08-06 12:00:00 AM	2025-08-07 12:00:00 AM	TWD?10,028

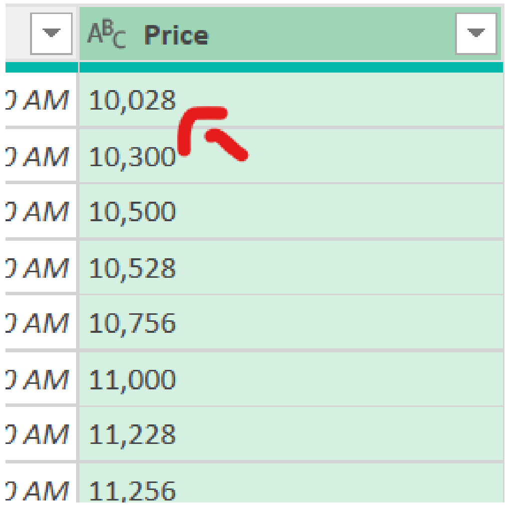
Replace Values

Replace one value with another in the selected columns.

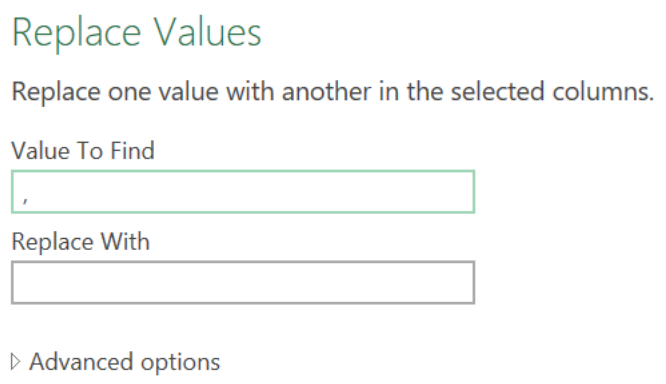
Value To Find: TWD?

Replace With:

2C Remove commas and adjust dates



	Price
AM	10,028
AM	10,300
AM	10,500
AM	10,528
AM	10,756
AM	11,000
AM	11,228
AM	11,256

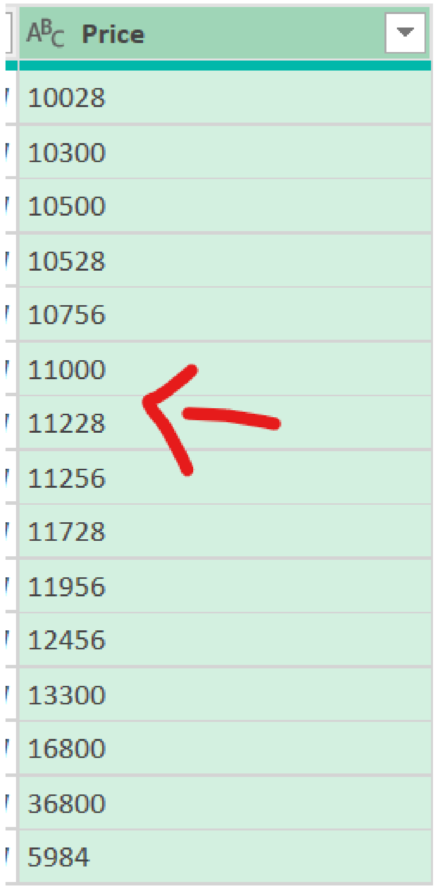


Replace Values

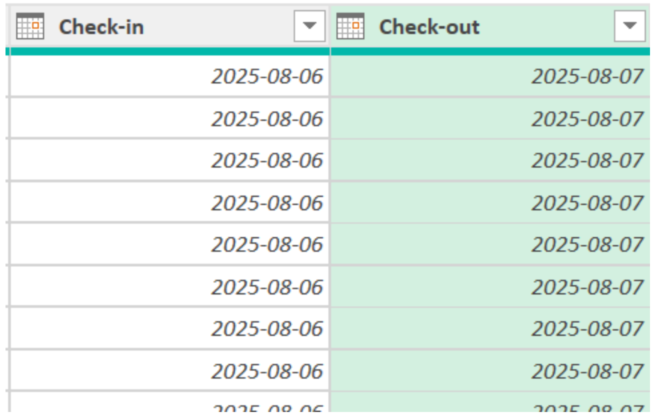
Replace one value with another in the selected columns.

Value To Find: ,

Replace With:



Price
10028
10300
10500
10528
10756
11000
11228
11256
11728
11956
12456
13300
16800
36800
5984

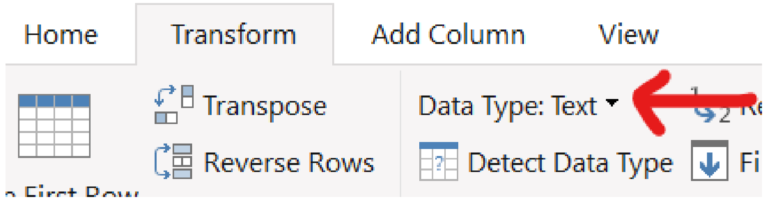


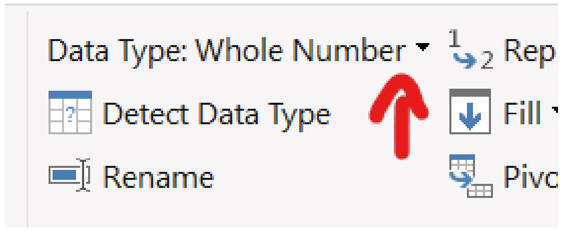
Check-in	Check-out
2025-08-06	2025-08-07
2025-08-06	2025-08-07
2025-08-06	2025-08-07
2025-08-06	2025-08-07
2025-08-06	2025-08-07
2025-08-06	2025-08-07
2025-08-06	2025-08-07
2025-08-06	2025-08-07

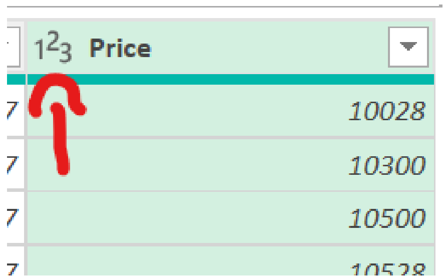
Date only, remove time, as it is not necessary for our analysis.

2D Change Data Type from Text to Number

- The scrapped price data is shown as Text.
- We use Power Query – Transform to change data type to Number, allowing future statistical analysis. Currency would also be an option.
- Once finished, we load the data.

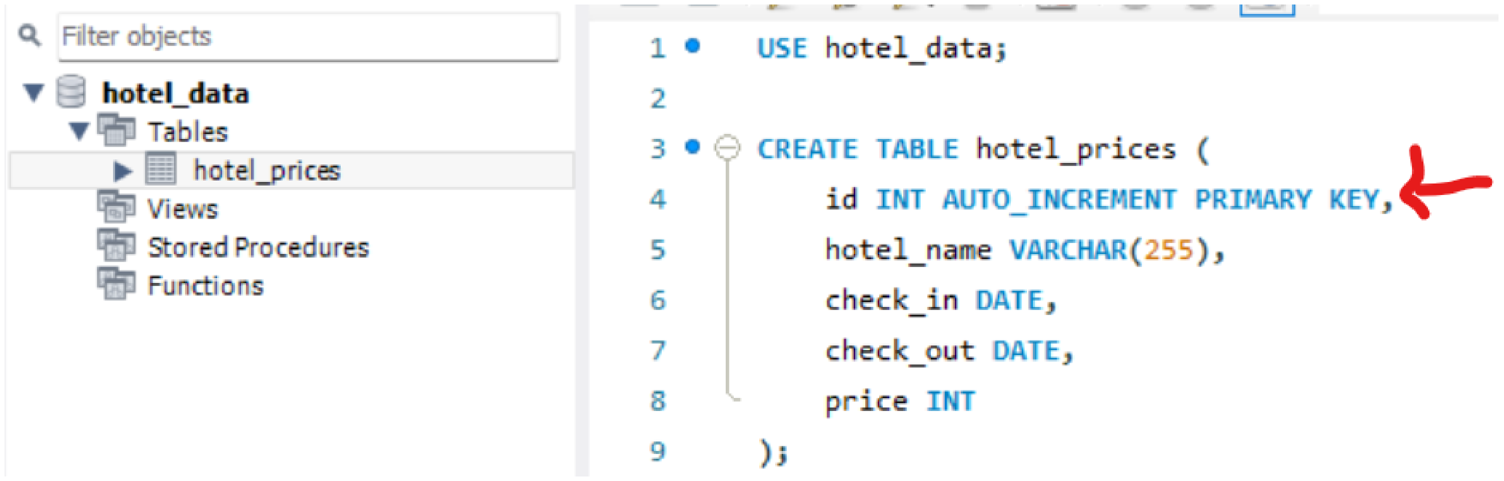






Price
10028
10300
10500
10528

3A Create new MySQL Database and Table

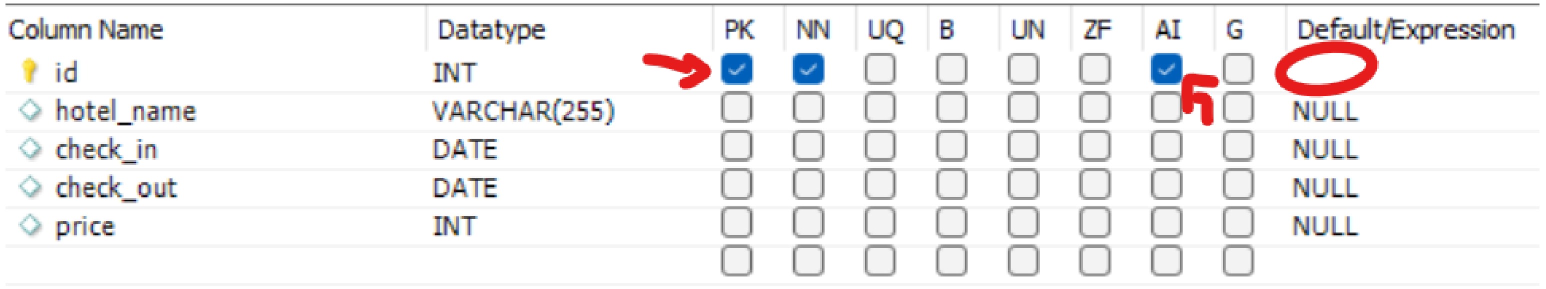


Filter objects

- hotel_data
  - Tables
    - hotel_prices
  - Views
  - Stored Procedures
  - Functions

```
USE hotel_data;

CREATE TABLE hotel_prices (
    id INT AUTO_INCREMENT PRIMARY KEY,
    hotel_name VARCHAR(255),
    check_in DATE,
    check_out DATE,
    price INT
);
```

hotel_name is VARCHAR as name length varies, price is Integer, just like we set in Power Query



Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	✓	✓					✓		
hotel_name	VARCHAR(255)									NULL
check_in	DATE									NULL
check_out	DATE									NULL
price	INT									NULL

Primary Key(PK) mandatory, automatically create id (AI), default non-NULL

3B Import Data from CSV into Table

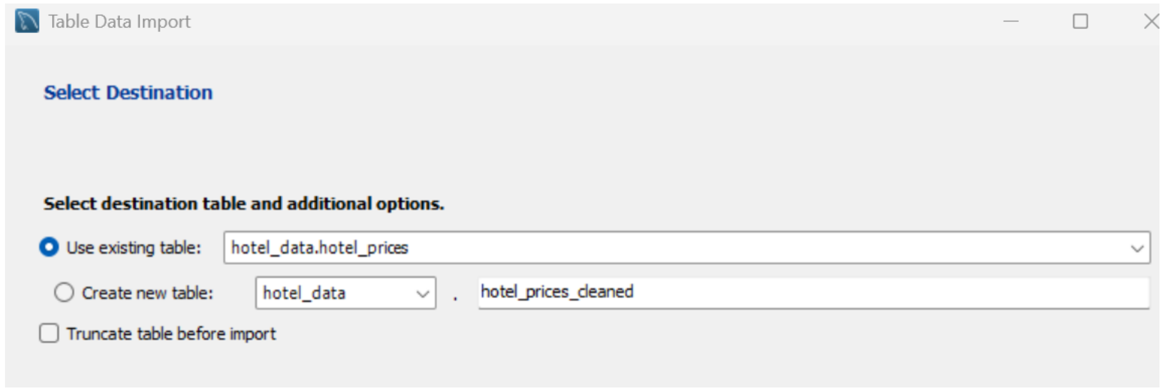


Table Data Import

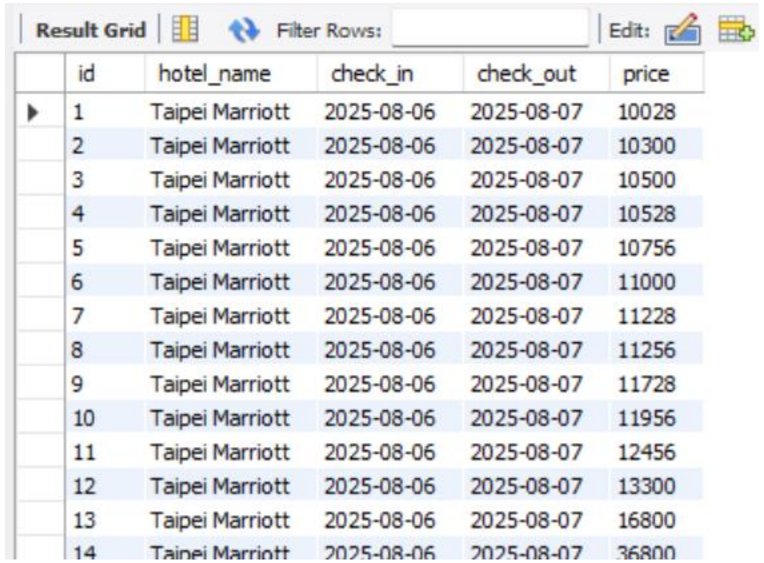
Select Destination

Select destination table and additional options.

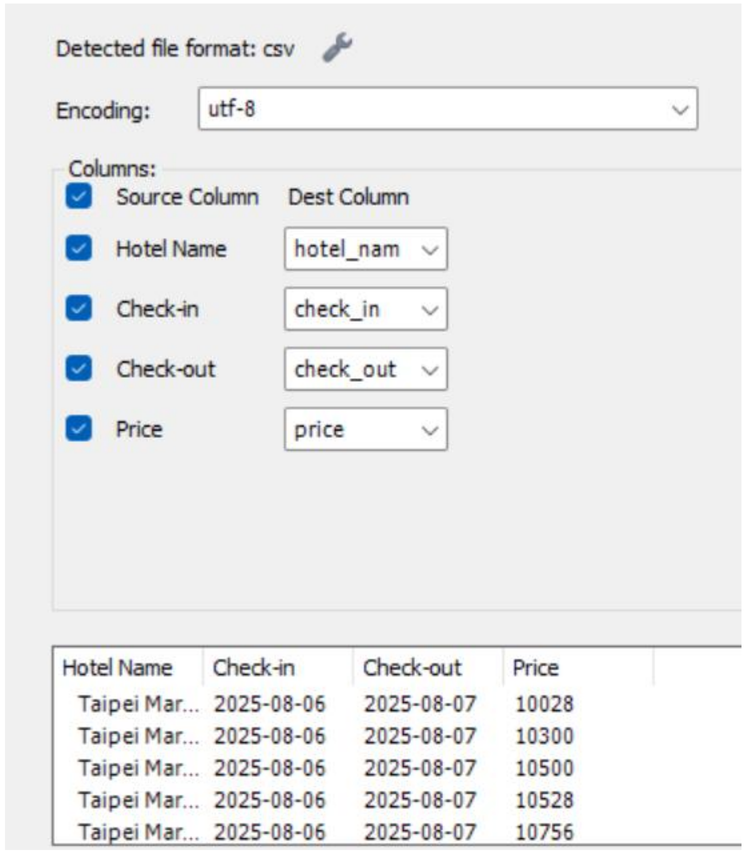
[✓ Use existing table:] hotel_data.hotel_prices

Create new table: hotel_data . hotel_prices_cleaned

☐ Truncate table before import



id	hotel_name	check_in	check_out	price
1	Taipei Marriott	2025-08-06	2025-08-07	10028
2	Taipei Marriott	2025-08-06	2025-08-07	10300
3	Taipei Marriott	2025-08-06	2025-08-07	10500
4	Taipei Marriott	2025-08-06	2025-08-07	10528
5	Taipei Marriott	2025-08-06	2025-08-07	10756
6	Taipei Marriott	2025-08-06	2025-08-07	11000
7	Taipei Marriott	2025-08-06	2025-08-07	11228
8	Taipei Marriott	2025-08-06	2025-08-07	11256
9	Taipei Marriott	2025-08-06	2025-08-07	11728
10	Taipei Marriott	2025-08-06	2025-08-07	11956
11	Taipei Marriott	2025-08-06	2025-08-07	12456
12	Taipei Marriott	2025-08-06	2025-08-07	13300
13	Taipei Marriott	2025-08-06	2025-08-07	16800
14	Taipei Marriott	2025-08-06	2025-08-07	36800


Detected file format: csv

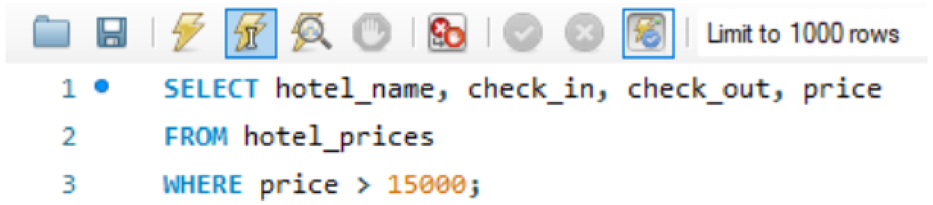
Encoding: utf-8

Columns:

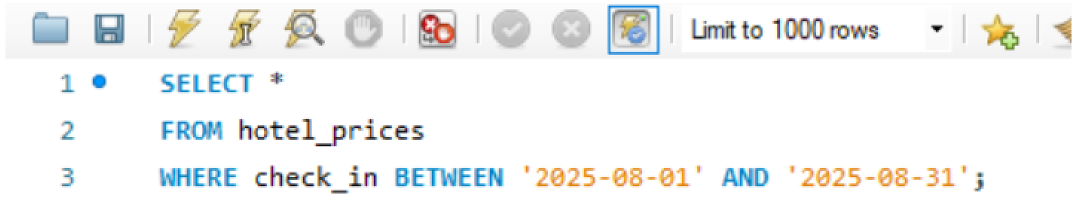
✓	Source Column	Dest Column
✓	Hotel Name	hotel_nam
✓	Check-in	check_in
✓	Check-out	check_out
✓	Price	price

Hotel Name	Check-in	Check-out	Price
Taipei Mar...	2025-08-06	2025-08-07	10028
Taipei Mar...	2025-08-06	2025-08-07	10300
Taipei Mar...	2025-08-06	2025-08-07	10500
Taipei Mar...	2025-08-06	2025-08-07	10528
Taipei Mar...	2025-08-06	2025-08-07	10756

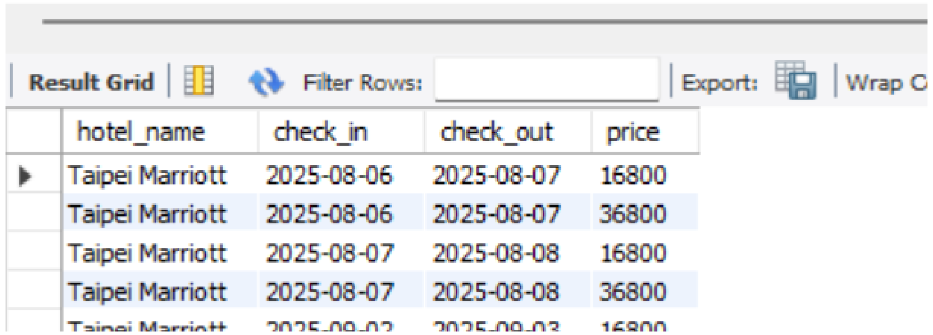
3C Query to show nights above amount or in specific month



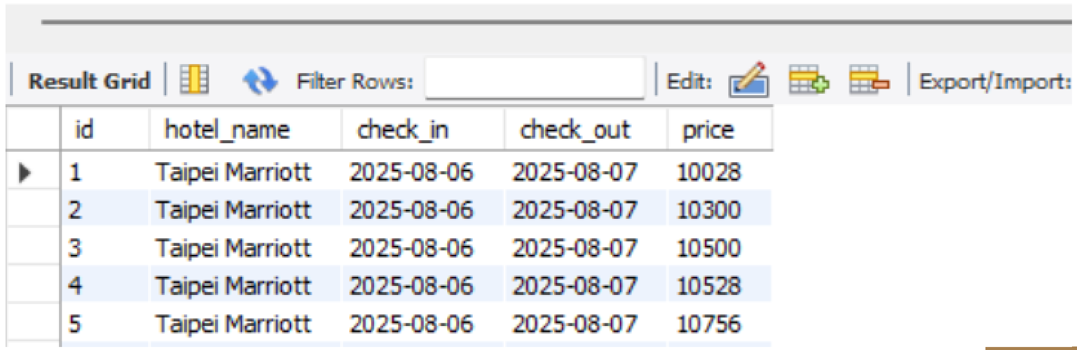
```sql
SELECT hotel_name, check_in, check_out, price
FROM hotel_prices
WHERE price > 15000;
```



```sql
SELECT *
FROM hotel_prices
WHERE check_in BETWEEN '2025-08-01' AND '2025-08-31';
```

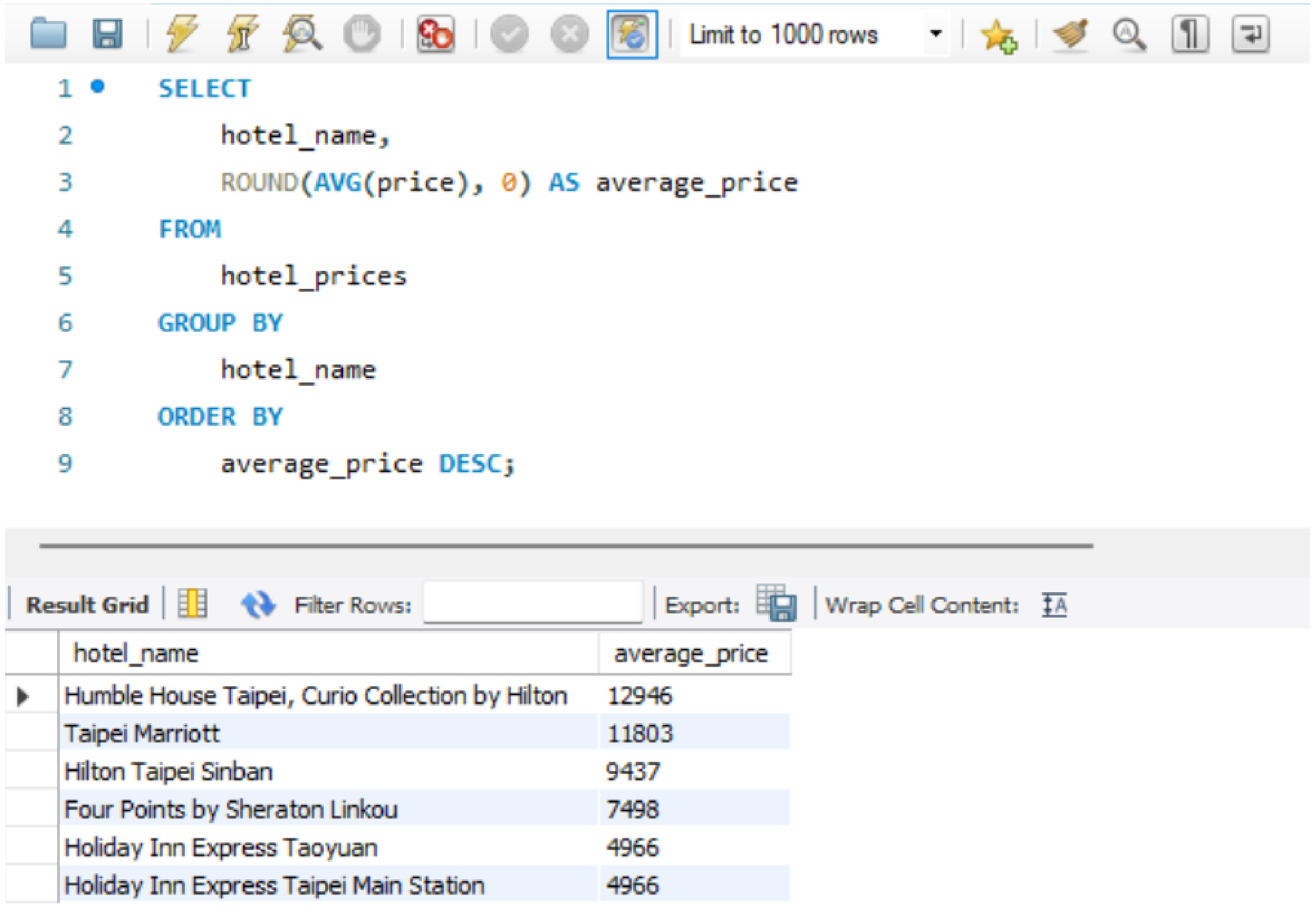


hotel_name	check_in	check_out	price
Taipei Marriott	2025-08-06	2025-08-07	16800
Taipei Marriott	2025-08-06	2025-08-07	36800
Taipei Marriott	2025-08-07	2025-08-08	16800
Taipei Marriott	2025-08-07	2025-08-08	36800
Taipei Marriott	2025-09-02	2025-09-03	16800



id	hotel_name	check_in	check_out	price
1	Taipei Marriott	2025-08-06	2025-08-07	10028
2	Taipei Marriott	2025-08-06	2025-08-07	10300
3	Taipei Marriott	2025-08-06	2025-08-07	10500
4	Taipei Marriott	2025-08-06	2025-08-07	10528
5	Taipei Marriott	2025-08-06	2025-08-07	10756

3D Average Price per Hotel



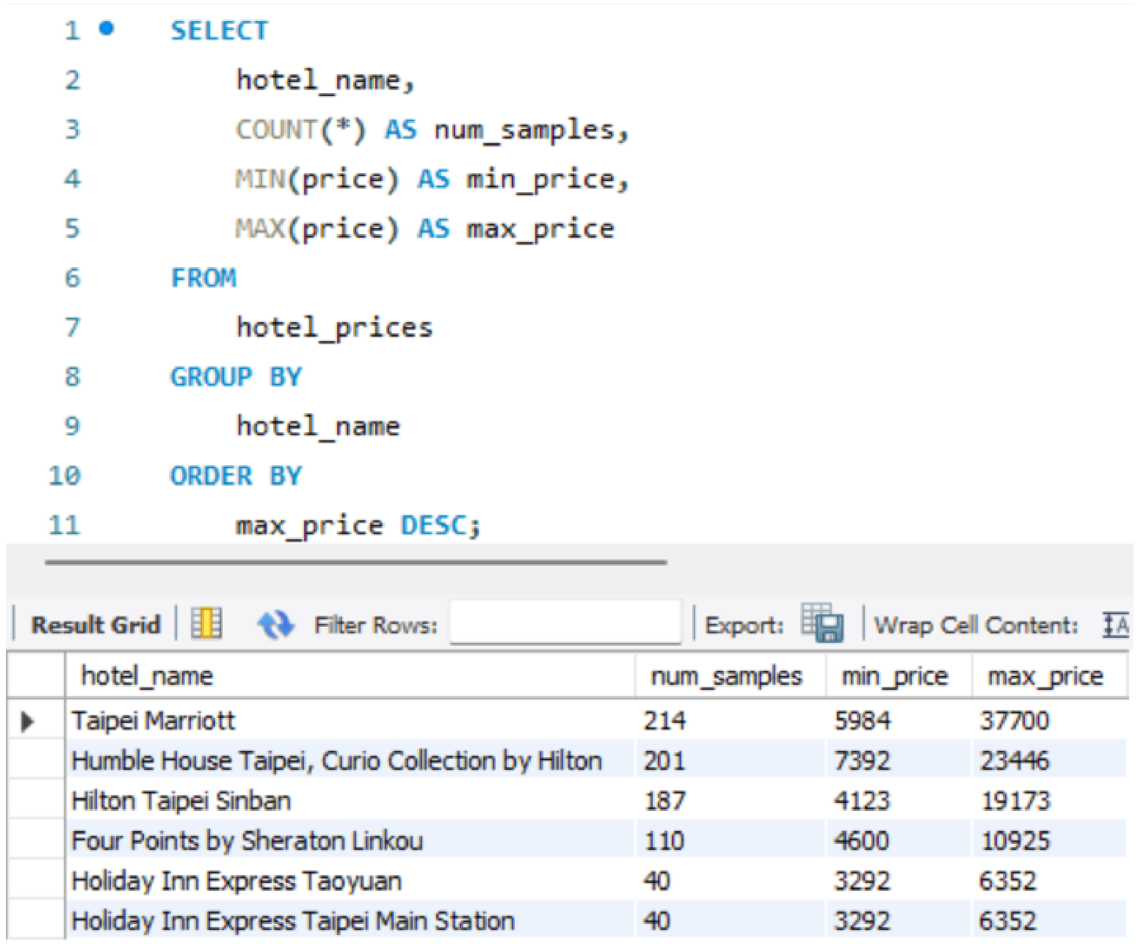
```
SELECT
    hotel_name,
    ROUND(AVG(price), 0) AS average_price
FROM
    hotel_prices
GROUP BY
    hotel_name
ORDER BY
    average_price DESC;
```

hotel_name	average_price
Humble House Taipei, Curio Collection by Hilton	12946
Taipei Marriott	11803
Hilton Taipei Sinban	9437
Four Points by Sheraton Linkou	7498
Holiday Inn Express Taoyuan	4966
Holiday Inn Express Taipei Main Station	4966

Humble House Curio by Hilton is on average the most expensive Hotel.

Holiday Inn Express are on average the cheapest Hotel(s)

3E Min Max Prices for each hotel

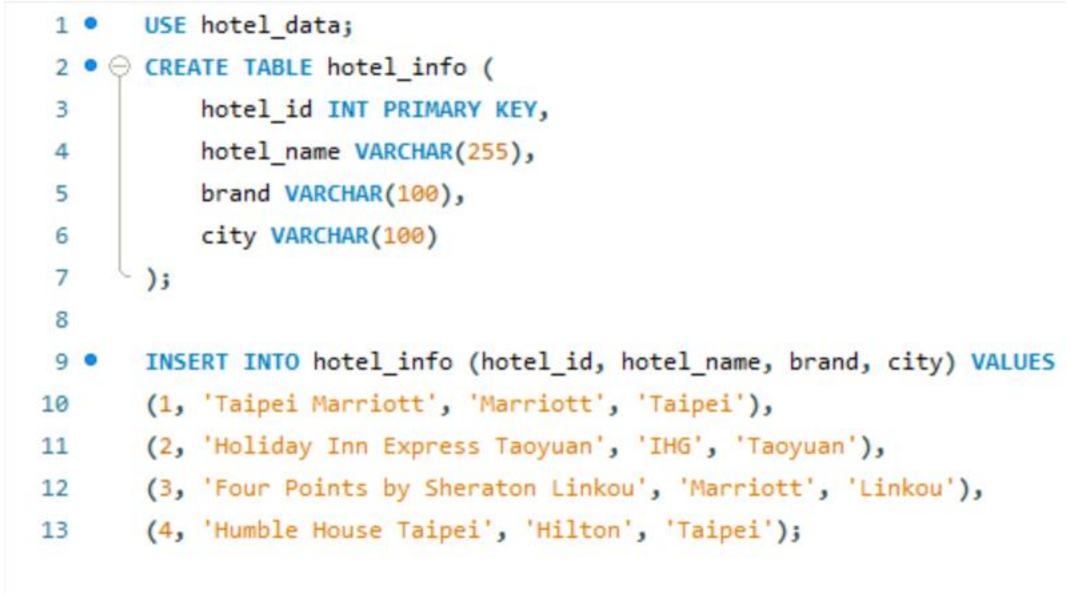


```
SELECT
    hotel_name,
    COUNT(*) AS num_samples,
    MIN(price) AS min_price,
    MAX(price) AS max_price
FROM
    hotel_prices
GROUP BY
    hotel_name
ORDER BY
    max_price DESC;
```

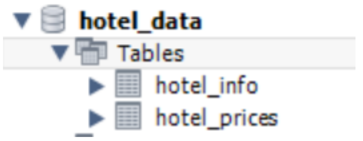
hotel_name	num_samples	min_price	max_price
Taipei Marriott	214	5984	37700
Humble House Taipei, Curio Collection by Hilton	201	7392	23446
Hilton Taipei Sinban	187	4123	19173
Four Points by Sheraton Linkou	110	4600	10925
Holiday Inn Express Taoyuan	40	3292	6352
Holiday Inn Express Taipei Main Station	40	3292	6352

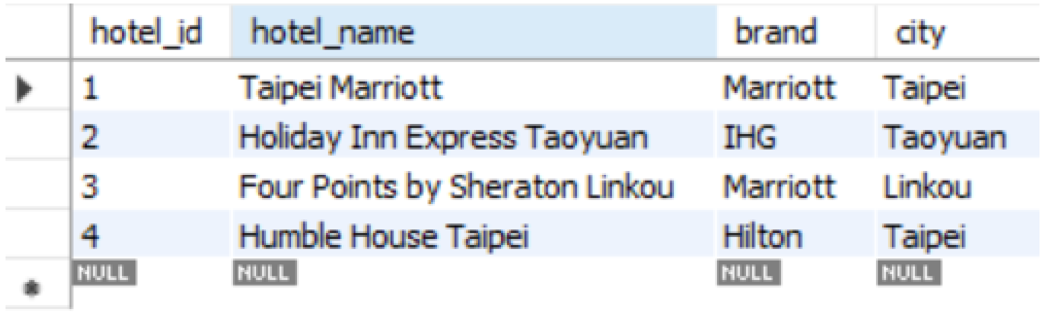
Allows us to decide if we get a good or a bad deal by booking at a given price point. e.g. if we book Taipei Marriott for 6100 NTD / night, that is a good deal.

3F Create Hotel Info Table with SQL



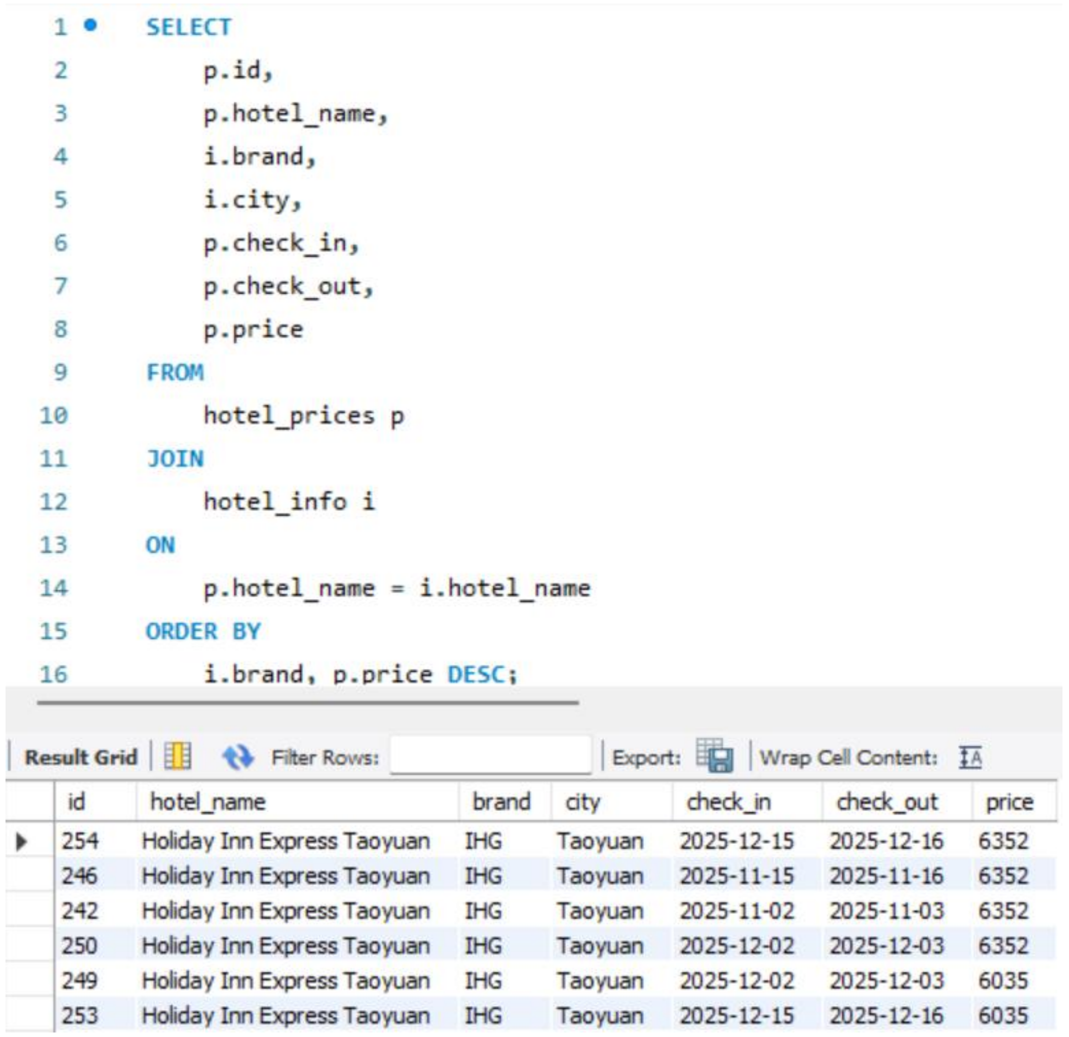
```sql
USE hotel_data;
CREATE TABLE hotel_info (
    hotel_id INT PRIMARY KEY,
    hotel_name VARCHAR(255),
    brand VARCHAR(100),
    city VARCHAR(100)
);

INSERT INTO hotel_info (hotel_id, hotel_name, brand, city) VALUES
(1, 'Taipei Marriott', 'Marriott', 'Taipei'),
(2, 'Holiday Inn Express Taoyuan', 'IHG', 'Taoyuan'),
(3, 'Four Points by Sheraton Linkou', 'Marriott', 'Linkou'),
(4, 'Humble House Taipei', 'Hilton', 'Taipei');
```





hotel_id	hotel_name	brand	city
1	Taipei Marriott	Marriott	Taipei
2	Holiday Inn Express Taoyuan	IHG	Taoyuan
3	Four Points by Sheraton Linkou	Marriott	Linkou
4	Humble House Taipei	Hilton	Taipei
NULL	NULL	NULL	NULL

3G Inner Join



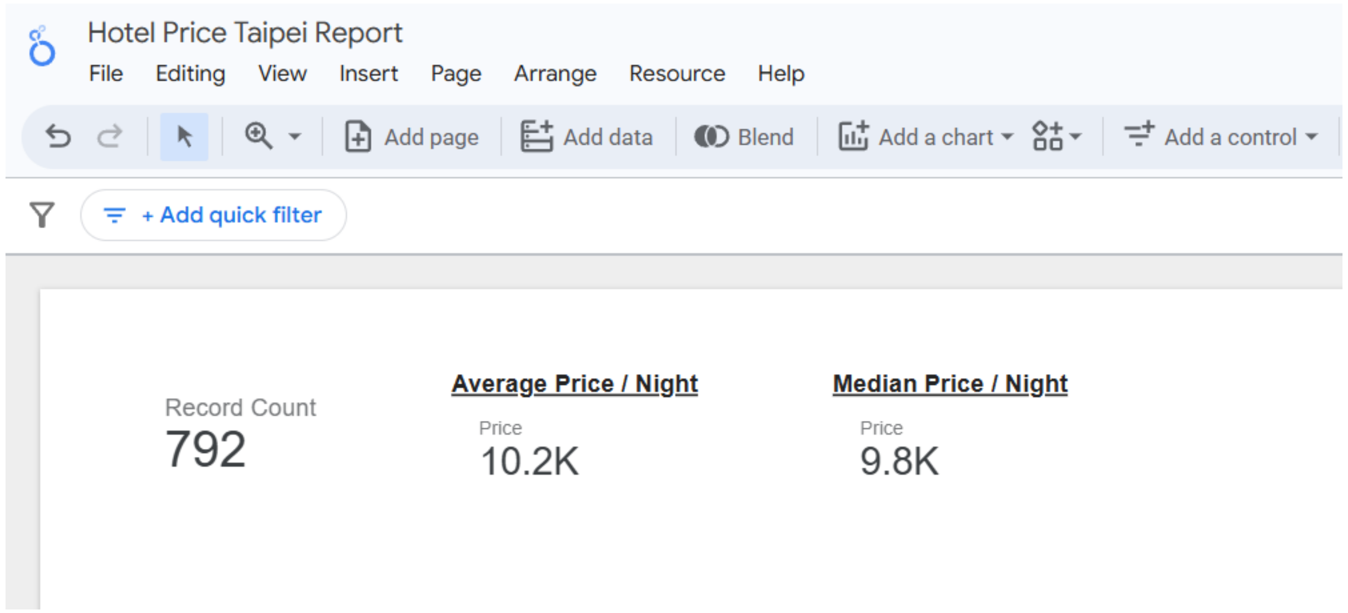
```
SELECT
    p.id,
    p.hotel_name,
    i.brand,
    i.city,
    p.check_in,
    p.check_out,
    p.price
FROM
    hotel_prices p
JOIN
    hotel_info i
ON
    p.hotel_name = i.hotel_name
ORDER BY
    i.brand, p.price DESC;
```

id	hotel_name	brand	city	check_in	check_out	price
254	Holiday Inn Express Taoyuan	IHG	Taoyuan	2025-12-15	2025-12-16	6352
246	Holiday Inn Express Taoyuan	IHG	Taoyuan	2025-11-15	2025-11-16	6352
242	Holiday Inn Express Taoyuan	IHG	Taoyuan	2025-11-02	2025-11-03	6352
250	Holiday Inn Express Taoyuan	IHG	Taoyuan	2025-12-02	2025-12-03	6352
249	Holiday Inn Express Taoyuan	IHG	Taoyuan	2025-12-02	2025-12-03	6035
253	Holiday Inn Express Taoyuan	IHG	Taoyuan	2025-12-15	2025-12-16	6035

Performed an INNER JOIN to combine price data with hotel metadata (brand and city). This enriches analysis while ensuring only matched records are returned.

4A Load CSV to Looker Studio

- Added basic stat information, such as count of total number of rows, average price per night across all hotels and dates, as well as median price per night.
- Median < Mean → The distribution is positively skewed.

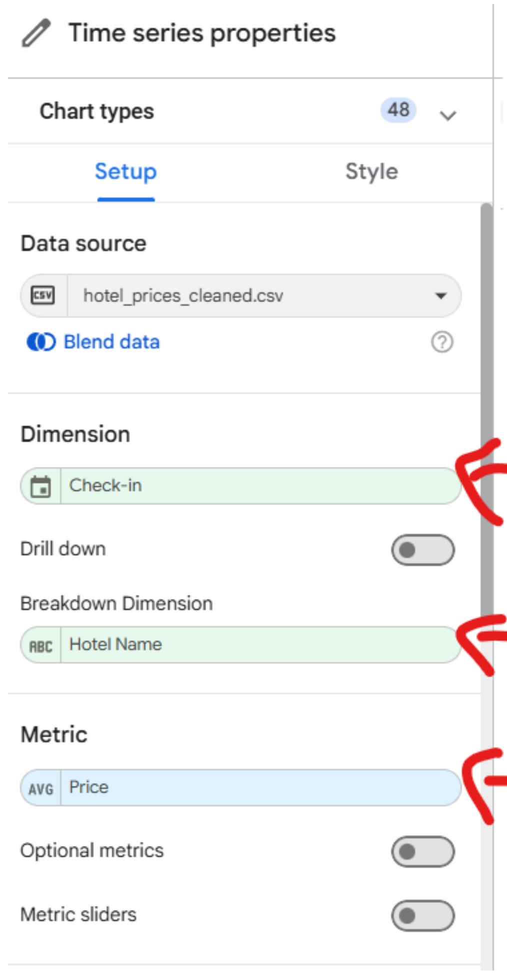


Record Count
792

Average Price / Night
Price
10.2K

Median Price / Night
Price
9.8K

4B Time Series with Breakdown Dimension



Time series properties

Chart types 48

Setup | Style

Data source

hotel_prices_cleaned.csv

Blend data

Dimension

Check-in

Drill down

Breakdown Dimension

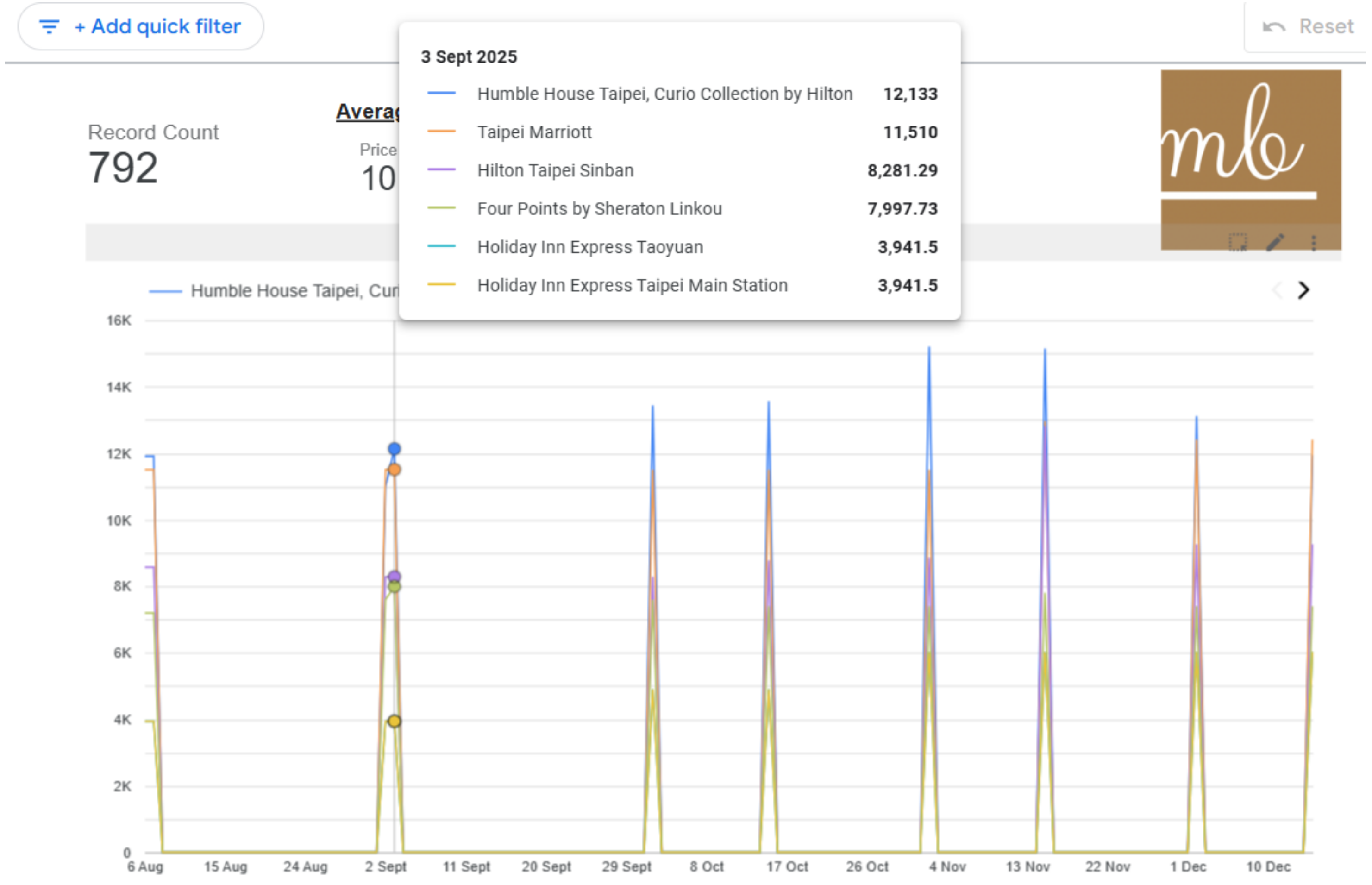
Hotel Name

Metric

AVG Price

Optional metrics

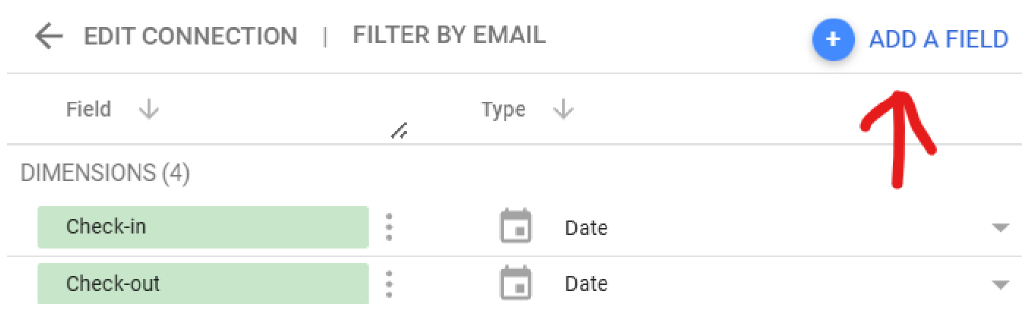
Metric sliders



3 Sept 2025

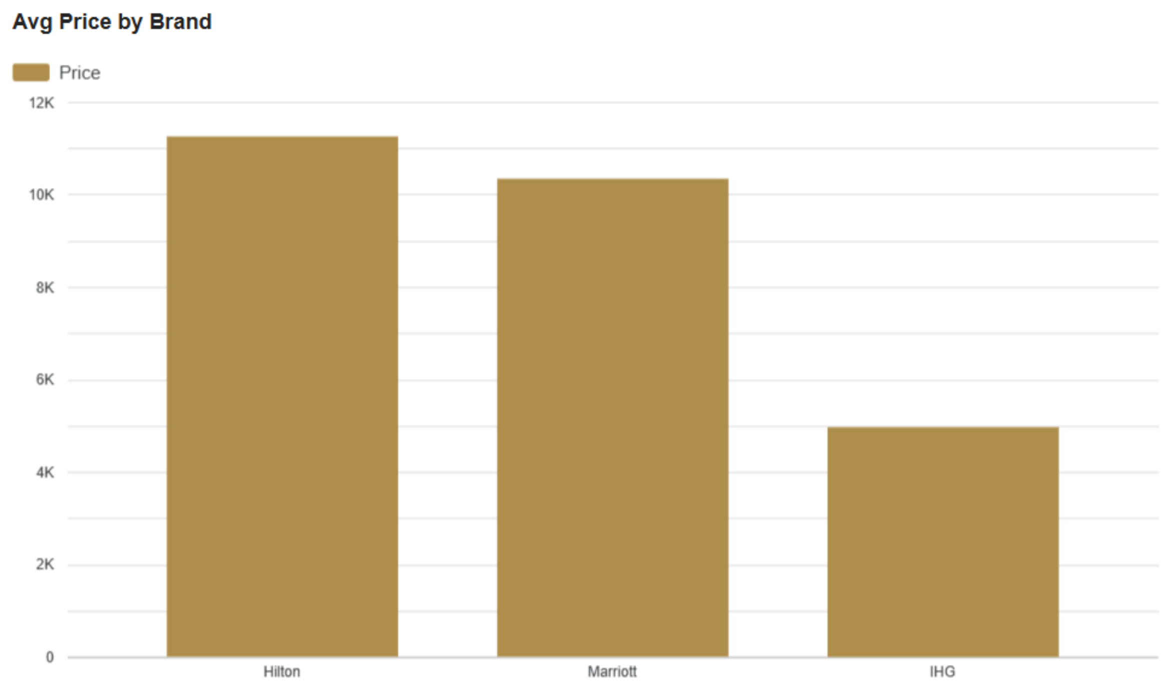
—	Humble House Taipei, Curio Collection by Hilton	12,133
—	Taipei Marriott	11,510
—	Hilton Taipei Sinban	8,281.29
—	Four Points by Sheraton Linkou	7,997.73
—	Holiday Inn Express Taoyuan	3,941.5
—	Holiday Inn Express Taipei Main Station	3,941.5

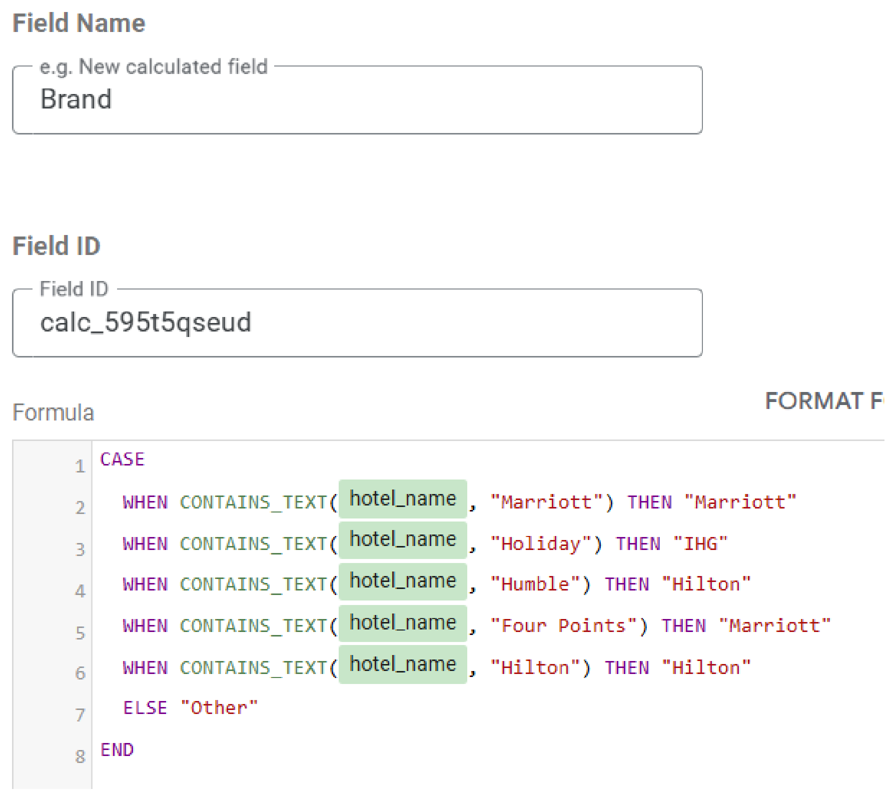
4C Edit Data Source by Adding Brand



DIMENSIONS (4)

Field	Type
Check-in	Date
Check-out	Date





Field Name

e.g. New calculated field: Brand

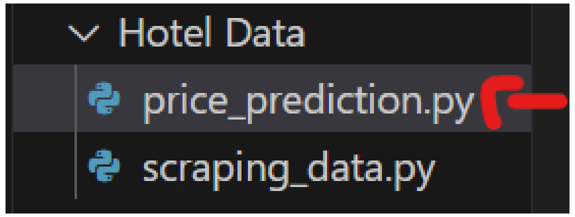
Field ID

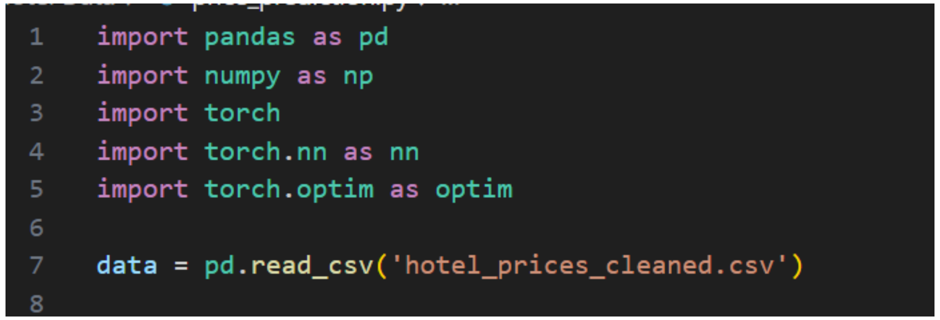
Field ID: calc_595t5qseud

Formula

```
CASE
  WHEN CONTAINS_TEXT( hotel_name , "Marriott") THEN "Marriott"
  WHEN CONTAINS_TEXT( hotel_name , "Holiday") THEN "IHG"
  WHEN CONTAINS_TEXT( hotel_name , "Humble") THEN "Hilton"
  WHEN CONTAINS_TEXT( hotel_name , "Four Points") THEN "Marriott"
  WHEN CONTAINS_TEXT( hotel_name , "Hilton") THEN "Hilton"
  ELSE "Other"
END
```

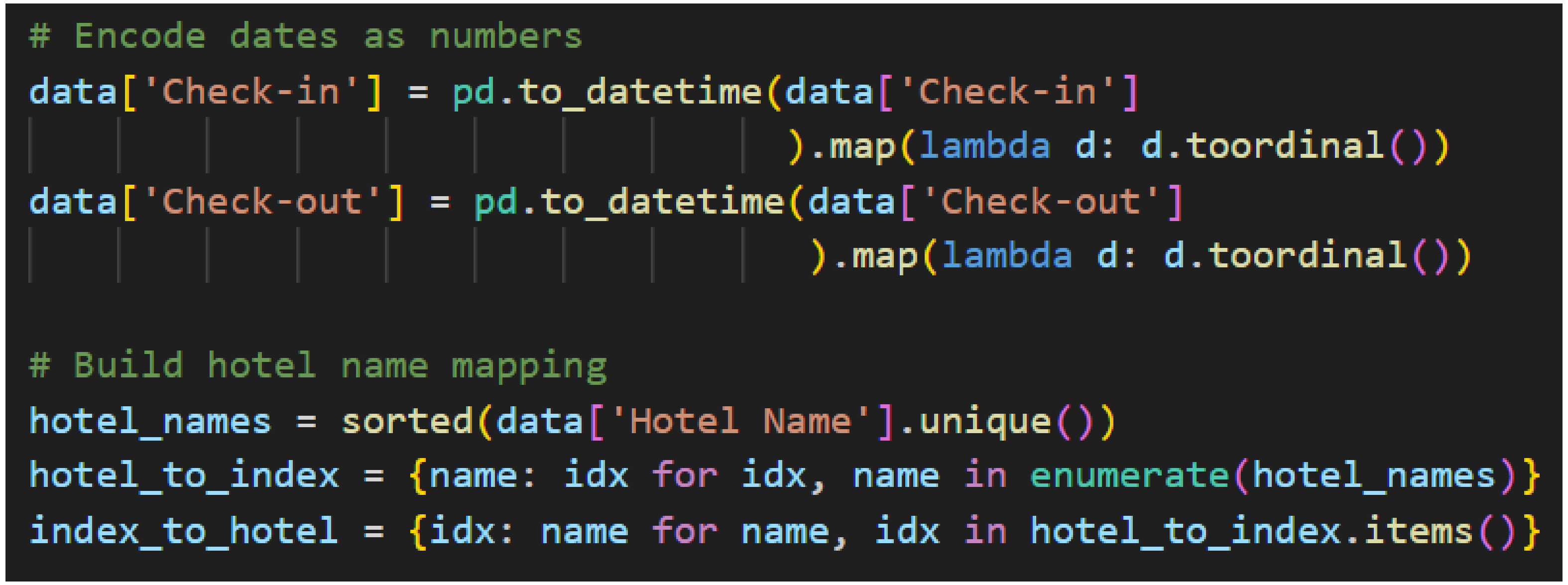
5A Create price_prediction.py for ML





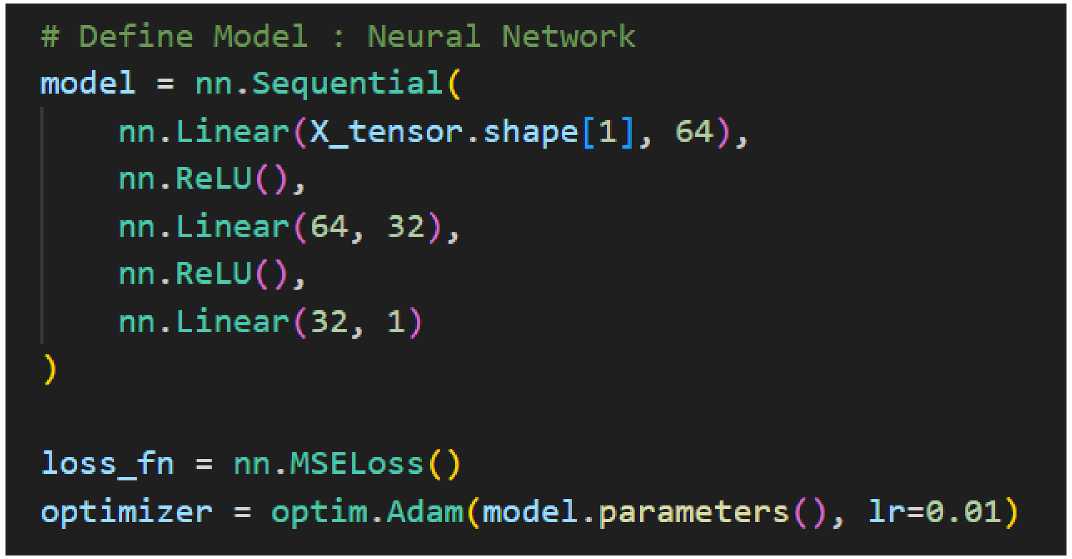
```
import pandas as pd
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim

data = pd.read_csv('hotel_prices_cleaned.csv')

```

Importing key tools: pandas, numpy, pytorch

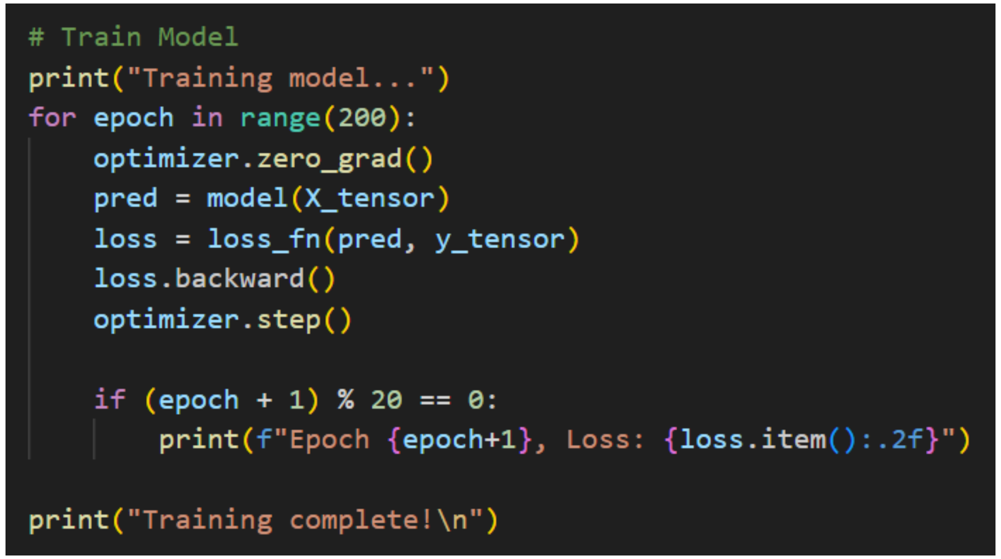


```
# Encode dates as numbers
data['Check-in'] = pd.to_datetime(data['Check-in']
                                  ).map(lambda d: d.toordinal())
data['Check-out'] = pd.to_datetime(data['Check-out']
                                   ).map(lambda d: d.toordinal())

# Build hotel name mapping
hotel_names = sorted(data['Hotel Name'].unique())
hotel_to_index = {name: idx for idx, name in enumerate(hotel_names)}
index_to_hotel = {idx: name for name, idx in hotel_to_index.items()}
```

5B Define Neural Network and Train it.



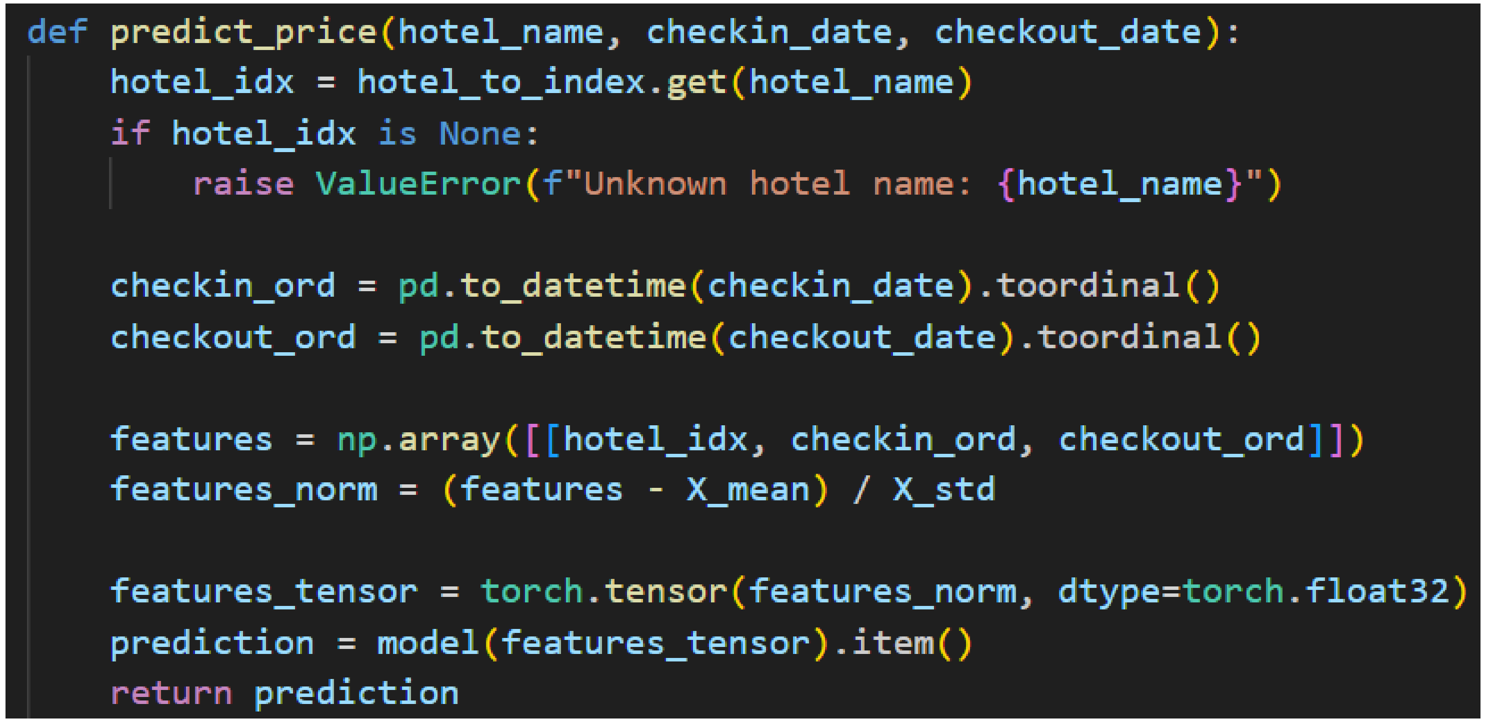
```python
# Define Model : Neural Network
model = nn.Sequential(
    nn.Linear(X_tensor.shape[1], 64),
    nn.ReLU(),
    nn.Linear(64, 32),
    nn.ReLU(),
    nn.Linear(32, 1)
)

loss_fn = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.01)
```



```python
# Train Model
print("Training model...")
for epoch in range(200):
    optimizer.zero_grad()
    pred = model(X_tensor)
    loss = loss_fn(pred, y_tensor)
    loss.backward()
    optimizer.step()

    if (epoch + 1) % 20 == 0:
        print(f"Epoch {epoch+1}, Loss: {loss.item():.2f}")

print("Training complete!\n")
```

Using Mean Squared Error loss and Adam optimizer to train for 200 epochs.

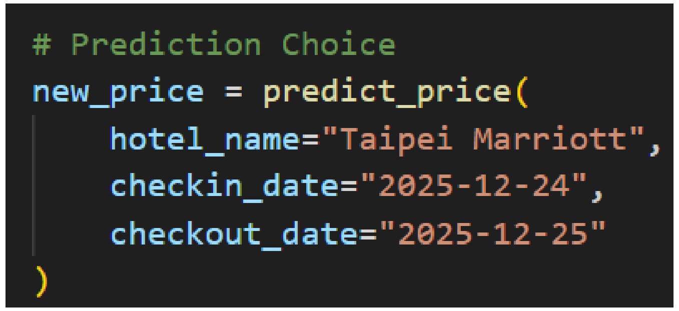
5C Create Price Prediction Function



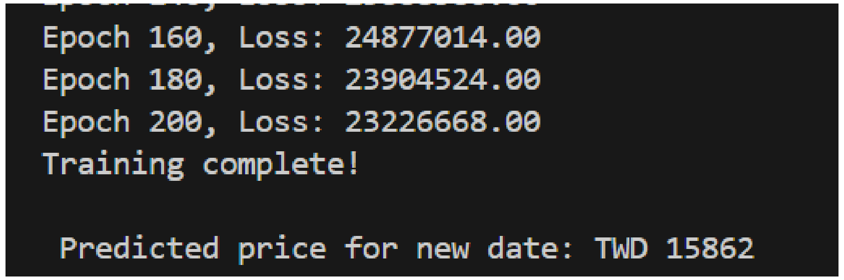
```python
def predict_price(hotel_name, checkin_date, checkout_date):
    hotel_idx = hotel_to_index.get(hotel_name)
    if hotel_idx is None:
        raise ValueError(f"Unknown hotel name: {hotel_name}")

    checkin_ord = pd.to_datetime(checkin_date).toordinal()
    checkout_ord = pd.to_datetime(checkout_date).toordinal()

    features = np.array([[hotel_idx, checkin_ord, checkout_ord]])
    features_norm = (features - X_mean) / X_std

    features_tensor = torch.tensor(features_norm, dtype=torch.float32)
    prediction = model(features_tensor).item()
    return prediction
```

Now it is possible to **predict** the price for any date, **even dates not in the original scrape.**

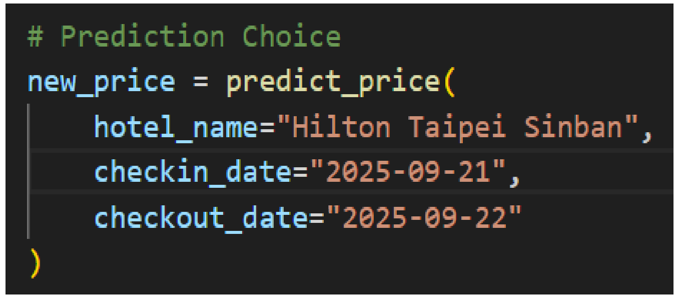
5D Prediction Examples



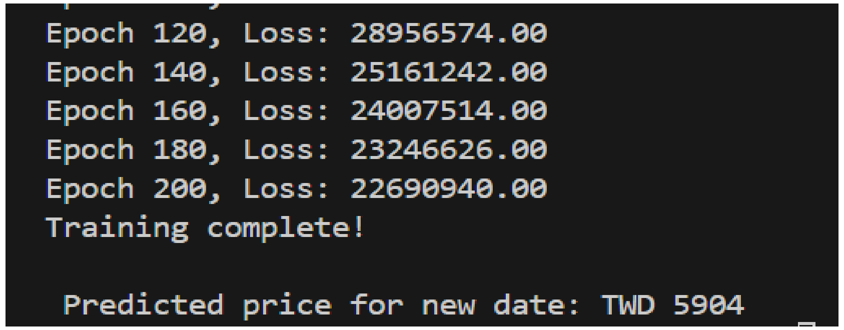
```python
# Prediction Choice
new_price = predict_price(
    hotel_name="Taipei Marriott",
    checkin_date="2025-12-24",
    checkout_date="2025-12-25"
)
```



```
Epoch 160, Loss: 24877014.00
Epoch 180, Loss: 23904524.00
Epoch 200, Loss: 23226668.00
Training complete!

 Predicted price for new date: TWD 15862
```



```python
# Prediction Choice
new_price = predict_price(
    hotel_name="Hilton Taipei Sinban",
    checkin_date="2025-09-21",
    checkout_date="2025-09-22"
)
```



```
Epoch 120, Loss: 28956574.00
Epoch 140, Loss: 25161242.00
Epoch 160, Loss: 24007514.00
Epoch 180, Loss: 23246626.00
Epoch 200, Loss: 22690940.00
Training complete!

 Predicted price for new date: TWD 5904
```

謝謝!