글로벌 서비스를 위한 Timezone/DST

SPINRG CAMP 2023 / 한화솔루션 김대겸

Introduce

Career

한화솔루션 / 소프트웨어개발팀
2021.05.24 ~ ing

롯데정보통신 / e커머스팀
2019.07.02 ~ 2021.05.21

Info

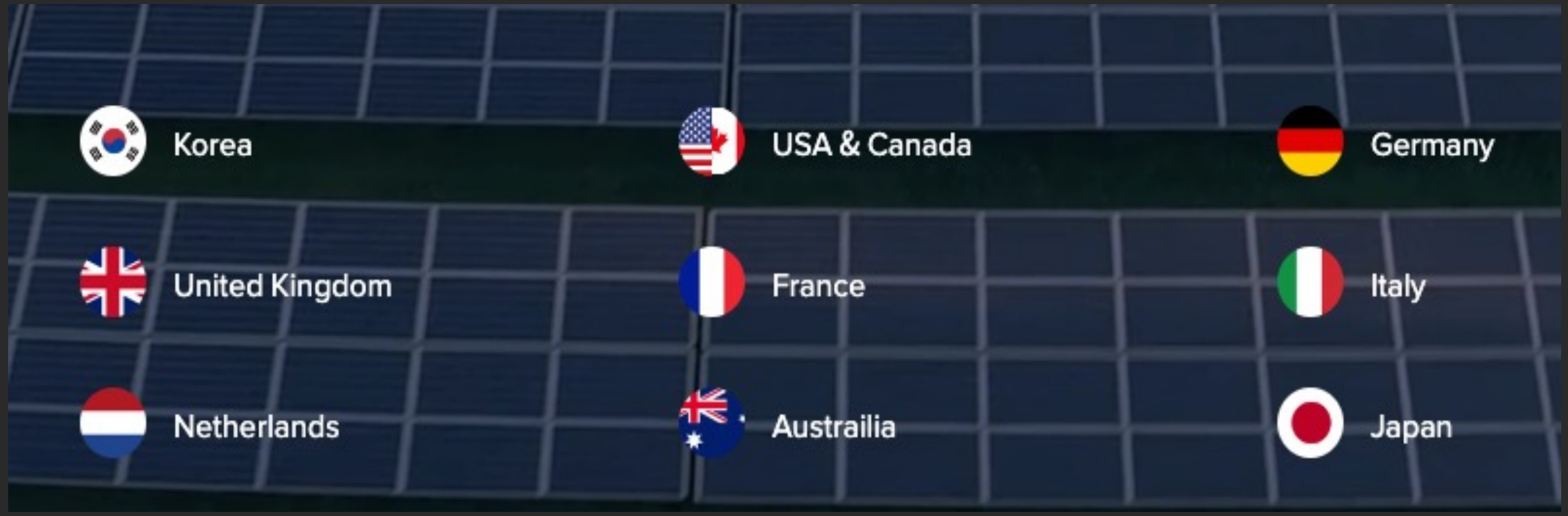
GitHub : https://github.com/gyeom
E-mail : koreatech93@naver.com

1. 서비스 소개 및 이슈배경

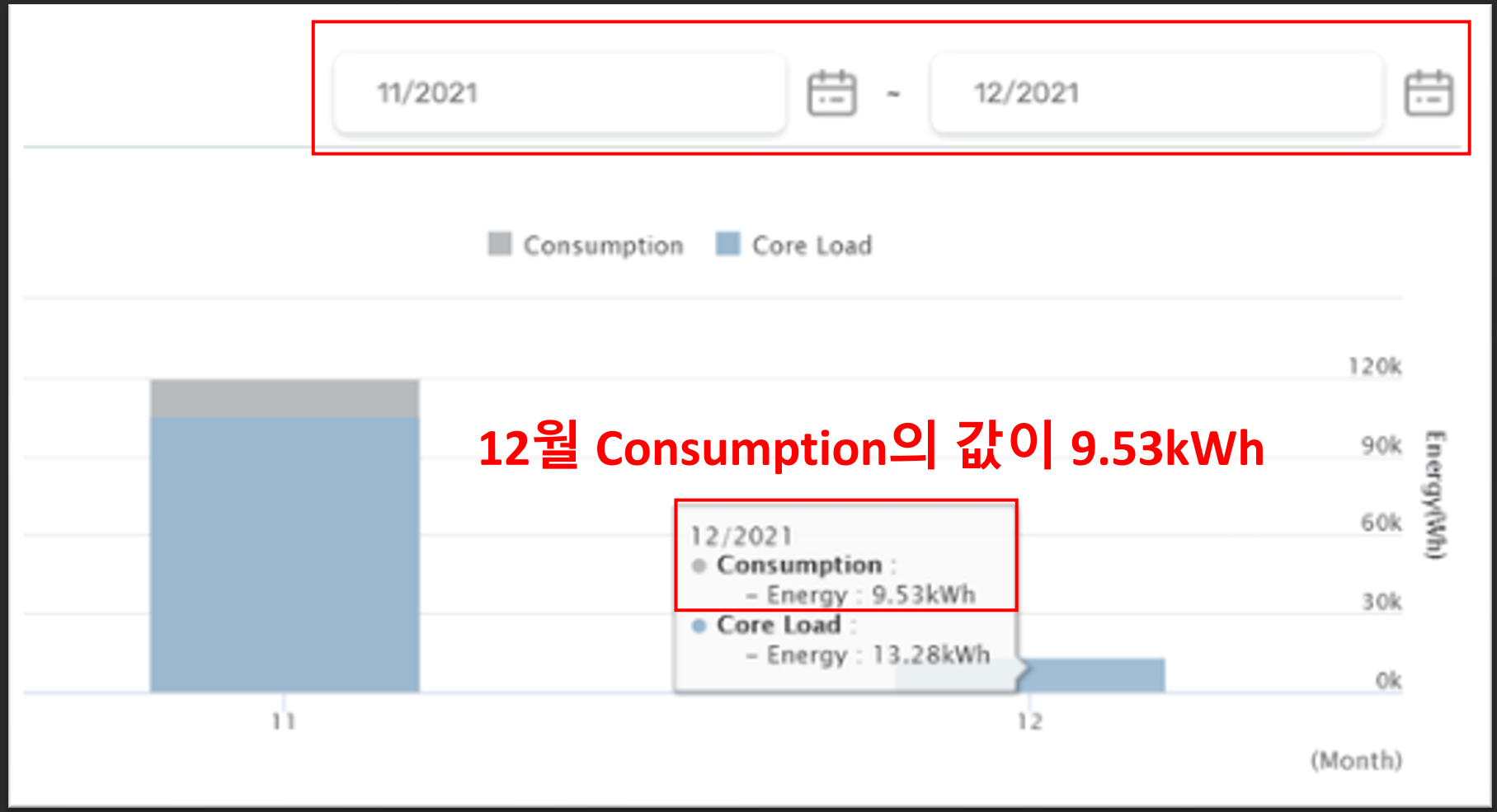
1. HEMS란?
2. 파리에서 생긴 일

HEMS (Home Energy Management System)

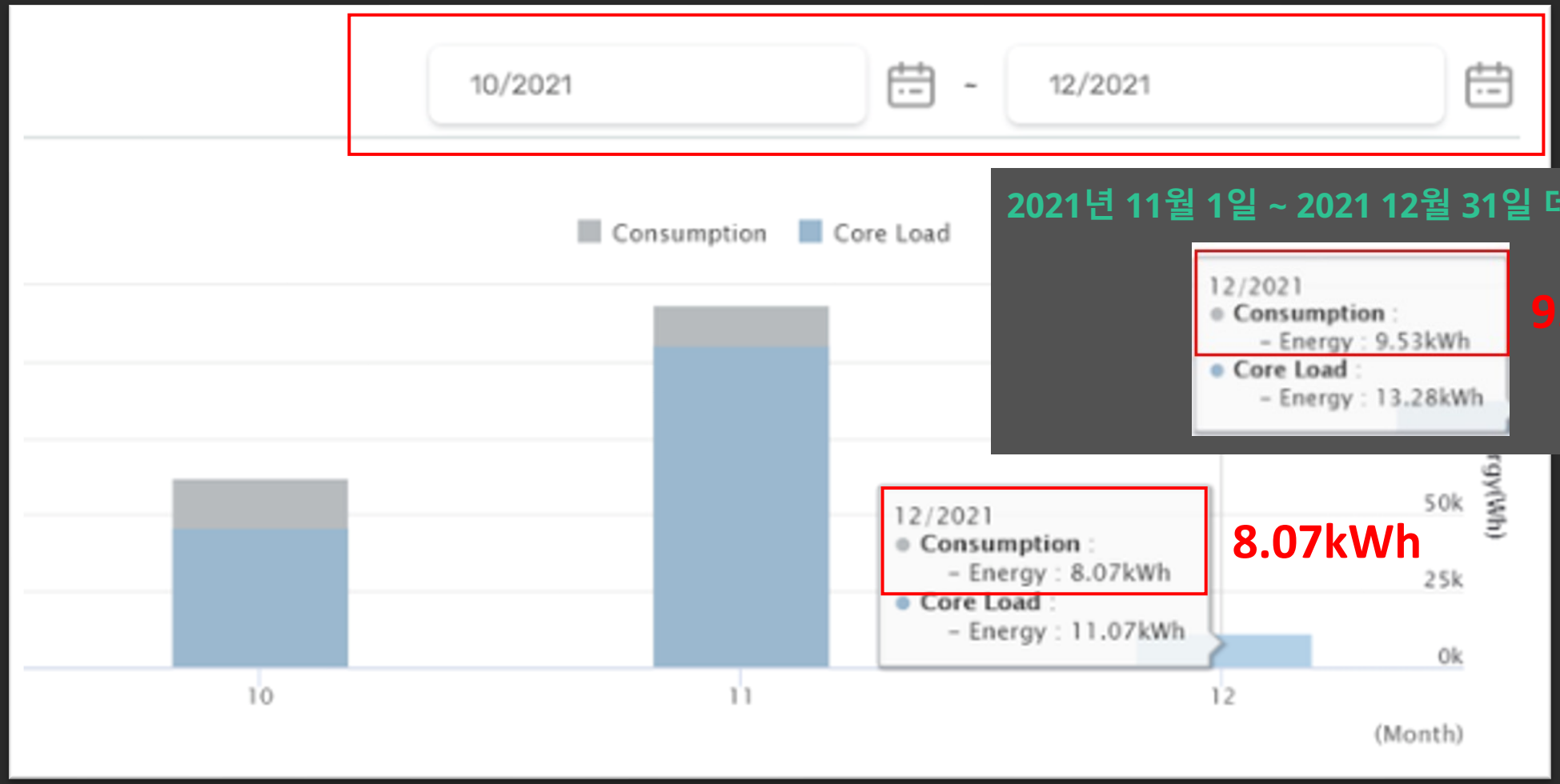
Global Service



2021년 11월 1일~ 2021 12월 31일 데이터 조회 (Europe/Paris)



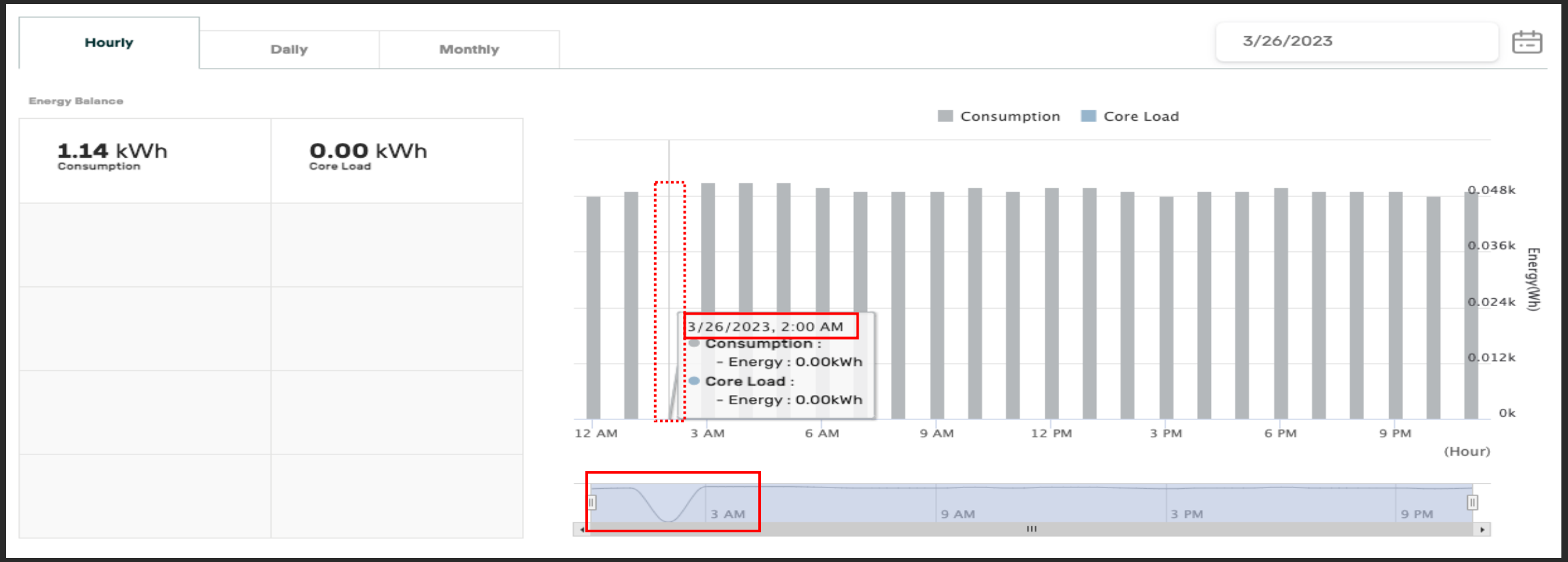
2021년 10월 1일 ~ 2021 12월 31일 데이터 조회 (Europe/Paris)



2021년 11월 1일 ~ 2021 12월 31일 데이터 조회

9.53kWh

2023년 3월 26일 데이터 조회 (Europe/Paris)



2시 데이터가 존재 하지 않음

2. 날짜 및 시간 관련 기본 지식 알아보기

1. GMT, UTC
2. Timezone
3. Offset
4. DST / SummerTime

Timezone Map

Timezone은 경도 0도에 있는 그리니치 천문대를 기준으로 한 시각의 차이를 말하고, 시간대라고 표현하기도 합니다.

Ref. https://en.wikipedia.org/wiki/Time_zone

GMT

Royal Observatory, Greenwich. Photo: National Maritime Museum, Greenwich.

그리니치 천문대

국제 표준시인 그리니치 평균시(Greenwich Mean Time, GMT)의 기준점으로 사용

UTC (협정 세계시)

세계 곳곳에서 동일한 기준 시간을 사용하기 위한 표준 시간대

2023-04-05T00:00:00Z (ISO 8601 형식)

↓

Zulu Time = UTC

Timezone

Region (time zone)	Converted timestamp 1680084900	Relative to UTC/GMT	Date in DST	Offset In seconds
Asia/Seoul (KST)	Mar 29 2023 19:15:00	GMT +09:00		+32400
Asia/Shanghai (CST)	Mar 29 2023 18:15:00	GMT +08:00		+28800
Asia/Singapore (+08)	Mar 29 2023 18:15:00	GMT +08:00		+28800
Asia/Srednekolymsk (+11)	Mar 29 2023 21:15:00	GMT +11:00		+39600
Asia/Taipei (CST)	Mar 29 2023 18:15:00	GMT +08:00		+28800
Australia				
Australia/Adelaide (ACDT)	Mar 29 2023 20:45:00	GMT +10:30	DST	+37800
Australia/Brisbane (AEST)	Mar 29 2023 20:15:00	GMT +10:00		+36000
Australia/Broken Hill (ACDT)	Mar 29 2023 20:45:00	GMT +10:30	DST	+37800
Australia/Darwin (ACST)	Mar 29 2023 19:45:00	GMT +09:30		+34200
Australia/Eucla (+0845)	Mar 29 2023 19:00:00	GMT +08:45		+31500
Australia/Hobart (AEDT)	Mar 29 2023 21:15:00	GMT +11:00	DST	+39600
Australia/Lindeman (AEST)	Mar 29 2023 20:15:00	GMT +10:00		+36000
Australia/Lord Howe (+11)	Mar 29 2023 21:15:00	GMT +11:00	DST	+39600
Australia/Melbourne (AEDT)	Mar 29 2023 21:15:00	GMT +11:00	DST	+39600
Australia/Perth (AWST)	Mar 29 2023 18:15:00	GMT +08:00		+28800
Australia/Sydney (AEDT)	Mar 29 2023 21:15:00	GMT +11:00	DST	+39600

Australia

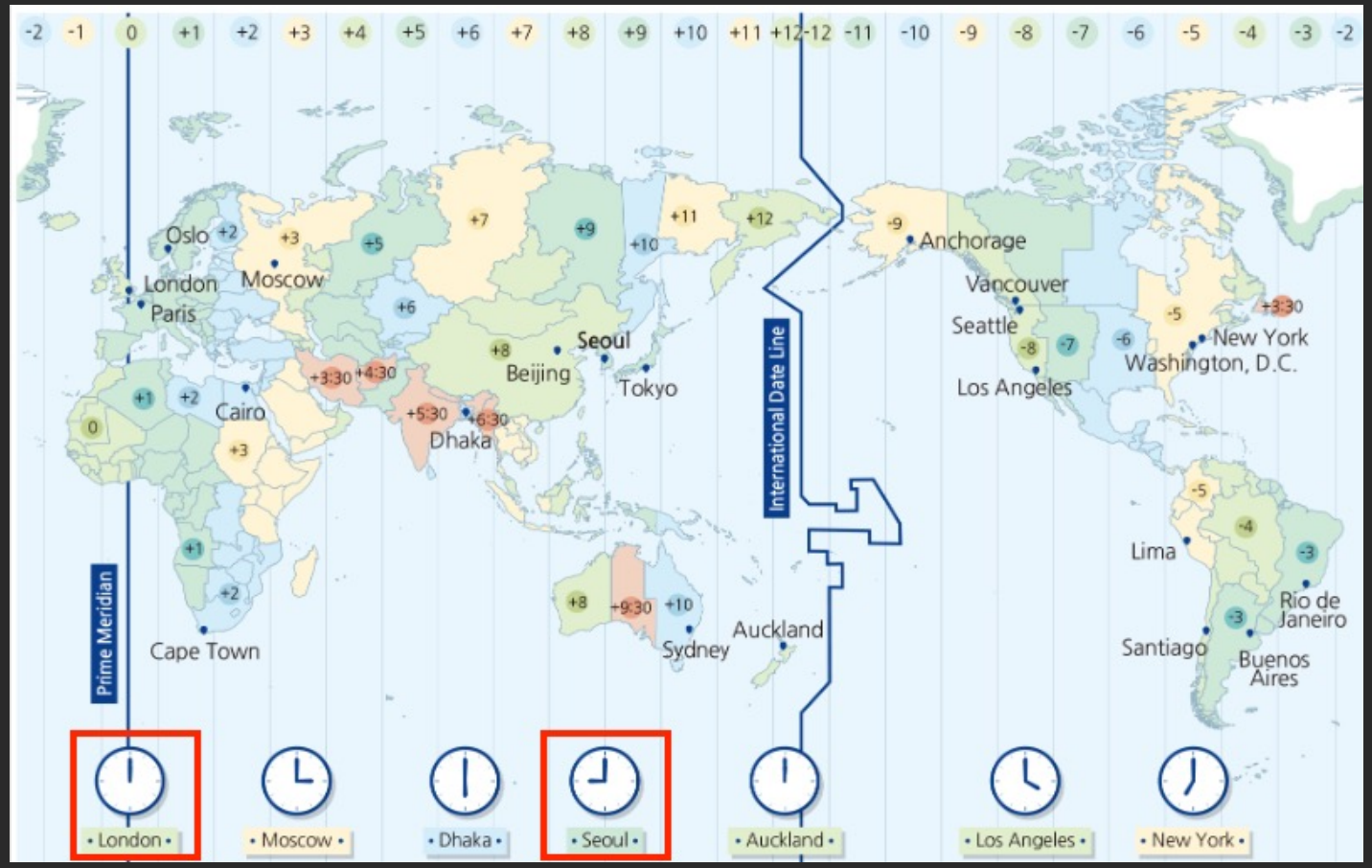
Ref. https://www.epochconverter.com/timezones

Offset

Region (time zone)	Converted timestamp 1680084900	Relative to UTC/GMT	Date in DST	Offset In seconds
Asia/Seoul (KST)	Mar 29 2023 19:15:00	GMT +09:00		+32400
Asia/Shanghai (CST)	Mar 29 2023 18:15:00	GMT +08:00		+28800
Asia/Singapore (+08)	Mar 29 2023 18:15:00	GMT +08:00		+28800
Asia/Srednekolymsk (+11)	Mar 29 2023 21:15:00	GMT +11:00		+39600
Asia/Taipei (CST)	Mar 29 2023 18:15:00	GMT +08:00		+28800
Australia				
Australia/Adelaide (ACDT)	Mar 29 2023 20:45:00	GMT +10:30	DST	+37800
Australia/Brisbane (AEST)	Mar 29 2023 20:15:00	GMT +10:00		+36000
Australia/Broken Hill (ACDT)	Mar 29 2023 20:45:00	GMT +10:30	DST	+37800
Australia/Darwin (ACST)	Mar 29 2023 19:45:00	GMT +09:30		+34200
Australia/Eucla (+0845)	Mar 29 2023 19:00:00	GMT +08:45		+31500
Australia/Hobart (AEDT)	Mar 29 2023 21:15:00	GMT +11:00	DST	+39600
Australia/Lindeman (AEST)	Mar 29 2023 20:15:00	GMT +10:00		+36000
Australia/Lord Howe (+11)	Mar 29 2023 21:15:00	GMT +11:00	DST	+39600
Australia/Melbourne (AEDT)	Mar 29 2023 21:15:00	GMT +11:00	DST	+39600
Australia/Perth (AWST)	Mar 29 2023 18:15:00	GMT +08:00		+28800
Australia/Sydney (AEDT)	Mar 29 2023 21:15:00	GMT +11:00	DST	+39600

Ref. https://www.epochconverter.com/timezones

Offset



London : 2023-04-05T00:00:00+0000

Seoul : 2023-04-05T09:00:00+0900

Ref. http://nationalatlas.ngii.go.kr/pages/page_1933.php

Offset

London : 2023-04-05T00:00:00+0000

Seoul : 2023-04-05T09:00:00+0900

↓ UTC 시간대 로 변환

London : 2023-04-05T00:00:00Z

Seoul : 2023-04-05T00:00:00Z

DST (Daily Saving Time) / Summer Time

Region (time zone)	Converted timestamp	Relative to UTC/GMT	Date in DST	Offset In seconds
Europe/Paris (CET)	Mar 26 2023 01:00:00	GMT +01:00		+3600
Europe/Paris (CEST)	Mar 27 2023 02:00:00	GMT +02:00	DST	+7200

Central European Time (CET)

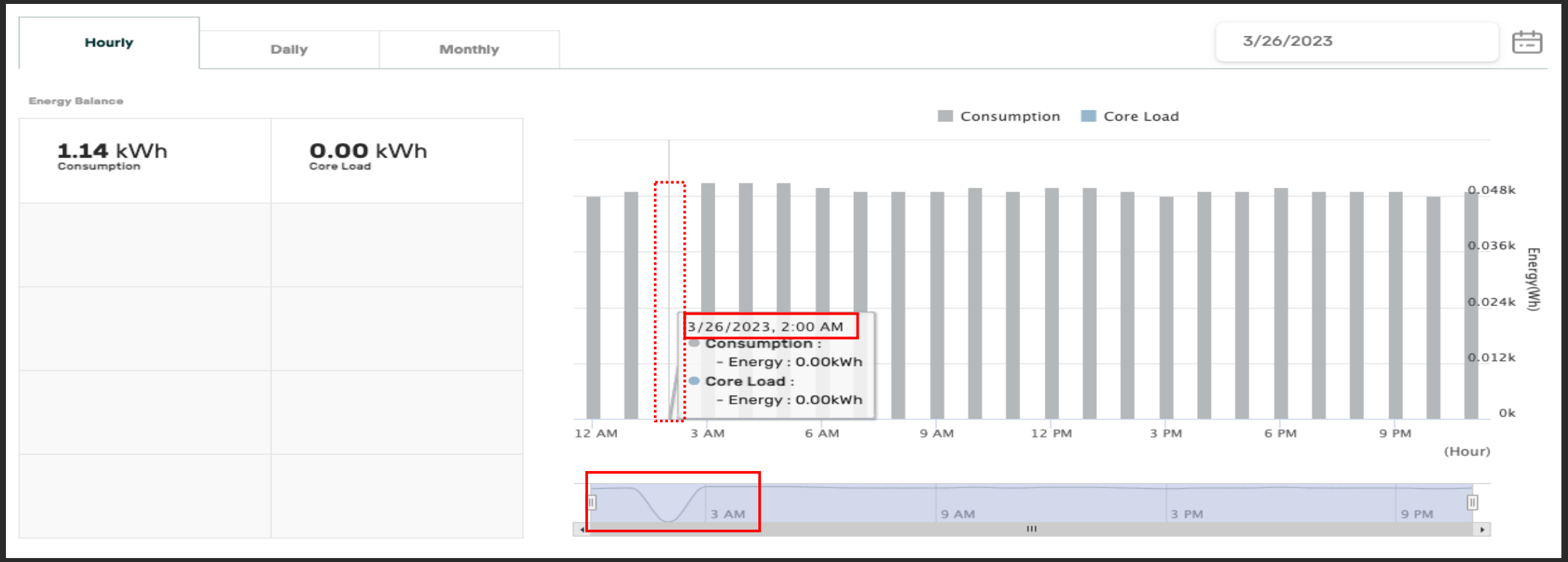
Central European Summer Time (CEST)

Ref. https://www.epochconverter.com/timezones

2023년 3월 26일 Summer Time 시작

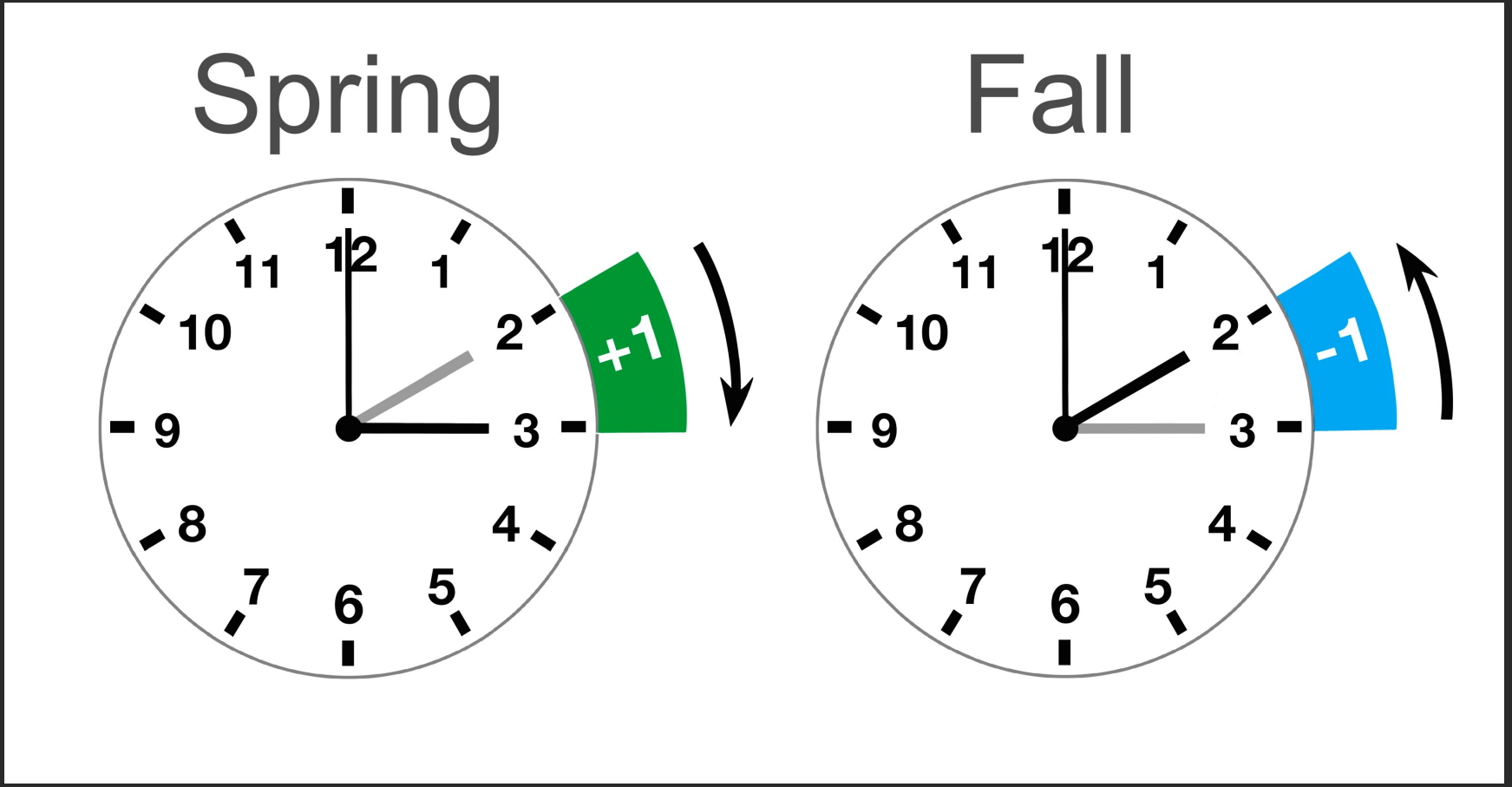
Ref. https://www.timeanddate.com/time/change/france/paris?year=2023

2023년 3월 26일 데이터 조회 (Europe/Paris)



사실 파리에서 2023년 3월 26일 2시는 존재하지 않음

DST (Daily Saving Time) / Summer Time



Ref. https://24timezones.com/daylight-saving-time

3. Java 시간, 날짜 클래스 변천사

1. Date (JAVA1.0)
2. Calendar (JAVA1.1)
3. Joda-Time
4. Java.Time (JAVA1.8)

java.util.Date (Java 1.0부터 도입)

- 간단한 날짜와 시간 조작이 가능하다
- mutable 객체로 인해 스레드 안전성이 없다
- 시간대와 로케일을 처리하는데 한계가 있다 (다른 시간대 및 로케일을 사용하려면 추가 작업이 필요)

java.util.Calendar (Java 1.1부터 도입)

- 시간대와 로케일을 지원한다.
- 다양한 날짜와 시간 조작 기능을 제공한다.
- mutable 객체로 인해 스레드 안전성이 없다.
- API 사용이 복잡하고 직관적이지 않다.
- 클래스 : Calendar, GregorianCalendar 등

Joda-Time 라이브러리

- 불변 객체로 스레드 안전성을 보장한다
- 직관적이고 사용하기 쉬운 API를 제공한다
- 시간대와 로케일을 쉽게 처리할 수 있다
- Java의 공식 표준 라이브러리는 아니다
- Java 8 이후의 새로운 날짜-시간 API가 도입되면서 더 이상 적극적으로 개발되지 않음
- 클래스 : DateTime, LocalDate, LocalTime, Duration, Period 등

java.time 패키지 (Java 8부터 도입, JSR-310)

- 불변 객체로 스레드 안전성을 보장한다.
- 직관적이고 사용하기 쉬운 API를 제공한다
- 클래스 : Instant , LocalDateTime, OffsetDateTime, ZonedDateTime 등

<table>
<tr><td colspan="4">ZonedDateTime</td></tr>
<tr><td colspan="3">OffsetDateTime</td><td></td></tr>
<tr><td colspan="2">LocalDateTime</td><td></td><td></td></tr>
<tr><td>LocalDate</td><td>LocalTime</td><td>ZoneOffset</td><td>ZoneId</td></tr>
<tr><td>2023-03-01</td><td>12:00:00</td><td>+01:00</td><td>Europe/Paris</td></tr>
</table>

ZonedDateTime / OffsetDateTime

```
 * @param localDateTime  the local date-time, not null
 * @param zone  the time-zone, not null
 * @return the zoned date-time, not null
 */
public static ZonedDateTime of(LocalDateTime localDateTime, ZoneId zone) {
    return ofLocal(localDateTime, zone, preferredOffset: null);
}
```

Params: dateTime – the local date-time, not null
offset – the zone offset, not null

Returns: the offset date-time, not null

```
public static OffsetDateTime of(LocalDateTime dateTime, ZoneOffset offset) {
    return new OffsetDateTime(dateTime, offset);
}
```

ZonedDateTime / OffsetDateTime

ZonedDateTime		
LocalDateTime		ZoneId
LocalDate	LocalTime	
2023-03-01	12:00:00	Europe/Paris

OffsetDateTime		
LocalDateTime		ZoneOffset
LocalDate	LocalTime	
2023-03-01	12:00:00	+01:00

4. Timezone/DST 적용해보기

1. Sample Data
2. Timezone 적용하기
3. DST 적용하기

Sample Data (24시간 편의점 매출액)

```
CREATE TABLE convenience_store_sales // 15분 단위 집계 테이블
  (
     id             SERIAL PRIMARY KEY,
     aggregate_date TIMESTAMP WITH time zone, // 15분 단위 집계 날짜
     sales          INTEGER // 매출액
  );
```

id	aggregate_date	sales
1	2021-12-31 23:00:00+00	30000
2	2021-12-31 23:15:00+00	12000
3	2021-12-31 23:30:00+00	19000
4	2021-12-31 23:45:00+00	32000
5	2022-01-01 00:00:00+00	17000
6	2022-01-01 00:15:00+00	10000
7	2022-01-01 00:30:00+00	17000

Timezone 적용하기 (Europe/Paris)

```java
SimpleDateFormat dateFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd'T'HH:mm:ss");
Date startTimeWithTimezone;
Date endTimeWithTimezone;
Date startTime;
Date endTime;

int offset = TimeZone.getTimeZone(timeZoneId).getRawOffset();

try {
    startTime = dateFormat.parse(startTimeStr);
    endTime = dateFormat.parse(endTimeStr);

    startTimeWithTimezone = dateFormat.parse(startTimeStr);
    endTimeWithTimezone = dateFormat.parse(endTimeStr);

    startTimeWithTimezone.setTime(startTime.getTime() - offset);
    endTimeWithTimezone.setTime(endTime.getTime() - offset);
} catch (ParseException e) {
    throw new IllegalArgumentException();
}
```

offset : 36000000 (1시간)

00시부터 01시 이전까지 집계한 매출액이 0

2022년 10월 30일

aggregated_date	total_sales
2022-10-30 00:00	0.0
2022-10-30 01:00	75000.0
2022-10-30 02:00	103000.0
2022-10-30 03:00	100000.0

2022년 10월 31일

aggregated_date	total_sales
2022-10-31 00:00	96000.0
2022-10-31 01:00	101000.0
2022-10-31 02:00	91000.0
2022-10-31 03:00	76000.0

ZoneRulesProvider

Evaluate

Expression:

```
ZoneRulesProvider.getRules("Europe/Paris", false)
```

Result:

- result = {ZoneRules@8816} "ZoneRules[currentStandardOffset=+01:00]"
  - standardTransitions = {long[4]@8817} [-1855958961, -932436000, -8
  - standardOffsets = {ZoneOffset[5]@8818}
  - savingsInstantTransitions = {long[102]@8819} [-1855958961, -168981 … 48342800, -1635123600, -1616893200, -1604
  - savingsLocalTransitions = {LocalDateTime[204]@8820}
  - wallOffsets = {ZoneOffset[103]@8821}
  - lastRules = {ZoneOffsetTransitionRule[2]@8822}
    - 0 = {ZoneOffsetTransitionRule@8834} "TransitionRule[Gap +01:00 to +02:00, SUNDAY on or after MARCH 25 at 01:00 UTC, standard offset +01:00]"
    - 1 = {ZoneOffsetTransitionRule@8835} "TransitionRule[Overlap +02:00 to +01:00, SUNDAY on or after OCTOBER 25 at 01:00 UTC, standard offset +01:00]"
  - lastRulesCache = {ConcurrentHashMap@8823} size = 0

October

Su	Mo	Tu	We	Th	Fr	Sa
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

TransitionRule 해석하기

```
"TransitionRule[Overlap +02:00 to +01:00, SUNDAY on or after OCTOBER 25 at 01:00 UTC, standard offset +01:00]"
```

UTC 기준 10월 25일 1시 이후에 오는 일요일에 offset의 값이 +02시간에서 +01시간으로 변경

Timezone 적용하기 (Europe/Paris)

2022년 10월 30일

Europe/Paris 기준
2022년 10월 30일 00시 ~ 2022년 10월 31일 00시

-01:00

UTC 기준
2022년 10월 29일 23시 ~ 2022년 10월 30일 23시

+02:00 (DST Offset)

Europe/Paris 기준
2022년 10월 30일 01시 ~ 2022년 10월 31일 00시

```
|-----------------|------------|
|aggregated_date  |total_sales |
|-----------------|------------|
|2022-10-30 00:00 |0.0         |
|2022-10-30 01:00 |18000.0     |
|2022-10-30 02:00 |29000.0     |
|2022-10-30 03:00 |32000.0     |
```

Timezone 적용하기 (Europe/Paris)

2022년 10월 31일

Europe/Paris 기준
2022년 10월 31일 00시 ~ 2022년 11월 01일 00시

-01:00

UTC 기준
2022년 10월 30일 23시 ~ 2022년 10월 31일 23시

+01:00 (Standard Offset)

Europe/Paris 기준
2022년 10월 30일 00시 ~ 2022년 10월 31일 00시

aggregated_date	total_sales
2022-10-31 00:00	27000.0
2022-10-31 01:00	31000.0
2022-10-31 02:00	8000.0
2022-10-31 03:00	5000.0

DST 적용하기 1 (Europe/Paris)

Timezone useDaylightTime(), inDaylightTime() 활용하기

```
int startOffset = timeZone.getRawOffset();
```

startOffset : 36000000 (1시간)

```
if (timeZone.useDaylightTime() && timeZone.inDaylightTime(startTime)) {
    startOffset += timeZone.getDSTSavings();
}
```

startOffset : 72000000 (2시간)

```
startTimeWithTimezone.setTime(startTime.getTime() - startOffset);
```

2022년 10월 30일

aggregated_date	total_sales
2022-10-30 00:00	1147000.0
2022-10-30 01:00	75000.0
2022-10-30 02:00	103000.0
2022-10-30 03:00	100000.0

DST 적용하기 2 (Europe/Paris)

```
    Params:   date – the date represented in milliseconds
              since January 1, 1970 00:00:00 GMT
    Returns:  the amount of time in milliseconds to add
              to UTC to get local time.
    Since:    1.4
    See Also: Calendar.ZONE_OFFSET,
              Calendar.DST_OFFSET
public int getOffset(long date) {
    if (inDaylightTime(new Date(date))) {
        return getRawOffset() + getDSTSavings();
    }
    return getRawOffset();
}
```

DST 적용하기 2 (Europe/Paris)

Timezone getOffset() 활용하기

```
try {
    TimeZone timeZone = TimeZone.getTimeZone(timeZoneId);

    startTime = dateFormat.parse(startTimeStr);
    endTime = dateFormat.parse(endTimeStr);

    startTimeWithTimezone = dateFormat.parse(startTimeStr);
    endTimeWithTimezone = dateFormat.parse(endTimeStr);

    startTimeWithTimezone.setTime(startTime.getTime() - timeZone.getOffset(startTime.getTime()));
    endTimeWithTimezone.setTime(endTime.getTime() - timeZone.getOffset(endTime.getTime()));

} catch (ParseException e) {
    throw new IllegalArgumentException();
}
```

2022년 10월 30일

aggregated_date	total_sales
2022-10-30 00:00	1147000.0
2022-10-30 01:00	75000.0
2022-10-30 02:00	103000.0
2022-10-30 03:00	100000.0

DST 적용하기 3 (Europe/Paris)

ZonedDateTime.of() 활용하기

2022년 10월 30일

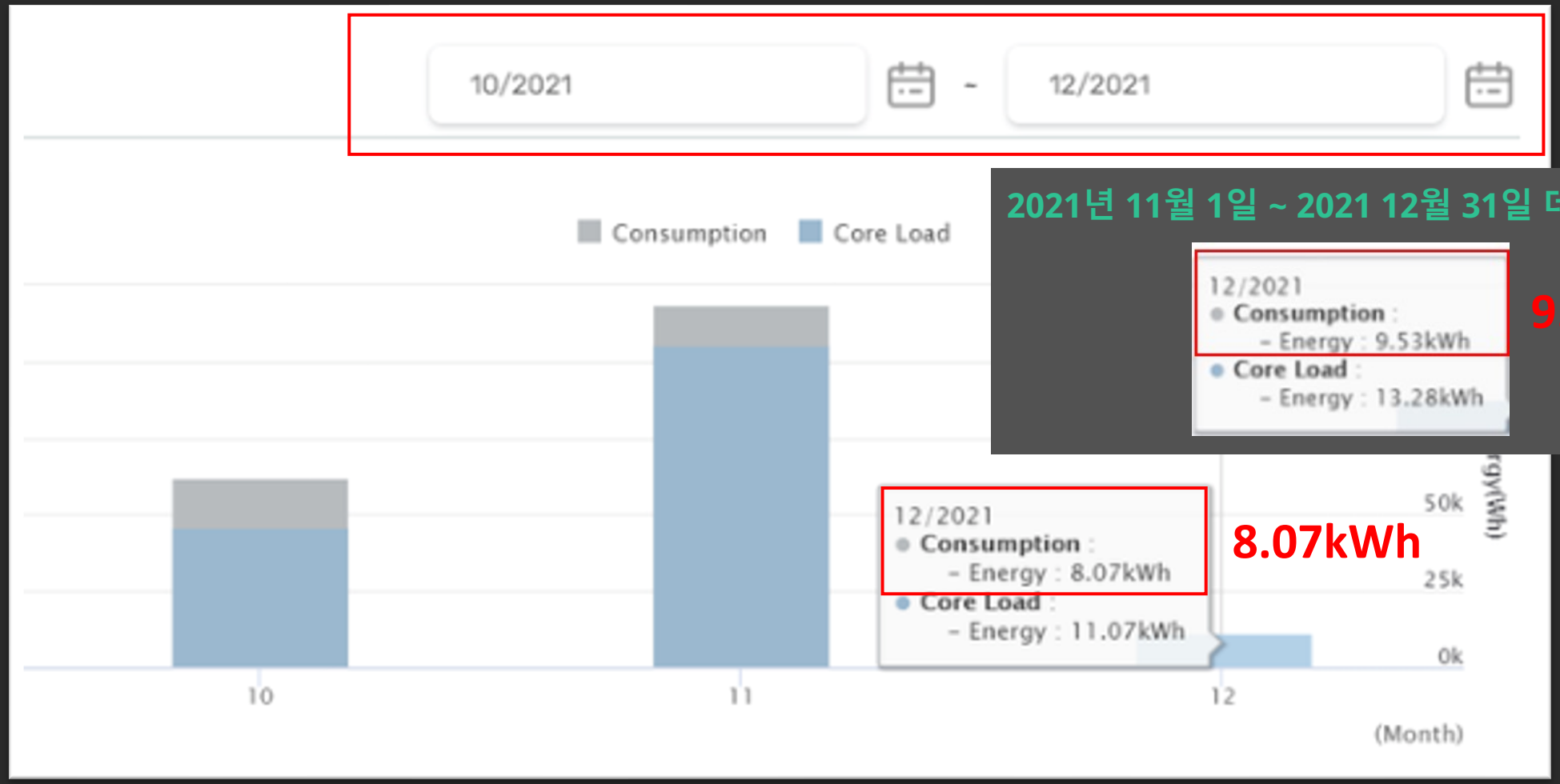
```
public List<SalesAggregate> getSalesAggregateV2(ZonedDateTime startTime, ZonedDateTime endTime, String timeZoneId) {

    return salesAggregateDao.getSalesAggregateV2(
            startTime,
            endTime,
            ZonedDateTime.of(startTime.toLocalDateTime(), ZoneId.of(timeZoneId)),
            ZonedDateTime.of(endTime.toLocalDateTime(), ZoneId.of(timeZoneId)),
            timeZoneId);
}
```

```
|------------------|------------|
|aggregated_date   |total_sales |
|------------------|------------|
|2022-10-30 00:00 |1147000.0    |
|2022-10-30 01:00 |75000.0      |
|2022-10-30 02:00 |103000.0     |
|2022-10-30 03:00 |100000.0     |
```

5. DST 관련 이슈 되짚어보기

1. 파리에서 생긴 일
2. 이슈 분석하기

2021년 10월 1일 ~ 2021 12월 31일 데이터 조회 (Europe/Paris)



2021년 11월 1일 ~ 2021 12월 31일 데이터 조회

9.53kWh

[기존 쿼리]

```
timescaledb_experimental.time_bucket_ng('24 hours'::interval, timezone('Europe/Paris', create_dt))
```

2021년 10월 1일 ~ 2021 12월 31일 데이터 조회 (Europe/Paris)

11월 1일	23	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22
11월 2일	23	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22
11월 3일	23	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22

00시부터 23시까지 aggregate되어야하는데, 데이터가 한시간씩 밀림

2021년 11월 1일 ~ 2021 12월 31일 데이터 조회 (Europe/Paris)

11월 1일	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
11월 2일	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
11월 3일	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23

[기존 쿼리]

```
timescaledb_experimental.time_bucket_ng('24 hours'::interval, timezone('Europe/Paris', create_dt))
```

2021년 10월 1일 ~ 2021 12월 31일 데이터 조회 (Europe/Paris)

10월 31일	00	01	02	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22
11월 1일	23	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22
11월 2일	23	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22
11월 3일	23	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22

2021년 11월 1일 ~ 2021 12월 31일 데이터 조회 (Europe/Paris)

11월 1일	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
11월 2일	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
11월 3일	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23

2021년 10월 1일 ~ 2021 12월 31일 데이터 조회 (Europe/Paris)

[기존 쿼리]

```
timescaledb_experimental.time_bucket_ng('24 hours'::interval, timezone('Europe/Paris', create_dt))
```

24 hour chunk

10월 31일	00	01	02	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22
11월 1일	23	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22
11월 2일	23	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22
11월 3일	23	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22

- 2021년 10월 31일에 Summer Time 종료
- 31일에는 **02**시가 두 번 존재 -> **하루가 총 25시간**

2021년 10월 1일 ~ 2021 12월 31일 데이터 조회 (Europe/Paris)

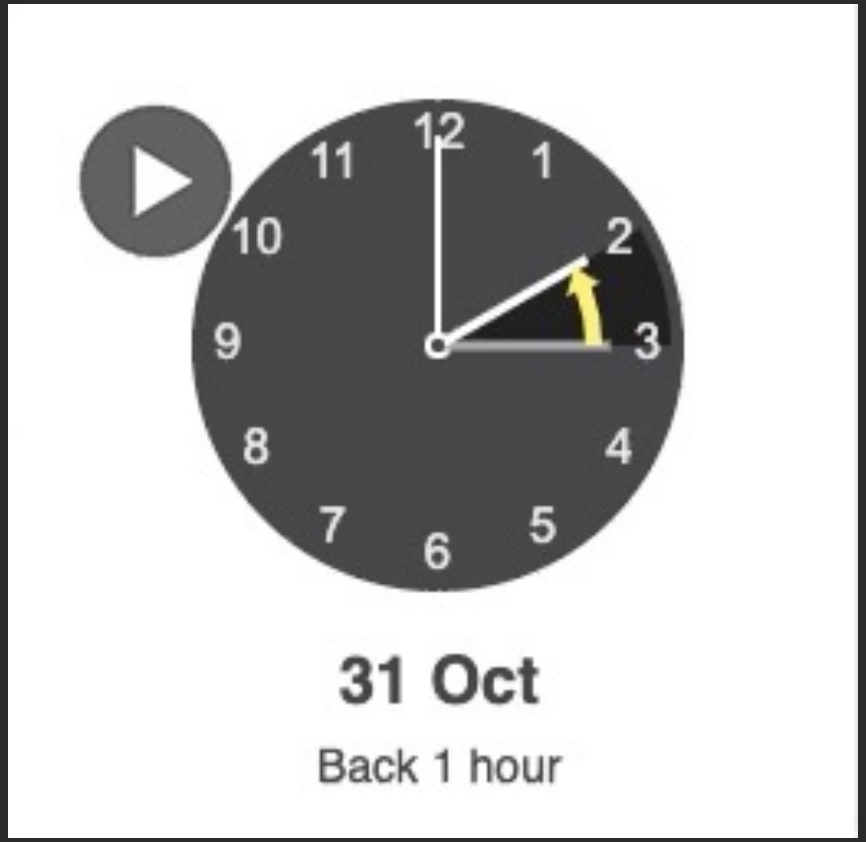
[기존 쿼리]

```
timescaledb_experimental.time_bucket_ng('1 day'::interval, timezone('Europe/Paris', create_dt))
```

1 day chunk

10월 31일	00	01	02	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
11월 1일	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
11월 2일	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	
11월 3일	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	

2021년 10월 31일 Summer Time 종료



Ref. https://www.timeanddate.com/time/change/france/paris?year=2021

Quiz

한국시간이 2023년 4월 22일 13시 00분일 때, 프랑스 파리 시간은 몇시일까요?

(프랑스 파리의 Summer Time 시작일은 3월 26일)

2023년 4월 22일 06시 00분

Good Bye, Spring Camp.

Welcome, Summer Time !