data-engineer-lgde-day1-answer

September 8, 2021

1 1. LGDE.com 1 ( )

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 13:44:26 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).
```

[1]:
```
<pyspark.sql.session.SparkSession at 0x7fcea130f5e0>
```

[2]:
```
user25 = spark.read.parquet("user/20201025")
user25.printSchema()
user25.show(truncate=False)
display(user25)
```

```
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|
+----+----------+--------+--------+
```

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

[3]:
```
purchase25 = spark.read.parquet("purchase/20201025")
purchase25.printSchema()
display(purchase25)
```

```
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|
+----------+-----+----+-------+--------+
```

[4]:
```
access25 = spark.read.option("inferSchema", "true").json("access/20201025")
access25.printSchema()
display(access25)
```

```
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|1603567200|2020-10-25 04:20:00.000|    1|
```

```
|logout|access|1603570200|2020-10-25 05:10:00.000|    2|
|logout|access|1603579200|2020-10-25 07:40:00.000|    3|
|logout|access|1603584200|2020-10-25 09:03:20.000|    4|
|logout|access|1603589500|2020-10-25 10:31:40.000|    5|
| login|access|1603565200|2020-10-25 03:46:40.000|    1|
| login|access|1603569200|2020-10-25 04:53:20.000|    2|
| login|access|1603573200|2020-10-25 06:00:00.000|    2|
| login|access|1603577200|2020-10-25 07:06:40.000|    3|
| login|access|1603580200|2020-10-25 07:56:40.000|    4|
| login|access|1603584500|2020-10-25 09:08:20.000|    4|
| login|access|1603586500|2020-10-25 09:41:40.000|    5|
+------+------+----------+-----------------------+-----+
```

1.1.2 5-2. ,

[5]:
```
user25.createOrReplaceTempView("user25")
purchase25.createOrReplaceTempView("purchase25")
access25.createOrReplaceTempView("access25")
spark.sql("show tables '*25'")
```

[5]:
```
+--------+----------+-----------+
|database| tableName|isTemporary|
+--------+----------+-----------+
|        |  access25|       true|
|        |purchase25|       true|
|        |    user25|       true|
+--------+----------+-----------+
```

1.1.3 5-3. SparkSQL

[6]:
```
u_signup_condition = "u_signup >= '20201025' and u_signup < '20201026'"
user = spark.sql("select u_id, u_name, u_gender from user25").
 ↪where(u_signup_condition)
user.createOrReplaceTempView("user")

p_time_condition = "p_time >= '2020-10-25 00:00:00' and p_time < '2020-10-26 00:
 ↪00:00'"
purchase = spark.sql("select from_unixtime(p_time) as p_time, p_uid, p_id,␣
 ↪p_name, p_amount from purchase25").where(p_time_condition)
purchase.createOrReplaceTempView("purchase")

access = spark.sql("select a_id, a_tag, a_timestamp, a_uid from access25")
access.createOrReplaceTempView("access")

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

[6]:
```
+--------+---------+-----------+
|database| tableName|isTemporary|
+--------+---------+-----------+
|        |    access|       true|
|        |  access25|       true|
|        |  purchase|       true|
|        |purchase25|       true|
|        |      user|       true|
|        |    user25|       true|
+--------+---------+-----------+
```

1.1.4 5-4. SQL

[7]:
```
whereCondition = "u_gender = ' '"
spark.sql("select * from user").where(whereCondition)
```

[7]:
```
+----+--------+--------+
|u_id|  u_name|u_gender|
+----+--------+--------+
|   1|    |       |
|   2|  |       |
|   4|    |       |
+----+--------+--------+
```

[8]:
```
selectClause = "select * from purchase where p_amount > 2000000"
spark.sql(selectClause)
```

[8]:
```
+-------------------+-----+----+-------+--------+
|             p_time|p_uid|p_id| p_name|p_amount|
+-------------------+-----+----+-------+--------+
|2020-10-25 09:32:35|    5|2004|LG Gram| 3500000|
|2020-10-25 09:35:55|    5|2004|  LG TV| 2500000|
+-------------------+-----+----+-------+--------+
```

[9]:
```
groupByClause="select a_id, count(1) from access group by a_id"
spark.sql(groupByClause)
```

[9]:
```
+------+--------+
|  a_id|count(1)|
+------+--------+
|logout|       5|
| login|       7|
+------+--------+
```

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|2020-10-25 04:20:00.000|    1|
|logout|access|2020-10-25 05:10:00.000|    2|
|logout|access|2020-10-25 07:40:00.000|    3|
|logout|access|2020-10-25 09:03:20.000|    4|
|logout|access|2020-10-25 10:31:40.000|    5|
| login|access|2020-10-25 03:46:40.000|    1|
| login|access|2020-10-25 04:53:20.000|    2|
| login|access|2020-10-25 06:00:00.000|    2|
| login|access|2020-10-25 07:06:40.000|    3|
| login|access|2020-10-25 07:56:40.000|    4|
| login|access|2020-10-25 09:08:20.000|    4|
| login|access|2020-10-25 09:41:40.000|    5|
+------+------+----------------------+-----+

+---+
|DAU|
+---+
|  5|
+---+
```

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|
+-------------------+-----+----+-------+--------+
|2020-10-25 05:32:30|    1|2000|LG DIOS| 2000000|
|2020-10-25 17:32:35|    1|2000|LG Gram| 1800000|
|2020-10-25 11:38:20|    2|2001|LG Cyon| 1400000|
|2020-10-25 05:42:35|    3|2002|  LG TV| 1000000|
|2020-10-25 09:32:35|    5|2004|LG Gram| 3500000|
|2020-10-25 09:35:55|    5|2004|  LG TV| 2500000|
+-------------------+-----+----+-------+--------+
```

```
+---+
| PU|
+---+
|  4|
+---+
```

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|
+--------+
|12200000|
+--------+
```

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 : 2440000.0
```

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

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

```
ARPPU : 3050000.0
```

1.3 7.

1.3.1 7-1.

1.3.2 7-2.

[15]:
```
access.printSchema()
countOfAccess = "select a_uid, count(a_uid) as a_count from access group by␣
 ↪a_uid order by a_uid asc"
accs = spark.sql(countOfAccess)
display(accs)
```

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


+-----+-------+
|a_uid|a_count|
+-----+-------+
|    1|      2|
|    2|      3|
|    3|      2|
|    4|      3|
|    5|      2|
+-----+-------+
```

1.3.3 7-3. ,

[16]:
```
purchase.printSchema()
sumOfCountAndAmount = "select p_uid, count(p_uid) as p_count, sum(p_amount) as␣
 ↪p_amount from purchase group by p_uid order by p_uid asc"
amts = spark.sql(sumOfCountAndAmount)
display(amts)
```

```
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_uid|p_count|p_amount|
+-----+-------+--------+
|    1|      2| 3800000|
|    2|      1| 1400000|
|    3|      1| 1000000|
|    5|      2| 6000000|
+-----+-------+--------+
```

1.3.4 7-4.

[17]:
```
accs.printSchema()
amts.printSchema()
joinCondition = accs.a_uid == amts.p_uid
joinHow = "left_outer"
dim1 = accs.join(amts, joinCondition, joinHow)
```

```
dim1.printSchema()
display(dim1.orderBy(asc("a_uid")))
```

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

root
 |-- p_uid: integer (nullable = true)
 |-- p_count: long (nullable = false)
 |-- p_amount: long (nullable = true)

root
 |-- a_uid: long (nullable = true)
 |-- a_count: long (nullable = false)
 |-- p_uid: integer (nullable = true)
 |-- p_count: long (nullable = true)
 |-- p_amount: long (nullable = true)


+-----+-------+-----+-------+--------+
|a_uid|a_count|p_uid|p_count|p_amount|
+-----+-------+-----+-------+--------+
|    1|      2|    1|      2| 3800000|
|    2|      3|    2|      1| 1400000|
|    3|      2|    3|      1| 1000000|
|    4|      3| null|   null|    null|
|    5|      2|    5|      2| 6000000|
+-----+-------+-----+-------+--------+
```

1.3.5 7-5.

[18]:
```
dim1.printSchema()
user.printSchema()
joinCondition = dim1.a_uid == user.u_id
joinHow = "left_outer"
dim2 = dim1.join(user, joinCondition, joinHow)
dim2.printSchema()
display(dim2.orderBy(asc("a_uid")))
```

```
root
 |-- a_uid: long (nullable = true)
 |-- a_count: long (nullable = false)
 |-- p_uid: integer (nullable = true)
 |-- p_count: long (nullable = true)
 |-- p_amount: long (nullable = true)

root
```

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

root
 |-- a_uid: long (nullable = true)
 |-- a_count: long (nullable = false)
 |-- p_uid: integer (nullable = true)
 |-- p_count: long (nullable = true)
 |-- p_amount: long (nullable = true)
 |-- u_id: integer (nullable = true)
 |-- u_name: string (nullable = true)
 |-- u_gender: string (nullable = true)


+-----+-------+-----+-------+--------+----+----------+--------+
|a_uid|a_count|p_uid|p_count|p_amount|u_id|    u_name|u_gender|
+-----+-------+-----+-------+--------+----+----------+--------+
|    1|      2|    1|      2| 3800000|   1|      |       |
|    2|      3|    2|      1| 1400000|   2|    |       |
|    3|      2|    3|      1| 1000000|   3|      |       |
|    4|      3| null|   null|    null|   4|      |       |
|    5|      2|    5|      2| 6000000|   5|   |       |
+-----+-------+-----+-------+--------+----+----------+--------+
```

1.3.6 7-6. ID , 0

[19]:
```
dim2.printSchema()
dim3 = dim2.drop("p_uid", "u_id")
fillDefaultValue = { "p_count":0, "p_amount":0 }
dim4 = dim3.na.fill(fillDefaultValue)
dim4.printSchema()
display(dim4.orderBy(asc("a_uid")))
```

```
root
 |-- a_uid: long (nullable = true)
 |-- a_count: long (nullable = false)
 |-- p_uid: integer (nullable = true)
 |-- p_count: long (nullable = true)
 |-- p_amount: long (nullable = true)
 |-- u_id: integer (nullable = true)
 |-- u_name: string (nullable = true)
 |-- u_gender: string (nullable = true)

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

```
 |-- p_amount: long (nullable = false)
 |-- u_name: string (nullable = true)
 |-- u_gender: string (nullable = true)


+-----+-------+-------+--------+----------+--------+
|a_uid|a_count|p_count|p_amount|    u_name|u_gender|
+-----+-------+-------+--------+----------+--------+
|    1|      2|      2| 3800000|      |       |
|    2|      3|      1| 1400000|    |       |
|    3|      2|      1| 1000000|      |       |
|    4|      3|      0|       0|      |       |
|    5|      2|      2| 6000000|   |       |
+-----+-------+-------+--------+----------+--------+
```

1.3.7 7-7.

[20]:
```
dim4.printSchema()
dim5 = (
    dim4
    .withColumnRenamed("a_uid", "d_uid")
    .withColumnRenamed("a_count", "d_acount")
    .withColumnRenamed("p_amount", "d_pamount")
    .withColumnRenamed("p_count", "d_pcount")
    .withColumnRenamed("u_name", "d_name")
    .withColumnRenamed("u_gender", "d_gender")
   .drop("a_uid", "a_count", "p_amount", "p_count", "u_name", "u_gender")
   .select("d_uid", "d_name", "d_gender", "d_acount", "d_pamount", "d_pcount")
)
display(dim5.orderBy(asc("d_uid")))
```

```
root
 |-- a_uid: long (nullable = true)
 |-- a_count: long (nullable = false)
 |-- p_count: long (nullable = false)
 |-- p_amount: long (nullable = false)
 |-- u_name: string (nullable = true)
 |-- u_gender: string (nullable = true)


+-----+----------+--------+--------+---------+--------+
|d_uid|    d_name|d_gender|d_acount|d_pamount|d_pcount|
+-----+----------+--------+--------+---------+--------+
|    1|      |       |       2|  3800000|       2|
|    2|    |       |       3|  1400000|       1|
|    3|      |       |       2|  1000000|       1|
|    4|      |       |       3|        0|       0|
|    5|   |       |       2|  6000000|       2|
+-----+----------+--------+--------+---------+--------+
```

1.3.8 7-8.

[21]:
```
purchase.printSchema()
selectFirstPurchaseTime = "select p_uid, min(p_time) as p_time from purchase␣
 ↪group by p_uid"

first_purchase = spark.sql(selectFirstPurchaseTime)
dim6 = dim5.withColumn("d_first_purchase", lit(None))
dim6.printSchema()

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

dim7 = (
    dim6.join(first_purchase, dim5.d_uid == first_purchase.p_uid, "left_outer")
    .withColumn("first_purchase", exprFirstPurchase)
    .drop("d_first_purchase", "p_uid", "p_time")
    .withColumnRenamed("first_purchase", "d_first_purchase")
)

dimension = dim7.orderBy(asc("d_uid"))
dimension.printSchema()
display(dimension)
```

```
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)

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

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

1.3.9 7-9.

[22]:
```
dimension.printSchema()
target_dir="dimension/dt=20201025"
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 = false)
 |-- d_pamount: long (nullable = false)
 |-- d_pcount: long (nullable = false)
 |-- d_first_purchase: string (nullable = true)
```

1.3.10 7-10.

[23]:
```
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|
```

```
+-----+----------+--------+--------+---------+--------+-------------------+
|    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|
+-----+----------+--------+--------+---------+--------+-------------------+
```

1.3.11 7-11. MySQL

[24]:
```
print("DT:{}, DAU:{}, PU:{}, DR:{}".format("2020-10-25", v_dau, v_pu, v_dr))
```

```
DT:2020-10-25, DAU:5, PU:4, DR:12200000
```

[41]:
```
today = "2020-10-25"
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 = lgde_origin.union(lgde_today)
lgde.write.mode("overwrite").jdbc("jdbc:mysql://mysql:3306/testdb", "testdb.
↪lgde", properties={"user": "sqoop", "password": "sqoop"})
```

[ ]:
```

```