data-engineer-lgde-day2-answer

September 8, 2021

1 1. LGDE.com 2 ( )

Alt+Enter	+	
Shift+Enter	+	
Ctrl+Enter	+	
Ctrl+/		Shift
Ctrl+s		-

(Windows)

1.0.1

- Code, Markdown, Raw 3 , ***Code***
- Menu ***Kernel - Interrupt Kernel***
- Menu ***Kernel - Restart Kernel..***

1.1 5.

1.1.1 5-1.

[1]:
```
from pyspark.sql import *
from pyspark.sql.functions import *
from pyspark.sql.types import *
from IPython.display import display, display_pretty, clear_output, JSON

spark = (
    SparkSession
    .builder
    .config("spark.sql.session.timeZone", "Asia/Seoul")
    .getOrCreate()
)

#
spark.conf.set("spark.sql.repl.eagerEval.enabled", True) # display enabled
spark.conf.set("spark.sql.repl.eagerEval.truncate", 100) # display output␣
↪columns size
```

```
#
home_jovyan = "/home/jovyan"
work_data = f"{home_jovyan}/work/data"
work_dir=!pwd
work_dir = work_dir[0]

#
spark.conf.set("spark.sql.shuffle.partitions", 5) # the number of partitions to␣
 ↪use when shuffling data for joins or aggregations.
spark.conf.set("spark.sql.streaming.forceDeleteTempCheckpointLocation", "true")
spark
```

```
21/09/08 13:51:40 WARN NativeCodeLoader: Unable to load native-hadoop library
for your platform… using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
21/09/08 13:51:42 WARN Utils: Service 'SparkUI' could not bind on port 4040.
Attempting port 4041.
```

[1]:
```
<pyspark.sql.session.SparkSession at 0x7f41142415e0>
```

[2]:
```
user26 = spark.read.parquet("user/20201026")
user26.printSchema()
user26.show(truncate=False)
display(user26)
```

```
root
 |-- u_id: integer (nullable = true)
 |-- u_name: string (nullable = true)
 |-- u_gender: string (nullable = true)
 |-- u_signup: integer (nullable = true)
```

```
+----+----------+--------+--------+
|u_id|u_name    |u_gender|u_signup|
+----+----------+--------+--------+
|1   |      |       |20201025|
|2   |    |       |20201025|
|3   |      |       |20201025|
|4   |      |       |20201025|
|5   |   |       |20201025|
|6   |    |       |20201026|
```

```
|7   |    |       |20201026|
|8   |    |       |20201026|
|9   |    |       |20201026|
+----+----------+--------+--------+


+----+----------+--------+--------+
|u_id|    u_name|u_gender|u_signup|
+----+----------+--------+--------+
|   1|      |       |20201025|
|   2|    |       |20201025|
|   3|      |       |20201025|
|   4|      |       |20201025|
|   5|   |       |20201025|
|   6|    |       |20201026|
|   7|    |       |20201026|
|   8|    |       |20201026|
|   9|    |       |20201026|
+----+----------+--------+--------+
```

[3]:
```
purchase26 = spark.read.parquet("purchase/20201026")
purchase26.printSchema()
purchase26.show(truncate=False)
display(purchase26)
```

```
root
 |-- p_time: string (nullable = true)
 |-- p_uid: integer (nullable = true)
 |-- p_id: integer (nullable = true)
 |-- p_name: string (nullable = true)
 |-- p_amount: integer (nullable = true)

+----------+-----+----+-----------+--------+
|p_time    |p_uid|p_id|p_name     |p_amount|
+----------+-----+----+-----------+--------+
|1603571550|1    |2000|LG DIOS    |2000000 |
|1603614755|1    |2000|LG Gram    |1800000 |
|1603593500|2    |2001|LG Cyon    |1400000 |
|1603572155|3    |2002|LG TV      |1000000 |
|1603585955|5    |2004|LG Gram    |3500000 |
|1603586155|5    |2004|LG TV      |2500000 |
|1603651550|1    |2001|LG Cyon    |1400000 |
|1603652155|5    |2002|LG TV      |1000000 |
|1603674500|6    |2003|LG Computer|4500000 |
|1603665955|7    |2004|LG Gram    |3500000 |
|1603666155|9    |2004|LG TV      |2500000 |
+----------+-----+----+-----------+--------+
```

```
+----------+-----+----+----------+--------+
|    p_time|p_uid|p_id|    p_name|p_amount|
+----------+-----+----+----------+--------+
|1603571550|    1|2000|   LG DIOS| 2000000|
|1603614755|    1|2000|   LG Gram| 1800000|
|1603593500|    2|2001|   LG Cyon| 1400000|
|1603572155|    3|2002|     LG TV| 1000000|
|1603585955|    5|2004|   LG Gram| 3500000|
|1603586155|    5|2004|     LG TV| 2500000|
|1603651550|    1|2001|   LG Cyon| 1400000|
|1603652155|    5|2002|     LG TV| 1000000|
|1603674500|    6|2003|LG Computer| 4500000|
|1603665955|    7|2004|   LG Gram| 3500000|
|1603666155|    9|2004|     LG TV| 2500000|
+----------+-----+----+----------+--------+
```

[4]:
```
access26 = spark.read.option("inferSchema", "true").json("access/20201026")
access26.printSchema()
access26.show(truncate=False)
display(access26)
```

```
root
 |-- a_id: string (nullable = true)
 |-- a_tag: string (nullable = true)
 |-- a_time: long (nullable = true)
 |-- a_timestamp: string (nullable = true)
 |-- a_uid: long (nullable = true)

+------+------+----------+-----------------------+-----+
|a_id  |a_tag |a_time    |a_timestamp            |a_uid|
+------+------+----------+-----------------------+-----+
|logout|access|1603647200|2020-10-26 02:33:20.000|1    |
|logout|access|1603650200|2020-10-26 03:23:20.000|2    |
|logout|access|1603659200|2020-10-26 05:53:20.000|3    |
|logout|access|1603664200|2020-10-26 07:16:40.000|4    |
|logout|access|1603669500|2020-10-26 08:45:00.000|5    |
|login |access|1603645200|2020-10-26 02:00:00.000|1    |
|login |access|1603649200|2020-10-26 03:06:40.000|2    |
|login |access|1603653200|2020-10-26 04:13:20.000|2    |
|login |access|1603657200|2020-10-26 05:20:00.000|3    |
|login |access|1603660200|2020-10-26 06:10:00.000|4    |
|login |access|1603664500|2020-10-26 07:21:40.000|4    |
|login |access|1603666500|2020-10-26 07:55:00.000|5    |
|login |access|1603670500|2020-10-26 09:01:40.000|6    |
|login |access|1603673500|2020-10-26 09:51:40.000|7    |
|login |access|1603674500|2020-10-26 10:08:20.000|8    |
|login |access|1603675500|2020-10-26 10:25:00.000|9    |
+------+------+----------+-----------------------+-----+
```

```
+------+------+----------+----------------------+-----+
|  a_id| a_tag|    a_time|           a_timestamp|a_uid|
+------+------+----------+----------------------+-----+
|logout|access|1603647200|2020-10-26 02:33:20.000|    1|
|logout|access|1603650200|2020-10-26 03:23:20.000|    2|
|logout|access|1603659200|2020-10-26 05:53:20.000|    3|
|logout|access|1603664200|2020-10-26 07:16:40.000|    4|
|logout|access|1603669500|2020-10-26 08:45:00.000|    5|
| login|access|1603645200|2020-10-26 02:00:00.000|    1|
| login|access|1603649200|2020-10-26 03:06:40.000|    2|
| login|access|1603653200|2020-10-26 04:13:20.000|    2|
| login|access|1603657200|2020-10-26 05:20:00.000|    3|
| login|access|1603660200|2020-10-26 06:10:00.000|    4|
| login|access|1603664500|2020-10-26 07:21:40.000|    4|
| login|access|1603666500|2020-10-26 07:55:00.000|    5|
| login|access|1603670500|2020-10-26 09:01:40.000|    6|
| login|access|1603673500|2020-10-26 09:51:40.000|    7|
| login|access|1603674500|2020-10-26 10:08:20.000|    8|
| login|access|1603675500|2020-10-26 10:25:00.000|    9|
+------+------+----------+----------------------+-----+
```

1.1.2 5-2. ,

[5]:
```
user26.createOrReplaceTempView("user26")
purchase26.createOrReplaceTempView("purchase26")
access26.createOrReplaceTempView("access26")
spark.sql("show tables '*26'")
```

[5]:
```
+--------+----------+-----------+
|database| tableName|isTemporary|
+--------+----------+-----------+
|        |  access26|       true|
|        |purchase26|       true|
|        |    user26|       true|
+--------+----------+-----------+
```

1.1.3 5-3. SparkSQL

[6]:
```
u_signup_condition = "u_signup >= '20201026' and u_signup < '20201027'"
user = spark.sql("select u_id, u_name, u_gender from user26").
 ↪where(u_signup_condition)
user.createOrReplaceTempView("user")

p_time_condition = "p_time >= '2020-10-26 00:00:00' and p_time < '2020-10-27 00:
 ↪00:00'"
```

```
purchase = spark.sql("select from_unixtime(p_time) as p_time, p_uid, p_id,␣
↪p_name, p_amount from purchase26").where(p_time_condition)
purchase.createOrReplaceTempView("purchase")

access = spark.sql("select a_id, a_tag, a_timestamp, a_uid from access26")
access.createOrReplaceTempView("access")

spark.sql("show tables")
```

[6]:
```
+--------+----------+-----------+
|database| tableName|isTemporary|
+--------+----------+-----------+
|        |    access|       true|
|        |  access26|       true|
|        |  purchase|       true|
|        |purchase26|       true|
|        |      user|       true|
|        |    user26|       true|
+--------+----------+-----------+
```

1.1.4 5-4. SQL

[7]:
```
spark.sql("describe user")
groupByCount = "select u_gender, count(1) from user group by u_gender"
spark.sql(groupByCount)
```

[7]:
```
+--------+--------+
|u_gender|count(1)|
+--------+--------+
|       |       1|
|       |       3|
+--------+--------+
```

[8]:
```
spark.sql("describe purchase")
selectClause = "select min(p_amount), max(p_amount) from purchase"
spark.sql(selectClause)
```

[8]:
```
+-------------+-------------+
|min(p_amount)|max(p_amount)|
+-------------+-------------+
|      1000000|      4500000|
+-------------+-------------+
```

[9]:
```
spark.sql("describe access")
countTop="select a_uid, count(1) as a_count from access where a_id = 'login'␣
↪group by a_uid order by a_count desc"
spark.sql(countTop)
```

[9]:
```
+-----+-------+
|a_uid|a_count|
+-----+-------+
|    2|      2|
|    4|      2|
|    6|      1|
|    7|      1|
|    3|      1|
|    1|      1|
|    8|      1|
|    5|      1|
|    9|      1|
+-----+-------+
```

1.2 6.

1.2.1 6-1. DAU (Daily Activer User)

[10]:
```
display(access)
distinctAccessUser = "select count(distinct a_uid) as DAU from access"
dau = spark.sql(distinctAccessUser)
display(dau)
```

```
+------+------+-------------------+-----+
|  a_id| a_tag|         a_timestamp|a_uid|
+------+------+-------------------+-----+
|logout|access|2020-10-26 02:33:20.000|    1|
|logout|access|2020-10-26 03:23:20.000|    2|
|logout|access|2020-10-26 05:53:20.000|    3|
|logout|access|2020-10-26 07:16:40.000|    4|
|logout|access|2020-10-26 08:45:00.000|    5|
| login|access|2020-10-26 02:00:00.000|    1|
| login|access|2020-10-26 03:06:40.000|    2|
| login|access|2020-10-26 04:13:20.000|    2|
| login|access|2020-10-26 05:20:00.000|    3|
| login|access|2020-10-26 06:10:00.000|    4|
| login|access|2020-10-26 07:21:40.000|    4|
| login|access|2020-10-26 07:55:00.000|    5|
| login|access|2020-10-26 09:01:40.000|    6|
| login|access|2020-10-26 09:51:40.000|    7|
| login|access|2020-10-26 10:08:20.000|    8|
| login|access|2020-10-26 10:25:00.000|    9|
+------+------+-------------------+-----+

+---+
|DAU|
+---+
|  9|
```

```
+---+
```

1.2.2 6-2. DPU (Daily Paying User)

[11]:
```
display(purchase)
distinctPayingUser = "select count(distinct p_uid) as PU from purchase"
pu = spark.sql(distinctPayingUser)
display(pu)
```

```
+-------------------+-----+----+-----------+---------+
|             p_time|p_uid|p_id|     p_name|p_amount|
+-------------------+-----+----+-----------+---------+
|2020-10-26 03:45:50|    1|2001|    LG Cyon| 1400000|
|2020-10-26 03:55:55|    5|2002|      LG TV| 1000000|
|2020-10-26 10:08:20|    6|2003|LG Computer| 4500000|
|2020-10-26 07:45:55|    7|2004|    LG Gram| 3500000|
|2020-10-26 07:49:15|    9|2004|      LG TV| 2500000|
+-------------------+-----+----+-----------+---------+

+---+
| PU|
+---+
|  5|
+---+
```

1.2.3 6-3. DR (Daily Revenue)

[12]:
```
display(purchase)
sumOfDailyRevenue = "select sum(p_amount) as DR from purchase"
dr = spark.sql(sumOfDailyRevenue)
display(dr)
```

```
DataFrame[p_time: string, p_uid: int, p_id: int, p_name: string, p_amount: int]
```

```
+--------+
|      DR|
+--------+
|12900000|
+--------+
```

1.2.4 6-4. ARPU (Average Revenue Per User)

[13]:
```
v_dau = dau.collect()[0]["DAU"]
v_pu = pu.collect()[0]["PU"]
v_dr = dr.collect()[0]["DR"]

print("ARPU : {}".format(v_dr / v_dau))
```

```
ARPU : 1433333.3333333333
```

1.2.5 6-5. ARPPU (Average Revenue Per Paying User)

[14]:
```
print("ARPPU : {}".format(v_dr / v_pu))
```

```
ARPPU : 2580000.0
```

1.3 7.

[15]:
```
yesterday_dimension="dimension/dt=20201025"
yesterday = spark.read.parquet(yesterday_dimension)
yesterday.printSchema()
display(yesterday)
```

```
root
 |-- d_uid: long (nullable = true)
 |-- d_name: string (nullable = true)
 |-- d_gender: string (nullable = true)
 |-- d_acount: long (nullable = true)
 |-- d_pamount: long (nullable = true)
 |-- d_pcount: long (nullable = true)
 |-- d_first_purchase: string (nullable = true)
```

```
+-----+---------+--------+--------+---------+--------+-------------------+
|d_uid|   d_name|d_gender|d_acount|d_pamount|d_pcount|   d_first_purchase|
+-----+---------+--------+--------+---------+--------+-------------------+
|    5|  |      |       2|  6000000|       2|2020-10-25 09:32:35|
|    2|    |      |       3|  1400000|       1|2020-10-25 11:38:20|
|    1|     |      |       2|  3800000|       2|2020-10-25 05:32:30|
|    3|     |      |       2|  1000000|       1|2020-10-25 05:42:35|
|    4|     |      |       3|        0|       0|               null|
+-----+---------+--------+--------+---------+--------+-------------------+
```

1.3.1 7-1. ID

[16]:
```
yesterday_uids = yesterday.select("d_uid")
today_uids = access.select("a_uid")

joinCondition = "<  uid    uid       >"
joinCondition = yesterday_uids.d_uid == today_uids.a_uid

joinHow = "<            >"
joinHow = "full_outer"

all_uids = (
    yesterday_uids.join(today_uids, joinCondition, joinHow)
```

```
    .withColumn("uid", when(yesterday.d_uid.isNull(), access.a_uid).
 ↪otherwise(yesterday.d_uid))
    .select("uid").distinct()
)

all_uids.printSchema()
display(all_uids.orderBy("uid"))
```

```
root
 |-- uid: long (nullable = true)


+---+
|uid|
+---+
|  1|
|  2|
|  3|
|  4|
|  5|
|  6|
|  7|
|  8|
|  9|
+---+
```

[17]:
```
uid1 = yesterday.select("d_uid").withColumnRenamed("d_uid", "a_uid")
uid2 = access.select("a_uid")
all_uids = uid1.union(uid2).distinct()
all_uids.printSchema()
display(all_uids)
```

```
root
 |-- a_uid: long (nullable = true)


+-----+
|a_uid|
+-----+
|    4|
|    6|
|    7|
|    8|
|    3|
|    5|
|    2|
|    1|
|    9|
```

```
+-----+
```

1.3.2 7-2.

[18]:
```
joinCondition = "uid   yesterday   uid       "
joinCondition = all_uids.a_uid == yesterday.d_uid
joinHow = "left_outer"

# uid      id      user_id   , drop         d_uid     , withColumnRenamed     ,␣
 ↪uid   d_uid
uids = (
    all_uids.join(yesterday, joinCondition, joinHow)
    .drop("d_uid")
    .withColumnRenamed("a_uid", "d_uid")
    .sort(asc("d_uid"))
)

uids.printSchema()
display(uids)
```

```
root
 |-- d_uid: long (nullable = true)
 |-- d_name: string (nullable = true)
 |-- d_gender: string (nullable = true)
 |-- d_acount: long (nullable = true)
 |-- d_pamount: long (nullable = true)
 |-- d_pcount: long (nullable = true)
 |-- d_first_purchase: string (nullable = true)
```

```
+-----+---------+--------+--------+---------+--------+-------------------+
|d_uid|   d_name|d_gender|d_acount|d_pamount|d_pcount|  d_first_purchase|
+-----+---------+--------+--------+---------+--------+-------------------+
|    1|         |        |       2|  3800000|       2|2020-10-25 05:32:30|
|    2|     |       |       3|  1400000|       1|2020-10-25 11:38:20|
|    3|      |       |       2|  1000000|       1|2020-10-25 05:42:35|
|    4|      |       |       3|        0|       0|               null|
|    5|   |       |       2|  6000000|       2|2020-10-25 09:32:35|
|    6|      null|    null|    null|     null|    null|               null|
|    7|      null|    null|    null|     null|    null|               null|
|    8|      null|    null|    null|     null|    null|               null|
|    9|      null|    null|    null|     null|    null|               null|
+-----+---------+--------+--------+---------+--------+-------------------+
```

1.3.3 7-3.

[19]:
```
user.show()
uids.show()
```

```
+----+--------+--------+
|u_id|  u_name|u_gender|
+----+--------+--------+
|   6|        |        |
|   7|        |        |
|   8|        |        |
|   9|        |        |
+----+--------+--------+
```

```
+-----+---------+--------+--------+---------+--------+-------------------+
|d_uid|   d_name|d_gender|d_acount|d_pamount|d_pcount|  d_first_purchase|
+-----+---------+--------+--------+---------+--------+-------------------+
|    1|         |        |       2|  3800000|       2|2020-10-25 05:32:30|
|    2|        |        |       3|  1400000|       1|2020-10-25 11:38:20|
|    3|         |        |       2|  1000000|       1|2020-10-25 05:42:35|
|    4|         |        |       3|        0|       0|               null|
|    5|        |        |       2|  6000000|       2|2020-10-25 09:32:35|
|    6|     null|    null|    null|     null|    null|               null|
|    7|     null|    null|    null|     null|    null|               null|
|    8|     null|    null|    null|     null|    null|               null|
|    9|     null|    null|    null|     null|    null|               null|
+-----+---------+--------+--------+---------+--------+-------------------+
```

[20]:
```
#  25) d_name   null   u_name           d_name
exprName = expr("case when d_name is null then u_name else d_name end")

#  26) d_gender   null   u_gender           d_gender
exprGender = expr("case when d_gender is null then u_gender else d_gender end")

dim1 = (
    uids.join(user, uids.d_uid == user.u_id, "left_outer")
    .withColumn("name", exprName)
    .withColumn("gender", exprGender)
    .drop("d_name", "d_gender", "u_id", "u_name", "u_gender")
    .withColumnRenamed("name", "d_name")
    .withColumnRenamed("gender", "d_gender")
).orderBy(asc("d_uid"))

display(dim1)
```

```
+-----+--------+---------+--------+-------------------+---------+--------+
|d_uid|d_acount|d_pamount|d_pcount|  d_first_purchase|    d_name|d_gender|
```

```
+-----+--------+---------+--------+-------------------+----------+--------+
|    1|       2|  3800000|       2|2020-10-25 05:32:30|      |       |
|    2|       3|  1400000|       1|2020-10-25 11:38:20|     |       |
|    3|       2|  1000000|       1|2020-10-25 05:42:35|      |       |
|    4|       3|        0|       0|               null|      |       |
|    5|       2|  6000000|       2|2020-10-25 09:32:35|   |       |
|    6|    null|     null|    null|               null|     |       |
|    7|    null|     null|    null|               null|     |       |
|    8|    null|     null|    null|               null|     |       |
|    9|    null|     null|    null|               null|     |       |
+-----+--------+---------+--------+-------------------+----------+--------+
```

1.3.4 7-4. 0

[21]:
```
#  27) d_acount, d_pamount, d_pcount       0
fillDefaultValue = { "d_acount":0, "d_pamount":0, "d_pcount":0 }

dim2 = dim1.na.fill(fillDefaultValue)
display(dim2)
```

```
+-----+--------+---------+--------+-------------------+----------+--------+
|d_uid|d_acount|d_pamount|d_pcount|   d_first_purchase|    d_name|d_gender|
+-----+--------+---------+--------+-------------------+----------+--------+
|    1|       2|  3800000|       2|2020-10-25 05:32:30|      |       |
|    2|       3|  1400000|       1|2020-10-25 11:38:20|     |       |
|    3|       2|  1000000|       1|2020-10-25 05:42:35|      |       |
|    4|       3|        0|       0|               null|      |       |
|    5|       2|  6000000|       2|2020-10-25 09:32:35|   |       |
|    6|       0|        0|       0|               null|     |       |
|    7|       0|        0|       0|               null|     |       |
|    8|       0|        0|       0|               null|     |       |
|    9|       0|        0|       0|               null|     |       |
+-----+--------+---------+--------+-------------------+----------+--------+
```

1.3.5 7-5.

[22]:
```
access_sum = spark.sql("select a_uid, count(a_uid) as a_count from access where␣
↪a_id = 'login' group by a_uid")
access_sum.printSchema()
access_sum.show()
dim2.printSchema()

#  28)      (a_count)  null       d_acount      ,       d_acount + a_count
sumOfAccess = expr("a_count")
sumOfAccess = expr("case when a_count is null then d_acount else d_acount +␣
↪a_count end")

dim3 = (
```

```
    dim2.join(access_sum, dim2.d_uid == access_sum.a_uid, "left_outer")
    .withColumn("sum_of_access", sumOfAccess)
    .drop("a_uid", "a_count", "d_acount")
    .withColumnRenamed("sum_of_access", "d_acount")
).orderBy(asc("d_uid"))
dim3.printSchema()
display(dim3)
```

```
root
 |-- a_uid: long (nullable = true)
 |-- a_count: long (nullable = false)

+-----+-------+
|a_uid|a_count|
+-----+-------+
|    4|      2|
|    6|      1|
|    7|      1|
|    8|      1|
|    3|      1|
|    2|      2|
|    5|      1|
|    1|      1|
|    9|      1|
+-----+-------+

root
 |-- d_uid: long (nullable = true)
 |-- d_acount: long (nullable = false)
 |-- d_pamount: long (nullable = false)
 |-- d_pcount: long (nullable = false)
 |-- d_first_purchase: string (nullable = true)
 |-- d_name: string (nullable = true)
 |-- d_gender: string (nullable = true)

root
 |-- d_uid: long (nullable = true)
 |-- d_pamount: long (nullable = false)
 |-- d_pcount: long (nullable = false)
 |-- d_first_purchase: string (nullable = true)
 |-- d_name: string (nullable = true)
 |-- d_gender: string (nullable = true)
 |-- d_acount: long (nullable = true)

+-----+---------+--------+------------------+----------+--------+--------+
|d_uid|d_pamount|d_pcount|   d_first_purchase|    d_name|d_gender|d_acount|
+-----+---------+--------+------------------+----------+--------+--------+
```

```
|    1|  3800000|       2|2020-10-25 05:32:30|      |       |       3|
|    2|  1400000|       1|2020-10-25 11:38:20|     |       |       5|
|    3|  1000000|       1|2020-10-25 05:42:35|      |       |       3|
|    4|        0|       0|               null|      |       |       5|
|    5|  6000000|       2|2020-10-25 09:32:35|   |       |       3|
|    6|        0|       0|               null|     |       |       1|
|    7|        0|       0|               null|     |       |       1|
|    8|        0|       0|               null|     |       |       1|
|    9|        0|       0|               null|     |       |       1|
+-----+---------+--------+-------------------+----------+--------+--------+
```

1.3.6 7-6.

[23]:
```
display(dim3)
dim3.printSchema()

purchase_sum = spark.sql("select p_uid, sum(p_amount) as pamount, count(p_uid)␣
↪as pcount from purchase group by p_uid")

display(purchase_sum)
purchase_sum.printSchema()
```

```
DataFrame[d_uid: bigint, d_pamount: bigint, d_pcount: bigint, d_first_purchase:␣
↪string, d_name: string, d_gender: string, d_acount: bigint]
```

```
root
 |-- d_uid: long (nullable = true)
 |-- d_pamount: long (nullable = false)
 |-- d_pcount: long (nullable = false)
 |-- d_first_purchase: string (nullable = true)
 |-- d_name: string (nullable = true)
 |-- d_gender: string (nullable = true)
 |-- d_acount: long (nullable = true)
```

```
+-----+-------+------+
|p_uid|pamount|pcount|
+-----+-------+------+
|    5|1000000|     1|
|    7|3500000|     1|
|    1|1400000|     1|
|    6|4500000|     1|
|    9|2500000|     1|
+-----+-------+------+
```

```
root
 |-- p_uid: integer (nullable = true)
 |-- pamount: long (nullable = true)
 |-- pcount: long (nullable = false)
```

[24]:
```
#  29)      (d_pamount + pamount)
sumOfAmount = expr("case when pamount is null then d_pamount else d_pamount +␣
 ↪pamount end")

#  30)      (d_pcount + pcount)
sumOfCount = expr("case when pcount is null then d_pcount else d_pcount +␣
 ↪pcount end")

dim4 = (
    dim3.join(purchase_sum, dim3.d_uid == purchase_sum.p_uid, "left")
    .withColumn("sum_of_amount", sumOfAmount)
    .withColumn("sum_of_count", sumOfCount)
    .drop("p_uid", "d_pamount", "d_pcount", "pamount", "pcount")
    .withColumnRenamed("sum_of_amount", "d_pamount")
    .withColumnRenamed("sum_of_count", "d_pcount")
).orderBy(asc("d_uid"))

dim4.printSchema()
display(dim4)
```

```
root
 |-- d_uid: long (nullable = true)
 |-- d_first_purchase: string (nullable = true)
 |-- d_name: string (nullable = true)
 |-- d_gender: string (nullable = true)
 |-- d_acount: long (nullable = true)
 |-- d_pamount: long (nullable = true)
 |-- d_pcount: long (nullable = true)
```

d_uid	d_first_purchase	d_name	d_gender	d_acount	d_pamount	d_pcount
1	2020-10-25 05:32:30			3	5200000	3
2	2020-10-25 11:38:20			5	1400000	1
3	2020-10-25 05:42:35			3	1000000	1
4	null			5	0	0
5	2020-10-25 09:32:35			3	7000000	3
6	null			1	4500000	1
7	null			1	3500000	1
8	null			1	0	0
9	null			1	2500000	1

1.3.7 7-7.

[25]:
```
# 31) group by p_uid , min(p_time)
selectFirstPurchaseTime = "select p_uid, min(p_time) as p_time from purchase␣
↪group by p_uid"

first_purchase = spark.sql(selectFirstPurchaseTime)
first_purchase.printSchema()
first_purchase.show()

# 32) (d_first_purchase) null p_time d_first_purchase
exprFirstPurchase = expr("case when d_first_purchase is null then p_time else␣
↪d_first_purchase end")

dimension = (
    dim4.join(first_purchase, dim4.d_uid == first_purchase.p_uid, "left")
    .withColumn("first_purchase", exprFirstPurchase)
    .drop("p_uid", "p_time", "d_first_purchase")
    .withColumnRenamed("first_purchase", "d_first_purchase")
).orderBy("d_uid")

dimension.printSchema()
display(dimension)
```

```
root
 |-- p_uid: integer (nullable = true)
 |-- p_time: string (nullable = true)

+-----+-------------------+
|p_uid|             p_time|
+-----+-------------------+
|    5|2020-10-26 03:55:55|
|    7|2020-10-26 07:45:55|
|    1|2020-10-26 03:45:50|
|    6|2020-10-26 10:08:20|
|    9|2020-10-26 07:49:15|
+-----+-------------------+

root
 |-- d_uid: long (nullable = true)
 |-- d_name: string (nullable = true)
 |-- d_gender: string (nullable = true)
 |-- d_acount: long (nullable = true)
 |-- d_pamount: long (nullable = true)
 |-- d_pcount: long (nullable = true)
 |-- d_first_purchase: string (nullable = true)
```

```
+-----+---------+--------+--------+---------+--------+-------------------+
|d_uid|   d_name|d_gender|d_acount|d_pamount|d_pcount|  d_first_purchase|
+-----+---------+--------+--------+---------+--------+-------------------+
|    1|         |        |       3|  5200000|       3|2020-10-25 05:32:30|
|    2|        |        |       5|  1400000|       1|2020-10-25 11:38:20|
|    3|         |        |       3|  1000000|       1|2020-10-25 05:42:35|
|    4|         |        |       5|        0|       0|               null|
|    5|        |        |       3|  7000000|       3|2020-10-25 09:32:35|
|    6|         |        |       1|  4500000|       1|2020-10-26 10:08:20|
|    7|         |        |       1|  3500000|       1|2020-10-26 07:45:55|
|    8|         |        |       1|        0|       0|               null|
|    9|         |        |       1|  2500000|       1|2020-10-26 07:49:15|
+-----+---------+--------+--------+---------+--------+-------------------+
```

1.3.8 7-8.

[26]:
```
#
today_uids = dimension.select("d_uid")
yesterday_uids = yesterday.select("d_uid")

nu = today_uids.subtract(yesterday_uids)
nu.printSchema()
nu.show()
v_nu = nu.count()
print("NU: {}".format(v_nu))
```

```
root
 |-- d_uid: long (nullable = true)

+-----+
|d_uid|
+-----+
|    6|
|    7|
|    8|
|    9|
+-----+

NU: 4
```

1.3.9 7-9.

[27]:
```
dimension.printSchema()
target_dir="dimension/dt=20201026"
dimension.write.mode("overwrite").parquet(target_dir)
```

```
root
 |-- d_uid: long (nullable = true)
```

```
 |-- d_name: string (nullable = true)
 |-- d_gender: string (nullable = true)
 |-- d_acount: long (nullable = true)
 |-- d_pamount: long (nullable = true)
 |-- d_pcount: long (nullable = true)
 |-- d_first_purchase: string (nullable = true)
```

1.3.10 7-10.

[28]:
```
newDimension = spark.read.parquet(target_dir)
newDimension.printSchema()
display(newDimension)
```

```
root
 |-- d_uid: long (nullable = true)
 |-- d_name: string (nullable = true)
 |-- d_gender: string (nullable = true)
 |-- d_acount: long (nullable = true)
 |-- d_pamount: long (nullable = true)
 |-- d_pcount: long (nullable = true)
 |-- d_first_purchase: string (nullable = true)
```

d_uid	d_name	d_gender	d_acount	d_pamount	d_pcount	d_first_purchase
9			1	2500000	1	2020-10-26 07:49:15
7			1	3500000	1	2020-10-26 07:45:55
8			1	0	0	null
3			3	1000000	1	2020-10-25 05:42:35
4			5	0	0	null
1			3	5200000	3	2020-10-25 05:32:30
2			5	1400000	1	2020-10-25 11:38:20
5			3	7000000	3	2020-10-25 09:32:35
6			1	4500000	1	2020-10-26 10:08:20

1.3.11 7-11. MySQL

, Append .

[29]:
```
print("DT:{}, DAU:{}, PU:{}, DR:{}".format("2020-10-26", v_dau, v_pu, v_dr))
```

```
DT:2020-10-26, DAU:9, PU:5, DR:12900000
```

[64]:
```
today = "2020-10-26"
lgde_origin = spark.read.jdbc("jdbc:mysql://mysql:3306/testdb", "testdb.lgde",␣
 ↪properties={"user": "sqoop", "password": "sqoop"}).where(col("dt") <␣
 ↪lit(today))
lgde_today = spark.createDataFrame([(today, v_dau, v_pu, v_dr)], ["DT", "DAU",␣
 ↪"PU", "DR"])
lgde_union = lgde_origin.union(lgde_today)
lgde_local = lgde_union.collect()
lgde = spark.createDataFrame(lgde_local)
lgde.write.mode("overwrite").jdbc("jdbc:mysql://mysql:3306/testdb", "testdb.
 ↪lgde", properties={"user": "sqoop", "password": "sqoop"})
```

[ ]:
```

```