databricks HW6_template (1)

(https://databricks.com)

Assignment 6

setup environment (must run first)

```
%python
users_df = spark.read.format("json").load("/FileStore/tables/users.json")
pictures_df = spark.read.format("json").load("/FileStore/tables/pictures.json")
items_df = spark.read.format("json").load("/FileStore/tables/items.json")
ads_df = spark.read.format("json").load("/FileStore/tables/ads.json")
ratings_df = spark.read.format("json").load("/FileStore/tables/ratings.json")

users_df.createOrReplaceTempView("users")
pictures_df.createOrReplaceTempView("pictures")
items_df.createOrReplaceTempView("items")
ads_df.createOrReplaceTempView("ads")
ratings_df.createOrReplaceTempView("ratings")

from pyspark.sql import functions as F
from pyspark.sql.functions import *
```

```
%sql
(select "users", count(*) from users)
union all
(select "pictures", count(*) from pictures)
union all
(select "items", count(*) from items)
union all
(select "ads", count(*) from ads)
union all
(select "ratings", count(*) from ratings)
```

Table

	users	count(1)
1	users	2000
2	pictures	20000
3	items	10000
4	ads	4436
5	ratings	2127

5 rows

Solution

```
#1.A
ads_df.printSchema()

#1.B seller data type is: A Struct with a string field called placed_date and a string field called user_id

root
 |-- _oid: string (nullable = true)
 |-- ad_id: string (nullable = true)
 |-- content: string (nullable = true)
 |-- item_id: string (nullable = true)
 |-- pic_num: long (nullable = true)
 |-- plan: string (nullable = true)
 |-- seller: struct (nullable = true)
 |    |-- placed_date: string (nullable = true)
```

```
 |    |-- user_id: string (nullable = true)
```

```
%sql
--1.C
DESCRIBE users;
--1.D phone data type is: an array of structs, with a kind field and a number field. Both fields are of type string.
```

Table

	col_name	data_type	comment
1	_oid	string	null
2	address	struct<city:string,state:string,street:string,zip:bigint>	null
3	categories	array<string>	null
4	email	string	null
5	is_buyer	boolean	null
6	is_seller	boolean	null
7	joined_date	string	null

10 rows

```
#2.A DF
answer = users_df.where("user_id = 'ZW64O'")
display(answer)
```

Table

	_oid	address
1	ac5be233-4483-7b39-b283-0e5af988c59c	▸ {"city": "New Sergiobury", "state": "Missouri", "street": "1039 Brianna Passage Suite 274", "

1 row

```
%sql
--2.A SQL
SELECT *
FROM users
WHERE user_id = "ZW64O";
```

Table

	_oid	address
1	ac5be233-4483-7b39-b283-0e5af988c59c	▸ {"city": "New Sergiobury", "state": "Missouri", "street": "1039 Brianna Passage Suite 274", "

1 row

```
#2.B DF
ads_df.filter("plan == 'gold'").join(users_df, users_df.user_id ==
ads_df.seller.user_id).groupBy(users_df.user_id).agg(F.count("*")).sort(F.desc('count(1)')).limit(5).display()
```

Table

	user_id	count(1)
1	KGD5Z	8
2	BJLLH	7
3	6IPKS	6
4	P8DKB	6
5	0YDV7	6

5 rows

```
%sql
--2.B SQL
SELECT u.user_id, COUNT(*)
FROM ads a, users u
WHERE a.seller.user_id = u.user_id AND a.plan = "gold"
GROUP BY u.user_id
ORDER BY COUNT(*) DESC
LIMIT 5;
```

Table

	user_id	count(1)
1	KGD5Z	8
2	BJLLH	7
3	6IPKS	6
4	P8DKB	6
5	0YDV7	6

5 rows

```
#2.C DF
items_df.where("buyer.purchase_date LIKE '2022-06-
%'").groupBy(items_df.category).agg(F.count("*")).sort(F.desc('count(1)')).limit(3).display()
```

Table

	category	count(1)
1	Toys & Games	12
2	Office Products	9
3	Arts, Crafts & Sewing	8

3 rows

```
%sql
--2.C SQL
SELECT i.category, count(*)
FROM items i
WHERE i.buyer.purchase_date LIKE "2022-06-%"
GROUP BY i.category
ORDER BY COUNT(*) DESC
LIMIT 3;
```

Table

	category	count(1)
1	Toys & Games	12
2	Office Products	9
3	Arts, Crafts & Sewing	8

3 rows

```
#2.D DF
users_df.withColumn("category", explode(users_df.categories)).groupBy("category").agg(F.count("*")).select("category",
"count(1)").sort(F.desc("count(1)")).limit(1).display()
```

Table

	category	count(1)
1	Beverages	685

1 row

```
%sql
--2.D SQL
SELECT category, COUNT(*)
FROM Users AS u LATERAL VIEW explode(u.categories) AS category
GROUP BY category
ORDER BY COUNT(*) DESC
LIMIT 1;
```

Table

	category	count(1)
1	Beverages	685

1 row

```
#2.E
items_df.filter("'2022-06-03' <= buyer.purchase_date").filter("buyer.purchase_date < '2022-06-18'").join(users_df,
users_df.user_id == items_df.buyer.user_id).sort(F.asc("buyer.purchase_date")).select("item_id", "email",
"buyer.purchase_date").limit(5).display()
```

Table

	item_id	email	purchase_date
1	YU5MK	taylor.nichole31@aol.com	2022-06-03
2	YR9IC	fox72442@gmail.com	2022-06-03
3	E6TD8	coleman13@gmail.com	2022-06-03
4	3F9TL	berger.david1497@aol.com	2022-06-04
5	1IJ4G	tre_vasquez68842@yahoo.com	2022-06-04

5 rows

```
%sql
--2.E SQL
SELECT i.item_id, u.email, i.buyer.purchase_date
FROM items as i, users as u
WHERE i.buyer.user_id = u.user_id AND i.buyer.purchase_date >= "2022-06-03" AND i.buyer.purchase_date < "2022-06-18"
ORDER BY i.buyer.purchase_date ASC
LIMIT 5;
```

Table

	item_id	email	purchase_date
1	YU5MK	taylor.nichole31@aol.com	2022-06-03
2	YR9IC	fox72442@gmail.com	2022-06-03
3	E6TD8	coleman13@gmail.com	2022-06-03
4	3F9TL	berger.david1497@aol.com	2022-06-04
5	1IJ4G	tre_vasquez68842@yahoo.com	2022-06-04

5 rows

```
#2.F DF
users_df.join(items_df, users_df.user_id == items_df.seller.user_id).join(pictures_df, items_df.item_id ==
pictures_df.item_id).groupBy("user_id", "email").agg(F.countDistinct(pictures_df.item_id)).select("email",
"count(item_id)").sort(F.desc("count(item_id)")).limit(3).display()
```

Table

	email	count(item_id)
1	melissa_Benjamin05626@hotmail.com	35
2	561tracy55@yahoo.com	34
3	michael829@gmail.com	34

3 rows

```
%sql
--2.F SQL
SELECT u.email, COUNT(DISTINCT p.item_id) as pic_num
FROM users u, items i, pictures p
WHERE u.user_id = i.seller.user_id AND i.item_id = p.item_id
GROUP BY u.user_id, u.email
ORDER BY pic_num DESC
LIMIT 3;
```

Table

	email	pic_num
1	melissa_Benjamin05626@hotmail.com	35
2	561tracy55@yahoo.com	34
3	michael829@gmail.com	34

3 rows

```
#2.G DF
active_buyers = items_df.groupBy("buyer.user_id").agg(F.count("*")).where("count(1) >= 2").select("user_id").collect()
ratings_df.where(ratings_df.quality.isNotNull()).filter(ratings_df.buyer_id.isin([ab['user_id'] for ab in
active_buyers])).join(users_df, ratings_df.seller_id == users_df.user_id).groupBy("seller_id", "name.first",
"name.last").agg(F.avg("quality"), F.count("*")).where("count(1) >= 2").sort(F.desc("avg(quality)")).select("seller_id",
"avg(quality)", "count(1)", "first", "last").limit(5).display()
```

Table

	seller_id	avg(quality)	count(1)	first	last
1	DAWJW	4.5	2	Kristi	Caldwell
2	4MSI6	3.5	2	Monica	Singleton
3	CJZRT	3	2	Corey	Quinn
4	QFR1K	3	2	Michael	Scott
5	PB6Z6	2.6666666666666665	3	Bradley	Anderson

5 rows

```
%sql
--2.G SQL

--Credible sellers
WITH active_buyers(user_id) AS (
  SELECT i.buyer.user_id
  FROM items i
  GROUP BY i.buyer.user_id
  HAVING COUNT(*) >= 2
)
SELECT r.seller_id, AVG(r.quality) as average_quality, COUNT(*), u.name.first, u.name.last
FROM ratings r, users u
WHERE u.user_id = r.seller_id AND r.quality IS NOT NULL AND r.buyer_id IN (SELECT user_id FROM active_buyers)
GROUP BY r.seller_id, u.name.first, u.name.last
HAVING COUNT(*) >= 2
ORDER BY average_quality DESC
LIMIT ;
```

Table

	seller_id	average_quality	count(1)	first	last
1	DAWJW	4.5	2	Kristi	Caldwell
2	4MSI6	3.5	2	Monica	Singleton
3	CJZRT	3	2	Corey	Quinn
4	QFR1K	3	2	Michael	Scott

5	PB6Z6	2.6666666666666665	3	Bradley	Anderson

5 rows