

逢甲大學學生報告 ePaper

報告題名：

逢甲商圈周邊路邊停車壓力分析：以端午節連假期間
為例

Analysis of On-Street Parking Pressure around Feng Chia Commercial
District during the Dragon Boat Festival Holiday

作者：陳鴻程、黃智佳
系級：大二
學號：D1345204、D1382901
開課老師：周永振教授
課程名稱：大數據分析與應用
開課系所：人工智慧技術與應用學士學位學程
開課學年：114 學年度 第 2 學期

中文摘要

本報告目的在於分析端午節連假期間逢甲商圈周邊路邊停車壓力的時間與空間變化，了解不同日期、時段及路段的占用情形，並檢查資料蒐集過程是否存在缺口或感測異常。研究使用臺中市政府路邊剩餘車位 API，每十分鐘蒐集一次車格狀態，先保存原始快照，再以 Azure VM、Docker Compose、collector 與 TimescaleDB 建立持續蒐集及儲存環境。分析時以逢甲商圈中心（24.17950, 120.64630）為中心，篩選半徑一公里內的資料，依日期、時段及路段進行彙整。2026 年 6 月 18 日至 21 日共取得 337,536 筆有效觀測，涵蓋 25 個路段與 146 個車格。結果顯示，整體占用率為 69.8%，週六最高，達 71.5%。夜市時段最高，達 77.7%。路段 6458001 的高壓力時段佔比最高，達 88.2%。每小時序列的一小時落後自我相關為 0.90，顯示短期停車壓力具有延續性。預期抓取批次覆蓋率為 100%，且未出現感測故障紀錄。雖然觀測期間只有四天，但仍可作為本次連假期間停車壓力描述、短期停車資訊規劃與後續分析的參考。

關鍵字：車位、時間序列、停車壓力、開放資料、智慧交通

Abstract

The purpose of this report is to analyze the temporal and spatial patterns of on-street parking pressure around Feng Chia Commercial District during the Dragon Boat Festival holiday, compare occupancy across dates, time periods, and road sections, and assess whether the data collection process contains gaps or sensor anomalies. Parking-space status was collected every ten minutes from the Taichung City Government open-data API. Raw snapshots were retained, and Azure VM, Docker Compose, a collector service, and TimescaleDB were used to establish the data collection and storage environment. Using the Feng Chia commercial district center (24.17950, 120.64630) as the reference point, we filtered parking spaces within a one-kilometer radius and aggregated the data by date, time period, and road section. From 18 to 21 June 2026, the study obtained 337,536 valid observations from 146 parking spaces across 25 road sections. The overall occupancy rate was 69.8%, with the highest daily rate observed on Saturday at 71.5%. The night-market period recorded the highest occupancy rate at 77.7%, and section 6458001 had the highest proportion of high-pressure periods at 88.2%. The one-hour lag autocorrelation was 0.90, indicating short-term persistence in parking pressure. All expected collection batches were completed, representing 100% batch coverage, and no sensor-failure records were observed. Although the observation period covered only four days, the findings can still serve as a reference for describing parking pressure during the holiday period, planning short-term parking information, and conducting further analysis.

Keyword: Open Data, Parking Pressure, Smart Transportation, Time Series, Urban Parking

目次

1 緒論	5
1.1 報告背景	5
1.2 研究問題	5
2 資料來源與系統建置	5
2.1 原始資料擷取	5
2.2 複合鍵問題與修正	7
2.3 API 回應、快照與儲存取捨	8
2.4 JSON 結構與欄位對應	8
2.5 資料標準化與 TimescaleDB	9
2.6 Docker Compose 執行環境	10
2.7 GitHub 與部署方式	11
2.8 Azure VM 上的持續蒐集	12
2.9 DBeaver 查詢與資料匯出	14
3 研究方法	15
3.1 報告範圍與資料量	15
3.2 ETLT 與品質規則	15
3.3 指標設計	16
4 分析結果	16
4.1 日期與整體占用率	16
4.2 時段差異	17
4.3 高占用路段	18
4.4 自我相關	20
4.5 資料品質	20
5 結果與討論	21
5.1 空間呈現	22
5.2 報告限制	22
6 結論	22
7 學習心得	23
7.1 陳鴻程	23
7.2 黃智佳	23

1 緒論

1.1 報告背景

逢甲商圈周邊同時涵蓋大學校園、住宅區、商業設施與逢甲夜市，停車需求來源較為複雜。平日時段可能受到學生、教職員、居民與通勤人口影響。晚間則隨夜市與商圈活動增加消費及遊憩車流。近期商業區停車研究也比較了平日、週末與假日期間的停車行為差異 (Sun et al., 2025)。

因此，不同日期與時段的停車需求可能存在差異，車位不足也未必平均分布在整個商圈，而可能集中於特定道路與時段。本報告使用臺中市政府「臺中市路邊剩餘車位」開放資料 (臺中市政府資料開放平臺, 2026)，連續蒐集週四至週日共四天的停車格狀態，再以逢甲商圈中心為中心，篩選半徑一公里內的車格，從區域與路段層級觀察連假期間停車壓力的時間與空間分布 (Balmer et al., 2021；Wu & Ma, 2022)。

1.2 研究問題

本報告聚焦於下列五項問題：

1. 本次觀察期間中，週四一般日、端午節當日與連假週末的整體停車占用率有何差異？
2. 週四至週日四天之中，哪一天的整體停車壓力最高？
3. 不同時段的停車壓力有何差異，壓力主要集中於哪些時段？
4. 逢甲商圈周邊哪些路段經常處於高占用狀態，其停車壓力是否具有不同的時段特性？
5. 本次蒐集資料是否存在感測異常、抓取缺口或時間覆蓋不足等資料品質問題？

2 資料來源與系統建置

本專案將資料蒐集流程分為兩層。第一層負責 API 呼叫、欄位解析、資料清理與資料庫寫入。第二層將同一套程式部署至 Azure VM，透過排程持續蒐集資料。以下先說明資料蒐集與系統建置，作為資料可追溯性與品質檢查的背景。實際的分析範圍、指標與結果將於後續研究方法章節說明。

2.1 原始資料擷取

資料來自臺中市政府開放資料平臺的路邊剩餘車位 API，資料集約每十分鐘產生九千筆資料。本專案保留每次 API 回傳內容，並補上 `collected_at` 作為蒐集時間。原始快照先保存，再寫入時間序列資料庫，以便日後核對資料或重新套用清理規則。

如圖 1 所示，資料集頁面提供 API 的基本識別資訊。圖 2 則列出欄位與更新頻率，是後續設計蒐集週期及資料表的依據。

(逢甲商圈周邊路邊停車壓力分析：以端午節連假期間為例)

預覽

資料名稱	臺中市路邊剩餘車位						
檔案名稱	臺中市路邊剩餘車位						
資料資源欄位	Section_ID、PS_ID、PS_type、Lat、Lng、status、CountyCode、AgencyCodes						
Section_ID	PS_ID	PS_type	Lat	Lng	status	CountyCode	AgencyCodes
0434704	00100	0	24.142640	120.679684	1	66000	387000000A
0434704	00300	0	24.142710	120.679756	1	66000	387000000A
0434704	00400	0	24.142676	120.679787	1	66000	387000000A
0434704	00500	0	24.142749	120.679797	1	66000	387000000A
0434704	00600	0	24.142716	120.679827	1	66000	387000000A
0434704	00800	0	24.142860	120.679972	1	66000	387000000A
0434704	00900	0	24.143376	120.680491	1	66000	387000000A
0434704	01000	0	24.142890	120.680009	1	66000	387000000A
0434704	01100	0	24.143399	120.680516	1	66000	387000000A
0434704	01200	0	24.142932	120.680047	1	66000	387000000A

圖 1 臺中市路邊剩餘車位開放資料集

建立日期	2022-10-04 13:18:02
使用對象	一般民眾
服務分類	交通及通訊
資料集上架方式	SERVICES
資料集提供機關	臺中市政府交通局
資料集提供機關聯絡人	陳小姐
資料集提供機關聯絡人電話	04-22289111#61361
更新頻率	即時(每10分鐘)
授權方式	政府資料開放授權條款-第1版
計費方式	免費
關鍵字	交通、路邊、剩餘車位
資料集語系	中文
資料資源欄位	Section_ID、PS_ID、PS_type、Lat、Lng、status、CountyCode、AgencyCodes
資料資源備註	
資料量	9000
上架日期	2022-10-05
空間範圍	全臺中市

圖 2 資料集欄位與更新頻率說明

程式先呼叫 JSON API，再檢查 HTTP 回應與實際取得的資料。確認 API 回應後，檢查重點進一步轉向資料筆數、鍵值與資料庫寫入結果。

如圖 3 所示，程式實際呼叫路邊剩餘車位 JSON API。這張圖呈現資料取得的入口，後續仍須透過資料筆數、鍵值與寫入結果檢查資料是否正確保存。

(逢甲商圈周邊路邊停車壓力分析：以端午節連假期間為例)

```
def main() -> int:
    load_dotenv(ENV_FILE)

    logger = configure_logging()

    api_url = os.getenv("TAICHUNG_PARKING_API_URL", "").strip()

    if not api_url:
        logger.error("找不到 TAICHUNG_PARKING_API_URL。")
        return 1

    collected_at = datetime.now(timezone.utc)
```

圖 3 呼叫臺中市政府路邊剩餘車位 JSON API

2.2 複合鍵問題與修正

初始設計以 (collected_at, ps_id) 作為複合鍵，但資料庫最後只留下 371 筆，明顯少於單次擷取約九千筆的規模。檢查後發現，不同路段可能使用相同的 ps_id，例如黎明路與逢甲路都可能有編號 110 的車格。只使用蒐集時間與車格編號，會使同一批資料中的不同路段互相覆寫。

組員討論並參考 LLM 提供的排查方向後，將識別鍵改為 (collected_at, section_id, ps_id)。加入路段編號後，同一時間、不同路段的同號車格不再衝突。LLM 只協助列出可能原因，修正是否有效仍以實際資料筆數與欄位關係確認。

如圖 4 所示，LLM 將異常筆數與複合鍵設計連結起來，提出加入 section_id 的排查方向。這項建議再以實際資料進行驗證。

```
原因找到了：不是你只抓到 371
筆，而是你的資料庫主鍵設計把同一批資料裡重複的 PS_ID 互相覆蓋了。

我確認到的事

你的 CSV 實際有：

- rows = 9000
- unique_ps_id = 371
- unique_section_ps = 9000

這代表：

- 一次 API 回來有 9000 筆資料
- 但只有 371 種不同的 PS_ID
- PS_ID 不是全市唯一
- 真正唯一比較像是 Section_ID + PS_ID
```

圖 4 利用 LLM 協助排查複合鍵覆寫問題

(逢甲商圈周邊路邊停車壓力分析：以端午節連假期間為例)

2.3 API 回應、快照與儲存取捨

API 回應成功後，程式會產生 CSV 與 JSON 快照。兩種格式同時保留有利於檢查，但也會增加儲存用量。由於 CSV 寫入 SQL 後較少再次使用，後續備份可優先保留 JSON。若分析需要表格格式，則由資料庫查詢或匯出 CSV。

如圖 5 所示，API 呼叫已成功完成。圖 6 則呈現同一批次產生的 CSV 與 JSON 檔案。兩張圖分別確認 API 回應與本地快照的產生，不只停留在 HTTP 成功狀態。

```
(parking_analysis) PS C:\Users\dolph\Desktop\School\Sophomore\BigData\parking_analysis> python -c "import json; print(len(json.load(open('data/raw/taichung_parking_20260616.json', encoding='utf-8'))))"
```

圖 5 API 回應驗證結果

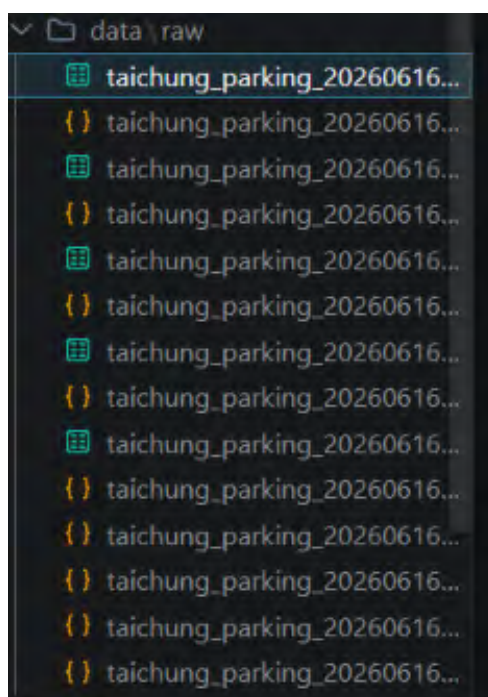


圖 6 同一批資料產生的 CSV 與 JSON 快照

2.4 JSON 結構與欄位對應

寫入資料庫前，我們先展開 JSON 結構，核對 API 實際回傳的欄位名稱，再建立 SQL schema，避免程式依照文件假設欄位，卻與實際資料不一致。

如圖 7(a) 所示，API 回應採用巢狀結構。如圖 7(b) 所示，展開後可檢查單筆記錄的實際欄位。這項檢查結果作為後續欄位映射的依據。

(逢甲商圈周邊路邊停車壓力分析：以端午節連假期間為例)

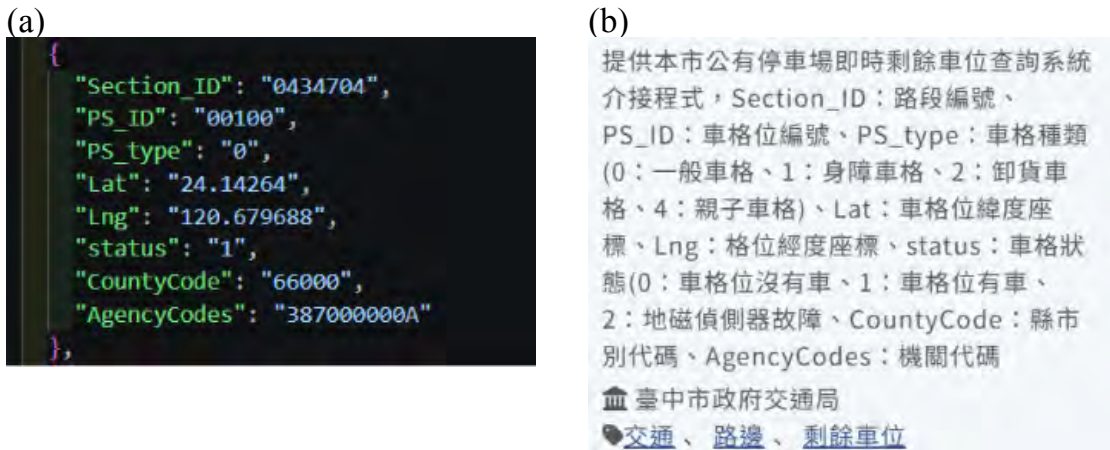


圖 7 JSON 結構與欄位檢查。(a) API 回傳的 JSON 結構。(b) 單筆 JSON 記錄的實際欄位。

根據前述檢查，如圖 8 所示，API 原始欄位逐一對應至固定的資料庫欄位，使 collector 與 SQL schema 使用一致的名稱。

```
sql > schema.sql
1 CREATE EXTENSION IF NOT EXISTS timescaledb;
2
3 CREATE TABLE IF NOT EXISTS parking_status_raw (
4   collected_at_utc TIMESTAMPTZ NOT NULL,
5   section_id TEXT NOT NULL,
6   ps_id TEXT NOT NULL,
7   ps_type SMALLINT,
8   lat DOUBLE PRECISION,
9   lng DOUBLE PRECISION,
10  status SMALLINT,
11  county_code TEXT,
12  agency_codes TEXT,
13  created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
14  PRIMARY KEY (collected_at_utc, section_id, ps_id)
15 );
16
17 SELECT create_hypertable(
18   'parking_status_raw',
19   'collected_at_utc',
20   if_not_exists => TRUE,
21   migrate_data => TRUE
22 );
23
24 CREATE INDEX IF NOT EXISTS idx_parking_status_raw_ps_id_time
25 ON parking_status_raw (ps_id, collected_at_utc DESC);
26
```

圖 8 API 回傳欄位與資料庫欄位的對應

2.5 資料標準化與 TimescaleDB

collector 會先將 API 欄位對應至統一資料欄位，並完成車格識別資訊、座標、狀態與車格類型的型別轉換。資料處理時會整理字串空白與空值。無法解析的數值設為 None，缺少 section_id 或 ps_id 的資料則排除。

寫入 TimescaleDB 的欄位統一為 collected_at_utc、section_id、ps_id、ps_type、lat、lng、status、county_code 與 agency_codes。本專案沒有在擷取階段先篩選分析中心一公里內的資料，而是先保存完整 raw data，再於分析階段依空間條件篩選。這樣既保留原始資料，也能在報告範圍改變時重新分析。

(逢甲商圈周邊路邊停車壓力分析：以端午節連假期間為例)

如圖 9 所示，圖中呈現 API 資料經欄位對應、型別轉換與缺值處理後，送入 TimescaleDB 的流程。



```
64
65 def normalize_records_for_db(
66     records: list[dict[str, Any]],
67     resolved_fields: dict[str, str | None],
68     collected_at: datetime,
69 ) -> list[tuple[object, ...]]:
70     validate_required_fields(resolved_fields)
71
72     normalized_rows: list[tuple[object, ...]] = []
73
74     section_id_field = resolved_fields["section_id"]
75     ps_id_field = resolved_fields["ps_id"]
76     ps_type_field = resolved_fields["ps_type"]
77     latitude_field = resolved_fields["latitude"]
78     longitude_field = resolved_fields["longitude"]
79     status_field = resolved_fields["status"]
80
81     for record in records:
82         section_id = _to_optional_text(record.get(section_id_field))
83         ps_id = _to_optional_text(record.get(ps_id_field))
84
85         if section_id is None or ps_id is None:
86             continue
87
88         normalized_rows.append(
89             (
90                 collected_at,
91                 section_id,
92                 ps_id,
93                 _to_optional_int(record.get(ps_type_field)),
94                 _to_optional_float(record.get(latitude_field)),
95                 _to_optional_float(record.get(longitude_field)),
96                 _to_optional_int(record.get(status_field)),
97                 _to_optional_text(record.get("CountyCode")),
98                 _to_optional_text(record.get("AgencyCodes")),
99             )
100     )
```

圖 9 標準化資料並寫入 TimescaleDB 的處理流程

2.6 Docker Compose 執行環境

資料庫與擷取程式的生命週期不同，因此以 Docker Compose 分成兩個容器：TimescaleDB container 負責儲存時間序列資料，collector container 負責呼叫 API、整理欄位並寫入資料庫。兩者透過 parking_network 通訊，TimescaleDB 資料則寫入 timescaledb_data volume，以避免容器重建後資料遺失。

如圖 10 所示，collector 程式涵蓋從 API 擷取到資料庫寫入的主要流程。如圖 11 所示，Docker Compose 定義了資料庫與 collector 服務。前者呈現程式流程，後者呈現服務的執行環境。

(逢甲商圈周邊路邊停車壓力分析：以端午節連假期間為例)

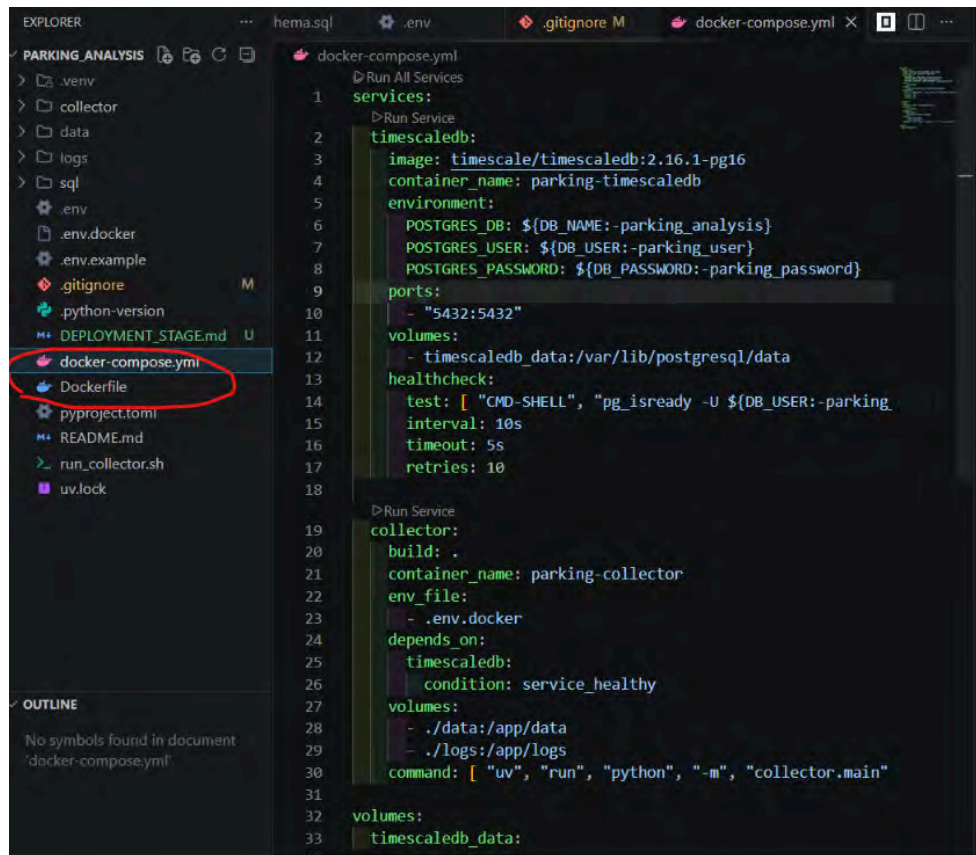


圖 10 collector 程式的主要執行內容

☐	☐	parking_analysis	-	-	-				
☐	☐	parking-timescaledb	d15175248e6f	timescale/	5432:5432				
☐	☐	parking-collector	ce3b1249334c	parking_an					

圖 11 Docker Compose 中的資料庫與 collector 服務

2.7 GitHub 與部署方式

專案程式先上傳至 GitHub，再由 Azure VM 使用 `git clone` 取得 (DolphinHuang, 2026)。部署流程依序為建立 VM、以 SSH 登入、clone repository、建立 `.env`、啟動 Docker Compose，最後設定 crontab，每十分鐘執行一次 collector。

如圖 12 所示，collector、資料庫設定與 Docker Compose 檔案均集中在同一個 repository，VM 可直接取得一致版本的部署檔案。

(逢甲商圈周邊路邊停車壓力分析：以端午節連假期間為例)

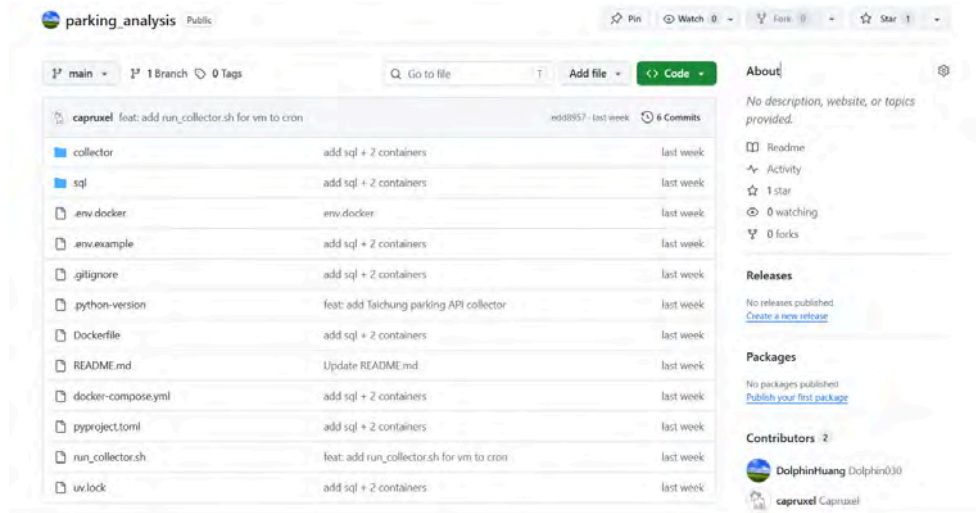


圖 12 供 Azure VM 部署使用的 GitHub repository

repository 設為 public，是為了省去 VM 上的 SSH key 或 GitHub token 設定。這項取捨只適用於程式碼與 `env.example` 等設定範例。資料庫密碼、API 金鑰及 `.env` 不得提交，敏感資料只保留在 VM 內。

2.8 Azure VM 上的持續蒐集

系統部署於 Azure 虛擬機，使用 Docker Compose 管理 TimescaleDB 與 collector。collector 由 crontab 每十分鐘啟動，讀取 API、清理欄位並寫入資料庫。flock 鎖則避免前一次工作尚未完成時重複執行。資料庫透過 Docker volume 持久化，兩個容器在專用網路內以服務名稱通訊。

如圖 13 所示，crontab 定期啟動 collector，collector 經由 Docker network 將資料寫入 TimescaleDB，資料庫內容再由 volume 保存。

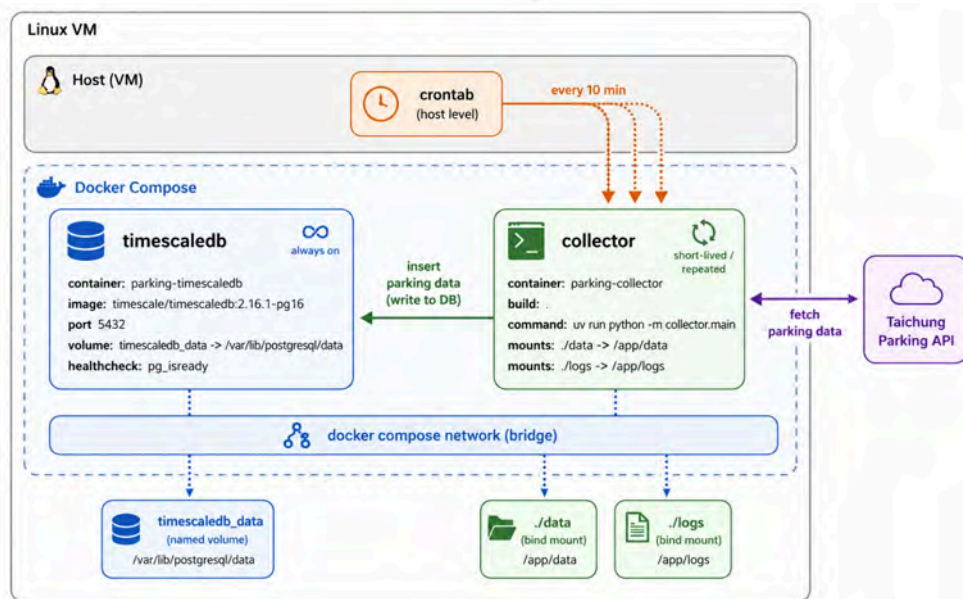


圖 13 Azure VM、Docker Compose、collector 與 TimescaleDB 架構

(逢甲商圈周邊路邊停車壓力分析：以端午節連假期間為例)

架構圖整理 VM、兩個 container、排程與 Docker network 的關係，讓部署方式可以在一張圖中呈現。

實際部署分為主機建立與程式下載兩步。如圖 14(a) 所示，先完成 Azure VM 設定。如圖 14(b) 所示，再以 SSH 登入並 clone repository。



圖 14 Azure VM 建立與程式部署。(a) 建立 Azure 虛擬機。(b) 以 SSH 登入並 clone 專案。

程式下載完成後，先啟動 TimescaleDB 並確認資料庫服務正常，再加入 collector 排程。如圖 15 所示，VM 已成功啟動資料庫服務。

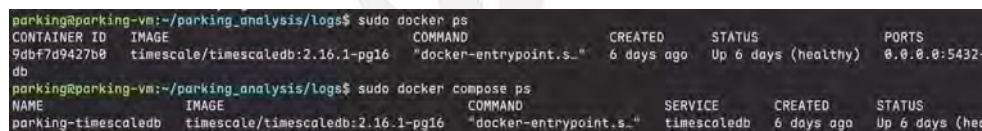


圖 15 在 VM 上先啟動 TimescaleDB 服務

如圖 16 所示，crontab 每十分鐘執行一次 collector，並透過 flock 避免前一次工作尚未結束時重複啟動。

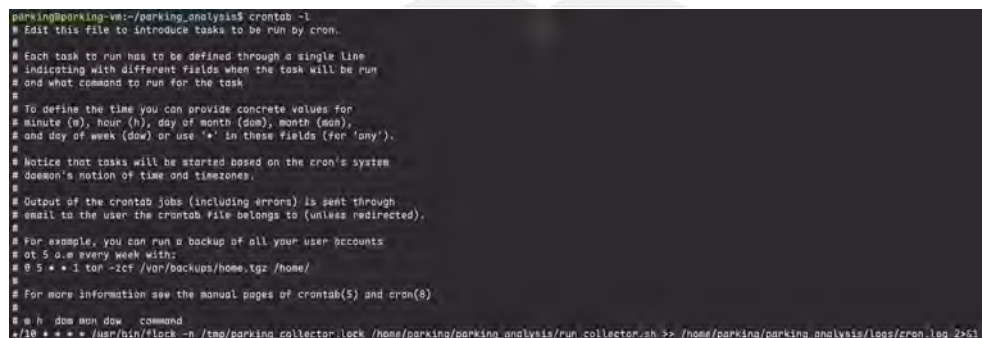


圖 16 每十分鐘執行 collector 的排程設定

排程啟用後，再以執行紀錄確認資料是否依序完成 API 呼叫、資料處理與資料庫寫入。如圖 17 所示，collector log 可用來核對這些步驟。

(逢甲商圈周邊路邊停車壓力分析：以端午節連假期間為例)

```
parking@parking-vm:~/parking_analysis/logs$ cat ./cron.log | head -30
[2026-06-17T07:47:01+08:00] collector started
Container parking_analysis-collector-run-003de8ba9c38 Creating
Container parking_analysis-collector-run-003de8ba9c38 Created
2026-06-17 07:47:04 | INFO | 開始呼叫臺中市政府停車資料 API。
2026-06-17 07:47:04 | INFO | API URL: https://newdatacenter.taichung.gov.tw/api/v1/no-auth/resource.download?rid
2026-06-17 07:47:06 | INFO | HTTP 狀態碼: 200
2026-06-17 07:47:06 | INFO | Content-Type: application/octet-stream
2026-06-17 07:47:06 | INFO | 回傳大小: 1,519,442 bytes
2026-06-17 07:47:06 | INFO | 最終 URL: https://newdatacenter.taichung.gov.tw/api/v1/no-auth/resource.download?rid
2026-06-17 07:47:06 | INFO | 成功取得資料: 9,931 筆
2026-06-17 07:47:06 | INFO | 主要欄位均已辨識。
2026-06-17 07:47:06 | INFO | 有效座標: 9,927 / 9,931
2026-06-17 07:47:06 | WARNING | 共有 4 筆座標無法解析或超出合理範圍。
2026-06-17 07:47:06 | INFO | status 為 0, 1, 2: 9,931 / 9,931

status 原始值分布:
status
0    2214
1    7717
2026-06-17 07:47:06 | INFO | 本次回應中重複的 PS_ID: 9,541 筆
2026-06-17 07:47:06 | INFO | 原始 JSON 已儲存: /app/data/raw/taichung_parking_20260617T074704Z.json
2026-06-17 07:47:10 | INFO | 成功寫入資料庫: 9,931 筆

Collector 執行完成。
資料筆數: 9,931
寫入資料庫: 9,931 筆
JSON 檔案: /app/data/raw/taichung_parking_20260617T074704Z.json
擷取時間 UTC: 2026-06-17T07:47:04.885760+08:00
[2026-06-17T07:47:10+08:00] collector finished, exit_code=0
```

圖 17 collector 實際執行紀錄

排程中的 `/usr/bin/flock -n /tmp/parking_collector.lock` 會在 collector 執行期間建立鎖。若某次工作超過十分鐘，下一次排程不會同時啟動另一個 collector，因而避免重複抓取與寫入。執行 log 則用來確認程式是否依排程完成 API 呼叫與資料庫寫入。

2.9 DBeaver 查詢與資料匯出

為降低資料庫直接暴露於公網的風險，本地端先以 SSH key 連到 VM，再透過 localhost 對映 5432 port，由 DBeaver 存取 TimescaleDB。蒐集到足夠資料後，先在 DBeaver 檢查 schema，再以 SQL 建立索引並篩選報告所需資料，最後匯出 CSV 供 notebook 分析。

如圖 18(a) 所示，本地端以 DBeaver 設定 SSH tunnel。如圖 18(b) 所示，連線後即可檢查資料庫 schema。這兩個畫面確認資料庫不需直接開放公網連線，也能由本地端管理。

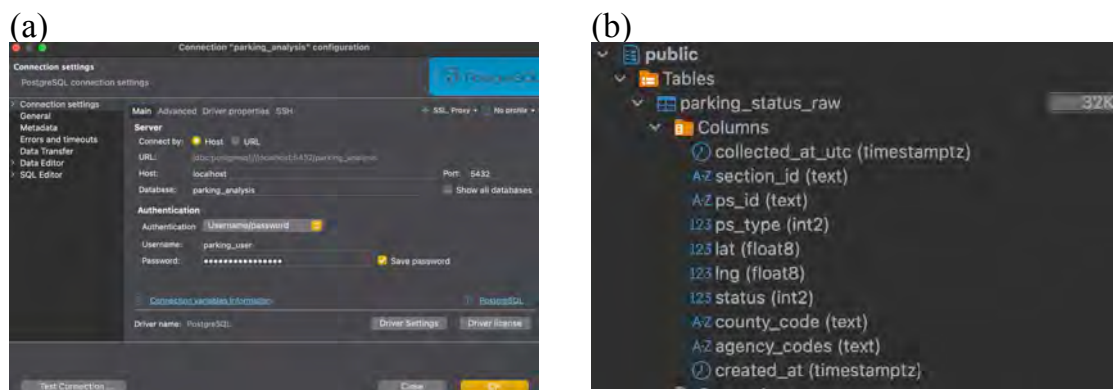


圖 18 DBeaver 遠端連線與資料表檢查。(a) DBeaver 的 SSH tunnel 連線設定。

(b) 連線後的 TimescaleDB schema。

資料確認無誤後，才產出分析資料。如圖 19(a) 所示，先以 SQL 建立索引並選取需要的資料。如圖 19(b) 所示，再將查詢結果匯出為 CSV，交由 notebook 分析。

(逢甲商圈周邊路邊停車壓力分析：以端午節連假期間為例)

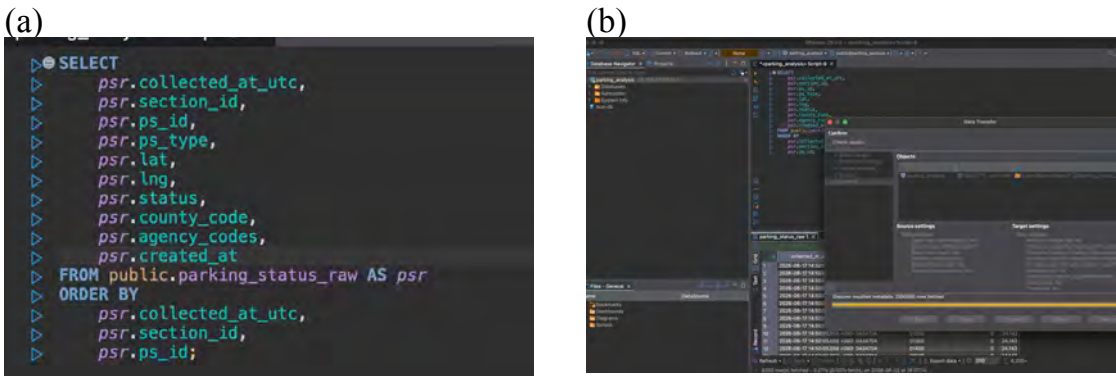


圖 19 分析資料的查詢與匯出。(a) 建立索引及擷取分析資料的 SQL。(b) 由 DBBeaver 匯出分析用 CSV。

3 研究方法

3.1 報告範圍與資料量

本報告以逢甲商圈中心 (24.17950, 120.64630) 為中心，設定半徑一公里的分析範圍。分析期間為 2026 年 6 月 18 日至 21 日，包含週四一般日、端午節當日與週末連假。

指標	數值
原始資料筆數	7,332,503
清理後資料筆數	7,287,485
快照完整時間範圍	2026-06-17 14:50 至 2026-06-22 18:00
報告範圍有效觀測	337,536
報告範圍路段數	25
報告範圍車格數	146

3.2 ETLT 與品質規則

報告採用 Extract–Transform–Load–Transform (ETLT) 流程。在 notebooks/analysis.py 中，資料時間以臺灣時間 (UTC+08:00) 解讀。流程先從 API 擷取資料，解析時間、座標與狀態，保存原始快照並載入 TimescaleDB，再建立日期、小時、星期、假日旗標、可停與占用等衍生欄位。空間篩選時，程式將緯度差與經度差換算為公里，再計算平面距離，保留距離不超過一公里的觀測，最後依日期及時段篩選分析資料。清理階段保留可解析時間、臺中範圍內的座標 (緯度 24.0–24.4、經度 120.5–121.0) 與合法 status 值。

狀態	意義	是否納入占用率分母
0	可停	是
1	已占用	是
2	感測故障	否

(逢甲商圈周邊路邊停車壓力分析：以端午節連假期間為例)

status 是 API 提供的官方狀態欄位：0 代表可停、1 代表已占用、2 代表感測故障。分析表再由這些狀態建立 is_available、is_occupied 與 is_fault 欄位，並將 status = 2 排除在占用率分母之外。座標或主要識別欄位缺失，以及時間或狀態無法解析的資料，也會在清理階段排除 (Arjona Martinez et al., 2021 ; Schneble & Kauermann, 2025)。

3.3 指標設計

考量資料期間僅四天，本報告先採用可解釋的描述性壓力指標。待後續累積更長期間的資料後，再評估 ARIMA、LSTM 等長期預測模型 (Hyndman & Athanasopoulos, 2021 ; Goodfellow et al., 2016 ; Hochreiter & Schmidhuber, 1997)。令 A 為可停車格觀測數、 O 為已占用觀測數、 V 為有效觀測數，則：

$$V = A + O$$

$$R_a = \frac{A}{V}, \quad R_o = \frac{O}{V}, \quad P = R_o = \frac{O}{A + O}$$

其中 R_a 為可停率， R_o 與 P 分別為占用率及停車壓力。路段排名另以高壓力時段佔比衡量：

$$H = \frac{N(P \geq 0.85)}{N_{\text{valid}}}$$

本研究將占用率達 85% 設為高壓力的操作性門檻，這是本次 EDA 的描述性分類，不是官方停車管理標準。實作上，整體、日期與時段占用率以篩選後資料的 is_occupied 平均值計算。路段排名先排除有效觀測數低於所有路段中位數 80% 的路段，再按 section_id 與十分鐘時間桶計算各路段的時段占用率，依高壓力時段佔比排序，若比例相同則以平均占用率作為第二排序。

4 分析結果

以下依序比較日期、時段與路段的停車壓力，再檢查短期自我相關與資料品質，對應前述研究問題。

4.1 日期與整體占用率

報告範圍的整體占用率為 **69.8%**，可停率為 **30.2%**。端午節當日與連假週末的占用率均高於週四一般日。

日期類型	資料筆數	可停率	占用率
一般日	84,384	32.0%	68.0%
端午節	84,384	29.8%	70.2%
端午節連假	168,768	29.6%	70.4%

逐日比較顯示，週六 6 月 20 日占用率最高，為 **71.5%**。週四最低，為 **68.0%**。

(逢甲商圈周邊路邊停車壓力分析：以端午節連假期間為例)

日期	星期	資料筆數	可停率	占用率
2026-06-18	週四	84,384	32.0%	68.0%
2026-06-19	週五	84,384	29.8%	70.2%
2026-06-20	週六	84,384	28.5%	71.5%
2026-06-21	週日	84,384	30.7%	69.3%

4.2 時段差異

為比較各時段與日期類型的差異，如圖 20(a) 所示，先比較各時段的平均停車壓力。如圖 20(b) 所示，再比較不同日期類型的逐小時變化。

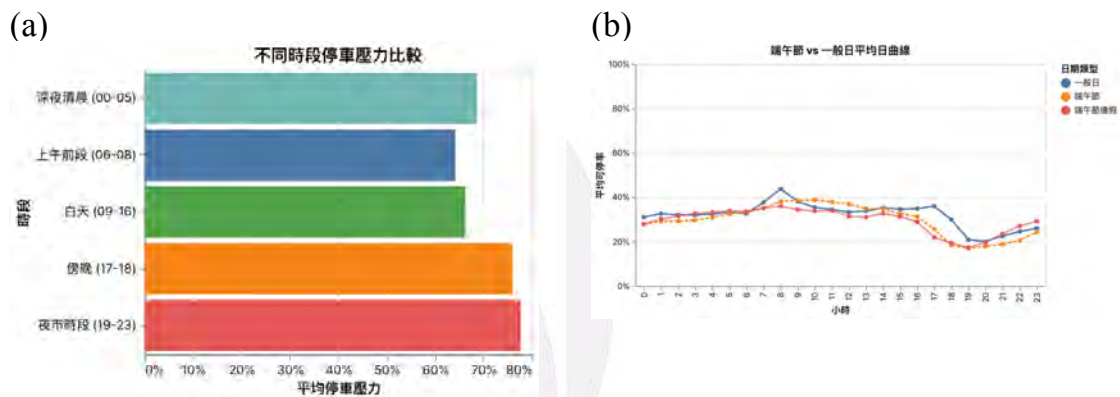


圖 20 時段與日期類型的停車占用情形。(a) 各時段平均停車壓力。(b) 不同日期類型的逐小時可停率變化。

時段	資料筆數	可停率	占用率
夜市時段 19-23 時	70,320	22.3%	77.7%
傍晚 17-18 時	28,128	24.1%	75.9%
深夜清晨 00-05 時	84,384	31.3%	68.7%
白天 09-16 時	112,512	33.8%	66.2%
上午前段 06-08 時	42,192	35.8%	64.2%

平均占用率最高的是夜市時段，其次為傍晚。單一小時的最高值出現在 6 月 20 日 19 時，占用率約 **85.1%**。最低值出現在 6 月 18 日 8 時，約 **56.2%**。

如圖 21 所示，日期與小時被放在同一張熱圖中，呈現各日期與小時的可停率變化。晚間色塊整體較深，表示可停率較低、占用率較高。

(逢甲商圈周邊路邊停車壓力分析：以端午節連假期間為例)

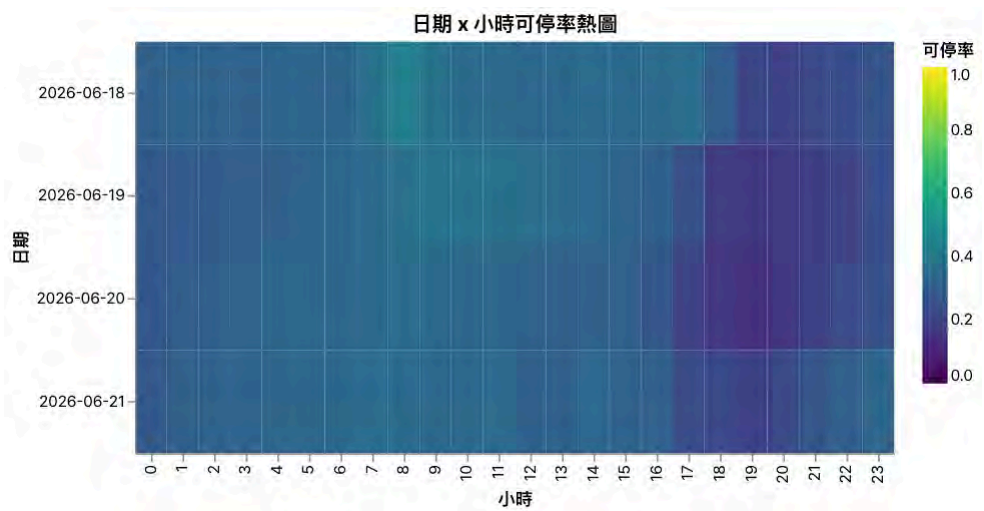


圖 21 日期與小時可停率熱圖

4.3 高占用路段

除了時間差異，本報告也比較各路段的壓力。如圖 22 所示，高占用主要集中在少數路段。

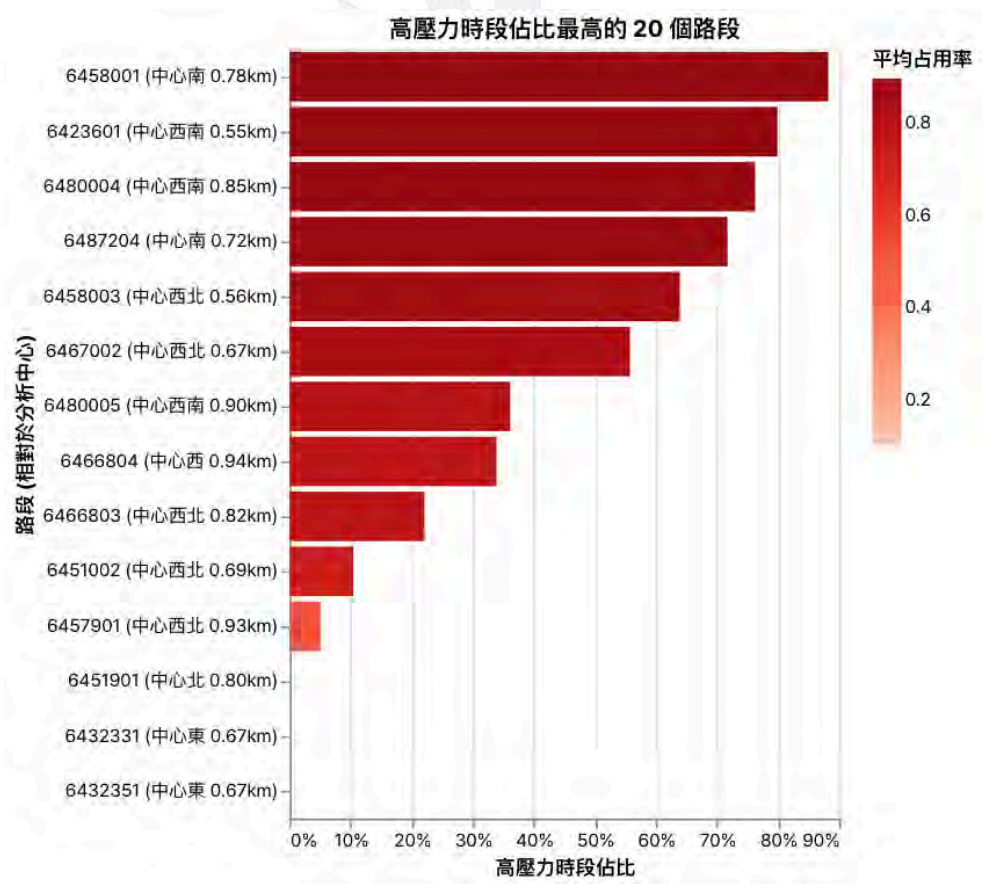


圖 22 高占用路段的初步比較

路段	高壓時段佔比	平均占用率	最高占用率	有效觀測數	車格數
6458001	88.2%	89.8%	97.1%	19,584	34
6423601	79.9%	89.3%	96.9%	55,872	97

(逢甲商圈周邊路邊停車壓力分析：以端午節連假期間為例)

路段	高壓時段佔比	平均占用率	最高占用率	有效觀測數	車格數
6480004	76.2%	88.8%	100.0%	11,520	20
6487204	71.7%	88.4%	100.0%	13,248	23
6458003	63.9%	86.0%	100.0%	12,672	22

路段 6458001 有 88.2% 的時段落在高壓力狀態，平均占用率亦達 89.8%，表示該路段不是偶爾爆滿，而是在多數觀測時段都偏滿。簡報圖中的「中心南」是相對於分析中心的方向標籤。依原始車格座標對照地圖，該路段主要位於臺中市西屯區福星路 656–693 號一帶，鄰近臺中市立圖書館西屯分館。部分座標接近河南路二段 301 巷周邊的停車空間。

如圖 23 所示，車格點位依平均可停率呈現於一公里分析範圍中，可觀察路段 6458001 所在的空間位置。

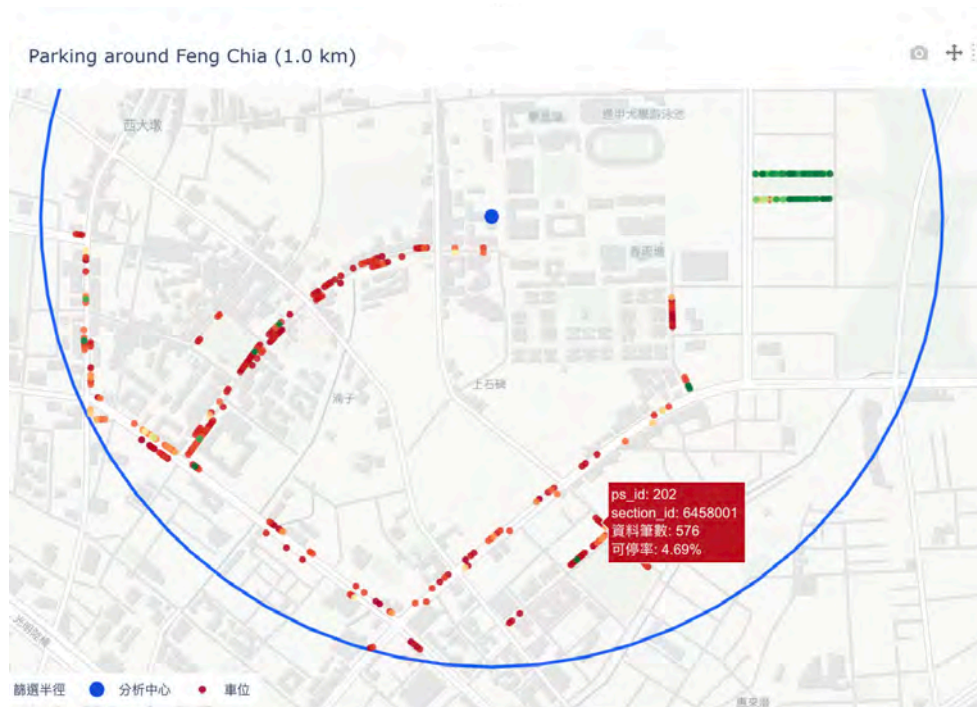


圖 23 車格平均可停率與空間分布

路段排名反映高壓力出現的頻率。路段與小時熱圖則進一步呈現壓力發生的時段。如圖 24 所示，有些路段全天偏滿，有些則主要在晚間升高。

(逢甲商圈周邊路邊停車壓力分析：以端午節連假期間為例)

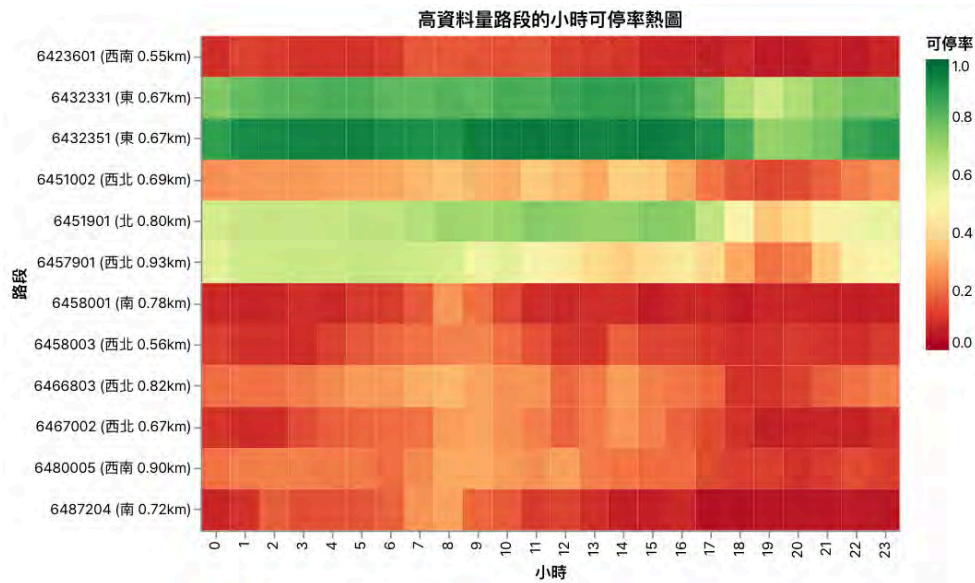


圖 24 路段與小時的停車壓力熱圖

4.4 自我相關

為檢查停車壓力的短期變化與自我相關，如圖 25(a) 所示，先觀察每小時停車壓力時間序列。如圖 25(b) 所示，再檢查 ACF 的各 lag 自我相關。

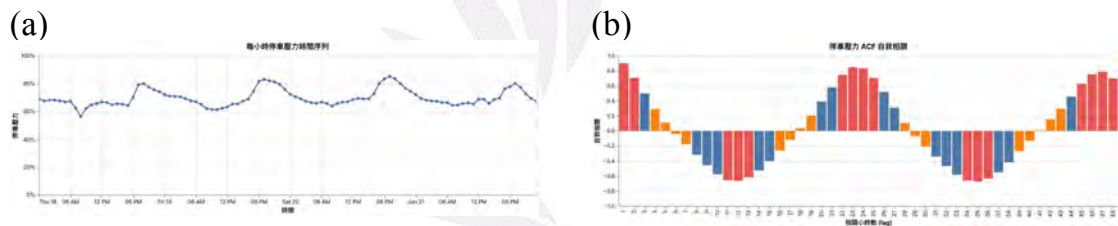


圖 25 每小時停車壓力的時間序列與自我相關。(a) 每小時停車壓力時間序列。
(b) 每小時停車壓力之 ACF。

一小時落後的自我相關為 0.90，23 與 24 小時落後分別為 0.85 與 0.84。47 與 46 小時落後則為 0.79 與 0.76。ACF 顯示短時間內壓力具有延續性。24 小時附近的相關性較高，表示本次資料可能存在相近時刻的重複節奏。這些結果可先用於檢查短期節奏。若要判斷穩定的日週期，後續仍需延長觀測期間 (Hyndman & Athanasopoulos, 2021)。

4.5 資料品質

資料品質檢查分為批次覆蓋率、抓取間隔與感測器狀態。如圖 26 所示，圖中呈現相鄰抓取間隔的分布。表格則列出預期批次、缺口與感測器狀態等指標。

(逢甲商圈周邊路邊停車壓力分析：以端午節連假期間為例)

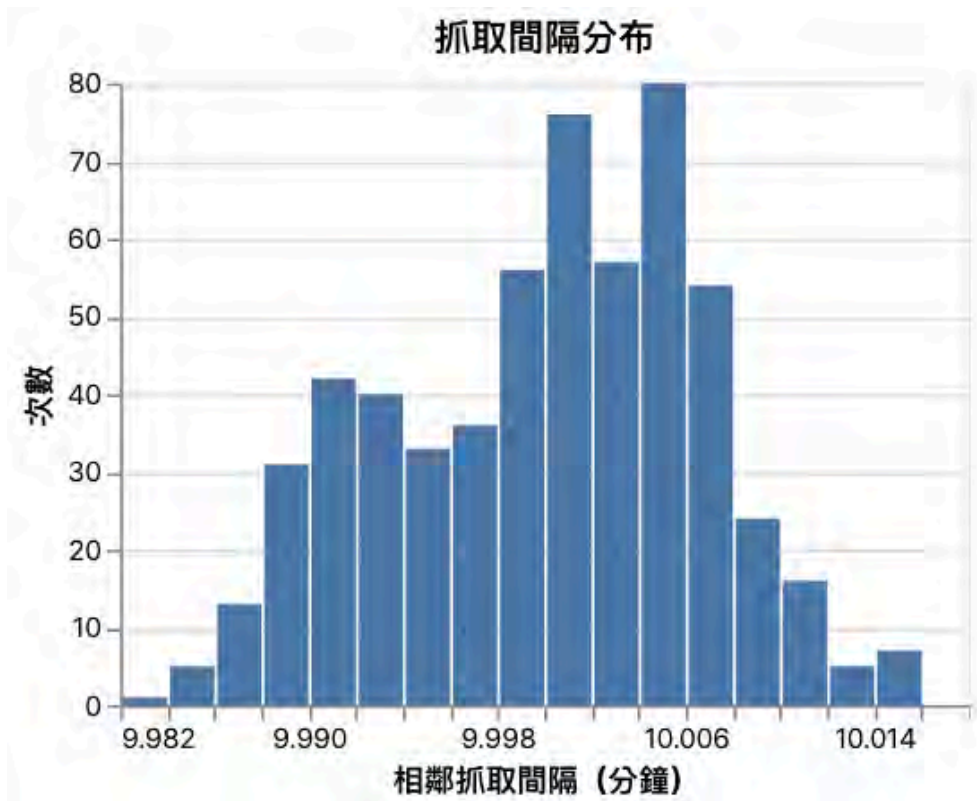


圖 26 相鄰抓取間隔分布

品質指標	結果
預期 / 實際抓取批次	576 / 576
抓取覆蓋率	100.0%
中位數抓取間隔	10.0 分鐘
最大抓取間隔	10.02 分鐘
超過 15 分鐘的缺口	0
感測器異常率	0.0%
有效狀態比例	100.0%

報告期間共完成預期的 576 個十分鐘抓取批次，未出現超過十五分鐘的資料缺口，也未出現 status = 2 的感測故障資料。這裡的 100% 覆蓋率指預期批次全部完成，不等同於所有原始欄位都沒有缺值。

5 結果與討論

在本次一公里、四天的觀測範圍內，晚間停車壓力較白天集中。週六、夜市時段與少數路段的占用率較高。這些結果可作為後續規劃分時停車資訊、熱點導引或延伸分析的資料基礎。

本報告以日期、時段與路段差異的描述為主。若要分析駕駛人身分、旅次目的或活動人流等因素，後續需補充相應的外部資料。

(逢甲商圈周邊路邊停車壓力分析：以端午節連假期間為例)

5.1 空間呈現

專案曾嘗試以地圖呈現車格分布。初次成果受地圖 API 與中文字型設定影響，後續調整設定後完成逢甲商圈周邊停車格的空間分布圖。本圖呈現車格的位置與分布 (OpenStreetMap contributors, 2026)。停車壓力則依占用率及高壓力時段佔比判讀。

如圖 27(a) 所示，初次結果出現底圖與中文字型顯示異常。調整 API 與字型設定後，如圖 27(b) 所示，已能辨識車格位置及其空間分布。

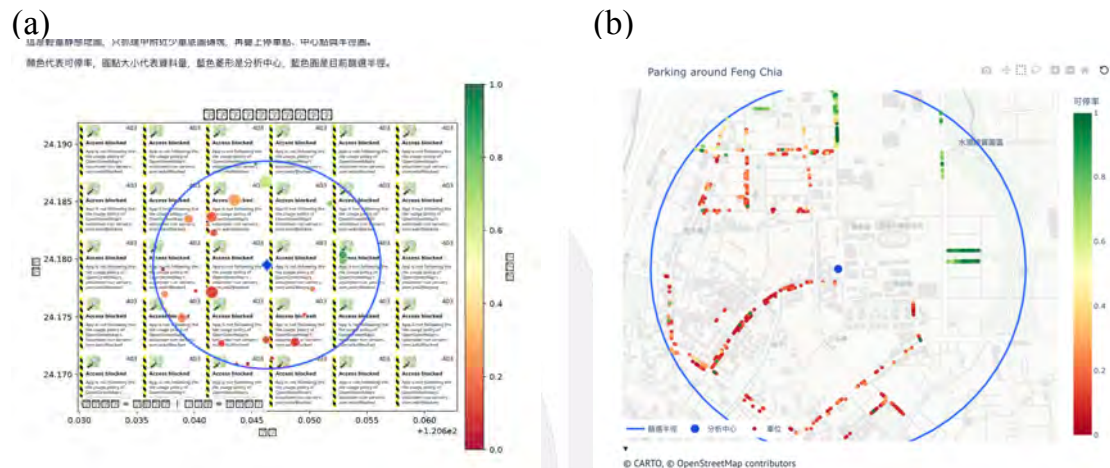


圖 27 停車格地圖修正前後比較。(a) 初次地圖繪製結果及顯示問題。(b) 修正後的停車格空間分布。

5.2 報告限制

1. 目前研究僅涵蓋端午節連假前後四天。後續需補充多個一般週、不同節日與其他季節資料，才能比較全年停車情形。
2. 目前資料僅涵蓋開放資料中有紀錄的路邊車格。若要估計商圈完整停車供給，後續需併入其他停車設施資料。
3. 未設置地磁感測器的道路不會出現在資料中。後續可透過其他資料來源補充這些路段。
4. API 更新時間與本專案蒐集時間可能不同步。後續可同時保存兩種時間戳，以比較資料更新與實際擷取的差異。
5. 感測器異常或資料缺漏可能影響占用率計算。本次觀測期間未發現相關紀錄，後續仍可持續監測資料狀態。
6. 目前資料反映取得當下的車位狀態。若要提供使用者抵達時的可用性資訊，後續需結合即時更新或預測模型。
7. 本研究聚焦區域與路段層級的停車壓力。若要進行單一車格即時預測，後續需累積更長時間序列並另行驗證模型。

6 結論

本專案以開放資料 API、Docker Compose、collector 與 TimescaleDB 建立可重現的蒐集流程，再將資料整理為可解釋的停車壓力指標。分析期間的 576 個預

(逢甲商圈周邊路邊停車壓力分析：以端午節連假期間為例)

期抓取批次全部完成。端午節當日與連假週末、週六及夜市時段的占用率相對較高，少數路段也經常處於高壓力狀態。

若要進一步建立穩健的管理或預測模型，後續可延長蒐集期間，納入多個一般週、不同節日、天候與活動資料，並比較季節性。累積足夠資料後，再評估因果關係、長期規律與更細緻的即時應用。近期研究已示範結合停車感測資料、空間特徵、LSTM 與圖形 Transformer 的停車預測方法 (Gong et al., 2023 ; Zhang et al., 2024 ; Zhou et al., 2024) 。

7 學習心得

7.1 陳鴻程

這次我主要負責 Azure VM、Docker Compose 部署，以及時段差異、路段分析、ACF 和資料品質檢查。這次真的把 TimescaleDB 和 collector 跑在 VM 上後，才比較清楚 Docker Compose 不是把所有東西塞進同一個容器，並且也對於它的操作和設定上有更進一步的了解。

在系統部分，我們用 crontab 每 10 分鐘啟動 collector，讓它去抓臺中市停車 API，再把資料寫進 TimescaleDB。這讓整個資料蒐集流程可以自動跑，不用一直手動執行程式。我也比較理解課堂上講的資料管道，要讓資料可以穩定進到資料庫。

分析結果中，我覺得比較有意思的是晚間和夜市時段的停車壓力明顯比較高，而且有些路段壓力特別集中。ACF 也顯示停車壓力有短期延續性，代表現在很滿的話，下一小時通常也還是偏滿。以上也讓我發現，許多觀念必須實際操作後，才能確認自己是否真正理解。

7.2 黃智佳

這次我負責本地端 sql schema、Docker Compose 初擬以及資料整理，以及端午節連假四天的占用率分析。一開始我以為只要把政府資料抓下來就能直接畫圖，但實際做才發現，資料量很大，而且還要先整理時間、篩選逢甲周邊 1 公里、確認欄位和狀態值，後面算出來的結果才有意義。

在分析四天資料時，我比較了週四、端午節、週六和週日的占用率。結果可以看出，逢甲周邊路邊停車格本來占用率就偏高，其中週六最高，夜市和連假時段也比較容易滿。這讓我覺得停車問題不能只靠感覺判斷，用資料整理後會更清楚看到哪一天壓力比較大。

這次最大的收穫是，分析前要先清理資料、定義指標，再解釋結果。實際操作也讓我更清楚資料整理、指標設計與結果解讀必須相互配合。

參考文獻

1. Arjona Martinez, J., Linares, M. P., & Casanovas, J. (2021). Characterizing parking systems from sensor data through a data-driven approach. *Transportation Letters*, 13(3), 183–192. <https://doi.org/10.1080/19427867.2020.1866331>
2. Balmer, M., Weibel, R., & Huang, H. (2021). Value of incorporating geospatial information into the prediction of on-street parking occupancy: A case study. *Geospatial Information Science*, 24(3), 438–457. <https://doi.org/10.1080/10095020.2021.1937337>
3. DolphinHuang. (2026). *parking_analysis* [Computer software]. GitHub. https://github.com/DolphinHuang/parking_analysis
4. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. <https://www.deeplearningbook.org/>
5. Gong, S., Qin, J., Xu, H., Cao, R., Liu, Y., Jing, C., Hao, Y., & Yang, Y. (2023). Spatio-temporal parking occupancy forecasting integrating parking sensing records and street-level images. *International Journal of Applied Earth Observation and Geoinformation*, 118, Article 103290. <https://doi.org/10.1016/j.jag.2023.103290>
6. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
7. Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and practice* (3rd ed.). OTexts. <https://otexts.com/fpp3/>
8. OpenStreetMap contributors. (2026). *OpenStreetMap* [地圖資料]. 路段代表座標地圖。
9. Schneble, M., & Kauermann, G. (2025). Statistical modelling of on-street parking spot occupancy in smart cities. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 74(4), 1128–1149. <https://doi.org/10.1093/jrssc/qlaf017>
10. Sun, Z., Wu, J., Wang, F., Tian, Z., & He, Q. (2025). Survival analysis of parking duration in urban commercial areas: A case study in Zhengzhou, China. *PLOS ONE*, 20(6), Article e0323207. <https://doi.org/10.1371/journal.pone.0323207>
11. Wu, F., & Ma, W. (2022). Clustering analysis of the spatio-temporal on-street parking occupancy data: A case study in Hong Kong. *Sustainability*, 14(13), Article 7957. <https://doi.org/10.3390/su14137957>
12. Zhang, F., Shang, K., Yan, L., Nan, H., & Miao, Z. (2024). Prediction of parking space availability using improved MAT-LSTM network. *ISPRS International Journal of Geo-Information*, 13(5), Article 151. <https://doi.org/10.3390/ijgi13050151>
13. Zhou, Y., Ye, X., Yan, X., Wang, T., & Chen, J. (2024). Multi-step parking demand prediction model based on multi-graph convolutional Transformer. *Systems*, 12(11), Article 487. <https://doi.org/10.3390/systems12110487>
14. 臺中市政府資料開放平臺. (2026). 臺中市路邊剩餘車位 [資料集]. 資料集網站。