



Lab 2 - Data Munging

Module 5 Assignment

In this exercise, you will be working with mock data meant to replicate data from an ecommerce mattress seller. Broadly, your work is to clean up and present this data so that it can be used to target geographic areas. Work through the tasks below and answer the challenge to produce the required report.

In this assignment you will:

- Work with hierarchical data
- Use common table expressions to display data
- Create new tables based on existing tables
- Manage working with null values and timestamps

As you work through the following tasks, you will be prompted to enter selected answers in Coursera. Find the quiz associated with this lab to enter your answers.

Run the cell below to prepare this workspace for the lab.

```
%run ../Includes/Classroom-Setup
```

Mounting course-specific datasets to **/mnt/training...**

Datasets are already mounted to **/mnt/training** from **s3a://databricks-corp-training/common**

```
res4: Boolean = false
```

```
res5: Boolean = false
```

Exercise 1: Create a table

Summary: Create a new table named `eventsRaw`

Use this path to access the data: `/mnt/training/ecommerce/events/events.parquet`

Steps to complete:

- Make sure this notebook is idempotent by first dropping the table named `eventsRaw`, if it exists already
- Use the provided path to read in the data

```
DROP TABLE IF EXISTS eventsRaw;
CREATE TABLE eventsRaw USING parquet OPTIONS (
  path "/mnt/training/ecommerce/events/events.parquet",
  header "true"
)
```

OK

```
SELECT * FROM eventsRaw LIMIT 5
```

	device ▲	ecommerce
1	macOS	▶ {"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items":
2	Windows	▶ {"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items":
3	macOS	▶ {"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items":
4	iOS	▶ {"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items":
5	Windows	▶ {"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items":

Showing all 5 rows.

Exercise 2: Understand the schema and metadata

Summary: Run a command to display this table's schema and other detailed table information

Notice that this table includes `ArrayType` and `StructType` data

Steps to complete:

- Run a single command to display the table information
- **Answer the corresponding question in Coursera, in the quiz for this module, regarding the location of this table**

DESCRIBE eventsRaw;

	col_name ▲	data_type
1	device	string
2	ecommerce	struct<purchase_revenue_in_usd:double,total_item_quantity:big
3	event_name	string
4	event_previous_timestamp	bigint
5	event_timestamp	bigint
6	geo	struct<city:string,state:string>
7	items	array<struct<coupon:string,item_id:string,item_name:string,item

Showing all 10 rows.

DESCRIBE EXTENDED eventsRaw;

	col_name ▲	data_type
1	device	string
2	ecommerce	struct<purchase_revenue_in_usd:double,total_item_quantity:bi
3	event_name	string
4	event_previous_timestamp	bigint
5	event_timestamp	bigint
6	geo	struct<city:string,state:string>
7	items	array<struct<coupon:string,item_id:string,item_name:string,item

Showing all 25 rows.

Exercise 3: Sample the table

Summary: Sample this table to get a closer look at the data

Steps to complete:

- Sample the table to display up to 1 percent of the records

```
SELECT * FROM eventsRaw TABLESAMPLE(1 percent)
```


	device ▲	ecommerce
1	Android	▶ {"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items":
2	Windows	▶ {"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items":
3	iOS	▶ {"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items":
4	Linux	▶ {"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items":
5	macOS	▶ {"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items":
6	Chrome OS	▶ {"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items":
7	macOS	▶ {"purchase_revenue_in_usd": null, "total_item_quantity": null, "unique_items":

Truncated results, showing first 1000 rows.

Exercise 4: Create a new table

Summary: Create a table `purchaseEvents` that includes event data *with* purchases that has the following schema:

ColumnName	DataType
purchases	double
previousEventDate	date
eventDate	date
city	string
state	string
userId	string

 The timestamps in this table are meant to match those used in Google Analytics, which measures time to the microsecond. To convert to unixtime, you must divide these values by 1000000 (10e6) before casting to a timestamp.

 **Hint:** Access values from StructType objects using dot notation

Steps to complete:

- Make sure this notebook is idempotent by first dropping the table, if it exists
- Create a table based on the existing table
- Use a common table expression to manipulate your data before writing the

`SELECT` statement that will define your table (*Recommended*)

Do not include records where the `previousEventDate` is `NULL`

```
SELECT * FROM purchaseEvents;
```

```
Error in SQL statement: AnalysisException: Table or view not found: purchase
Events; line 1 pos 14;
'Project [*]
+- 'UnresolvedRelation [purchaseEvents], [], false
```

```

DROP TABLE IF EXISTS purchaseEvents2;
CREATE TABLE purchaseEvents2 AS (
  SELECT
    ecommerce.purchase_revenue_in_usd AS purchases,
    event_previous_timestamp/1000000 AS previousEventDate,
    event_timestamp/1000000 AS eventDate,
    geo.city AS city,
    geo.state AS state,
    user_id AS userId
  FROM
    eventsRaw
  WHERE
    ecommerce.purchase_revenue_in_usd IS NOT NULL
  ORDER BY ecommerce.purchase_revenue_in_usd
  DESC
);

```

Query returned no results

```
SELECT * FROM purchaseEvents2;
```

	purchases ▲	previousEventDate ▲	eventDate ▲	city
1	5830	1592582223.66408	1592582226.664186	Amarillo
2	5485	1592862660.164622	1592862669.375088	Tampa
3	5289	1593543276.577961	1593543308.044722	Buffalo
4	5219.1	1593523099.009913	1593523682.510157	Miami
5	5180	1593534453.169522	1593535539.976752	Sarasota
6	5175	1592580995.454087	1592581153.490607	Bakersfield
7	5125	1592672945.778788	1592673251.796709	Corpus Christi

Truncated results, showing first 1000 rows.

Exercise 5: Count the records

Summary: Count all the records in your new table.

Steps to complete:

- Write a `SELECT` statement that counts the records in `purchaseEvents`
- Answer the corresponding quiz question in Coursera

```
SELECT count(*) FROM purchaseEvents2;
```

	count(1) ▲	
1	180678	

Showing all 1 rows.

Exercise 6: Find the location with the top purchase

Summary: Write a query to produce the city and state where the top purchase amount originated.

Steps to complete:

- Write a query, sorted by `purchases`, that shows the city and state of the top purchase
- **Answer the corresponding quiz question in Coursera**

```
SELECT city, state FROM purchaseEvents2 LIMIT 1;
```

	city ▲	state ▲	
1	Amarillo	TX	

Showing all 1 rows.

Challenge: Produce reports

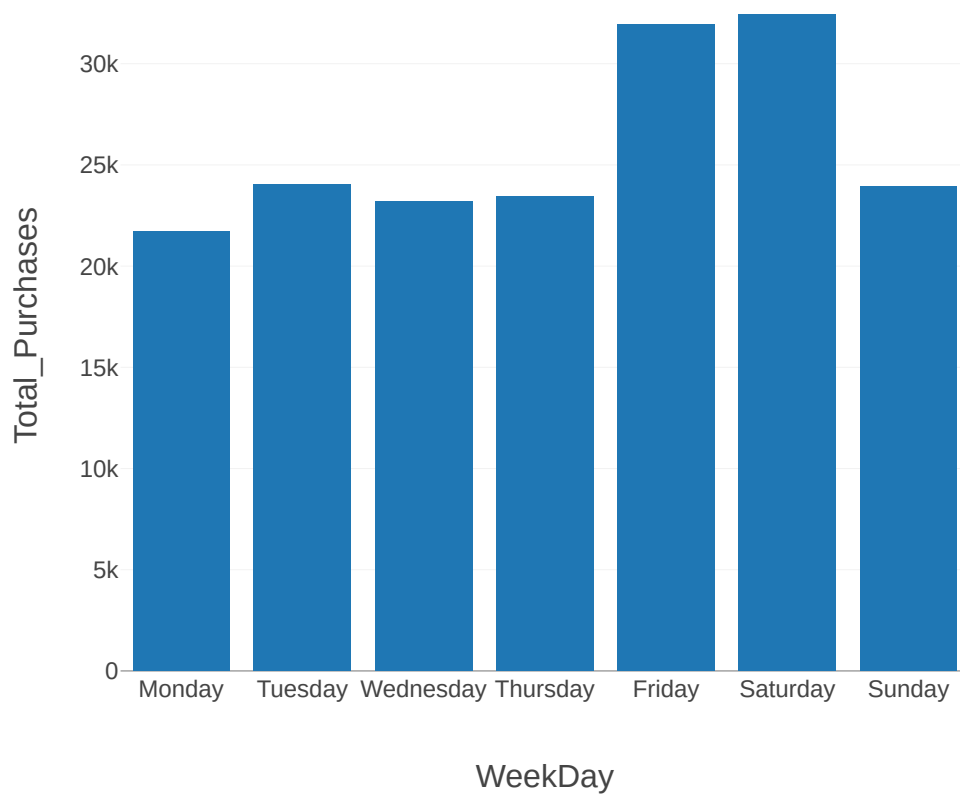
Summary: Use the `purchaseEvents` table to produce queries that explore purchase patterns in the table. Add visualizations to a dashboard to produce one comprehensive customer report.

Steps to complete:

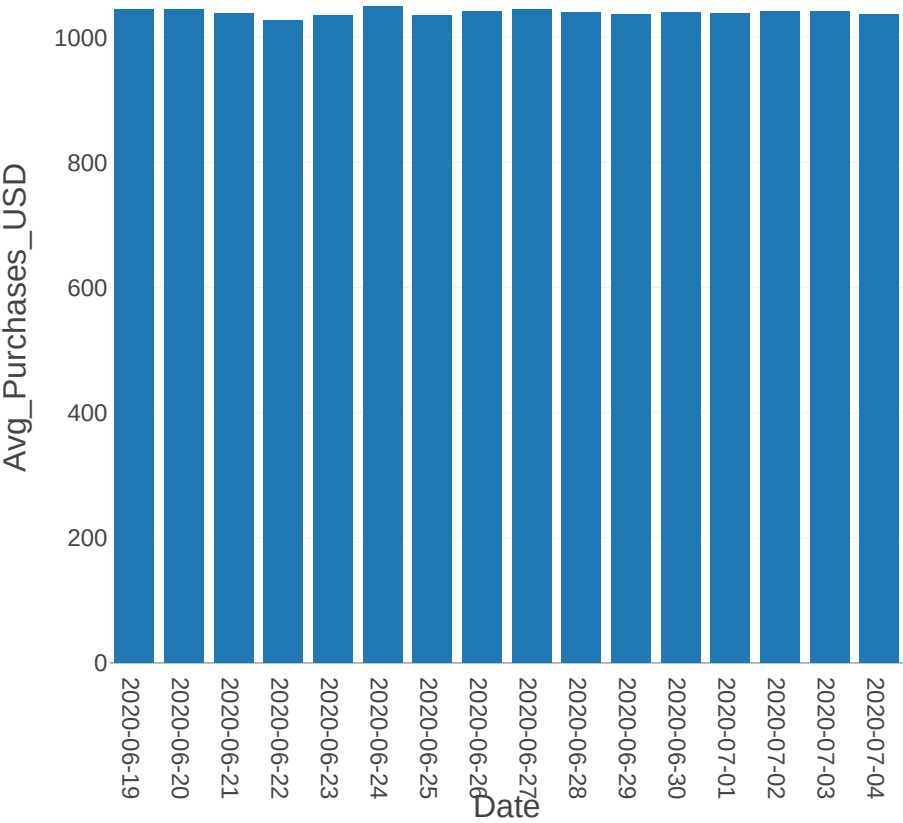
- Create visualizations to report on:
 - total purchases by day of week
 - average purchases by date of purchase
 - total purchases by state
 - Any other patterns you can find in the data
- Join your table with the data at the path listed below to get list of customers with confirmed email addresses
- **Answer the corresponding quiz question in Coursera**



```
--Total purchases by day of week
SELECT WeekDay, Total_Purchases
FROM
(
  SELECT date_format(split(from_unixtime(eventDate), ' ')[0], 'EEEE') AS WeekDay,
    count(purchases) AS Total_Purchases,
  CASE date_format(split(from_unixtime(eventDate), ' ')[0], 'EEEE')
    when 'Monday' then 0
    when 'Tuesday' then 1
    when 'Wednesday' then 2
    when 'Thursday' then 3
    when 'Friday' then 4
    when 'Saturday' then 5
    else 6
  END AS Index_
FROM purchaseEvents2
GROUP BY
date_format(split(from_unixtime(eventDate), ' ')[0], 'EEEE')
ORDER BY
Index_
)
```

```
--Average purchases by date of purchase
SELECT Date, ROUND(sum(purchases)/count(purchases) ,2) AS Avg_Purchases_USD
FROM
  (
    SELECT split(from_unixtime(eventDate), ' ')[0] AS Date,
    purchases
    FROM purchaseEvents2
    ORDER BY
    Date
  )
GROUP BY
Date
```



--TOD

%run ../Includes/Classroom-Cleanup

© 2020 Databricks, Inc. All rights reserved.
Apache, Apache Spark, Spark and the Spark logo are trademarks of the Apache Software Foundation (<http://www.apache.org/>).

