Seoul Land Consulting

2조
김호연, 김현재, 이세훈, 허인경

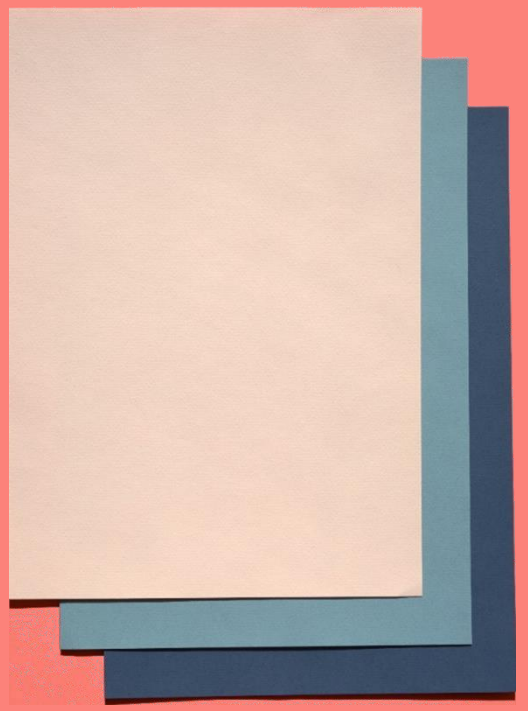
01
■ 목적

02
■ 환경

03
■ 분석 및 과정

04
■ 결과

05
■ 차후 계획



Part 1. 목적

PART 1

목 적

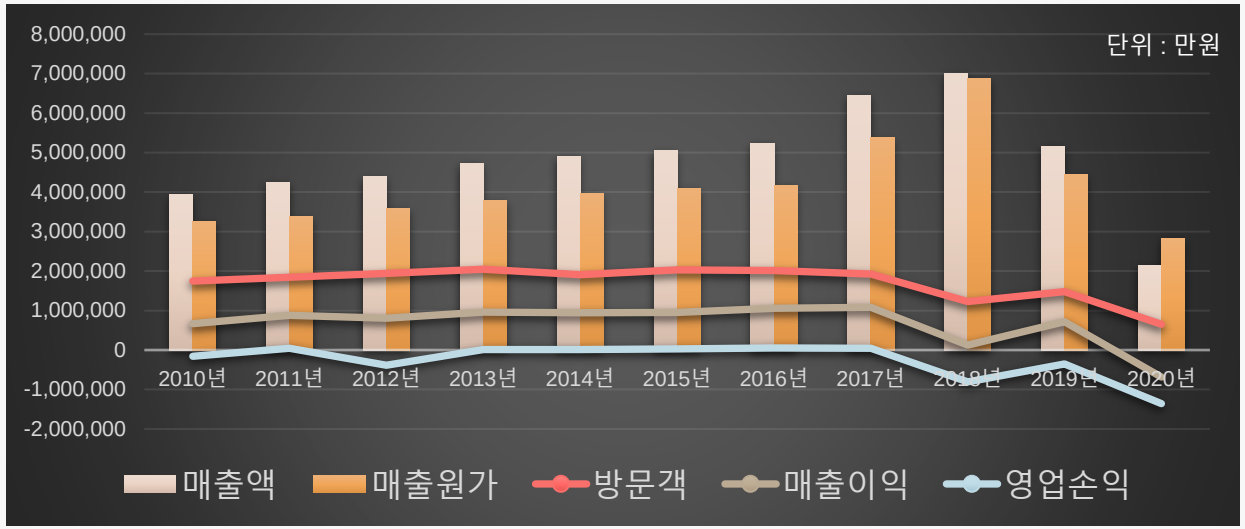
Cause Analysis

Consulting

feedback

PART 1

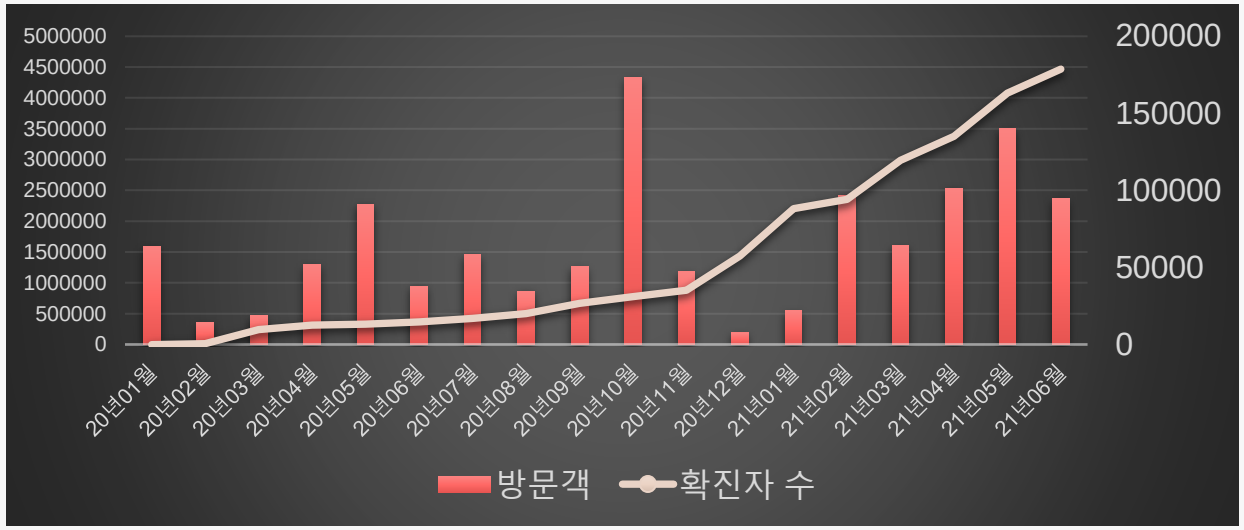
목적



원인 분석 Cause Analysis

1) 방문객은 왜 떨어졌는가?

- 서울 랜드를 방문했던 방문객은 서울 랜드를 어떻게 평가를 하는가?



2) 코로나의 영향은 있는가?

- 20년 1월 부터 코로나 여파로 인하여 방문자 수 가 영향이 있는가?

Part 2. 환경

PART 2

환경

1) 서버 환경

OS : Linux(CentOS 7)
MEM : 8GB
HDD : 1TB
S/W : Jupyter Lab, anaconda, SQL(mariadb),
Elastic Search & kibana

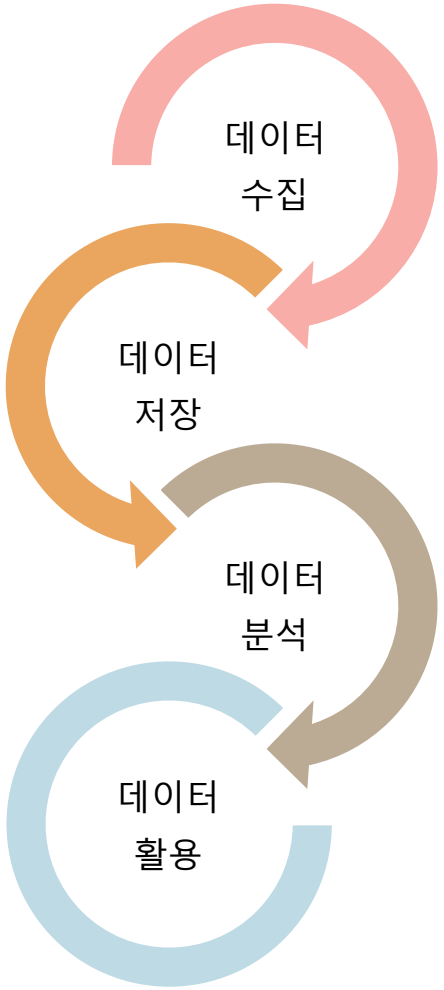
2) 언어 및 기술

Language : Python
Library : ML, NLP, pandas, numpy, matplotlib …

Part 3. 분석 및 과정

PART 3

분석 및 과정



1) 분석 과정

① 데이터 수집
- 크롤링
- 공공데이터
- 서울랜드

② 데이터 저장
- DB
- Excel

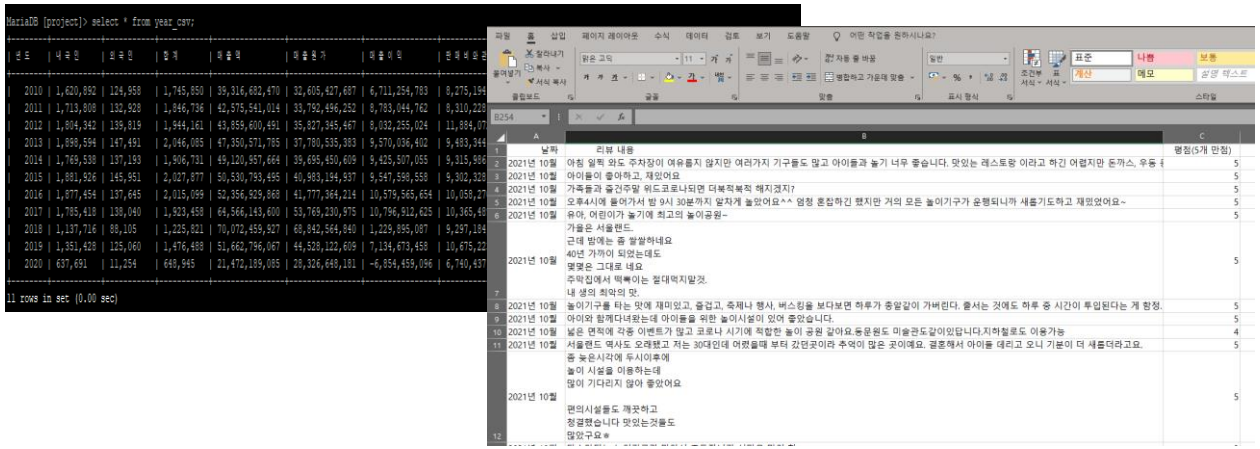
③ 데이터 분석
- 알고리즘
- NLP, ML

④ 데이터 활용
- 시각화

분석 및 과정

2) 데이터 수집 및 저장

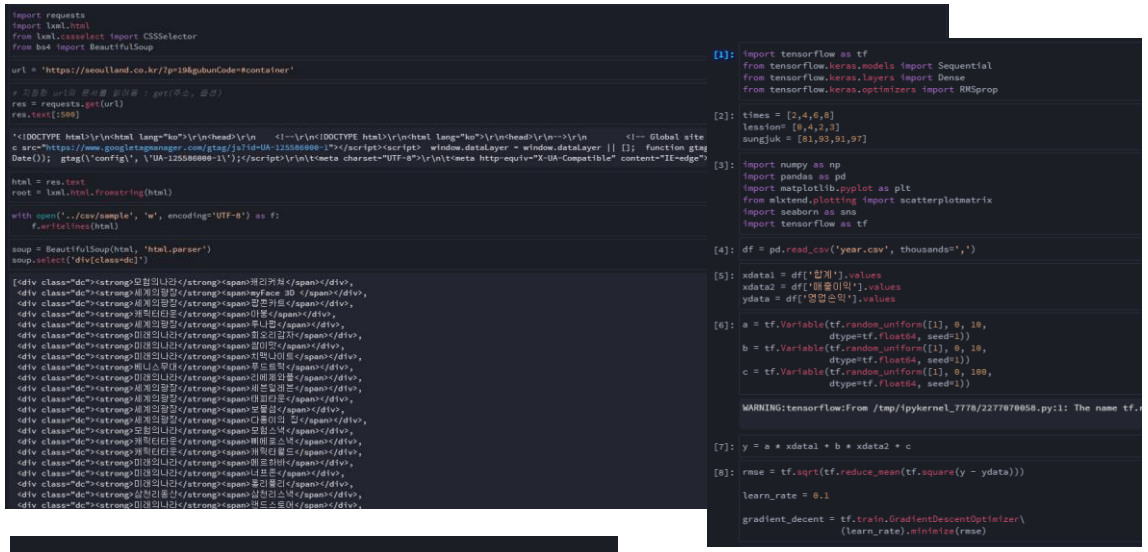
① 데이터 수집
- 크롤링 – 구글, 네이버, 구글스토어
- 공공데이터 – 관광지식정보시스템, DART, 기상청
- 서울랜드 – 놀이기구 정보

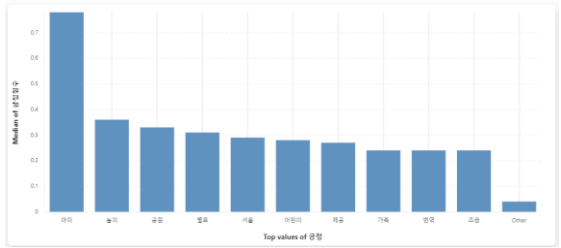


② 데이터 저장
- DB - MariaDB
- Excel - CSV

PART 3

분석 및 과정





3) 데이터 분석 및 활용

③ 데이터 분석
- 알고리즘 – 추가 코스 용 알고리즘
- NLP, ML – 감정분석 및 상관분석

④ 데이터 활용
- 시각화 – Elastic Search & kibana, Jupyter lab

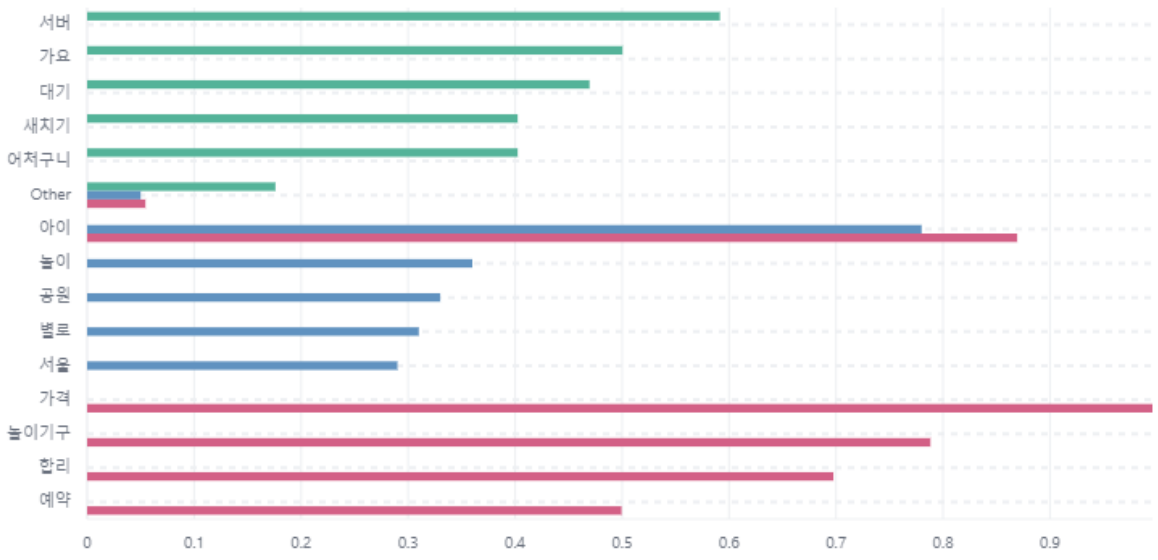
Part 4. 결과

PART 4

결과

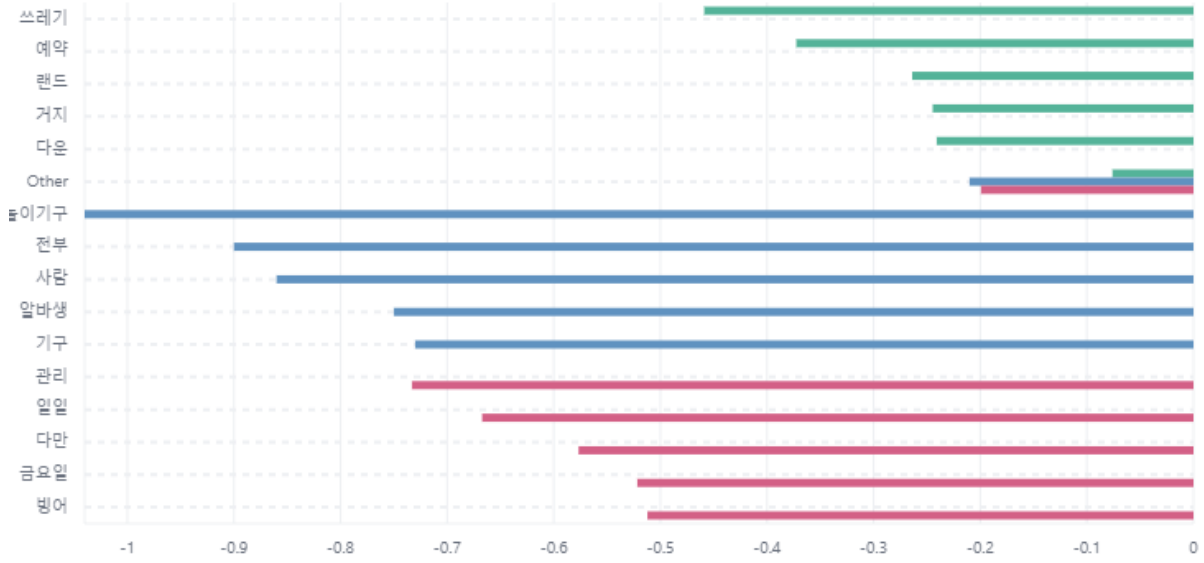
1. 리뷰 감정 분석

긍정



1) 구글 긍정 : **아이**, 놀이, 공원
2) 네이버 긍정 : 가격, **아이**, 놀이기구

부정

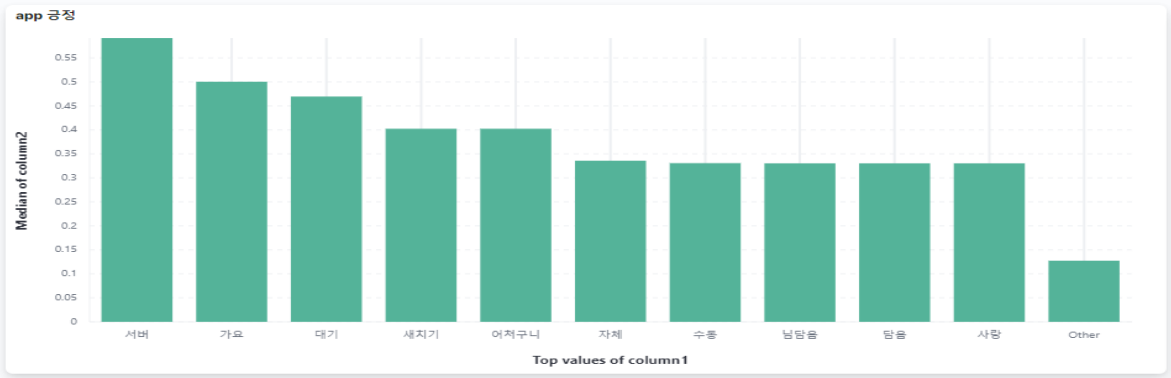


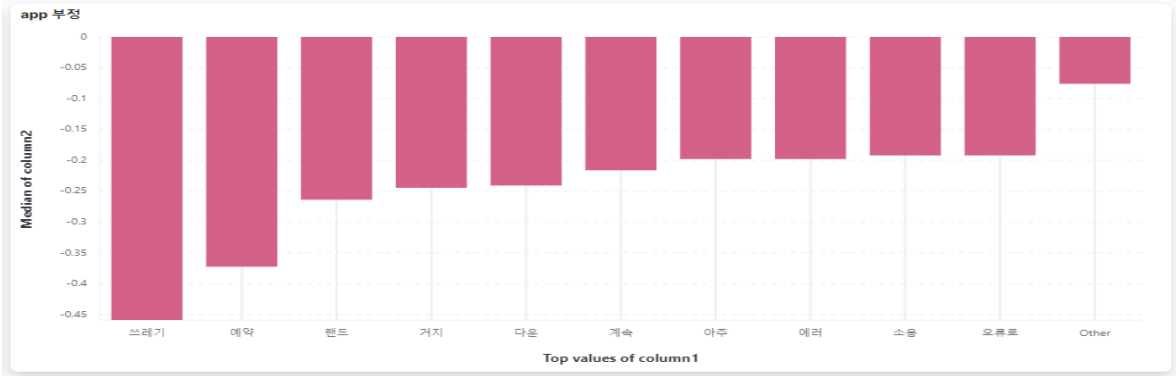
1) 구글 부정 : **놀이기구**, 사람, 알바생
2) 네이버 부정 : 관리, 금요일, 주차장, 할로윈데이

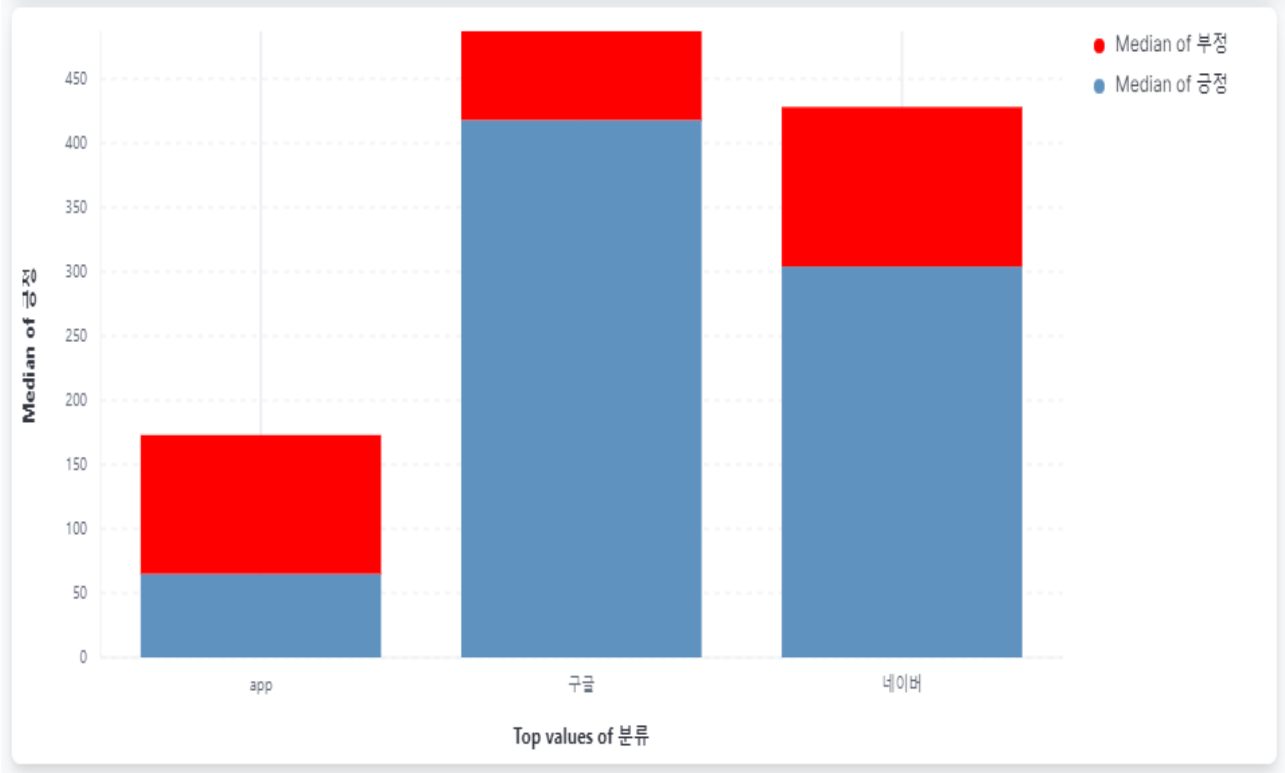
PART 4

결과

1. 리뷰 감정 분석





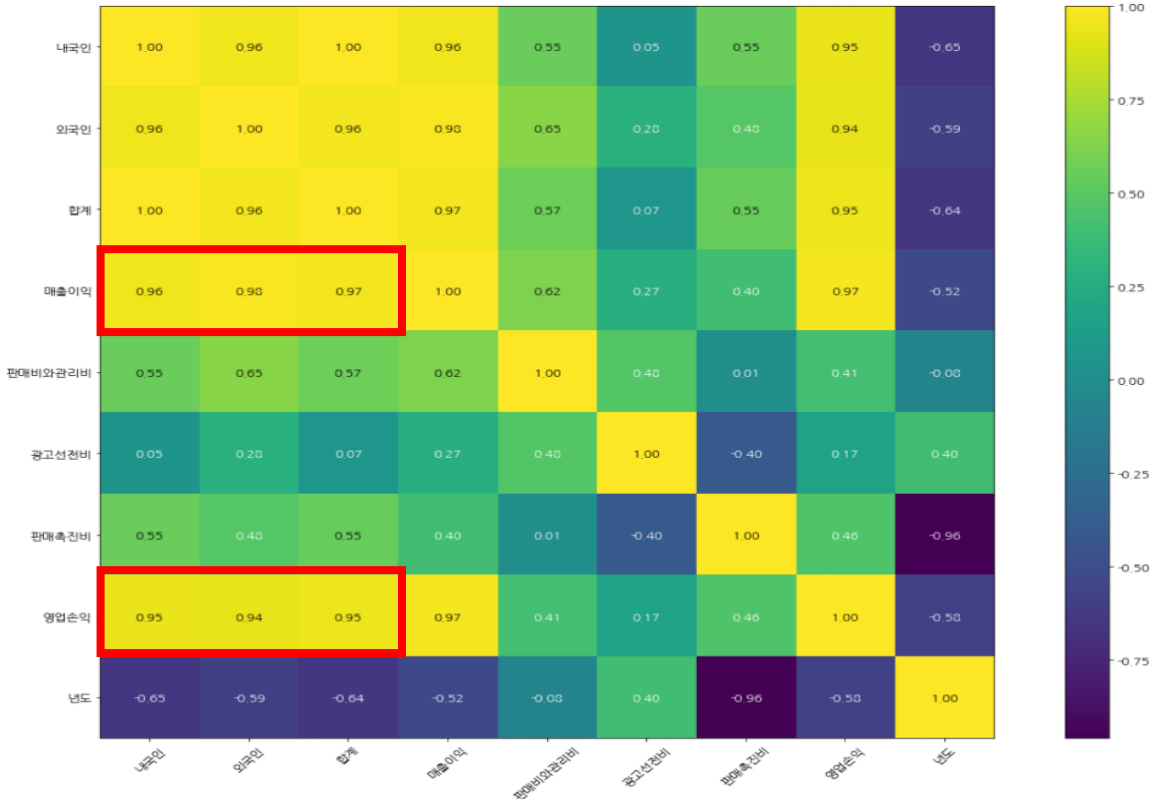


1) 구글app 긍정 : 가요, **대기**
2) 구글app 부정 : **쓰레기**, 예약, 랜드

1) 총 긍정 리뷰 : 72.3 %
2) 총 부정 리뷰 : 27.7 %

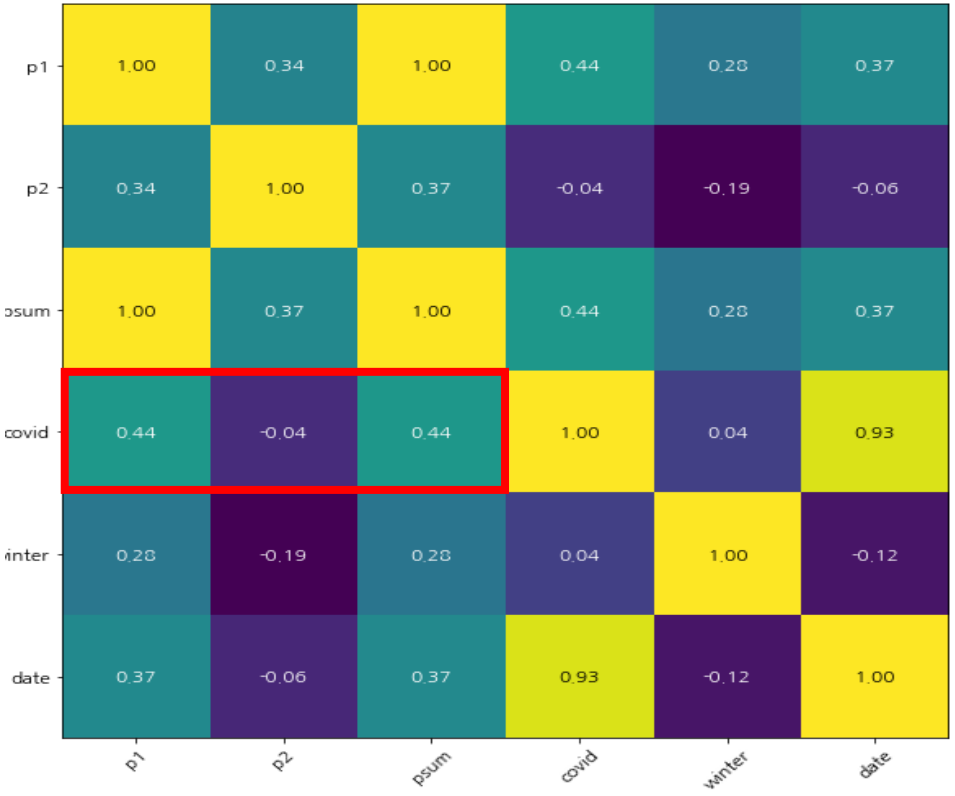
PART 4

결과



1) Columes : 내국인,외국인,합계,매출이익,판매비와관리비,
광고선전비,판매촉진비,영업손익,년도
2) 상관도 분석평 : 방문객(내국인,외국인) 총 합은
매출이익과 영업손익에 상관분석
관계도가 높다.

2. 상관분석



1) Columes : 내국인, 외국인, 합계, 코로나확진자, 날씨, 월별
2) 상관도 분석평 : 방문객(내국인,외국인) 총 합은
코로나와 관계도가 비교적 높지는 않지만
영향은 받으며, 날씨와는 관계가 없는
걸로 나옵니다.

PART 4

결과

```
for i in range(len(ridsn)):
    # 길찾기
    chrome.find_element_by_xpath('/html/body/div[5]/div[1]/div/div/ul/li[2]/a').send_keys(Keys.RETURN)
    ti.sleep(2)
    # 출발지
    chrome.find_element_by_xpath('/html/body/div[5]/div[2]/div[2]/div[1]/div/div[1]/div[2]/form/input[1]').send_keys('서울랜드',rids1.name[0])
    chrome.find_element_by_xpath('/html/body/div[5]/div[2]/div[2]/div[1]/div/div[1]/div[2]/form/input[1]').send_keys(Keys.RETURN)
    # 목적지
    ti.sleep(2)
    chrome.find_element_by_xpath('/html/body/div[5]/div[2]/div[2]/div[1]/div/div[2]/div[2]/form/input[1]').send_keys('서울랜드',rids1.name[i+1])
    chrome.find_element_by_xpath('/html/body/div[5]/div[2]/div[2]/div[1]/div/div[2]/div[2]/form/input[1]').send_keys(Keys.RETURN)
    ti.sleep(2)
    # 길찾기 시작
    chrome.find_element_by_xpath('/html/body/div[5]/div[2]/div[2]/div[2]/div/a[3]').send_keys(Keys.RETURN)
    ti.sleep(2)
    # 걸리는 시간
    time1 = chrome.find_element_by_xpath('/html/body/div[5]/div[2]/div[2]/div[5]/div[3]/div[1]/ul/li[1]/div[1]/div/p/span[1]').text
    # 거리 m
    m = chrome.find_element_by_xpath('/html/body/div[5]/div[2]/div[2]/div[5]/div[3]/div[1]/ul/li[1]/div[1]/div/p/span[2]').text
    # 결과 출력
    print('출발지 : ', ridsn[0], ', 목적지 : ',ridsn[i+1], ', 시간 : ',time1, ', 거리 m : ',m)
    # 결과 저장 csv
    word1 =[ridsn[0], ridsn[i+1], time1, m]
    with open('ridestm.csv', 'a', newline='') as f:
        wr = csv.writer(f)
        wr.writerow(word1)

    ti.sleep(1)
```

```
출발지 :  은하열차888 , 목적지 :  동심비행선 , 시간 :  2분 , 거리 m :  136m
출발지 :  은하열차888 , 목적지 :  빅회전목마 , 시간 :  6분 , 거리 m :  400m
출발지 :  은하열차888 , 목적지 :  월드컵 , 시간 :  3분 , 거리 m :  226m
출발지 :  은하열차888 , 목적지 :  도깨비바람 , 시간 :  6분 , 거리 m :  418m
출발지 :  은하열차888 , 목적지 :  쥬라기랜드 , 시간 :  4분 , 거리 m :  256m
출발지 :  은하열차888 , 목적지 :  킹바이킹 , 시간 :  9분 , 거리 m :  573m
출발지 :  은하열차888 , 목적지 :  블랙홀 2000 , 시간 :  3분 , 거리 m :  223m
출발지 :  은하열차888 , 목적지 :  라바 트위스터 , 시간 :  4분 , 거리 m :  264m
출발지 :  은하열차888 , 목적지 :  은하열차888 , 시간 :  , 거리 m :
출발지 :  은하열차888 , 목적지 :  베스트키즈 , 시간 :  7분 , 거리 m :  471m
출발지 :  은하열차888 , 목적지 :  급류타기 , 시간 :  9분 , 거리 m :  603m
출발지 :  은하열차888 , 목적지 :  티키목열차 , 시간 :  8분 , 거리 m :  525m
출발지 :  은하열차888 , 목적지 :  출동! 슈퍼윙스 , 시간 :  2분 , 거리 m :  152
출발지 :  은하열차888 , 목적지 :  브루미즈 동산 , 시간 :  5분 , 거리 m :  321m
출발지 :  은하열차888 , 목적지 :  착각의 집 , 시간 :  3분 , 거리 m :  219m
출발지 :  은하열차888 , 목적지 :  터닝메카드 레이싱 , 시간 :  6분 , 거리 m :
출발지 :  은하열차888 , 목적지 :  피터팬 , 시간 :  6분 , 거리 m :  376m
출발지 :  은하열차888 , 목적지 :  엑스플라이어 , 시간 :  3분 , 거리 m :  209m
출발지 :  은하열차888 , 목적지 :  알포소윙 , 시간 :  5분 , 거리 m :  349m
출발지 :  은하열차888 , 목적지 :  깜짝모험관 , 시간 :  15분 , 거리 m :  982m
출발지 :  은하열차888 , 목적지 :  달나라열차 , 시간 :  2분 , 거리 m :  116m
출발지 :  은하열차888 , 목적지 :  니나노고카트 , 시간 :  9분 , 거리 m :  588m
출발지 :  은하열차888 , 목적지 :  타임머신 , 시간 :  2분 , 거리 m :  105m
출발지 :  은하열차888 , 목적지 :  터닝메카드 레이싱 , 시간 :  6분 , 거리 m :
출발지 :  은하열차888 , 목적지 :  록까페 , 시간 :  3분 , 거리 m :  205m
출발지 :  은하열차888 , 목적지 :  샷드롭 , 시간 :  15분 , 거리 m :  995m
출발지 :  은하열차888 , 목적지 :  잠부비행기 , 시간 :  5분 , 거리 m :  339m
출발지 :  은하열차888 , 목적지 :  스카이엑스 , 시간 :  2분 , 거리 m :  147m
출발지 :  은하열차888 , 목적지 :  뭉게꽁항액션존 , 시간 :  3분 , 거리 m :  210
출발지 :  은하열차888 , 목적지 :  개구리만세 , 시간 :  5분 , 거리 m :  351m
출발지 :  은하열차888 , 목적지 :  미니바이킹 , 시간 :  5분 , 거리 m :  351m
출발지 :  은하열차888 , 목적지 :  해적소굴 , 시간 :  9분 , 거리 m :  575m
출발지 :  은하열차888 , 목적지 :  도래미악단 , 시간 :  3분 , 거리 m :  228m
출발지 :  은하열차888 , 목적지 :  구름빵 , 시간 :  5분 , 거리 m :  355m
출발지 :  은하열차888 , 목적지 :  춤추는 요술집 , 시간 :  5분 , 거리 m :  351m
출발지 :  은하열차888 , 목적지 :  카트라이더범퍼 , 시간 :  5분 , 거리 m :  349
출발지 :  은하열차888 , 목적지 :  붕붕카 , 시간 :  3분 , 거리 m :  229m
출발지 :  은하열차888 , 목적지 :  또봇트레인 , 시간 :  5분 , 거리 m :  329m
출발지 :  은하열차888 , 목적지 :  캐니멀 서커스 , 시간 :  3분 , 거리 m :  189
```

```
rids.head()
```

	name	imit	ppc	time(초)	우산	스릴	키즈	예상인원/탑승객 = 몫	예상대기시간 = 몫 * 놀이기구 시간(초)	놀이기구이름
0	동심비행선	제한없음	64	150	NaN	NaN	1.0	1	2분 30초	동심비행선
1	빅회전목마	100cm ~	68	150	1.0	NaN	1.0	1	2분 30초	빅회전목마
2	월드컵	130cm ~	40	150	NaN	1.0	NaN	2	5분	월드컵
3	도깨비바람	140cm ~ 185cm	40	180	NaN	1.0	NaN	2	6분	도깨비바람
4	쥬라기랜드	제한없음	100	없음	1.0	NaN	1.0	1	없음	쥬라기랜드

```
ridstm.head()
```

	출발지	목적지	시간(분)	거리(m)
0	도깨비바람	동심비행선	8분	506.0
1	도깨비바람	빅회전목마	6분	373.0
2	도깨비바람	월드컵	9분	596.0
3	도깨비바람	도깨비바람	NaN	NaN
4	도깨비바람	쥬라기랜드	10분	633.0

```
# 결측값 제거
ridstm = ridstm[ridstm['시간(분)'].notnull()]
ridstm = ridstm[ridstm['거리(m)'].notnull()]

ridstm = ridstm.sort_values(by='거리(m)')

ridstm.head(10)
```

	출발지	목적지	시간(분)	거리(m)
58	샷드롭	깜짝모험관	1분	80.0
9	도깨비바람	베스트키즈	1분	84.0
100	은하열차888	타임머신	2분	105.0
98	은하열차888	달나라열차	2분	116.0
30	도깨비바람	미니바이킹	2분	117.0
34	도깨비바람	춤추는 요술집	2분	117.0
29	도깨비바람	개구리만세	2분	117.0

3. 코스 알고리즘

1) 지도맵을 크롤링하여 놀이기구와 놀이기구 사이에 거리 및 도보 시간을 저장 함

2) 사용자가 임의의 놀이기구를 정하면, 거리에 따라 최소 시간에 맞춰 다음 놀이기구를 선택 함

3) 예상시간은 놀이기구 (운영인원 * 운영시간)/예상인원 을 시간으로 나타냄

4) 도보시간 및 예상시간을 결정해서 총 시간을 계산함

```
a = '도깨비바람'
b = '은하열차888'
c = '샷드롭'

ridsn = rids['놀이기구이름']
ridso = ridstm['출발지']
ridsp = ridstm['목적지']

result1 = ridstm[(ridso == a)&(ridsp == c)|(ridsp == b)].head(1)

result1
```

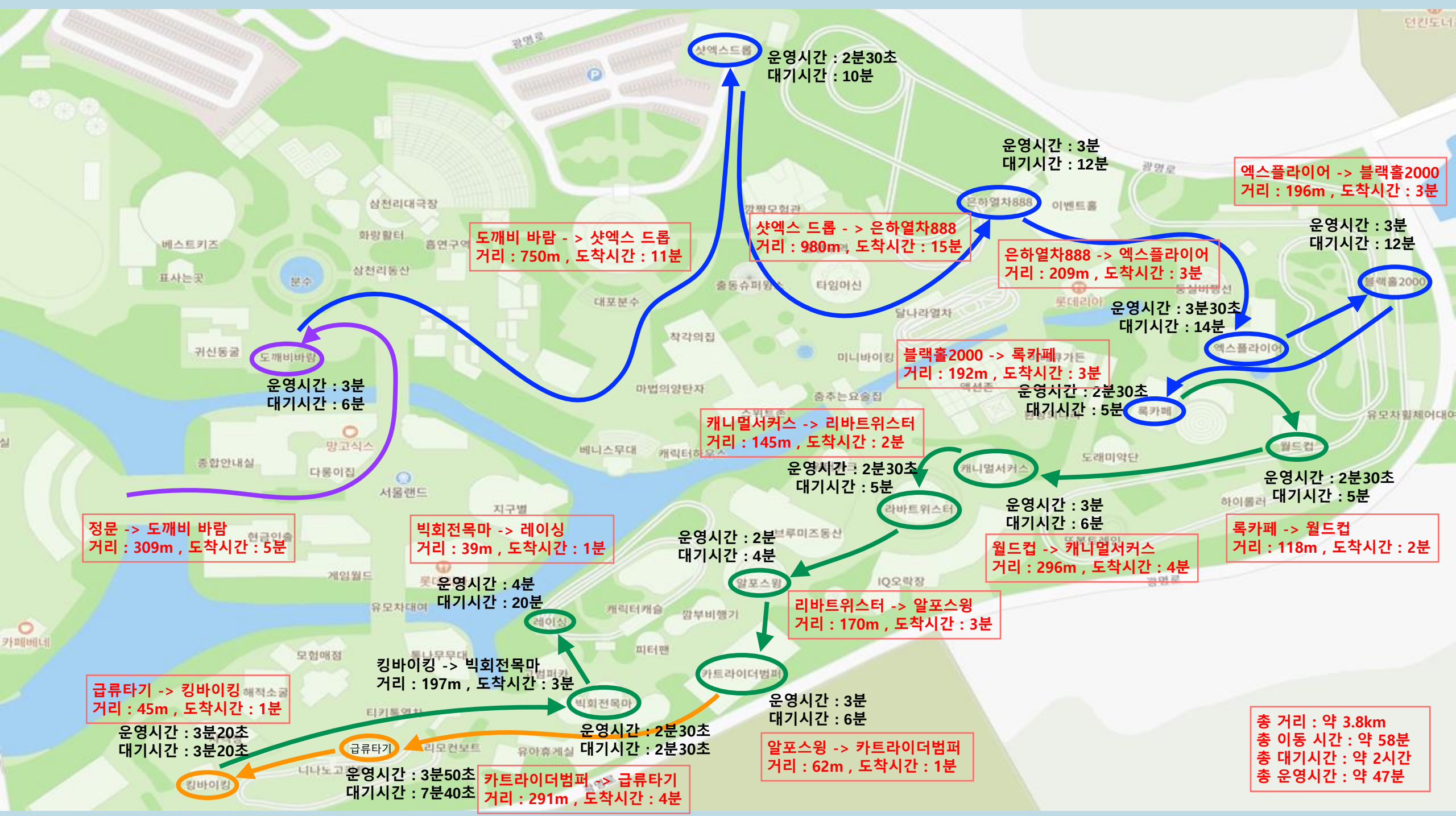
	출발지	목적지	시간(분)	거리(m)
25	도깨비바람	샷드롭	11분	750.0

```
result2 = rids[(ridsn == a)|(ridsn == b)|(ridsn == c)]

result2
```

	name	imit	ppc	time(초)	우산	스릴	키즈	예상인원/탑승객 = 몫	예상대기시간 = 몫 * 놀이기구 시간(초)	놀이기구이름
3	도깨비바람	140cm ~ 185cm	40	180	NaN	1.0	NaN	2	6분	도깨비바람
8	은하열차888	120cm ~	24	180	NaN	1.0	NaN	4	12분	은하열차888
25	샷드롭	140cm ~ 195cm	24	150	NaN	NaN	NaN	4	10분	샷드롭

#커플코스


샷엑스드롭
운영시간 : 2분30초
대기시간 : 10분
은하열차888
운영시간 : 3분
대기시간 : 12분
엑스플라이어 -> 블랙홀2000
거리 : 196m , 도착시간 : 3분
블랙홀2000
운영시간 : 3분
대기시간 : 12분
엑스플라이어
운영시간 : 3분30초
대기시간 : 14분
은하열차888 -> 엑스플라이어
거리 : 209m , 도착시간 : 3분
샷엑스 드롭 - > 은하열차888
거리 : 980m , 도착시간 : 15분
도깨비 바람 - > 샷엑스 드롭
거리 : 750m , 도착시간 : 11분
도깨비바람
운영시간 : 3분
대기시간 : 6분
블랙홀2000 -> 록카페
거리 : 192m , 도착시간 : 3분
록카페
운영시간 : 2분30초
대기시간 : 5분
캐니멀서커스 -> 리바트위스터
거리 : 145m , 도착시간 : 2분
캐니멀서커스
운영시간 : 2분30초
대기시간 : 5분
월드컵
운영시간 : 2분30초
대기시간 : 5분
록카페 -> 월드컵
거리 : 118m , 도착시간 : 2분
월드컵 -> 캐니멀서커스
거리 : 296m , 도착시간 : 4분
운영시간 : 3분
대기시간 : 6분
리바트위스터
운영시간 : 2분
대기시간 : 4분
리바트위스터 -> 알포스윙
거리 : 170m , 도착시간 : 3분
알포스윙
운영시간 : 3분
대기시간 : 6분
알포스윙 -> 카트라이더범퍼
거리 : 62m , 도착시간 : 1분
카트라이더범퍼
운영시간 : 2분30초
대기시간 : 2분30초
카트라이더범퍼 -> 급류타기
거리 : 291m , 도착시간 : 4분
빅회전목마
운영시간 : 4분
대기시간 : 20분
빅회전목마 -> 레이싱
거리 : 39m , 도착시간 : 1분
레이싱
킹바이킹 -> 빅회전목마
거리 : 197m , 도착시간 : 3분
킹바이킹
운영시간 : 3분50초
대기시간 : 7분40초
급류타기
운영시간 : 3분20초
대기시간 : 3분20초
급류타기 -> 킹바이킹
거리 : 45m , 도착시간 : 1분
정문 -> 도깨비 바람
거리 : 309m , 도착시간 : 5분
총 거리 : 약 3.8km
총 이동 시간 : 약 58분
총 대기시간 : 약 2시간
총 운영시간 : 약 47분

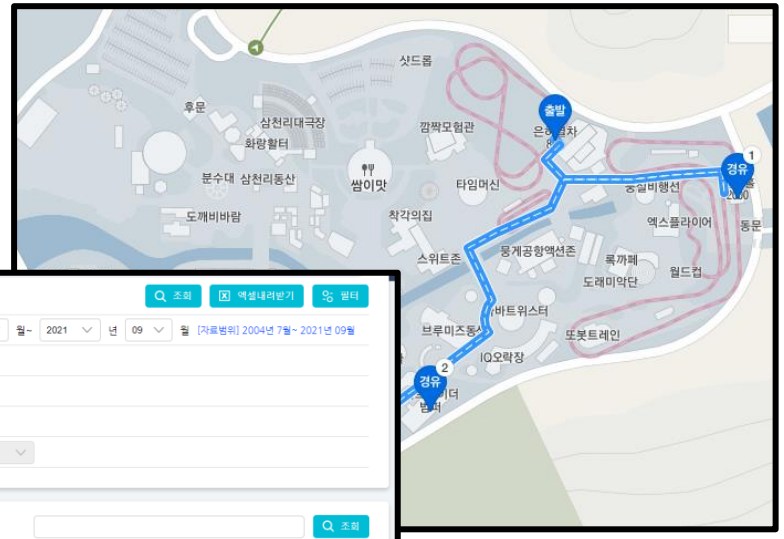
#아이와함께코스

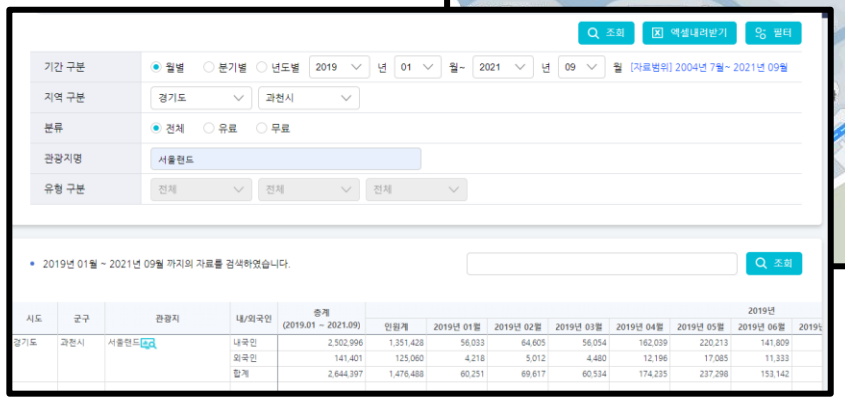

베스트키즈
운영시간 : 없음
대기시간 : 없음
슬동슈퍼윙스
운영시간 : 3분
대기시간 : 8분
개구리만세
미니바이킹
개구리만세
운영시간 : 2분40초
대기시간 : 37분20초
미니바이킹
운영시간 : 1분30초
대기시간 : 12분
둥실비행선
운영시간 : 2분30초
대기시간 : 2분30초
붕붕카
운영시간 : 2분40초
대기시간 : 13분
도래미악단
운영시간 : 2분30초
대기시간 : 7분30초
또봇트레인
운영시간 : 50초
대기시간 : 4분4.초
깡부비행기
운영시간 : 3분
대기시간 : 24분
피터팬
운영시간 : 3분
대기시간 : 15분
고범퍼카
운영시간 : 3분
대기시간 : 6분
회전목마
운영시간 : 2분30초
대기시간 : 2분30초
티키톡열차
운영시간 : 1분20초
대기시간 : 8분
해적소굴
운영시간 : 2분30초
대기시간 : 20분
급류타기
운영시간 : 3분50초
대기시간 : 7분40초
총 거리 : 약 2.4km
총 이동 시간 : 약 38분
총 대기시간 : 약 2시간46분
총 운영시간 : 약 35분

Part 5. 차후 계획

PART 5

차후계획





1) 추가 내용

1- 맵 API가 유료여서 자동으로 선을 이어 그리는 부분이 없었습니다.

2- 상관분석시 데이터가 많이 부족하여 분석이 많이 아쉬웠습니다.

2) 필요 데이터

1- 일별 방문객 데이터
(데이터를 요청했으나 보안 상 공개 불가)

2- 일별 매출액
(년도별 매출액만 공개-공시서류)

Thank You