databricks3.4 Lab: Exploratory Data Analysis

Lab 1 - Exploratory Data Analysis

Module 3 Assignment

In this assignment you will:

- Create tables
- Create temporary views
- Write basic SQL queries to explore, manipulate, and present data
- Join two views and visualize the result

As you work through these exercises, 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**

```
res1: Boolean = false
```

```
res2: Boolean = false
```

Working with Retail Data

For this assignment, we'll be working with a generated dataset meant to mimic data collected for online retail transactions. It was added to your workspace when you ran the previous cell. You can use the following path to access the data: `/mnt/training/online_retail/data-001/data.csv`

Exercise 1: Create a table

Summary: Create a new table named `outdoorProducts` with the following schema:

Column Name	Type
invoiceNo	STRING
stockCode	STRING
description	STRING
quantity	INT
invoiceDate	STRING
unitPrice	DOUBLE
customerID	INT
countryName	STRING

Steps to complete:

- Make sure this notebook is idempotent by dropping any tables that have the name `outdoorProducts`
- Use `csv` as the specified data source
- Use the path provided above to access the data
- This data contains a header; include that in your table creation statement

```
DROP TABLE IF EXISTS outdoorProducts;
CREATE TABLE outdoorProducts (
  invoiceNo STRING,
  stockCode STRING,
  description STRING,
  quantity INT,
  invoiceDate STRING,
  unitPrice DOUBLE,
  customerID INT,
  countryName STRING
) USING csv OPTIONS (
  PATH "/mnt/training/online_retail/data-001/data.csv",
  header "true"
);
```

OK

```
SELECT * FROM outdoorProducts;
```

	invoiceNo	stockCode	description	qua
1	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6
2	536365	71053	WHITE METAL LANTERN	6
3	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8
4	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6
5	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6
6	536365	22752	SET 7 BABUSHKA NESTING BOXES	2
7	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6

Truncated results, showing first 1000 rows.

Exercise 2: Explore the data

Summary: Count the number of items that have a negative `quantity`

This table keeps track of online transactions, including returns. Some of the quantities in the `quantity` column show a negative number. Run a query that counts then number of negative values in the `quantity` column.

Steps to complete:

- Write a query that reports the number of values less than 0 in the `quantity` column

- **Report the answer in the corresponding quiz in Coursera**

```
SELECT count(*) FROM outdoorProducts
WHERE quantity < 0;
```

	count(1)
1	1192

Showing all 1 rows.

Exercise 3: Create a temporary view

Summary: Create a temporary view that includes only the specified columns and rows, and uses math to create a new column.

Steps to complete:

- Create a temporary view named `sales`
- Create a new column, `totalAmount`, by multiplying `quantity` times `unitPrice` and rounding to the nearest cent
- Include columns: `stockCode`, `quantity`, `unitPrice`, `totalAmount`, `countryName`
- Include only rows where `quantity` is greater than 0

```
CREATE
OR REPLACE TEMPORARY VIEW sales AS
SELECT
  stockCode,
  quantity, unitPrice,
  ROUND(quantity*unitPrice, 2) AS totalAmount,
  countryName
FROM
  outdoorProducts
WHERE
  quantity > 0;
```

OK

Exercise 4: Display ordered view

Summary: Show the view you created with `totalAmount` sorted greatest to least

Steps to complete:

- Select all columns form the view `sales`
- Order the `totalAmount` column from greatest to least
- **Report the `countryName` from the row with the greatest `totalAmount` of sales in the corresponding answer area in Coursera**

```
SELECT * FROM sales
ORDER BY totalAmount
DESC;
```

	stockCode	quantity	unitPrice	totalAmount	countryName
1	23166	74215	1.04	77183.6	United Kingdom
2	AMAZONFEE	1	13541.33	13541.33	United Kingdom
3	21108	3114	2.1	6539.4	United Kingdom
4	85123A	1930	2.55	4921.5	United Kingdom
5	48185	670	6.75	4522.5	United Kingdom
6	22470	1284	3.21	4121.64	United Kingdom
7	21623	600	6.38	3828	United Kingdom

Truncated results, showing first 1000 rows.

Exercise 5: View countries

Summary: Show a list of all unique `countryName` values in the `sales` view

Steps to complete:

- Write a query that returns only distinct `countryName` values
- **Answer the corresponding question in Coursera**

```
SELECT DISTINCT countryName FROM sales
ORDER BY countryName ASC;
```

	countryName

1	Australia
2	Austria
3	Bahrain
4	Belgium
5	Channel Islands
6	Cyprus
7	Denmark
8	EIRE
9	Finland
10	France
11	Germany
12	Iceland
13	Israel
14	Italy
15	Japan
16	Lithuania
17	Netherlands

Exercise 6: Create a temporary view: `salesQuants`

Summary: Create a temporary view that shows total `quantity` of items purchased from each `countryName`

Steps to complete:

- Create a temporary view named `salesQuants`
- Display the sum of all `quantity` values grouped by `countryName`. Name that column `totalQuantity`
- Order the view by `totalQuantity` from greatest to least
- **Answer the corresponding question in Coursera**

```
CREATE
OR REPLACE TEMPORARY VIEW salesQuants AS
SELECT
  countryName,
  SUM(quantity) AS totalQuantity
FROM
  outdoorProducts
GROUP BY
  countryName
ORDER BY
  totalQuantity DESC;
```

OK

```
SELECT * FROM salesQuants;
```

	countryName	totalQuantity
1	United Kingdom	455147
2	Netherlands	21381
3	EIRE	13177
4	France	11559
5	Germany	11218
6	Australia	5988
7	Japan	4048
8	Sweden	4006
9	Spain	3958
10	Norway	3582
11	Switzerland	3367
12	Portugal	2786
13	Belgium	2430
14	Finland	1254
15	Cyprus	1061
16	Italy	1038
17	Lithuania	652
18	Denmark	454
19	Poland	428

Exercise 7: Read in a new parquet table

Summary: Create a new table named `countryCodes`.

Steps to complete:

- Drop any existing tables named `countryCodes` from your database
- Use this path: `/mnt/training/countries/ISOCountryCodes/ISOCountryLookup.parquet` to create a new table using parquet as the data source. Name it `countryCodes`
- Include options to indicate that there **is** a header for this table

```
DROP TABLE IF EXISTS countryCodes;
CREATE TABLE countryCodes USING parquet OPTIONS (
  PATH "/mnt/training/countries/ISOCountryCodes/ISOCountryLookup.parquet",
  header "true"
);
```

OK

```
SELECT * FROM countryCodes;
```

	EnglishShortName	alpha2Code	alpha3Code
1	Afghanistan	AF	AFG
2	Åland Islands	AX	ALA
3	Albania	AL	ALB
4	Algeria	DZ	DZA
5	American Samoa	AS	ASM
6	Andorra	AD	AND
7	Angola	AO	AGO

Showing all 249 rows.

Exercise 8: View metadata

Summary: View column names and data types in this table.

Steps to complete:

- Use the `DESCRIBE` command to display all of column names and their data types

- **Answer the corresponding question in Coursera**

```
DESCRIBE countryCodes;
```

	col_name	data_type	comment
1	EnglishShortName	string	null
2	alpha2Code	string	null
3	alpha3Code	string	null
4	numericCode	string	null
5	ISO31662SubdivisionCode	string	null
6	independentTerritory	string	null

Showing all 6 rows.

Exercise 9: Join and Visualize

Summary: Use the `salesQuants` view and the `countryCodes` table to display a pie chart that shows total sales by country, and identifies the country by its 3-letter id.

Steps to complete:

- Write a query that results in two columns: `totalQuantity` from `salesQuants` and `alpha3Code` from `countryCodes`
- Join `countryCodes` with `salesQuants` on the name of country listed in each table
- Visualize your results as a pie chart that shows the percent of sales from each country

```
--TODO
```

Sanity Check

It's always smart to do a sanity check when manipulating and joining datasets.

- Compare your chart to the table you displayed in task #4

- Try the challenge problem to figure out what may have gone wrong

Challenge: Find the problem

Click here for a hint

```
-- TODO
```

```
-- TODO
```

```
CREATE
OR REPLACE TEMPORARY VIEW modCountryCodes AS
SELECT
  alpha3code,
  REPLACE (
    EnglishShortName,
    "United Kingdom of Great Britain and Northern Ireland",
    "United Kingdom"
  ) AS EnglishShortName
FROM
  countryCodes;
```

```
Error in SQL statement: AnalysisException: cannot resolve '`alpha3code`' giv
en input columns: [countrycodes.countryName, countrycodes.totalQuantity]; li
ne 4 pos 2;
'Project ['alpha3code, 'REPLACE('EnglishShortName, United Kingdom of Great B
ritain and Northern Ireland, United Kingdom) AS EnglishShortName#1278]
+- SubqueryAlias countrycodes
   +- View (`countryCodes`, [countryName#1280,totalQuantity#1281L])
      +- Project [cast(countryName#956 as string) AS countryName#1280, cast
(totalQuantity#1279L as bigint) AS totalQuantity#1281L]
         +- Sort [totalQuantity#1279L DESC NULLS LAST], true
            +- Aggregate [countryName#956], [countryName#956, sum(cast(quant
ity#952 as bigint)) AS totalQuantity#1279L]
               +- SubqueryAlias spark_catalog.default.outdoorproducts
                  +- Relation[invoiceNo#949,stockCode#950,description#951,qu
antity#952,invoiceDate#953,unitPrice#954,customerID#955,countryName#956] csv
```

```
-- TODO
```

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