In [0]:

```
from pyspark.sql.types import *

from pyspark import SparkContext
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("my project 1").getOrCreate()
sc = spark.sparkContext

# Read a CSV into a dataframe
# There is a smarter version, that will first check if there is a Parquet file and use it
def load_PD_file(filename_or_dir, schema) :
    dataPath = "/mnt/ddscoursedatastorage/fwm-stb-data/" + filename_or_dir
    df = spark.read.format("csv")\
      .option("header","false")\
      .option("delimiter", "|")\
      .schema(schema)\
      .load(dataPath)
    return df
```

In [0]:

```
df1 = spark.read.csv("/mnt/ddscoursedatastorage/dds-students/test.csv")
```

Reading the data files

Reading Refrence Data

In [0]:

```
# Reading the Reference Parquet files

ref_data = spark.read.parquet('/ref_data_raw').withColumnRenamed("_device-id","device_id"
)\
                                              .withColumnRenamed("_dma","dma")\
                                              .withColumnRenamed("_dma-code","dma_cod
e")\
                                              .withColumnRenamed("_household-id","hou
sehold_id")\
                                              .withColumnRenamed("_household-type","h
ousehold_type")\
                                              .withColumnRenamed("_system-type","syst
em_type")\
                                              .withColumnRenamed("_zipcode","zipcode"
)
ref_data_count = ref_data.count()
print(ref_data_count)
ref_data
```

```
203581233
Out[6]: DataFrame[device_id: string, dma: string, dma_code: bigint, household_id: bigint,
household_type: string, system_type: string, zipcode: bigint]
```

Reading Daily Program Data

In [0]:

```
# Reading the Daily Programs CSV file

daily_prog_schema =  StructType([StructField('prog_code',StringType()),
                     StructField('title',StringType()),
                     StructField('genre',StringType()),
                     StructField('air_date',StringType()),
```

```
                    StructField('air_time',StringType()),
                    StructField('Duration',FloatType())
                                        ])
daily_prog_data = load_PD_file("Daily program data/" , daily_prog_schema  )
```

In [0]:

```
daily_prog_data.show()
```

```
+---------+-----+-----+--------+--------+--------+
|prog_code|title|genre|air_date|air_time|Duration|
+---------+-----+-----+--------+--------+--------+
+---------+-----+-----+--------+--------+--------+
```

Reading Program Viewing Data

In [0]:

```
#Reading the 1% sample of the viewing data from a Parquet file

viewing_data = spark.read.parquet('/sample_viewing_2_5percent')

print(f'There are {viewing_data.count():,} entries in viewing_data dataframe!')
```

```
There are 130,289,194 entries in viewing_data dataframe!
```

Reading Demographic Data

In [0]:

```
# Reading the Demographic CSV file

demographic_schema =  StructType([StructField('household_id',StringType()),
                    StructField('household_size',IntegerType()),
                    StructField('num_adults',IntegerType()),
                    StructField('num_generations',IntegerType()),
                    StructField('adult_range',StringType()),
                    StructField('marital_status',StringType()),
                    StructField('race_code',StringType()),
                    StructField('presence_children',StringType()),
                    StructField('num_children',IntegerType()),
                    StructField('age_children',StringType()), #format like range - 'bi
twise'
                    StructField('age_range_children',StringType()),
                    StructField('dwelling_type',StringType()),
                    StructField('home_owner_status',StringType()),
                    StructField('length_residence',IntegerType()),
                    StructField('home_market_value',StringType()),
                    StructField('num_vehicles',IntegerType()),
                    StructField('vehicle_make',StringType()),
                    StructField('vehicle_model',StringType()),
                    StructField('vehicle_year',IntegerType()),
                    StructField('net_worth',IntegerType()),
                    StructField('income',StringType()),
                    StructField('gender_individual',StringType()),
                    StructField('age_individual',IntegerType()),
                    StructField('education_highest',StringType()),
                    StructField('occupation_highest',StringType()),
                    StructField('education_1',StringType()),
                    StructField('occupation_1',StringType()),
                    StructField('age_2',IntegerType()),
                    StructField('education_2',StringType()),
                    StructField('occupation_2',StringType()),
                    StructField('age_3',IntegerType()),
                    StructField('education_3',StringType()),
                    StructField('occupation_3',StringType()),
                    StructField('age_4',IntegerType()),
```

```
                        StructField('education_4',StringType()),
                        StructField('occupation_4',StringType()),
                        StructField('age_5',IntegerType()),
                        StructField('education_5',StringType()),
                        StructField('occupation_5',StringType()),
                        StructField('polit_party_regist',StringType()),
                        StructField('polit_party_input',StringType()),
                        StructField('household_clusters',StringType()),
                        StructField('insurance_groups',StringType()),
                        StructField('financial_groups',StringType()),
                        StructField('green_living',StringType())
                                            ])
demographic_data = load_PD_file("demographic/" , demographic_schema)
```

Question 1.1

In [0]:

```
from pyspark.sql.functions import *

# Removimg non-relevant columns and duplicated entries
ref_data_Q1 = ref_data.distinct().select('device_id', 'household_id', 'dma')
daily_prog_data_Q1 = daily_prog_data.distinct().drop('title')
viewing_data_Q1 = viewing_data.distinct().select('prog_code', 'device_id', 'event_date')

# changing string to a number according the requries we've been given
# changing null strings to None so that when converting the 'income' column values to int
they will still considered null
demographic_data = demographic_data.withColumn('income', when(demographic_data.income ==
'A', '10')
                                                         .when(demographic_data.income =
= 'B', '11')
                                                         .when(demographic_data.income =
= 'C', '12')
                                                         .when(demographic_data.income =
= 'D', '13')
                                                         .when(demographic_data.income =
= 'null', None)
                                                         .otherwise(demographic_data.inc
ome)          )\
                                      .withColumn('income', col('income').cast("int"))

# Removimg non-relevant columns and duplicated entries
demographic_data_Q1 = demographic_data.distinct().select('household_id', 'household_size'
, 'num_adults', 'net_worth', 'income')
```

In [0]:

```
from pyspark.sql.column import *
from pyspark.sql.types import StringType
from pyspark.sql.functions import *

# Removimg entries that contain null values or null strings in rellevant fields/columns (
fields that are used in the queries)
# prepering helpful dataframes that will help us later in the queries
#1st condition
viewing_data_Q1_no_null = viewing_data_Q1.dropna().filter((col('prog_code') != 'null') |
                                                          (col('device_id') != 'null')
|
                                                          (col('event_date') != 'null')
).cache()
viewing_data_Q1_no_null1 = viewing_data_Q1.filter((col('device_id') != 'null') |
                                                          (col('event_date') != 'null')
)\
                                              .filter((col('device_id').isNotNull()) |
                                                          (col('event_date').isNotNull())
).cache()
```

```
avg_daily_events_per_device = spark.sql("""SELECT device_id, COUNT(*)/COUNT(DISTINCT even
t_date) AS avg_daily
                                           FROM viewing_data_Q1_no_null1
                                           GROUP BY device_id, num_dates
                                           """)
avg_daily_events_per_device = avg_daily_events_per_device.join(viewing_data_Q1_no_null, '
device_id', 'inner').cache()

#2nd condition
ref_data_Q1_no_null = ref_data_Q1.filter((col('device_id') != 'null') |
                                         (col('dma') != 'null') ) \
                                 .filter((col('device_id').isNotNull()) |
                                         (col('dma').isNotNull()) ).cache()
ref_data_is_z_in_dma = ref_data_Q1_no_null.withColumn("dma_contains_z", ref_data_Q1_no_nu
ll.dma.ilike('%z%')).cache()

# 3rd condition
viewing_data_Q1_no_null3 = viewing_data_Q1.filter((col('prog_code') != 'null') |
                                                  (col('device_id') != 'null') )\
                                          .filter((col('prog_code').isNotNull()) |
                                                  (col('device_id').isNotNull()) ).cache(
)
viewing_data_Q1_no_null3.createOrReplaceTempView("viewing_data_Q1_no_null3")

ref_data_Q1_no_null = ref_data_Q1.filter((col('household_id') != 'null') |
                                         (col('device_id') != 'null') )\
                                 .filter((col('household_id').isNotNull()) |
                                         (col('device_id').isNotNull()) ).cache()
ref_data_Q1_no_null.createOrReplaceTempView("ref_data_Q1_no_null")

demographic_data_Q1_no_null = demographic_data_Q1.filter((col('household_id') != 'null')
|
                                                         (col('num_adults') != 'null')
|
                                                         (col('net_worth') != 'null') )
\
                                                  .filter((col('household_id').isNotNull
()) |
                                                         (col('num_adults').isNotNull()
) |
                                                         (col('net_worth').isNotNull())
).cache()
demographic_data_Q1_no_null.createOrReplaceTempView("demographic_data_Q1_no_null")

# 4th condition
daily_prog_data_Q1_no_null = daily_prog_data_Q1.filter((col('prog_code') != 'null') |
                                                       (col('air_date') != 'null') |
                                                       (col('air_time') != 'null') )\
                                               .filter((col('prog_code').isNotNull())
|
                                                       (col('air_date').isNotNull()) |
                                                       (col('air_time').isNotNull()) )
.cache()

prog_with_dayofweek = daily_prog_data_Q1_no_null.withColumn('day_of_week',
                                                            dayofweek(to_date(daily_prog_data_Q
1_no_null['air_date'], 'yyyyMMdd'))).cache()

demographic_data_Q1_no_null4 = demographic_data_Q1.filter((col('household_id') != 'null')
|
                                                         (col('household_size') != 'null
' ))\
                                                  .filter((col('household_id').isNotNull(
)) |
                                                         (col('household_size').isNotNul
l()) ).cache()

# 5th condition
demographic_data_Q1_no_null5 = demographic_data_Q1.filter((col('household_id') != 'null')
|
                                                         (col('income') != 'null' ))\
                                                  .filter((col('household_id').isNotNull(
```

```
)) |
                                                    (col('income').isNotNull()) ).c
ache()

ref_data_Q1_no_null5 = ref_data_Q1_no_null.drop('dma')
num_devices_per_household = ref_data_Q1_no_null5.distinct()\
                                                .groupBy('household_id').agg(countDisti
nct('device_id').alias('num_devices')).cache()
ref_data_with_count = ref_data_Q1_no_null5.join(num_devices_per_household, on='household_
id', how='inner').cache()
household_data_with_count = ref_data_with_count.join(demographic_data_Q1_no_null5, 'house
hold_id', 'inner').cache()

# 6th condition
daily_prog_data_Q1_no_null6 = daily_prog_data_Q1.filter((col('prog_code') != 'null') |
                                                    (col('genre') != 'null') |
                                                    (col('Duration') != 'null') )\
                                                .filter((col('prog_code').isNotNull())
|
                                                    (col('genre').isNotNull()) |
                                                    (col('Duration').isNotNull()) )
.cache()

prog_genres = daily_prog_data_Q1_no_null6.select('prog_code', 'genre', 'Duration')\
                                 .withColumn("single_genres", explode(split(col("genre"),
',').cache()
```

In [0]:

```
joined_1 = household_data_with_count.join(ref_data_is_z_in_dma, on=['household_id', 'devi
ce_id'], how='inner')
joined_2 = prog_with_dayofweek.join(prog_genres, on=['prog_code', 'genre', 'Duration'],
how='inner')
joined_3 = avg_daily_events_per_device.join(joined_1, on='device_id', how='inner')
joined = joined_3.join(joined_2, on='prog_code', how='inner')
joined.display()
```

Question 1.2

Part A - in the PDF attached

Part B

In [0]:

```
viewing_num_dates.show()
```

In [0]:

```
# taking the requred programs as asked in condition 1
cond_1 = avg_daily_events_per_device.where(col('avg_daily') < 5)\
                                    .select('prog_code').distinct().cache()

print(cond_1.count())
```

```
9138
```

In [0]:

```
# taking the requred programs as asked in condition 2
cond_2 = ref_data_is_z_in_dma.where(ref_data_is_z_in_dma['dma_contains_z'] == True)\
                             .select('device_id')\
                             .join(viewing_data_Q1_no_null3, 'device_id', 'inner')\
                             .select('prog_code').distinct().cache()

print(cond_2.count())
```

```
235428
```

In [0]:

```
# taking the requred programs as asked in condition 3
cond_3 = spark.sql("""SELECT DISTINCT prog_code FROM viewing_data_Q1_no_null3
                  WHERE device_id in (SELECT device_id
                                      FROM demographic_data_Q1_no_null D INNER JOIN ref
_data_Q1_no_null R ON D.household_id == R.household_id
                                      WHERE D.num_adults < 3 and D.net_worth > 8)
                                      """)

cond_3.cache()
print(cond_3.count())
```

```
180061
```

In [0]:

```
# dataframe of the programs (prog_code) that was aired between Friday at 6PM and Saturday
at 7PM
prog_filtered_date_time = prog_with_dayofweek.filter(( (col('day_of_week') == 6) &
                                                       (daily_prog_data_Q1_no_null["air_
time"] >= 180000) ) |
                                                     ( (col('day_of_week') == 7) &
                                                      (daily_prog_data_Q1_no_null["air_t
ime"] <= 190000) ))\
                                            .select("prog_code").distinct().cache()

# combining each program with the device it was watched from
devices_prog_date_time = prog_filtered_date_time\
                         .join(viewing_data_Q1_no_null3.select("prog_code", "device_id"),
on='prog_code', how='inner').cache()

# combining each device with the households they are located at
households_prog_date_time = devices_prog_date_time.join(ref_data_Q1_no_null, "device_id",
"inner")\
                                                  .select('prog_code', 'household_id').
distinct().cache()

# taking households with size higher than or equal to 8
household_above_8 = demographic_data_Q1_no_null4.filter(demographic_data_Q1_no_null4['hou
sehold_size'] >= 8).select('household_id').distinct().cache()

# taking the requred programs as asked in condition 4 by combining each household with th
e programs they watched
cond_4 = household_above_8.join(households_prog_date_time, on='household_id', how='inner
')\
                          .select("prog_code").distinct().cache()

prog_filtered_date_time.unpersist()
devices_prog_date_time.unpersist()
households_prog_date_time.unpersist()
household_above_8.unpersist()

print(cond_4.count())
```

```
67646
```

In [0]:

```
# calculatig the average household income in the data and turning it into a list
average = demographic_data_Q1_no_null5.select(avg('income')).collect()
# splitting the requred value of the average in the list
avg_income = float(str(average[0]).split('=')[1].split(')')[0])

# taking the requred programs as asked in condition 5
cond_5 = household_data_with_count.filter((col('num_devices') > 3) & (col('income') < av
g_income))\
                                  .select('device_id').distinct()\
                                  .join(viewing_data_Q1_no_null3.drop('event_date'), 'de
vice_id', 'inner')\
                                  .select('prog_code').distinct().cache()
```

```
print(cond_5.count())
```

```
255489
```

In [0]:

```
from pyspark.sql.functions import *
# list of the genres that at least one of them should be contained in the program genres
genres = ['Talk', 'Politics', 'News', 'Community', 'Crime']

# creating a dataframe of these genres (from the list above)
genres_df = spark.createDataFrame(genres, StringType())
# renaming the column of these genres
genres_df = genres_df.withColumnRenamed('value', 'single_genres')

# taking the requred programs as asked in condition 6
cond_6 = prog_genres.join(genres_df, on='single_genres', how='inner')\
                    .select('prog_code', 'Duration')
                    .where(col('Duration') > 35)\
                    .select('prog_code').distinct()\.cache()

print(cond_6.count())
```

```
33648
```

In [0]:

```
from pyspark.sql.functions import *

# Adding a counting of conditions column and adding 1 to all entries that are true for co
ndition 1.
counter_1 = daily_prog_data_Q1.select('prog_code').distinct()\
                              .filter((col('prog_code') != 'null') | (col('prog_code').i
sNotNull())).cache()
counter_1 = counter_1.alias('all_programs')\
                     .join(cond_1.alias('cond_1'), col("all_programs.prog_code") == col(
"cond_1.prog_code"), 'left')\
                     .select("all_programs.*", when(col("cond_1.prog_code").isNotNull(),
1)\
                                                .otherwise(0)\
                                                .alias("num_conditions_met"))\
                     .distinct().cache()
```

In [0]:

```
# A function that automates the counter of conditions column.
def conditions_counter(cond, curr_counter):
    counter = curr_counter.alias('current_programs')\
                          .join(cond.alias('cond'), col("current_programs.prog_code") ==
col("cond.prog_code"), 'left')\
                          .select("current_programs.prog_code",
                                  when(col("cond.prog_code").isNotNull(), curr_counter.n
um_conditions_met + 1)\
                                 .otherwise(curr_counter.num_conditions_met)\
                                 .alias("num_conditions_met"))\
                          .distinct().cache()
    return counter
```

In [0]:

```
# Adding 1 in the counter column for each entry that is true for any conditions.
counter_2 = conditions_counter(cond_2, counter_1)
counter_3 = conditions_counter(cond_3, counter_2)
counter_4 = conditions_counter(cond_4, counter_3)
counter_5 = conditions_counter(cond_5, counter_4)
counter_6 = conditions_counter(cond_6, counter_5)
counter_1.unpersist()
counter_2.unpersist()
counter_3.unpersist()
counter_4.unpersist()
```

```
counter_5.unpersist()

# Filtering out entries that are true for less than 4 conditions - these are the maliciou
s entries.
malicious = counter_6.filter(col('num_conditions_met') >= 4)\
                     .select('prog_code').distinct().cache()
```

In [0]:

```
malicious.count()
```

Out[30]: 71765

In [0]:

```
# Adding all the details of the malicious programs.
output = daily_prog_data.join(malicious, on='prog_code', how='inner')\
                        .orderBy('prog_code').cache()

output.display()
output.show(150, truncate=False)
```

prog_code	title	genre	air_date	air_time	Duration
EP000000211576	20/20	Newsmagazine	20151226	030000	60.0
EP000000211576	20/20	Newsmagazine	20151226	040000	60.0
EP000000211576	20/20	Newsmagazine	20151226	060000	60.0
EP000000211614	20/20	Newsmagazine	20150104	030000	60.0
EP000000211614	20/20	Newsmagazine	20150104	040000	60.0
EP000000211614	20/20	Newsmagazine	20150104	060000	60.0
EP000000211639	20/20	Newsmagazine	20150704	020000	60.0
EP000000211639	20/20	Newsmagazine	20150704	030000	60.0
EP000000211639	20/20	Newsmagazine	20150704	040000	60.0
EP000000211639	20/20	Newsmagazine	20150704	050000	60.0

```
+--------------+-----+------------+--------+--------+--------+
|prog_code     |title|genre       |air_date|air_time|Duration|
+--------------+-----+------------+--------+--------+--------+
|EP000000211576|20/20|Newsmagazine|20151226|030000  |60.0    |
|EP000000211576|20/20|Newsmagazine|20151226|040000  |60.0    |
|EP000000211576|20/20|Newsmagazine|20151226|060000  |60.0    |
|EP000000211614|20/20|Newsmagazine|20150104|030000  |60.0    |
|EP000000211614|20/20|Newsmagazine|20150104|040000  |60.0    |
|EP000000211614|20/20|Newsmagazine|20150104|060000  |60.0    |
|EP000000211639|20/20|Newsmagazine|20150704|020000  |60.0    |
|EP000000211639|20/20|Newsmagazine|20150704|030000  |60.0    |
|EP000000211639|20/20|Newsmagazine|20150704|040000  |60.0    |
|EP000000211639|20/20|Newsmagazine|20150704|050000  |60.0    |
|EP000000211645|20/20|Newsmagazine|20150110|030100  |59.0    |
|EP000000211645|20/20|Newsmagazine|20150110|040100  |59.0    |
|EP000000211645|20/20|Newsmagazine|20150110|060100  |59.0    |
|EP000000211646|20/20|Newsmagazine|20150117|030100  |59.0    |
|EP000000211646|20/20|Newsmagazine|20150117|040100  |59.0    |
|EP000000211646|20/20|Newsmagazine|20150117|060100  |59.0    |
|EP000000211647|20/20|Newsmagazine|20150124|030100  |59.0    |
|EP000000211647|20/20|Newsmagazine|20150124|040100  |59.0    |
|EP000000211647|20/20|Newsmagazine|20150124|060100  |59.0    |
|EP000000211648|20/20|Newsmagazine|20150131|030100  |59.0    |
|EP000000211648|20/20|Newsmagazine|20150131|040100  |59.0    |
|EP000000211648|20/20|Newsmagazine|20150131|060100  |59.0    |
|EP000000211650|20/20|Newsmagazine|20150207|030100  |59.0    |
|EP000000211650|20/20|Newsmagazine|20150207|040100  |59.0    |
|EP000000211650|20/20|Newsmagazine|20150207|060100  |59.0    |
|EP000000211654|20/20|Newsmagazine|20150429|020100  |59.0    |
|EP000000211654|20/20|Newsmagazine|20150429|030100  |59.0    |
|EP000000211654|20/20|Newsmagazine|20150429|040100  |59.0    |
```

EP0000000211654	20/20	Newsmagazine	20150429	050100	59.0
EP0000000211659	20/20	Newsmagazine	20150411	020100	59.0
EP0000000211659	20/20	Newsmagazine	20150411	030100	59.0
EP0000000211659	20/20	Newsmagazine	20150411	040100	59.0
EP0000000211659	20/20	Newsmagazine	20150411	050100	59.0
EP0000000211661	20/20	Newsmagazine	20150418	020100	59.0
EP0000000211661	20/20	Newsmagazine	20150418	030100	59.0
EP0000000211661	20/20	Newsmagazine	20150418	040100	59.0
EP0000000211661	20/20	Newsmagazine	20150418	050100	59.0
EP0000000211665	20/20	Newsmagazine	20150502	020100	59.0
EP0000000211665	20/20	Newsmagazine	20150502	030100	59.0
EP0000000211665	20/20	Newsmagazine	20150502	040100	59.0
EP0000000211665	20/20	Newsmagazine	20150502	050100	59.0
EP0000000211666	20/20	Newsmagazine	20150509	020100	59.0
EP0000000211666	20/20	Newsmagazine	20150509	030100	59.0
EP0000000211666	20/20	Newsmagazine	20150509	040100	59.0
EP0000000211666	20/20	Newsmagazine	20150509	050100	59.0
EP0000000211667	20/20	Newsmagazine	20150516	020100	59.0
EP0000000211667	20/20	Newsmagazine	20150516	030100	59.0
EP0000000211667	20/20	Newsmagazine	20150516	040100	59.0
EP0000000211667	20/20	Newsmagazine	20150516	050100	59.0
EP0000000211669	20/20	Newsmagazine	20150530	020000	60.0
EP0000000211669	20/20	Newsmagazine	20150530	030000	60.0
EP0000000211669	20/20	Newsmagazine	20150530	040000	60.0
EP0000000211669	20/20	Newsmagazine	20150530	050000	60.0
EP0000000211670	20/20	Newsmagazine	20150610	050100	59.0
EP0000000211670	20/20	Newsmagazine	20150606	020000	60.0
EP0000000211670	20/20	Newsmagazine	20150606	030000	60.0
EP0000000211670	20/20	Newsmagazine	20150606	040000	60.0
EP0000000211670	20/20	Newsmagazine	20150606	050000	60.0
EP0000000211672	20/20	Newsmagazine	20150620	020000	60.0
EP0000000211672	20/20	Newsmagazine	20150620	030000	60.0
EP0000000211672	20/20	Newsmagazine	20150620	040000	60.0
EP0000000211672	20/20	Newsmagazine	20150620	050000	60.0
EP0000000211675	20/20	Newsmagazine	20150718	020000	60.0
EP0000000211675	20/20	Newsmagazine	20150718	030000	60.0
EP0000000211675	20/20	Newsmagazine	20150718	040000	60.0
EP0000000211675	20/20	Newsmagazine	20150718	050000	60.0
EP0000000211676	20/20	Newsmagazine	20150725	020000	60.0
EP0000000211676	20/20	Newsmagazine	20150725	030000	60.0
EP0000000211676	20/20	Newsmagazine	20150725	040000	60.0
EP0000000211676	20/20	Newsmagazine	20150725	050000	60.0
EP0000000211679	20/20	Newsmagazine	20150815	020000	60.0
EP0000000211679	20/20	Newsmagazine	20150815	023000	30.0
EP0000000211679	20/20	Newsmagazine	20150815	040000	60.0
EP0000000211679	20/20	Newsmagazine	20150815	050000	60.0
EP0000000211679	20/20	Newsmagazine	20150815	050700	60.0
EP0000000211679	20/20	Newsmagazine	20150815	063600	60.0
EP0000000211679	20/20	Newsmagazine	20150815	080000	60.0
EP0000000211679	20/20	Newsmagazine	20150815	020000	60.0
EP0000000211679	20/20	Newsmagazine	20150815	023000	30.0
EP0000000211679	20/20	Newsmagazine	20150815	040000	60.0
EP0000000211679	20/20	Newsmagazine	20150815	050000	60.0
EP0000000211679	20/20	Newsmagazine	20150815	050700	60.0
EP0000000211679	20/20	Newsmagazine	20150815	063600	60.0
EP0000000211679	20/20	Newsmagazine	20150815	080000	60.0
EP0000000211679	20/20	Newsmagazine	20150816	210000	60.0
EP0000000211680	20/20	Newsmagazine	20150823	053500	60.0
EP0000000211680	20/20	Newsmagazine	20150822	020000	60.0
EP0000000211680	20/20	Newsmagazine	20150822	030000	60.0
EP0000000211680	20/20	Newsmagazine	20150822	040000	60.0
EP0000000211680	20/20	Newsmagazine	20150822	050000	60.0
EP0000000211680	20/20	Newsmagazine	20150822	050700	60.0
EP0000000211680	20/20	Newsmagazine	20150822	080100	59.0
EP0000000211680	20/20	Newsmagazine	20150822	020000	60.0
EP0000000211680	20/20	Newsmagazine	20150822	040000	60.0
EP0000000211680	20/20	Newsmagazine	20150822	050000	60.0
EP0000000211680	20/20	Newsmagazine	20150822	050700	60.0
EP0000000211680	20/20	Newsmagazine	20150822	080100	59.0
EP0000000211680	20/20	Newsmagazine	20150823	053500	60.0
EP0000000211681	20/20	Newsmagazine	20150829	020000	60.0
EP0000000211681	20/20	Newsmagazine	20150829	040000	60.0

```
|EP000000211681|20/20|Newsmagazine|20150829|050000  |60.0    |
|EP000000211681|20/20|Newsmagazine|20150829|050700  |60.0    |
|EP000000211681|20/20|Newsmagazine|20150829|063600  |60.0    |
|EP000000211681|20/20|Newsmagazine|20150829|080000  |60.0    |
|EP000000211681|20/20|Newsmagazine|20150829|090100  |59.0    |
|EP000000211681|20/20|Newsmagazine|20150830|053500  |60.0    |
|EP000000211681|20/20|Newsmagazine|20150830|053500  |60.0    |
|EP000000211681|20/20|Newsmagazine|20150829|020000  |60.0    |
|EP000000211681|20/20|Newsmagazine|20150829|030000  |60.0    |
|EP000000211681|20/20|Newsmagazine|20150829|040000  |60.0    |
|EP000000211681|20/20|Newsmagazine|20150829|050000  |60.0    |
|EP000000211681|20/20|Newsmagazine|20150829|050700  |60.0    |
|EP000000211681|20/20|Newsmagazine|20150829|063600  |60.0    |
|EP000000211681|20/20|Newsmagazine|20150829|080000  |60.0    |
|EP000000211681|20/20|Newsmagazine|20150829|090100  |59.0    |
|EP000000211682|20/20|Newsmagazine|20150905|020000  |60.0    |
|EP000000211682|20/20|Newsmagazine|20150905|030000  |60.0    |
|EP000000211682|20/20|Newsmagazine|20150905|040000  |60.0    |
|EP000000211682|20/20|Newsmagazine|20150905|050000  |60.0    |
|EP000000211683|20/20|Newsmagazine|20150912|020100  |59.0    |
|EP000000211683|20/20|Newsmagazine|20150912|030100  |59.0    |
|EP000000211683|20/20|Newsmagazine|20150912|040100  |59.0    |
|EP000000211683|20/20|Newsmagazine|20150912|050100  |59.0    |
|EP000000211684|20/20|Newsmagazine|20150919|020100  |59.0    |
|EP000000211684|20/20|Newsmagazine|20150919|020200  |59.0    |
|EP000000211684|20/20|Newsmagazine|20150919|030100  |59.0    |
|EP000000211684|20/20|Newsmagazine|20150919|040100  |59.0    |
|EP000000211684|20/20|Newsmagazine|20150919|050100  |59.0    |
|EP000000211685|20/20|Newsmagazine|20150926|020100  |59.0    |
|EP000000211685|20/20|Newsmagazine|20150926|030100  |59.0    |
|EP000000211685|20/20|Newsmagazine|20150926|040100  |59.0    |
|EP000000211685|20/20|Newsmagazine|20150926|050100  |59.0    |
|EP000000211686|20/20|Newsmagazine|20151003|020100  |59.0    |
|EP000000211686|20/20|Newsmagazine|20151003|030100  |59.0    |
|EP000000211686|20/20|Newsmagazine|20151003|040100  |59.0    |
|EP000000211686|20/20|Newsmagazine|20151003|050100  |59.0    |
|EP000000211688|20/20|Newsmagazine|20151017|020100  |59.0    |
|EP000000211688|20/20|Newsmagazine|20151017|030100  |59.0    |
|EP000000211688|20/20|Newsmagazine|20151017|040100  |59.0    |
|EP000000211688|20/20|Newsmagazine|20151017|050100  |59.0    |
|EP000000211689|20/20|Newsmagazine|20151024|020100  |59.0    |
|EP000000211689|20/20|Newsmagazine|20151024|030100  |59.0    |
|EP000000211689|20/20|Newsmagazine|20151024|040100  |59.0    |
|EP000000211689|20/20|Newsmagazine|20151024|050100  |59.0    |
|EP000000211690|20/20|Newsmagazine|20151031|020100  |59.0    |
|EP000000211690|20/20|Newsmagazine|20151031|030100  |59.0    |
|EP000000211690|20/20|Newsmagazine|20151031|040100  |59.0    |
|EP000000211690|20/20|Newsmagazine|20151031|050100  |59.0    |
|EP000000211691|20/20|Newsmagazine|20151107|030100  |59.0    |
|EP000000211691|20/20|Newsmagazine|20151107|040100  |59.0    |
+--------------+-----+------------+--------+--------+--------+
only showing top 150 rows
```