PRESENTATION

LEAGUE OF LEGENDS

APACHE SPARK

Yan Zhu

LEAGUE OF LEGENDS

League of Legends is a team-based strategy game where two teams of five powerful champions face off to destroy the other's base.

WHY LEAGUE ?

1.MY FAVORITE GAME

2.INSPIRED BY COLONEL KI

https://www.mindshareworld.com/china/work/kfc-colonel-ki

PROJECT PURPOSE

PURPOSE 1

The goal of this project is to create an ETL process to extract data from semi-structured data and download CSV from spark. The purpose is to help players better analyze game data.

MORE

PURPOSE2

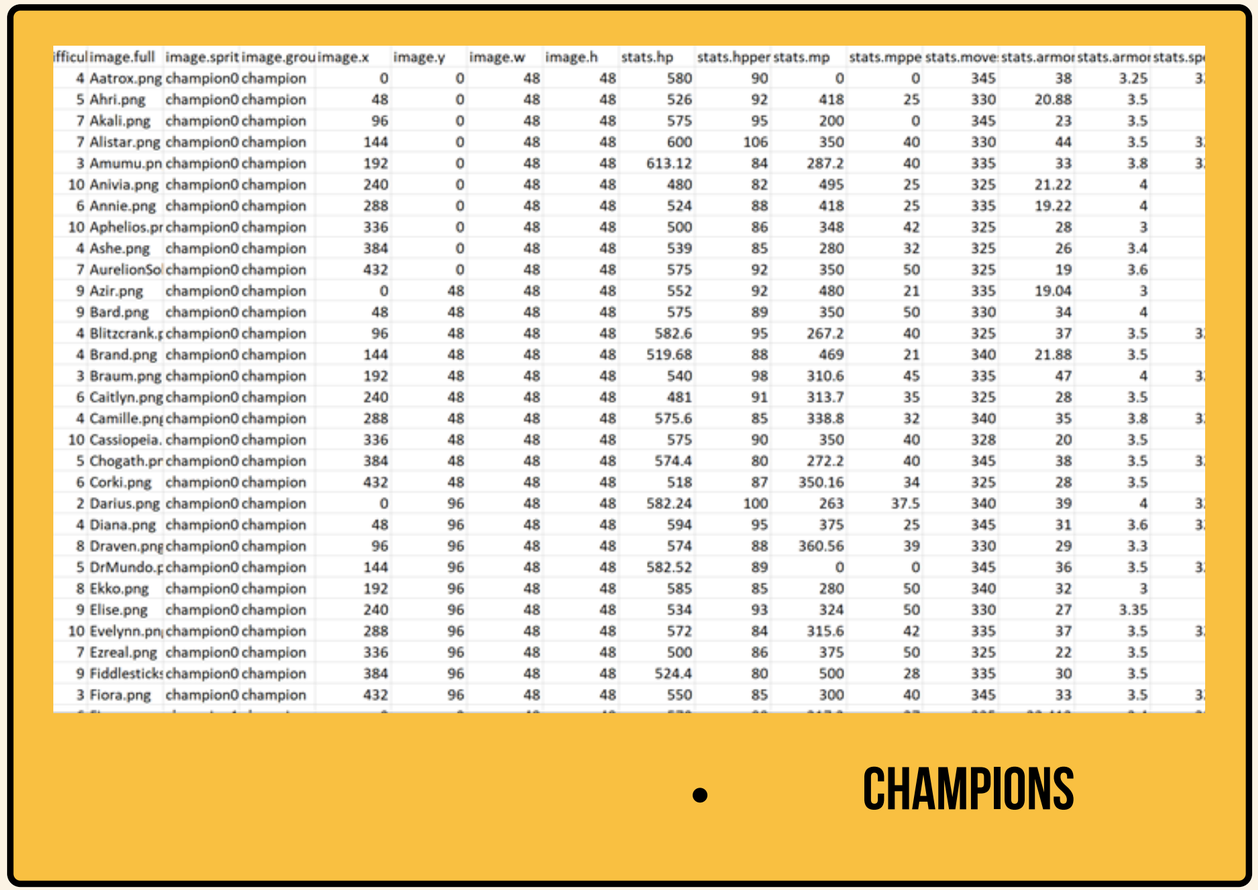
2.From structured and semi-structured data, find the most used items, highest win rate champions,etc.

MORE

DATASET INTRODUCTION:

RIOT_CHAMPION.CSV

It contains all the champions information, including ID, name, backstory, skills, movement speed, etc



ifficul	image.full	image.sprit	image.grou	image.x	image.y	image.w	image.h	stats.hp	stats.hpper	stats.mp	stats.mppe	stats.move	stats.armor	stats.armor	stats.sp
4	Aatrox.png	champion0	champion	0	0	48	48	580	90	0	0	345	38	3.25	3
5	Ahri.png	champion0	champion	48	0	48	48	526	92	418	25	330	20.88	3.5	
7	Akali.png	champion0	champion	96	0	48	48	575	95	200	0	345	23	3.5	
7	Alistar.png	champion0	champion	144	0	48	48	600	106	350	40	330	44	3.5	3
3	Amumu.pn	champion0	champion	192	0	48	48	613.12	84	287.2	40	335	33	3.8	3
10	Anivia.png	champion0	champion	240	0	48	48	480	82	495	25	325	21.22	4	
6	Annie.png	champion0	champion	288	0	48	48	524	88	418	25	335	19.22	4	
10	Aphelios.pr	champion0	champion	336	0	48	48	500	86	348	42	325	28	3	
4	Ashe.png	champion0	champion	384	0	48	48	539	85	280	32	325	26	3.4	
7	AurelionSo	champion0	champion	432	0	48	48	575	92	350	50	325	19	3.6	
9	Azir.png	champion0	champion	0	48	48	48	552	92	480	21	335	19.04	3	
9	Bard.png	champion0	champion	48	48	48	48	575	89	350	50	330	34	4	
4	Blitzcrank.p	champion0	champion	96	48	48	48	582.6	95	267.2	40	325	37	3.5	3
4	Brand.png	champion0	champion	144	48	48	48	519.68	88	469	21	340	21.88	3.5	
3	Braum.png	champion0	champion	192	48	48	48	540	98	310.6	45	335	47	4	3
6	Caitlyn.png	champion0	champion	240	48	48	48	481	91	313.7	35	325	28	3.5	
4	Camille.pn	champion0	champion	288	48	48	48	575.6	85	338.8	32	340	35	3.8	3
10	Cassiopeia.	champion0	champion	336	48	48	48	575	90	350	40	328	20	3.5	
5	Chogath.pr	champion0	champion	384	48	48	48	574.4	80	272.2	40	345	38	3.5	3
6	Corki.png	champion0	champion	432	48	48	48	518	87	350.16	34	325	28	3.5	
2	Darius.png	champion0	champion	0	96	48	48	582.24	100	263	37.5	340	39	4	3
4	Diana.png	champion0	champion	48	96	48	48	594	95	375	25	345	31	3.6	3
8	Draven.pn	champion0	champion	96	96	48	48	574	88	360.56	39	330	29	3.3	
5	DrMundo.p	champion0	champion	144	96	48	48	582.52	89	0	0	345	36	3.5	3
8	Ekko.png	champion0	champion	192	96	48	48	585	85	280	50	340	32	3	
9	Elise.png	champion0	champion	240	96	48	48	534	93	324	50	330	27	3.35	
10	Evelynn.pn	champion0	champion	288	96	48	48	572	84	315.6	42	335	37	3.5	3
7	Ezreal.png	champion0	champion	336	96	48	48	500	86	375	50	325	22	3.5	
9	Fiddlestick	champion0	champion	384	96	48	48	524.4	80	500	28	335	30	3.5	
3	Fiora.png	champion0	champion	432	96	48	48	550	85	300	40	345	33	3.5	3

- **CHAMPIONS**

DATASET INTRODUCTION:

RIOT_ITEMS.CSV

It contains all item information, including ID, name, damage,price, etc.

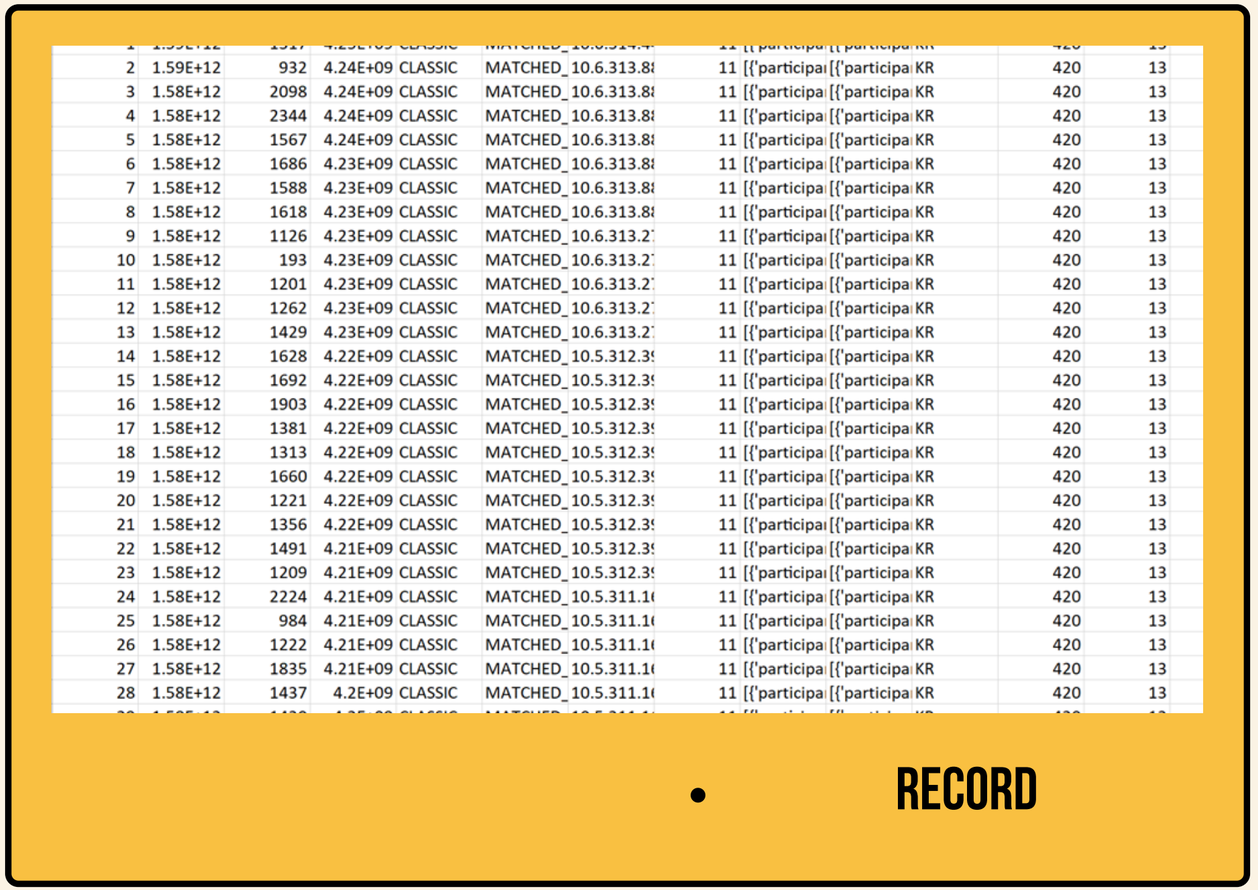
0	1026	Blasting W	['3135', '30	Moderately	850	595	SpellDamage
0	1027	Sapphire Cr	['3057', '30	Increases M	350	245	Mana
0	1028	Ruby Cryst	['1011', '30	Increases H	400	280	Health
0	1029	Cloth Armo	['3047', '10	Slightly incr	300	210	Armor
0	1031	Chain Vest	['3068', '31	Greatly incr	500	560	Armor
0	1033	Null-Magic	['3111', '32	Slightly incr	450	315	SpellBlock
0	1036	Long Sworc	['3077', '31	Slightly incr	350	245	Damage
0	1037	Pickaxe	['3124', '30	Moderately	875	613	Damage
0	1038	B. F. Sword	['3095', '30	Greatly incr	1300	910	Damage
0	1039	Hunter's Ta	['3706', '37	Provides da	350	245	LifeSteal
0	1041	Hunter's M	['3706', '37	Provides da	350	245	LifeSteal
0	1042	Dagger	['1043', '30	Slightly incr	300	210	AttackSpeed
0	1043	Recurve Bo	['3091', '31	Greatly incr	400	700	AttackSpeed
0	1052	Amplifying	['3108', '31	Slightly incr	435	305	SpellDamage
0	1053	Vampiric Sc	['3072', '30	Basic attacl	550	630	Damage
0	1054	Doran's Shield		Good defer	450	180	Health
0	1055	Doran's Blade		Good starti	450	180	Damage
0	1056	Doran's Ring		Good starti	400	160	Health
0	1057	Negatron C	['3091', '31	Moderately	270	504	SpellBlock

- ITEMS

DATASET INTRODUCTION:

MATCH_RECORD.CSV:

The match record contains all the information related to the match, such as duration, players, wins and losses, etc

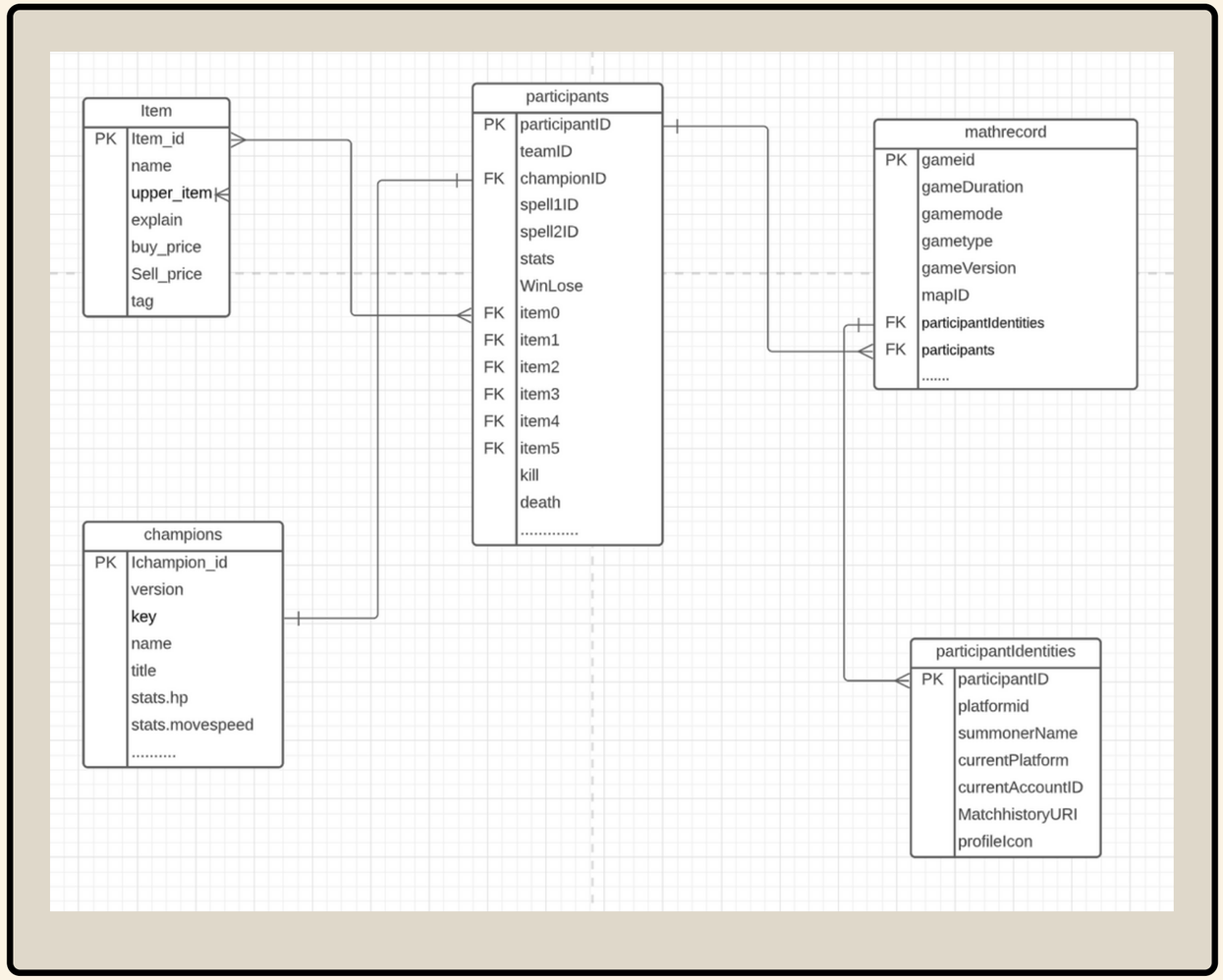


2	1.59E+12	932	4.24E+09	CLASSIC	MATCHED_10.6.313.8	11	[{'participa	KR	420	13
3	1.58E+12	2098	4.24E+09	CLASSIC	MATCHED_10.6.313.8	11	[{'participa	KR	420	13
4	1.58E+12	2344	4.24E+09	CLASSIC	MATCHED_10.6.313.8	11	[{'participa	KR	420	13
5	1.58E+12	1567	4.24E+09	CLASSIC	MATCHED_10.6.313.8	11	[{'participa	KR	420	13
6	1.58E+12	1686	4.23E+09	CLASSIC	MATCHED_10.6.313.8	11	[{'participa	KR	420	13
7	1.58E+12	1588	4.23E+09	CLASSIC	MATCHED_10.6.313.8	11	[{'participa	KR	420	13
8	1.58E+12	1618	4.23E+09	CLASSIC	MATCHED_10.6.313.8	11	[{'participa	KR	420	13
9	1.58E+12	1126	4.23E+09	CLASSIC	MATCHED_10.6.313.2	11	[{'participa	KR	420	13
10	1.58E+12	193	4.23E+09	CLASSIC	MATCHED_10.6.313.2	11	[{'participa	KR	420	13
11	1.58E+12	1201	4.23E+09	CLASSIC	MATCHED_10.6.313.2	11	[{'participa	KR	420	13
12	1.58E+12	1262	4.23E+09	CLASSIC	MATCHED_10.6.313.2	11	[{'participa	KR	420	13
13	1.58E+12	1429	4.23E+09	CLASSIC	MATCHED_10.6.313.2	11	[{'participa	KR	420	13
14	1.58E+12	1628	4.22E+09	CLASSIC	MATCHED_10.5.312.3	11	[{'participa	KR	420	13
15	1.58E+12	1692	4.22E+09	CLASSIC	MATCHED_10.5.312.3	11	[{'participa	KR	420	13
16	1.58E+12	1903	4.22E+09	CLASSIC	MATCHED_10.5.312.3	11	[{'participa	KR	420	13
17	1.58E+12	1381	4.22E+09	CLASSIC	MATCHED_10.5.312.3	11	[{'participa	KR	420	13
18	1.58E+12	1313	4.22E+09	CLASSIC	MATCHED_10.5.312.3	11	[{'participa	KR	420	13
19	1.58E+12	1660	4.22E+09	CLASSIC	MATCHED_10.5.312.3	11	[{'participa	KR	420	13
20	1.58E+12	1221	4.22E+09	CLASSIC	MATCHED_10.5.312.3	11	[{'participa	KR	420	13
21	1.58E+12	1356	4.22E+09	CLASSIC	MATCHED_10.5.312.3	11	[{'participa	KR	420	13
22	1.58E+12	1491	4.21E+09	CLASSIC	MATCHED_10.5.312.3	11	[{'participa	KR	420	13
23	1.58E+12	1209	4.21E+09	CLASSIC	MATCHED_10.5.312.3	11	[{'participa	KR	420	13
24	1.58E+12	2224	4.21E+09	CLASSIC	MATCHED_10.5.311.1	11	[{'participa	KR	420	13
25	1.58E+12	984	4.21E+09	CLASSIC	MATCHED_10.5.311.1	11	[{'participa	KR	420	13
26	1.58E+12	1222	4.21E+09	CLASSIC	MATCHED_10.5.311.1	11	[{'participa	KR	420	13
27	1.58E+12	1835	4.21E+09	CLASSIC	MATCHED_10.5.311.1	11	[{'participa	KR	420	13
28	1.58E+12	1437	4.2E+09	CLASSIC	MATCHED_10.5.311.1	11	[{'participa	KR	420	13

- RECORD

MATCH RECORD

These two sets of data contain all the data of the 10 players and are stored in match-record as an array structure. So it's going to be very difficult for us to extract this semi-structured data in SQL.



ERD

This is a simple ERD diagram designed to help us understand the relationship between match data.

```python
def jsonToDataFrame(json_input, schema=None):
    # SparkSessions are available with Spark 2.0+
    reader = spark.read
    if schema:
        reader.schema(schema)
    return reader.json(sc.parallelize([json_input]))
```

```python
def flatten_df(nested_df):
    stack = [((), nested_df)]
    columns = []

    while len(stack) > 0:
        parents, df = stack.pop()

        flat_cols = [
            col(".".join(parents +
(c[0],))).alias("_".join(parents + (c[0],)))
            for c in df.dtypes
            if c[1][:6] != "struct"
        ]

        nested_cols = [
            c[0]
            for c in df.dtypes
            if c[1][:6] == "struct"
        ]

        columns.extend(flat_cols)

        for nested_col in nested_cols:
            projected_df = df.select(nested_col + ".*")
            stack.append((parents + (nested_col,), projected_df))

    return nested_df.select(columns)
```

```python
def transform_colum(df, column):
    df_select = df.select(col(column))
    str_ = df_select.take(1)[0].asDict()[column]
    df_select = jsonToDataFrame(json.dumps(eval(str_)))
    schema = df_select.schema

    eval_column = udf(lambda x : eval(x), ArrayType(schema))

    df = df.withColumn(column, eval_column(col(column)))

    return df, schema
```

https://docs.databricks.com/_static/notebooks/transform-complex-data-types-scala.html

---

FUNCTION 1

FUNCTION2

FUNCTION3

EXAMPLE

JSON TO DATA FRAME

```
val events = jsonToDataFrame("""
{
  "a": {
     "b": 1,
     "c": 2
  }
}
""")

display(events.select("a.*"))
```

	b	c
1	1	2

select all of the subfields in a struct.

EXAMPLE TRANSFORM COLUM

BEFORE

```
root
 |-- _c0: integer (nullable = true)
 |-- gameCreation: double (nullable = true)
 |-- gameDuration: double (nullable = true)
 |-- gameId: double (nullable = true)
 |-- gameMode: string (nullable = true)
 |-- gameType: string (nullable = true)
 |-- gameVersion: string (nullable = true)
 |-- mapId: double (nullable = true)
 |-- participantIdentities: string (nullable = true)
 |-- participants: string (nullable = true)
 |-- platformId: string (nullable = true)
 |-- queueId: double (nullable = true)
 |-- seasonId: double (nullable = true)
```

AFTER

```
|-- gameType: string (nullable = true)
|-- gameVersion: string (nullable = true)
|-- mapId: double (nullable = true)
|-- participantIdentities: array (nullable = true)
|    |-- element: struct (containsNull = true)
|    |    |-- participantId: long (nullable = true)
|    |    |-- player: struct (nullable = true)
|    |    |    |-- accountId: string (nullable = true)
|    |    |    |-- currentAccountId: string (nullable = true)
|    |    |    |-- currentPlatformId: string (nullable = true)
|    |    |    |-- matchHistoryUri: string (nullable = true)
|    |    |    |-- platformId: string (nullable = true)
|    |    |    |-- profileIcon: long (nullable = true)
|    |    |    |-- summonerId: string (nullable = true)
|    |    |    |-- summonerName: string (nullable = true)
|-- participants: array (nullable = true)
|    |-- element: struct (containsNull = true)
|    |    |-- championId: long (nullable = true)
|    |    |-- participantId: long (nullable = true)
|    |    |-- spell1Id: long (nullable = true)
```

EXAMPLE

FLATTEN FUNCTION

```
root
 |-- _c0: integer (nullable
 |-- gameCreation: double (n
 |-- gameDuration: double (n
 |-- gameId: double (nullabl
 |-- gameMode: string (nulla
 |-- gameType: string (nulla
 |-- gameVersion: string (nu
 |-- mapId: double (nullable
 |-- participantIdentities:
 |-- participants: string (n
 |-- platformId: string (nul
 |-- queueId: double (nullab
 |-- seasonId: double (nulla
```

BEFORE

```
root
 |-- _c0: integer (nullable = true)
 |-- gameCreation: double (nullable = true
 |-- gameDuration: double (nullable = true
 |-- gameId: double (nullable = true)
 |-- gameMode: string (nullable = true)
 |-- gameType: string (nullable = true)
 |-- gameVersion: string (nullable = true)
 |-- mapId: double (nullable = true)
 |-- platformId: string (nullable = true)
 |-- queueId: double (nullable = true)
 |-- seasonId: double (nullable = true)
 |-- participants_info: struct (nullable =
 |    |-- ids: struct (nullable = true)
 |    |    |-- championId: long (nullable
 |    |    |-- participantId: long (nullab
 |    |    |-- spell1Id: long (nullable =
 |    |    |-- spell2Id: long (nullable =
 |    |    |-- stats: struct (nullable = t
 |    |    |    |-- assists: long (nullabl
 |    |    |    |-- champLevel: long (null
 |    |    |    |-- combatPlayerScore: lon
```

AFTER

FUNCTION

Struct_type: PySpark provides from pyspark.sql.types import StructType class to define the structure of the DataFrame.

sc.parallelize(): When spark parallelize method is applied on a Collection (with elements), a new distributed data set is created with specified number of partitions and the elements of the collection are copied to the distributed dataset

StructField(): provides spark.sql.types.StructField class to define the column name(String), column type.

Arraytype(): is used to define an array data type column on DataFrame

distinct().collect(): distinct value in the data frame column

lambda(): A lambda function is a small anonymous function

explode():Returns a new row for each element in the given array

```
summonerchampWR = sqlContext.sql("""
    SELECT victorys.id as user, victorys.name_champion as name_champion, victorys.won_matches,
    matches.total_matches, victorys.won_matches/matches.total_matches as win_rate
    FROM
    (SELECT record.participants_info_info_player_accountId as id, record.name_champion as name_champion, COUNT(DISTINCT(record.matchId))
    as     won_matches
    FROM record
    WHERE record.participants_info_ids_stats_win == true
    GROUP BY record.participants_info_info_player_accountId, record.name_champion) as victorys
    LEFT JOIN (SELECT record.participants_info_info_player_accountId as id, record.name_champion as name_champion,
    COUNT(DISTINCT(record.matchId)) as total_matches
    FROM record
    GROUP BY record.participants_info_info_player_accountId, record.name_champion) as matches
    ON victorys.id=matches.id AND victorys.name_champion = matches.name_champion
```

	user	name_champion	won_matches	total_matches	win_ra
1	zpdpHmvHki76wDo0Tkrhnd7sYyDWaTCD48UmFSvcqETlALU	Miss Fortune	1	1	1
2	zpdpHmvHki76wDo0Tkrhnd7sYyDWaTCD48UmFSvcqETlALU	Ashe	1	1	1
3	zpdpHmvHki76wDo0Tkrhnd7sYyDWaTCD48UmFSvcqETlALU	Kalista	1	1	1
4	zpdpHmvHki76wDo0Tkrhnd7sYyDWaTCD48UmFSvcqETlALU	Kai'Sa	1	1	1
5	zkYtjdZq7DuYLDNAt0TLl_jAiHLf_BuQkIkDlS3XRYbz	Shaco	2	3	0.6666
6	zkYtjdZq7DuYLDNAt0TLl_jAiHLf_BuQkIkDlS3XRYbz	Taliyah	1	1	1
7	zkYtjdZq7DuYLDNAt0TLl_jAiHLf_BuQkIkDlS3XRYbz	Ivern	2	2	1

Truncated results, showing first 1000 rows.

QUERY THE WIN RATE OF ALL CHAMPIONS PLAYED BY EACH PLAYER

```
champwinrate = sqlContext.sql("""
    SELECT victorys.name_champion as name_champion, victorys.won_matches, matches.total_matches,
    victorys.won_matches/matches.total_matches as win_rate
    FROM
    (SELECT record.name_champion as name_champion, COUNT(DISTINCT(record.matchId)) as won_matches
    FROM record
    WHERE record.participants_info_ids_stats_win == true
    GROUP BY record.name_champion) as victorys
    LEFT JOIN (SELECT record.name_champion as name_champion, COUNT(DISTINCT(record.matchId)) as total_matches
    FROM record
    GROUP BY record.name_champion) as matches
    ON victorys.name_champion = matches.name_champion
```

	name_champion	won_matches	total_matches	win_rate
1	Lee Sin	175	359	0.4874651810584958 4
2	Sett	186	340	0.5470588235294118
3	Ezreal	135	289	0.4671280276816609
4	Miss Fortune	140	283	0.4946996466431095 6

QUERY THE WIN RATE OF ALL CHAMPIONS PLAYED BY EACH PLAYER

```
championFI = sqlContext.sql("""
    SELECT FI.championName, FI.FI_name as first_item, COUNT(FI.FI_name) as total_matches
    FROM
    (SELECT record.name_champion as championName, record.item0  as FI_name
    FROM record
    WHERE record.participants_info_info_player_accountId
    IN (
    SELECT summonerchampWR.user
    FROM summonerchampWR \
    WHERE summonerchampWR.win_rate > 0.5
```

	championName	first_item	total_matches
1	Aatrox	Doran's Shield	37
2	Ahri	Hextech GLP-800	14
3	Akali	Hextech Gunblade	35

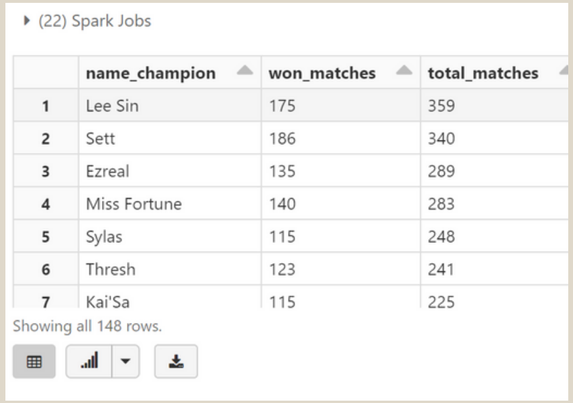
QUERY THE FIRST ITEM PURCHASED BY EACH CHAMPION WITH A WIN RATE GREATER THAN 0.5

SAVE FILE TO LOCAL

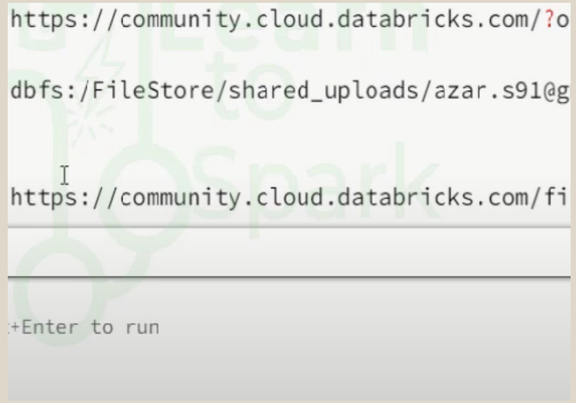
1.Using display()

2.using web URL

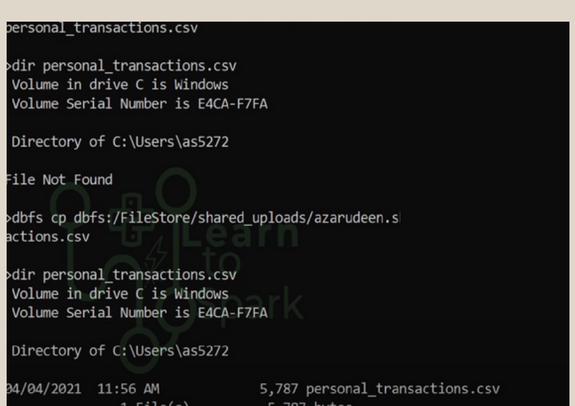
3.using databricks CLI



METHOD 1



METHOD 2



METHOD 3

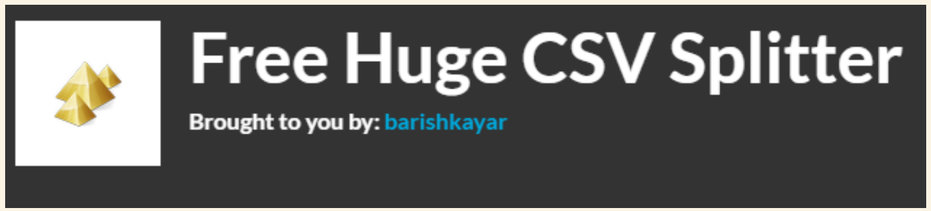
challenge

1. The first challenge is that the data is too large, ten thousands of game records are stored in CSV, and each line of game records contains a large amount of game data, resulting in a file size of about 7 g, making it impossible to open CSV.

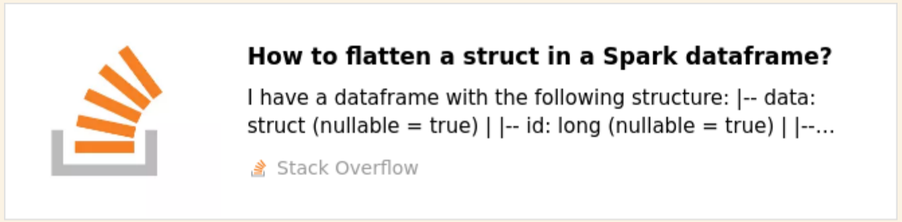
2. After I finished data transform, the data was story into array format.

3. The data stored in the file exists in the way of displaying 10 players in one row, but for the purpose of query or clear display of data, we need to map the ten rows to 10 players.

Solution

1. 

https://sourceforge.net/p/splitcsv/wiki/Home/

2. 

3.

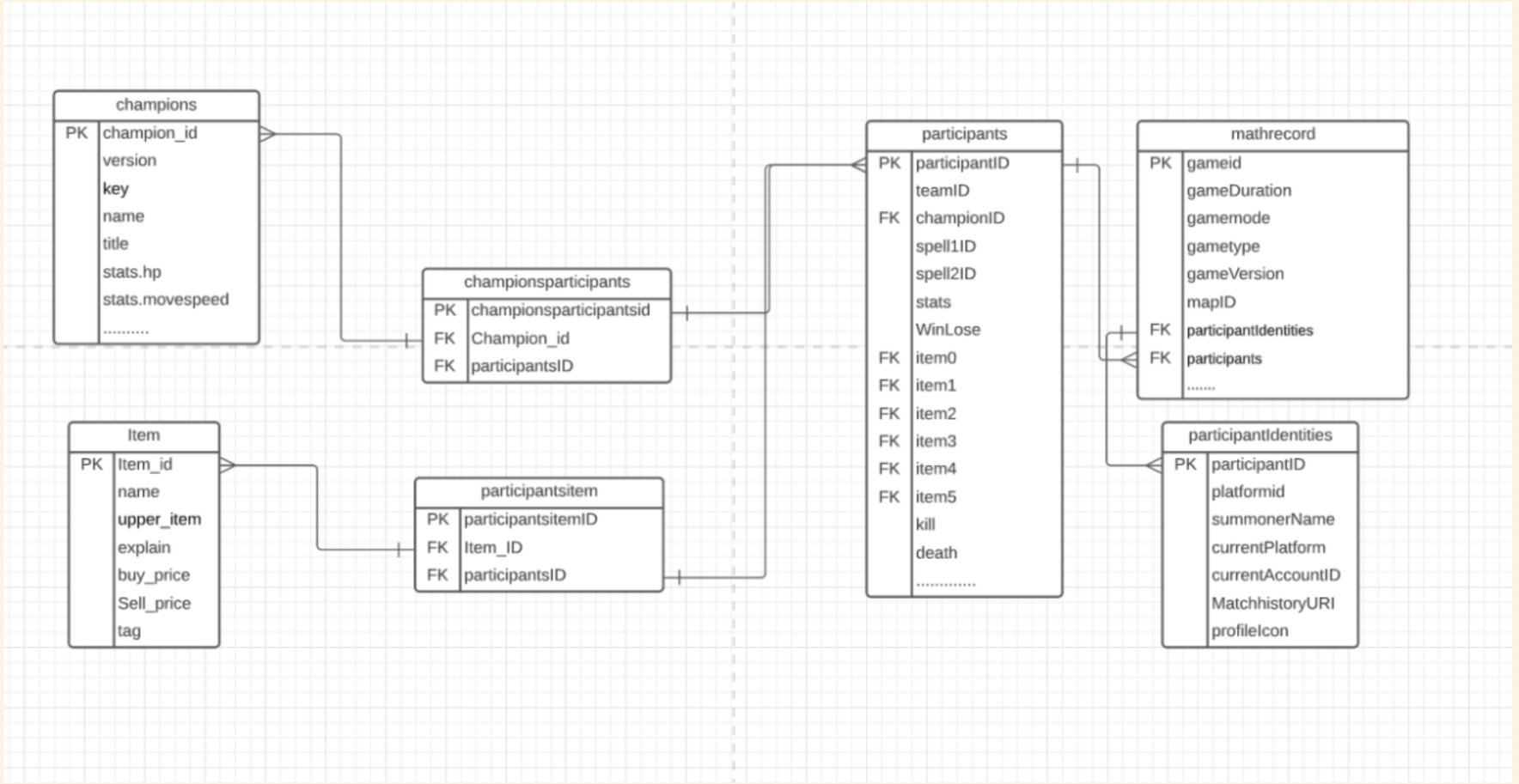
```
record = record.withColumn("participants_info", explode("participants_info")
```

```
10

from pyspark.sql.functions import col


def flatten_df(nested_df):
    stack = [((), nested_df)]
    columns = []

    while len(stack) > 0:
        parents, df = stack.pop()

        flat_cols = [
            col(".".join(parents + (c[0],))).alias("_".join(parents + (c[0],)))
            for c in df.dtypes
            if c[1][:6] != "struct"
        ]

        nested_cols = [
            c[0]
            for c in df.dtypes
            if c[1][:6] == "struct"
        ]

        columns.extend(flat_cols)

        for nested_col in nested_cols:
            projected_df = df.select(nested_col + ".*")
            stack.append((parents + (nested_col,), projected_df))
```

https://stackoverflow.com/questions/38753898/how-to-flatten-a-struct-in-a-spark-dataframe

IMPROVEMENT

ERD:




Thank you!