

# 5조

배진경, 손원장, 오민석, 최우린

# 팀 정보



배진경



손원장




오민석



최우린

# 목차

1. 프로젝트 목표
2. 적용 기술
3. 작업 과정
4. 구현 내역

A laptop screen is shown with a data dashboard. The dashboard features a line graph at the top with a blue line and a green line, and a pie chart below it. The text '프로젝트 목표:' is overlaid on the screen. The background is dark and blurry.

프로젝트 목표:  
최근 1년 영화 데이터  
수집 및 분석

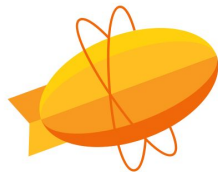
# 적용기술

./jq

**.Jq**



# Hive



# ZEPLIN

# Zeplin



## 영화관입장권통합전산망 오픈API



JAR(json-serde-1.1.4.jar)

- API
- JAR
- Shell Script

A close-up photograph of a person's hands working on a circuit board. The person is using a soldering iron to solder components. The background is blurred, showing some electronic components and a workbench.

# 작업 과정

1. 데이터 수집
2. 데이터 처리
3. 데이터 분석

# 데이터 수집

영화관 입장권 통합 전산망 오픈 API

raw data(json 형식):

- 일별 박스오피스
- 영화 상세정보



The screenshot shows the KOFIC Open API website. The header includes the KOFIC logo and the text '영화관입장권통합전산망 오픈API'. Navigation tabs include '이용안내', 'OPEN API', '키발급/관리', and '이용약관'. Below these are links for '제공 서비스', '다운로드', and '튜토리얼'. The main content area is titled '제공 서비스' and includes a description: '영화관입장권통합전산망이 제공하는 오픈API 서비스 모음입니다. 사용 가능한 서비스를 확인하고 서비스별 인터페이스 정보를 조회합니다.' Below this is a table listing the services.

번호	서비스명	서비스 설명
1	박스오피스	• 일별 박스오피스 • 주간/주말 박스오피스
2	공통코드조회	• 공통코드 조회
3	영화정보	• 영화목록 • 영화 상세정보
4	영화사정보	• 영화사목록 • 영화사 상세정보
5	영화인정보	• 영화인목록 • 영화인 상세정보

# 데이터 수집

## jq 설치 및 설정

- Json 형식으로 수집된 데이터의 패턴처리를 쉽게 다루기 위해 'JQ' 사용 (<https://stedolan.github.io/jq/>)

- JQ v1.4 설치

```
[root@dn01 ~]# cd /usr/local/bin
```

```
[root@dn01 bin]# wget http://stedolan.github.io/jq/download/linux64/jq
```

- 권한설정

```
[root@dn01 bin]# chmod a+x jq
```

- 테스트 실행

```
[root@dn01 bin]# jq
```



```
jq - commandline JSON processor [version 1.4]
Usage: jq [options] <jq filter> [file...]

For a description of the command line options and
how to write jq filters (and why you might want to)
see the jq manpage, or the online documentation at
http://stedolan.github.com/jq
```



# 데이터 수집

영화정보 *downloader shellscript*

- 원하는 날짜의 데이터 수집을 위해 간단한 **bash** 파일 작성으로 편의성을 향상 시켰다

bash 파일: [https://github.com/filoscoder/movie\\_data\\_analisys/blob/master/dataset-downloader-1.1.sh](https://github.com/filoscoder/movie_data_analisys/blob/master/dataset-downloader-1.1.sh)

- Windows에서 작성한 **shell script**는 **enter** 입력이 포함되어 있어, '**sed**' (stream editor)을 이용하여 줄 앞으로 이동하는 **carriage return** 기호('\r')를 모든 라인에 치환 ('/g')

```
[hadoop@dn01 ~]$ sed -i 's/\r$//g' /home/hadoop/movies_data/dataset_dlr.sh
```

- **bash** 파일을 모든 계정에서 실행 가능하도록 퍼미션을 수정

```
[hadoop@dn01 ~]$ chmod 777 /home/hadoop/movies_data/dataset_dlr.sh
```

- 실행

```
[hadoop@dn01 ~]$ /home/hadoop/movies_data/dataset_dlr.sh
```

---

# 데이터 수집

영화정보 downloader shellscript 2

- 조회 하고 싶은 1) 날짜 / 2)기간 / 3)취소 #? 옵션 선택

```
*****
*****
***** +-----+ *****
***** | MOVIE BOX OFFICE DATA CRAWLER | *****
***** +-----+ *****
*****
*****
*****

Please select your looking DATE format=

1) [SPECIFIC DATE] (YYYYMMDD)          3) [Cancel]
2) [DATE RANGE] (YYYYMMDD - YYYYMMDD)
#? █
```

# 데이터 수집

영화정보 *downloader shellscript 3*

- 조회 날짜를 형식에 맞게 입력 -> 'YYYYMMDD'

```
You selected : [SPECIFIC DATE] (YYYYMMDD)
```

```
Enter your specific date (YYYYMMDD):
```

```
You selected : [DATE RANGE] (YYYYMMDD - YYYYMMDD)
```

```
Enter the starting date (YYYYMMDD):
```


```
Enter the ending date (YYYYMMDD):
```

- 2018/08/01~2019/07/31 기간의 영화 데이터  
일별 json 파일 다운로드 자동화
- 각 파일의 영화코드 ('movieCd') 키값을 조회  
하여 각 영화의 상세 정보 자동 다운로드


# 데이터 수집

## 영화정보 downloader shellscript 4

### -일별 박스오피스 raw data

/home/hadoop/movies_data/datelist/				
이름	크기	수정한 날짜	권한	소유자
		2019-08-22 오후 4:59:59	rw-rw-r--x	hadoop
<input type="checkbox"/> 20190731	4 KB	2019-08-22 오전 1:39:28	rw-rw-r--	hadoop
<input type="checkbox"/> 20190730	4 KB	2019-08-22 오전 1:39:27	rw-rw-r--	hadoop
<input type="checkbox"/> 20190729	4 KB	2019-08-22 오전 1:39:26	rw-rw-r--	hadoop
<input type="checkbox"/> 20190728	4 KB	2019-08-22 오전 1:39:25	rw-rw-r--	hadoop
<input type="checkbox"/> 20190727	4 KB	2019-08-22 오전 1:39:25	rw-rw-r--	hadoop
<input type="checkbox"/> 20190726	4 KB	2019-08-22 오전 1:39:24	rw-rw-r--	hadoop
<input type="checkbox"/> 20190725	4 KB	2019-08-22 오전 1:39:23	rw-rw-r--	hadoop
<input type="checkbox"/> 20190724	4 KB	2019-08-22 오전 1:39:21	rw-rw-r--	hadoop
<input type="checkbox"/> 20190723	4 KB	2019-08-22 오전 1:39:20	rw-rw-r--	hadoop
<input type="checkbox"/> 20190722	4 KB	2019-08-22 오전 1:39:19	rw-rw-r--	hadoop
<input type="checkbox"/> 20190721	4 KB	2019-08-22 오전 1:39:18	rw-rw-r--	hadoop
<input type="checkbox"/> 20190720	4 KB	2019-08-22 오전 1:39:17	rw-rw-r--	hadoop
<input type="checkbox"/> 20190719	4 KB	2019-08-22 오전 1:39:16	rw-rw-r--	hadoop
<input type="checkbox"/> 20190718	4 KB	2019-08-22 오전 1:39:15	rw-rw-r--	hadoop
<input type="checkbox"/> 20190717	4 KB	2019-08-22 오전 1:39:13	rw-rw-r--	hadoop
<input type="checkbox"/> 20190716	4 KB	2019-08-22 오전 1:39:12	rw-rw-r--	hadoop
<input type="checkbox"/> 20190715	4 KB	2019-08-22 오전 1:39:11	rw-rw-r--	hadoop
<input type="checkbox"/> 20190714	4 KB	2019-08-22 오전 1:39:11	rw-rw-r--	hadoop
<input type="checkbox"/> 20190713	4 KB	2019-08-22 오전 1:39:10	rw-rw-r--	hadoop
<input type="checkbox"/> 20190712	4 KB	2019-08-22 오전 1:39:09	rw-rw-r--	hadoop

### -영화별 상세정보 raw data

/home/hadoop/movies_data/movie_detail/				
이름	크기	수정한 날짜	권한	소유자
		2019-08-24 오후 2:48:41	rw-rw-r--x	hadoop
<input type="checkbox"/> 20199954	2 KB	2019-08-22 오전 1:38:17	rw-rw-r--	hadoop
<input type="checkbox"/> 20199951	2 KB	2019-08-22 오전 1:38:53	rw-rw-r--	hadoop
<input type="checkbox"/> 20199949	2 KB	2019-08-22 오전 1:37:59	rw-rw-r--	hadoop
<input type="checkbox"/> 20199923	2 KB	2019-08-22 오전 1:37:59	rw-rw-r--	hadoop
<input type="checkbox"/> 20199822	2 KB	2019-08-22 오전 1:38:15	rw-rw-r--	hadoop
<input type="checkbox"/> 20199814	1 KB	2019-08-22 오전 1:37:20	rw-rw-r--	hadoop
<input type="checkbox"/> 20199736	2 KB	2019-08-22 오전 1:37:50	rw-rw-r--	hadoop
<input type="checkbox"/> 20199623	2 KB	2019-08-22 오전 1:37:58	rw-rw-r--	hadoop
<input type="checkbox"/> 20199448	3 KB	2019-08-22 오전 1:37:47	rw-rw-r--	hadoop
<input type="checkbox"/> 20199189	2 KB	2019-08-22 오전 1:37:42	rw-rw-r--	hadoop
<input type="checkbox"/> 20199063	2 KB	2019-08-22 오전 1:38:31	rw-rw-r--	hadoop
<input type="checkbox"/> 20198982	2 KB	2019-08-22 오전 1:37:36	rw-rw-r--	hadoop
<input type="checkbox"/> 20198912	2 KB	2019-08-22 오전 1:37:42	rw-rw-r--	hadoop
<input type="checkbox"/> 20198844	2 KB	2019-08-22 오전 1:37:28	rw-rw-r--	hadoop
<input type="checkbox"/> 20198603	2 KB	2019-08-22 오전 1:36:51	rw-rw-r--	hadoop
<input type="checkbox"/> 20198598	2 KB	2019-08-22 오전 1:37:52	rw-rw-r--	hadoop
<input type="checkbox"/> 20198597	2 KB	2019-08-22 오전 1:38:05	rw-rw-r--	hadoop
<input type="checkbox"/> 20198526	1 KB	2019-08-22 오전 1:38:48	rw-rw-r--	hadoop
<input type="checkbox"/> 20198453	3 KB	2019-08-22 오전 1:38:23	rw-rw-r--	hadoop
<input type="checkbox"/> 20198403	2 KB	2019-08-22 오전 1:37:13	rw-rw-r--	hadoop

# 데이터 처리

수집 data - 일별 박스오피스 data를 hive를 이용해 table로 저장

```
-- 일별 박스오피스 테이블 생성 (json구조)
create external table if not exists movies_js(
  boxOfficeResult struct<
    boxofficeType : String,
    showRange:String,
    dailyBoxOfficeList: Array <
      struct< rnum :String,
        rank : String,
        rankInten : String,
        rankOldAndNew : String,
        movieCd : String,
        movieNm : String,
        openDt : String,
        salesAmt : String,
        salesShare : String,
        salesInten : String,
        salesChange : String,
        salesAcc : String,
        audiCnt : String,
        audiInten : String,
        audiChange : String,
        audiAcc : String,
        scrnCnt : String,
        showCnt : String>
    >
  )
Row format serde 'org.apache.hive.hcatalog.data.JsonSerDe';

-- 일별 박스오피스 파일이 들어 있는 폴더 경로로 movies_js 테이블에 로딩.
load data local inpath '/home/hadoop/movies_data/datalist' into table movies_js;
```

```
hive (moviedata)> select * from movies_js
OK
```

```
{ "boxofficetype": "일별 박스오피스", "showrange": "20180801-20180801", "dailyboxofficelist": [{"rnum": "1", "rank": "1", "rankinten": "7", "rankoldandnew": "OLD", "moviecd": "20186202", "movienm": "신과함께-인화연", "opendt": "2018-08-01", "salesamt": "9684810000", "salesshare": "69.5", "salesacc": "9635814000", "saleschange": "19666.5", "salesacc": "9874188100", "audicnt": "1246603", "audiinten": "1240572", "audichange": "20569.9", "audiacc": "1268352", "scrncnt": "1967", "showcnt": "9825"}, {"rnum": "2", "rank": "2", "rankinten": "1", "rankoldandnew": "OLD", "moviecd": "20181181", "movienm": "미션 임파서블: 폴아웃", "opendt": "2018-07-25", "salesamt": "2091268500", "salesshare": "15.0", "salesacc": "1187331500", "saleschange": "-36.2", "salesacc": "36906473254", "audicnt": "258551", "audiinten": "145136", "audichange": "-36", "audiacc": "4394473", "scrncnt": "998", "scrncnt": "3827"}, {"rnum": "3", "rank": "3", "rankinten": "27", "rankoldandnew": "OLD", "moviecd": "20186501", "movienm": "극강의 챔피언: 박찬욱 감독", "opendt": "2018-08-01", "salesamt": "1145177500", "salesshare": "8.2", "salesacc": "51953.5", "salesacc": "11990500", "audicnt": "162242", "audiinten": "162022", "audichange": "73646.4", "audiacc": "166724", "scrncnt": "721", "showcnt": "1740"}, {"rnum": "4", "rank": "4", "rankinten": "-2", "rankoldandnew": "OLD", "moviecd": "20183361", "movienm": "엔크레타를 2", "opendt": "2018-07-18", "salesamt": "5607500", "salesshare": "4.0", "salesacc": "441566000", "saleschange": "-44.1", "salesacc": "21327771058", "audicnt": "76950", "audiinten": "-60073", "audichange": "-43.8", "audiacc": "2616266", "scrncnt": "583", "showcnt": "1284"}, {"rnum": "5", "rank": "5", "rankinten": "-2", "rankoldandnew": "OLD", "moviecd": "20185242", "movienm": "신비라파트 : 굿잇 도깨비와 비밀의 동굴", "opendt": "2018-07-25", "salesamt": "273460900", "salesshare": "2.0", "salesacc": "149442700", "saleschange": "-35.3", "salesacc": "3571395300", "audicnt": "39482", "audiinten": "21515", "audichange": "-35.3", "audiacc": "484967", "scrncnt": "490", "showcnt": "752"}, {"rnum": "6", "rank": "6", "rankinten": "1", "rankoldandnew": "OLD", "moviecd": "20186121", "movienm": "여는 겨울", "opendt": "2018-07-26", "salesamt": "48266200", "salesshare": "0.3", "salesacc": "21320900", "saleschange": "30.6", "salesacc": "513721500", "audicnt": "6192", "audiinten": "2754", "audichange": "30.8", "audiacc": "61909", "scrncnt": "75", "showcnt": "175"}, {"rnum": "7", "rank": "7", "rankinten": "-3", "rankoldandnew": "OLD", "moviecd": "20170942", "movienm": "미션 : 임파", "opendt": "2018-07-25", "salesamt": "47501700", "salesshare": "0.3", "salesacc": "340089700", "saleschange": "87.7", "salesacc": "6759590106", "audicnt": "6072", "audiinten": "43853", "audichange": "87.8", "audiacc": "869157", "scrncnt": "274", "showcnt": "393"}, {"rnum": "8", "rank": "8", "rankinten": "0", "rankoldandnew": "NEW", "moviecd": "208071", "movienm": "도언을 위한 나라는 없다", "opendt": "2008-02-21", "salesamt": "18190000", "salesshare": "0.1", "salesacc": "18190000", "saleschange": "100", "salesacc": "479122400", "audicnt": "1819", "audiinten": "1819", "audichange": "100", "audiacc": "69081", "scrncnt": "17", "showcnt": "1"}, {"rnum": "9", "rank": "9", "rankinten": "-4", "rankoldandnew": "OLD", "moviecd": "20180522", "movienm": "엔트렌드와 스포츠", "opendt": "2018-07-04", "salesamt": "11181200", "salesshare": "0.1", "salesacc": "123519100", "saleschange": "91.7", "salesacc": "47397390085", "audicnt": "1475", "audiinten": "15704", "audichange": "-91.4", "audiacc": "5439554", "scrncnt": "37", "showcnt": "44"}, {"rnum": "10", "rank": "10", "rankinten": "2", "rankoldandnew": "OLD", "moviecd": "20110027", "movienm": "여의도 열대", "opendt": "2011-01-20", "salesamt": "5024300", "salesshare": "0.0", "salesacc": "2321000", "saleschange": "-31.6", "salesacc": "336951661", "audicnt": "705", "audiinten": "286", "audichange": "-28.9", "audiacc": "42806", "scrncnt": "22", "showcnt": "33"}] }
```



# 데이터 처리

수집 data - 영화상세정보data를 hive를 이용해 table로 저장

```
-- 영화코드로 조회한 영화 정보 테이블 생성(json 구조)
create external table if not exists movies_feature(
  movieInfoResult struct <
    movieInfo : struct <
      movieCd : String,
      movieNm : String,
      movieNmEn : String,
      movieNmOg : String,
      showTm : String,
      prdtYear : String,
      openDt : String,
      prdtStatNm : String,
      typeNm : String,
      nations : Array<string>,
      genres : Array<struct <
        genreNm : String
      >>,
      directors : Array<string>,
      actors : Array<string>,
      showTypes : Array<string>,
      companys : Array<string>,
      audits : Array<string>,
      staffs : Array<string>
    >
  >
  source : String
)
Row format serde 'org.apache.hive.hcatalog.data.JsonSerDe';

--영화 정보 파일이 들어 있는 폴더 통째로 movies_feature 테이블에 로딩.
load data local inpath '/home/hadoop/movies_data/movie_detail' into table movies_feature;
```

```
hive (moviedata)> select * from movies_feature
OK
```

```
{
  "movieinfo": {
    "moviecd": "19880001",
    "movienm": "이웃집 토토로",
    "movienmen": "My Neighbor Totoro",
    "movienmog": "となりのトトロ",
    "prdtyear": "1988",
    "opendt": "20010728",
    "prdtstatnm": "개봉",
    "typenm": "장편",
    "nations": [
      {
        "nationNm": "일본"
      }
    ],
    "genres": [
      {
        "genreNm": "드라마"
      }
    ],
    "directors": [
      {
        "peopleNm": "미야자키 하야오",
        "peopleNmEn": "Hayao Miyazaki"
      }
    ],
    "actors": [
      {
        "peopleNm": "히다카 노리코",
        "peopleNmEn": "Noriko Hidaka",
        "cast": "",
        "castEn": ""
      },
      {
        "peopleNm": "사카모토 치카",
        "peopleNmEn": "Chika Sakamoto",
        "cast": "",
        "castEn": ""
      },
      {
        "peopleNm": "다치야마 미키",
        "peopleNmEn": "Miki Takizawa",
        "cast": "",
        "castEn": ""
      }
    ],
    "showtypes": [
      {
        "showTypeNm": "2D",
        "showTypeNmEn": "2D",
        "showTypeNmOg": "2D",
        "showTypeNmEnOg": "2D"
      }
    ],
    "companys": [
      {
        "companyCd": "2011081",
        "companyNm": "(주) 스마일 엔터테인먼트",
        "companyNmEn": "Smile Ent.",
        "companyPartNm": "제작사",
        "companyPartNmEn": "Production Company"
      },
      {
        "companyCd": "20168728",
        "companyNm": "씨네 그루 (주) 카다미엔티",
        "companyNmEn": "Cine Groove (Co.) Kadamy Entertainment",
        "companyPartNm": "배급사",
        "companyPartNmEn": "Distribution Company"
      },
      {
        "companyCd": "2010065",
        "companyNm": "대원미디어 (주)",
        "companyNmEn": "Daewon Media Co., Ltd.",
        "companyPartNm": "수입사",
        "companyPartNmEn": "Import Company"
      },
      {
        "companyCd": "2011091",
        "companyNm": "(주) 스마일엔터테인먼트",
        "companyNmEn": "Smile Ent.",
        "companyPartNm": "제작사",
        "companyPartNmEn": "Production Company"
      }
    ],
    "audits": [
      {
        "auditNm": "2001-1992",
        "watchGradeNm": "전체관람가",
        "staffs": [
          {
            "source": "영화진흥위원회"
          }
        ]
      }
    ]
  },
  "movieinfo": {
    "moviecd": "19990703",
    "movienm": "노팅 힐",
    "movienmen": "Notting Hill",
    "movienmog": "Notting Hill",
    "showTm": "124",
    "prdtYear": "1999",
    "opendt": "19990703",
    "prdtstatnm": "개봉",
    "typenm": "장편",
    "nations": [
      {
        "nationNm": "미국"
      },
      {
        "nationNm": "영국"
      }
    ],
    "genres": [
      {
        "genreNm": "멜로/로맨스"
      },
      {
        "genreNm": "코미디"
      }
    ],
    "directors": [
      {
        "peopleNm": "로저 미첼",
        "peopleNmEn": "Roger Michell"
      }
    ],
    "actors": [
      {
        "peopleNm": "줄리아 로버츠",
        "peopleNmEn": "Julia Roberts",
        "cast": "",
        "castEn": ""
      },
      {
        "peopleNm": "휴 그랜트",
        "peopleNmEn": "Hugh Grant",
        "cast": "",
        "castEn": ""
      },
      {
        "peopleNm": "폴리얼 다쳐",
        "peopleNmEn": "Polya Dache",
        "cast": "",
        "castEn": ""
      }
    ],
    "showtypes": [
      {
        "showTypeNm": "2D",
        "showTypeNmEn": "2D",
        "showTypeNmOg": "2D",
        "showTypeNmEnOg": "2D"
      }
    ],
    "companys": [
      {
        "companyCd": "20104509",
        "companyNm": "폴리그램 필름즈 인터내셔널",
        "companyNmEn": "Polygram Filmed Entertainment",
        "companyPartNm": "제작사",
        "companyPartNmEn": "Production Company"
      },
      {
        "companyCd": "20101439",
        "companyNm": "윌링 타이를 필름즈",
        "companyNmEn": "Willing Tails Films",
        "companyPartNm": "배급사",
        "companyPartNmEn": "Distribution Company"
      },
      {
        "companyCd": "20100603",
        "companyNm": "유니버설 픽처스 인터내셔널 코리아 (유)",
        "companyNmEn": "Universal Pictures International Korea",
        "companyPartNm": "수입사",
        "companyPartNmEn": "Import Company"
      },
      {
        "companyCd": "20101507",
        "companyNm": "유니버설 픽처스 인터내셔널 코리아 (유)",
        "companyNmEn": "Universal Pictures International Korea",
        "companyPartNm": "배급사",
        "companyPartNmEn": "Distribution Company"
      }
    ],
    "audits": [
      {
        "auditNm": "2001-1992",
        "watchGradeNm": "전체관람가",
        "staffs": [
          {
            "source": "영화진흥위원회"
          }
        ]
      }
    ]
  }
}
```

# 데이터 처리

각 *data*에서 필요한 정보만 *select*하여 해당 *data*를 CSV 파일로 추출.

```
[hadoop@dn01 ~]$ hive -e 'use moviedata;  
> select a.boxOfficeResult.showRange as showRange,  
> b.movieNm as movieNm,  
> b.movieCd as movieCd,  
> b.audiCnt as audiCnt,  
> b.showcnt as showcnt,  
> b.salesAcc as salesAcc,  
> b.audiAcc as audiAcc  
> from movies_js a LATERAL VIEW OUTER inline (a.boxOfficeResult.dailyBoxOfficeList) b  
> ' | sed 's/[\t]/,/g' > /home/hadoop/movies_data/movie_by_day.csv
```

```
[hadoop@dn01 ~]$ hive -e 'use moviedata;  
> select a.movieInfoResult.movieInfo.movieCd as movieCd,  
> a.movieInfoResult.movieInfo.genres[0].genreNm as genres  
> from movies_feature a' | sed 's/[\t]/,/g' > /home/hadoop/movies_data/movie_detail.csv
```

# 데이터 처리

csv 파일 로드 하여 필요한 정보만 포함한 table 생성

```
-- 날짜별 영화데이터 csv 파일 로드하여, 새로운 table에 저장.
```

```
create table movie_list_total
```

```
{
showRange string,
movieNm string,
movieCd string,
audiCnt int,
showcnt int,
salesAcc int,
audiAcc int
}
```

```
Row format delimited fields terminated by ',';
```

```
load data local inpath '/home/hadoop/movies_data/movie_by_day.csv' into table movie_list_total;
```

```
-- 영화 장르 csv파일 로드하여, 새로운 table에 저장.
```

```
create table movie_gen
```

```
{
movieCd string,
genres string
}
```

```
Row format delimited fields terminated by ',';
```

```
load data local inpath '/home/hadoop/movies_data/movie_detail.csv' into table movie_gen;
```

```
hive (moviedata)> select * from movie_list_total limit 60;
OK
```

20180801-20180801	신과함께-영과 연	20186202	1246603	9825	NULL	1268352	
20180801-20180801	미션 임파서블 : 플러웃	20181181	258561	3827	NULL	4394473	
20180801-20180801	극장판 헬로카봇 : 백악기 시대	20186501	162242	1740	1192090500	166724	
20180801-20180801	엔크레더블 2	20183361	76950	1284	NULL	2616266	
20180801-20180801	신비아파트 : 금빛 도깨비와 비밀의 동굴	20185242	39482	752	NULL	484967	
20180801-20180801	어느 가족	20186121	6192	175	513721500	61909	
20180801-20180801	연랑	20170942	6072	393	NULL	869157	
20180801-20180801	노인을 위한 나라는 없다	20080071	1819	17	479122400	69081	
20180801-20180801	렌트밀과 와스크	20180522	1475	44	NULL	5439554	
20180801-20180801	마이 엠 러브	20110027	705	33	336951661	42806	
20180801-20180801	신과함께-영과 연	20186202	1246603	9825	NULL	1268352	
20180801-20180801	미션 임파서블 : 플러웃	20181181	258561	3827	NULL	4394473	
20180801-20180801	극장판 헬로카봇 : 백악기 시대	20186501	162242	1740	1192090500	166724	
20180801-20180801	엔크레더블 2	20183361	76950	1284	NULL	2616266	
20180801-20180801	신비아파트 : 금빛 도깨비와 비밀의 동굴	20185242	39482	752	NULL	484967	
20180801-20180801	어느 가족	20186121	6192	175	513721500	61909	
20180801-20180801	연랑	20170942	6072	393	NULL	869157	
20180801-20180801	노인을 위한 나라는 없다	20080071	1819	17	479122400	69081	
20180801-20180801	렌트밀과 와스크	20180522	1475	44	NULL	5439554	
20180801-20180801	마이 엠 러브	20110027	705	33	336951661	42806	
20180802-20180802	신과함께-영과 연	20186202	1078012	10079	NULL	2346364	
20180802-20180802	미션 임파서블 : 플러웃	20181181	249826	3600	NULL	4644299	
20180802-20180802	극장판 헬로카봇 : 백악기 시대	20186501	127108	1825	2092884000	293832	
20180802-20180802	엔크레더블 2	20183361	74011	1256	NULL	2690277	



# 데이터 처리

csv 파일 로드 하여 필요한 정보만 포함한 table 생성

```
-- 날짜별 영화데이터 csv 파일 로드하여, 새로운 table에 저장.
```

```
create table movie_list_total
```

```
{
showRange string,
movieNm string,
movieCd string,
audiCnt int,
showcnt int,
salesAcc int,
audiAcc int
}
```

```
Row format delimited fields terminated by ',';
```

```
load data local inpath '/home/hadoop/movies_data/movie_by_day.csv' into table movie_list_total;
```

```
-- 영화 장르 csv파일 로드하여, 새로운 table에 저장.
```

```
create table movie_gen
```

```
{
movieCd string,
genres string
}
```

```
Row format delimited fields terminated by ',';
```

```
load data local inpath '/home/hadoop/movies_data/movie_detail.csv' into table movie_gen;
```

```
hive (moviedata)> select * from movie_list_total limit 60;
OK
```

20180801-20180801	신과함께-영과 연	20186202	1246603	9825	NULL	1268352	
20180801-20180801	미션 임파서블 : 플러웃	20181181	258561	3827	NULL	4394473	
20180801-20180801	극장판 헬로카봇 : 백악기 시대	20186501	162242	1740	1192090500	166724	
20180801-20180801	엔크레더블 2	20183361	76950	1284	NULL	2616266	
20180801-20180801	신비아파트 : 금빛 도깨비와 비밀의 동굴	20185242	39482	752	NULL	484967	
20180801-20180801	어느 가족	20186121	6192	175	513721500	61909	
20180801-20180801	연랑	20170942	6072	393	NULL	869157	
20180801-20180801	노인을 위한 나라는 없다	20080071	1819	17	479122400	69081	
20180801-20180801	렌트밀과 와스크	20180522	1475	44	NULL	5439554	
20180801-20180801	마이 엠 러브	20110027	705	33	336951661	42806	
20180801-20180801	신과함께-영과 연	20186202	1246603	9825	NULL	1268352	
20180801-20180801	미션 임파서블 : 플러웃	20181181	258561	3827	NULL	4394473	
20180801-20180801	극장판 헬로카봇 : 백악기 시대	20186501	162242	1740	1192090500	166724	
20180801-20180801	엔크레더블 2	20183361	76950	1284	NULL	2616266	
20180801-20180801	신비아파트 : 금빛 도깨비와 비밀의 동굴	20185242	39482	752	NULL	484967	
20180801-20180801	어느 가족	20186121	6192	175	513721500	61909	
20180801-20180801	연랑	20170942	6072	393	NULL	869157	
20180801-20180801	노인을 위한 나라는 없다	20080071	1819	17	479122400	69081	
20180801-20180801	렌트밀과 와스크	20180522	1475	44	NULL	5439554	
20180801-20180801	마이 엠 러브	20110027	705	33	336951661	42806	
20180802-20180802	신과함께-영과 연	20186202	1078012	10079	NULL	2346364	
20180802-20180802	미션 임파서블 : 플러웃	20181181	249826	3600	NULL	4644299	
20180802-20180802	극장판 헬로카봇 : 백악기 시대	20186501	127108	1825	2092884000	293832	
20180802-20180802	엔크레더블 2	20183361	74011	1256	NULL	2690277	

# 데이터 처리

날짜별 영화 데이터 파일과 영화정보 파일을join하여 최종 전처리 완료 table 완성

```
create table movie_join as
select a.*, b.genres
from (select to_date(from_unixtime(UNIX_TIMESTAMP(SUBSTRING_INDEX(showRange, '~', 1)), 'yyyyMMdd')) showdate,
movieNm, movieCd, audiCnt, showcnt, salesAcc, audiAcc
from movie_list_total) a
left outer join movie_gen b
on a.movieCd = b.movieCd;
```

```
hive (moviedata)> select * from movie_join limit 20;
OK
```

2018-08-01	신과함께-인과연	20186202	1246603	9825	NULL	1268352	판타지	
2018-08-01	미션 임파서블 : 폴아웃	20181181	258561	3827	NULL	4394473	액션	
2018-08-01	극장판 헬로카봇 : 백악기 시대	20186501		162242	1740	1192090500		166724 애니메이션
2018-08-01	인크레더블 2	20183361	76950	1284	NULL	2616266	애니메이션	
2018-08-01	신비아파트 : 금빛 도깨비와 비밀의 동굴		20185242	39482	752	NULL		484967 애니메이션
2018-08-01	어느 가족	20186121	6192	175	513721500	61909	드라마	
2018-08-01	인랑	20170942	6072	393	NULL	869157	SF	
2018-08-01	노인을 위한 나라는 없다	20080071	1819	17	479122400	69081	스릴러	
2018-08-01	엔트맨과 와스프	20180522	1475	44	NULL	5439554	액션	
2018-08-01	아이 엠 러브	20110027	705	33	336951661	42806	드라마	
2018-08-01	신과함께-인과연	20186202	1246603	9825	NULL	1268352	판타지	
2018-08-01	미션 임파서블 : 폴아웃	20181181	258561	3827	NULL	4394473	액션	
2018-08-01	극장판 헬로카봇 : 백악기 시대	20186501		162242	1740	1192090500		166724 애니메이션
2018-08-01	인크레더블 2	20183361	76950	1284	NULL	2616266	애니메이션	
2018-08-01	신비아파트 : 금빛 도깨비와 비밀의 동굴		20185242	39482	752	NULL		484967 애니메이션



# 구현 내역

1. 월별 인기 장르
2. 요일별 관객수
3. 장르별 평균 관객수
4. 영화별 최대  
관객수/매출액

# 1. 월별 인기 장르

## - movie\_join table 확인

```
hive (moviedata)> select * from movie_join;
```

```
OK
2018-08-01   신 과 함 께 -인 과 연      20186202      1246603  9825
NULL      1268352   판 타 지
2018-08-01   미 션 임 파 서 블 : 플 아 웃      20181181      258561   3827
NULL      4394473   액 션
2018-08-01   극 장 판 헬 로 카 붓 : 백 악 기 시 대      20186501      1622
42      1740      1192090500      166724   애 니 메 이 션
```

## - movie\_join 에서 select 로 월별 인기 장르 를 가져온다

```
hive (moviedata)> select a.month, a.genres, sum(a.audiCnt) audiCnt
> from (select date_format(showdate,'yyyy-MM')month,
> audiCnt , genres from movie_join) a
> where a.genres is not null
> group by a.genres, a.month
> order by a.month
```

```
2018-10 SF      649714
2018-8 SF      54102
2018-9 SF      190090
2018-10 가 족      148975
2018-11 가 족      118831
2018-12 가 족      28941
2019-3 가 족      236330
2019-4 가 족      91734
2019-1 공 연      262679
2019-2 공 연      79687
2018-10 공 포 (호 러 )      100521
2018-11 공 포 (호 러 )      131307
2018-9 공 포 (호 러 )      965975
2019-2 공 포 (호 러 )      397929
2019-3 공 포 (호 러 )      936041
2019-4 공 포 (호 러 )      863061
2019-5 공 포 (호 러 )      165875
2019-6 공 포 (호 러 )      459234
2019-7 공 포 (호 러 )      294870
```

## 2. 요일별 관객수

- movie\_join data에서 주,요일별로 관객수 합계를 select하여 result\_by\_week\_of\_day 테이블로 복제 생성.

```
create table result_by_week_of_day as
select a.week,a.wod
,sum(a.audiCnt) audiCnt from (select date_format(showdate,'W') week, case date_format(showdate,'u')
when 1 then "일"
when 2 then "월"
when 3 then "화"
when 4 then "수"
when 5 then "목"
when 6 then "금"
when 7 then "토"
end wod,audiCnt from movie_join) a group by a.week,a.wod order by week, wod;
```

- result\_by\_week\_of\_day 테이블 에서 요일별 관객수 합계 추출 결과 csv로 저장

```
hive -e 'use moviedata; set hive.cli.print.header=true;
select wod, sum(audicnt)sum from result_by_week_of_day group by wod' | sed 's/[\t]/,/g' > /home/hadoop/movies_data/result2.csv
```

### 3. 장르별 평균 관객수

- **movie\_join** 테이블에서 장르별 평균 관객수를 **select**하여 **result\_by\_genres** 테이블로 복제 생성.

```
hive (moviedata)> CREATE TABLE result_by_genres as
> SELECT genres, ROUND(AVG(audicnt), 2) AS audiavg
> FROM movie_join
> GROUP BY genres
> HAVING genres IS NOT null
> ORDER BY audiavg DESC
```

- **show tables;** > **result\_by\_genres** 테이블 생성 확인.

```
hive (moviedata)> show tables;
result_by_genres
```

- **zeplin**으로 시각화를 위해 **csv** 파일로 변환.

```
hive -e 'use moviedata; set hive.cli.print.header=true; SELECT
result_by_genres.genres as genres, result_by_genres.audiavg as
audiavg FROM result_by_genres' | sed 's/[\t]/./g' >
/home/hadoop/movies_data/result_by_genres.csv
```



## 4. 영화별 최대 관객수/매출액

- hive(moviedata)에 moives\_ranking 테이블 생성

```
hive (moviedata)> CREATE TABLE movies_ranking(  
    > movieNm String,  
    > salesAcc String,  
    > audiAcc String)  
    > row format delimited fields terminated by ',';
```

- movies\_ranking 테이블 생성 확인

```
hive (moviedata)> show tables;  
OK  
movies_js  
movies_ranking
```

- 저장한 csv파일을 테이블 movies\_ranking에 inpath해 주기

```
[hadoop@dn01 movies_data]$ load data local inpath '/home/hadoop/movies_data/movies_ranking.csv' into table movies_ranking;
```

- 영화별 최대 관객수/매출액 찾기

```
hive (moviedata)> select movieNm, max(audiAcc) as audiAcc, max(salesAcc) as salesAcc from movies_ranking group by movieNm;
```

.....

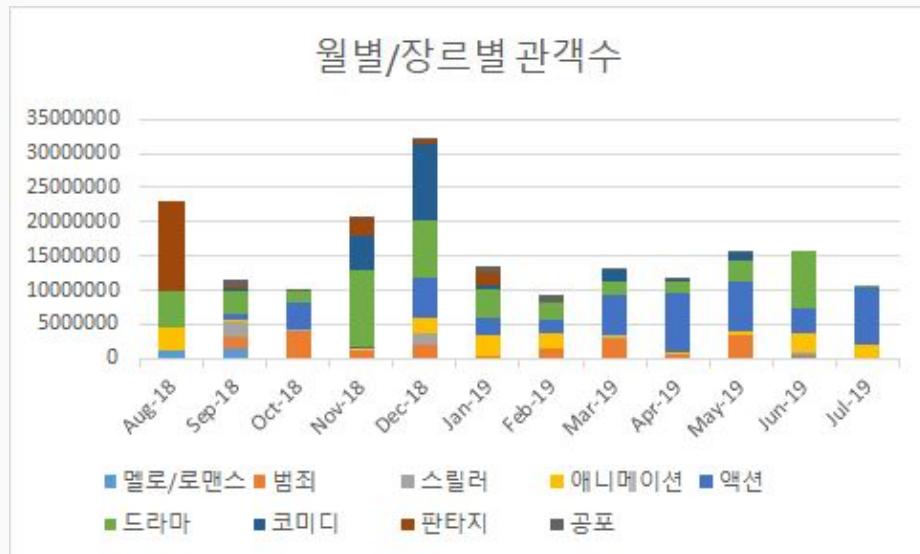
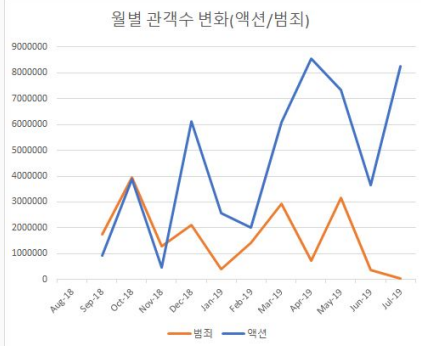
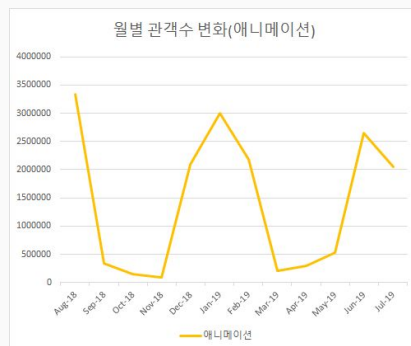
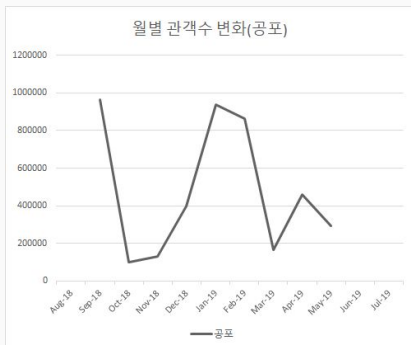
A close-up photograph of a person's hand holding a stylus and drawing on a tablet. The background is blurred, showing some bokeh lights. The text '결과 시각화' is overlaid on the image in white.

# 결과 시각화

1. 월별 인기 장르
2. 요일별 관객수
3. 장르별 평균 관객수
4. 영화별 최대  
관객수/매출액

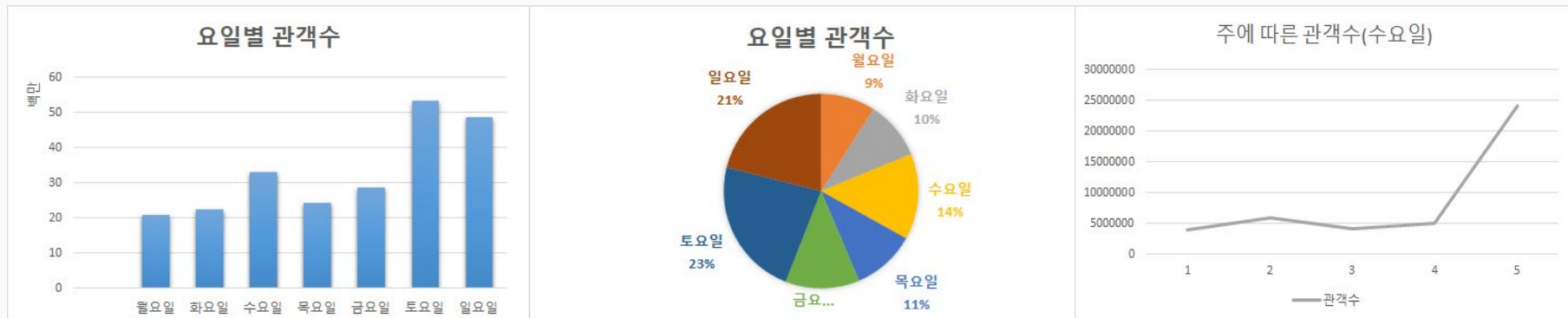


# 월별 인기장르



- 계절 영향을 가장 많이 받는 장르: 공포영화
- 공포영화를 제외한 다른 장르들은 계절의 영향을 받기 보다는 당월에 개봉된 영화의 영향을 받는 것으로 판단됨

# 요일별 관객수



- 주말인 토요일, 일요일에 관객수가 제일 많은 것으로 확인
- 평일 중 수요일에 관객수가 많이 나타나는 이유는 '문화의 날 할인 행사' 등의 영향으로 추정.

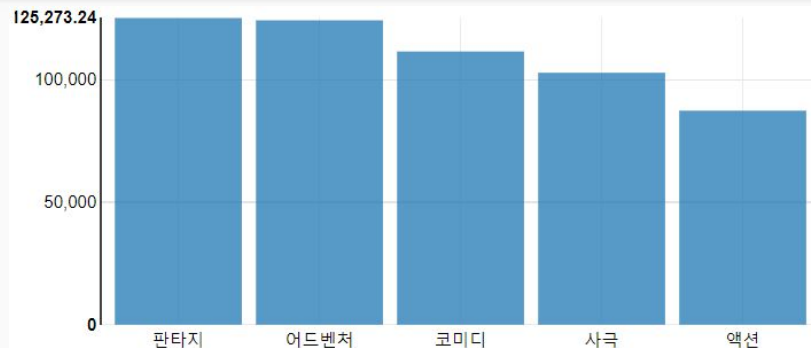
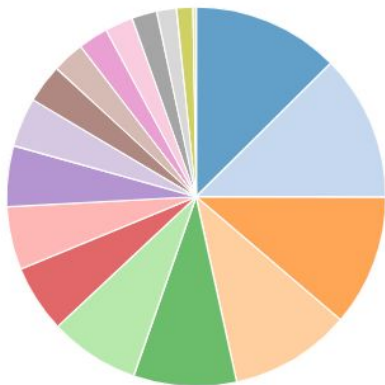
# 장르별 평균 관객수

%hive  
select \* from result\_by\_genres

FINISHED ▶ ⌵ ⌵ ⌵ ⌵

settings ▼

판타지 어드벤처 코미디 사극 액션 범죄  
 미스터리 드라마 뮤지컬 스릴러 멜로/로맨스 애니메이션  
 SF 공포(호러) 공연 기타 가족 다큐멘터리



2018년 8월 ~ 2019년 8월

인기 장르 5위!

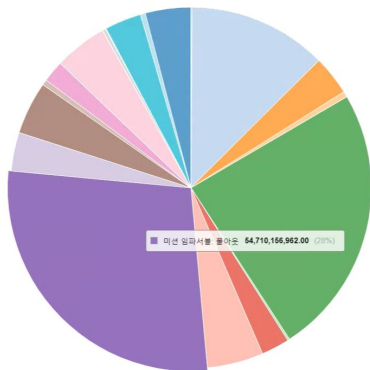
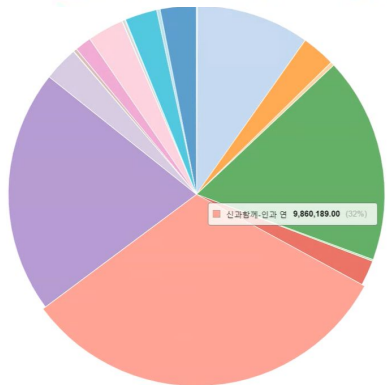
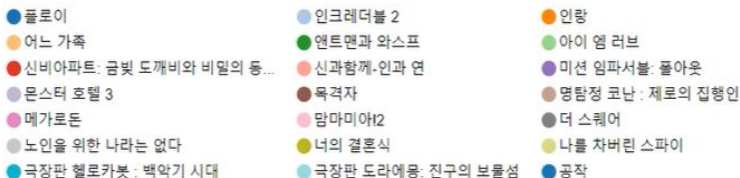
- 1) 판타지
- 2) 어드벤처
- 3) 코미디
- 4) 사극
- 5) 액션

# 영화별 최대 관객수/매출액

```
%hive
select movienm, max(audiAcc) as audiAcc, max(salesAcc) as salesAcc from movies_ranking group by movienm
```



settings



keys

movienm x

values

audiacc SUM x

values

salesacc SUM x

- 최대 관객수 1위!  
신과함께-인과 연
- 최대 매출액 1위!  
미션 임파서블: 폴아웃

감사합니다.