System Requirements Specification Index

For

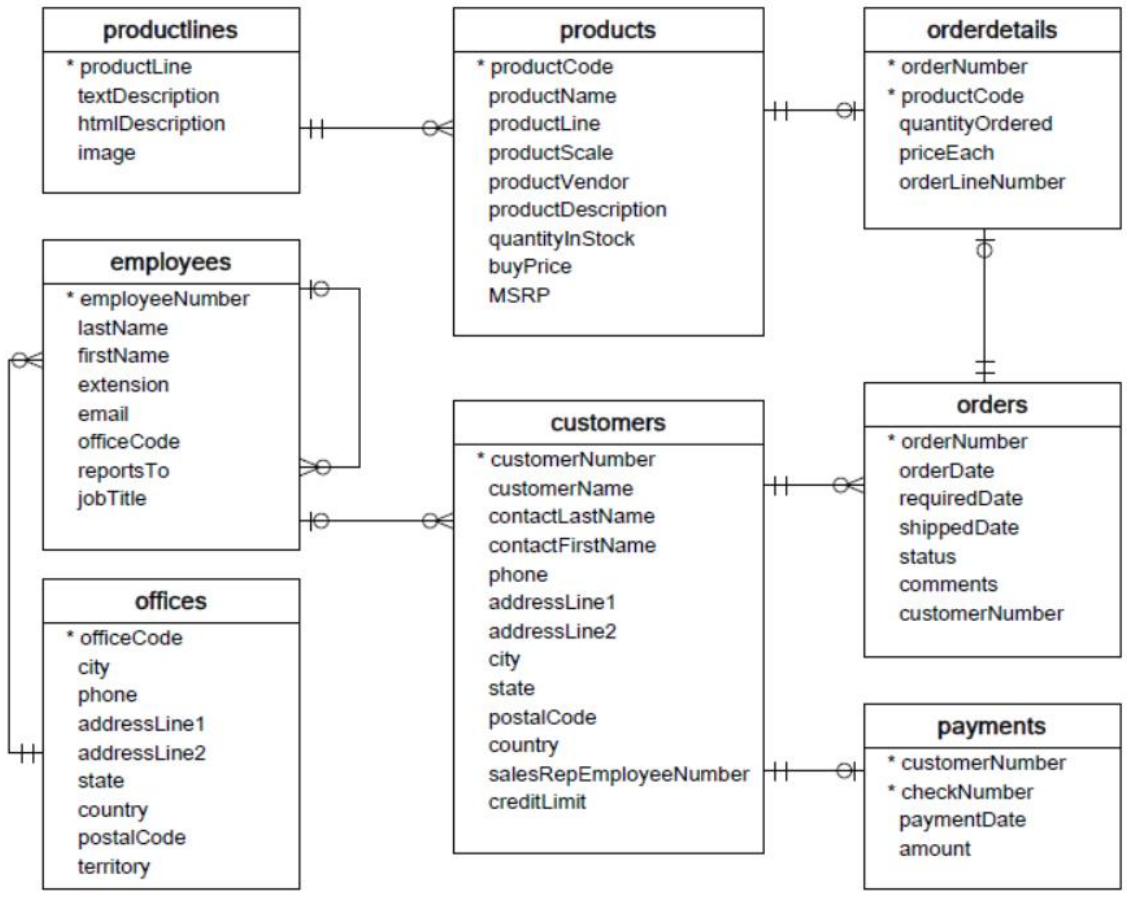
Data Engineering with Pyspark

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

Problem Statement : Perform basic load, transformation and data analytics using Pyspark.

You are given a database that you need to extract from multiple sources and perform some transformations and analysis. The database has the following the schema.



Customers: stores customer's data.
Products: stores a list of scale model cars.
ProductLines: stores a list of product line categories.
Orders: stores sales orders placed by customers.
OrderDetails: stores sales order line items for each sales order.
Payments: stores payments made by customers based on their accounts.
Employees: stores all employee information as well as the organisation structure such as who reports to whom.
Offices: stores sales office data.

The assessment contains the following folder structure.

DE_with_pyspark |

|--src

|--__init__.py

|-- **DE_with_pyspark.ipynb** – your code goes here

|--constants.py – defines few constants

|-- requirements.txt

|--data

|--data.csv – data files for the problem

|--docs – contains documents

|--DE with Pyspark Document.docx – this document

queries.ipynb

The **DE_with_pyspark.ipynb** has the following methods that you need to implement.

Important instructions:

- Please DO NOT change any method signature/filename, as it would result in failed submission.
- Please return the dataframe columns as specified in the comments with same names, order doesn't matter.
- For questions involving transformations just return the transformed data. Kindly not overwrite original data files.
- Your system comes with pyspark installed.
- Do not initiate multiple spark sessions. If terminal doesn't exit press CTRL+C

You will also find below signatures in the code notebook. Scroll down for execution instructions.

Load data:

```
Q1) def load_data_from_mysql(spark :pyspark.sql.SparkSession, db_name :str,
    table_name :str) -> pyspark.sql.DataFrame:
    '''
    - load data from MySQL table 'table_name' from database 'db_name'
      and return the table as a spark dataframe


     For the MySQL connection use below information:
```

```
    jdbc driver: 'com.mysql.cj.jdbc.Driver'
    Hostname: 'localhost'
    Port: 3306
    Database: 'classicmodels'
    Table_name: 'order_details'
    Username: 'mysql_user'
    Password: 'user'
    '''
```

```
Q2) def load_data_from_csv(spark :pyspark.sql.SparkSession, csv_file_name:
    str) -> pyspark.sql.DataFrame:
    '''
    Load data from CSV file 'csv_path' and return a spark Dataframe
    PS: The data files for this assignment are in 'data' folder
    You can access full path of 'data/' folder using 'DATA_FOLDER' variable
    from constants.py.
    '''
```

```
Q3) def load_data_from_flatfile(spark :pyspark.sql.SparkSession,
    txt_file_name: str) -> pyspark.sql.DataFrame:
    '''
    Load data from flat file 'txt_file_name' separated with ':' and return a
    spark Dataframe.
    PS: The data files for this assignment are in 'data' folder
    You can access full path of 'data/' folder using 'DATA_FOLDER' variable
    from constants.py
    '''
```

Transformations:

```
Q4) def clean_product_MSRP_column(spark :pyspark.sql.SparkSession) ->
    pyspark.sql.DataFrame:
    '''
    Due to a data entry issues MSRP, the selling price is lower than its
    buyPrice for some products. Change MSRP to 1.4 times of the buyPrice for
    such products and cast it to two decimal places.
    Note: Please do not change original file/dataframe.

    Return a spark dataframe with following columns.
    |productCode|productName|productLine|productScale|productVendor|productDes
    cription|quantityInStock|buyPrice|MSRP|
    '''
```

```
Q5) def explode_productLines_description(spark :pyspark.sql.SparkSession) ->
    pyspark.sql.DataFrame:
    '''
    Split the productLines description into words and explode the it.
```

```
    Return a spark dataframe with following columns.
    |productLine|description|
    +-----------+------------+
    |Motorcycles|         Our|
    |Motorcycles| motorcycles|
    |Motorcycles|         are|
       ... so on
    '''

Q6) def get_customer_info(spark :pyspark.sql.SparkSession) ->
    pyspark.sql.DataFrame:
    '''
    Return a consolidated customer info using structs.

    Return a spark dataframe with following columns.
    |custID|custName|country|#orders|totalMoneySpent|creditLimit|
    '''

Q7) def retrun_product_category(spark :pyspark.sql.SparkSession) ->
    pyspark.sql.DataFrame:
    '''
    Add a new column 'productCategory' using above udf and return the updated
    dataframe
    Note: Please do not change original file/dataframe.

    Return a spark dataframe with following columns.
    |productCode|productName|productLine|productScale|productVendor|productDes
    cription|quantityInStock|buyPrice|MSRP|productCategory|
    '''

Q8) def return_orders_by_day_by_status(spark :pyspark.sql.SparkSession) ->
    pyspark.sql.DataFrame:
    '''
    Return No. of orders by weekday(1 to 7) by order status.

    Return a spark dataframe with following columns.
    +-------+---------+--------+----------+-------+--------+-------+
    |weekDay|Cancelled|Disputed|In Process|On Hold|Resolved|Shipped|
    +-------+---------+--------+----------+-------+--------+-------+
    |      1|     null|    null|         2|      1|    null|      8|
           2
           3 and so on
    '''

Q9) def clean_productlines_description(spark :pyspark.sql.SparkSession) ->
    pyspark.sql.DataFrame:
    '''
    Take testDescription from productLines table and clean it.
    1. Convert to lower-case
    2. Remove all special characters except a-z and 0-9.
```

```
    3. Remove stop words (using
     StopWordsRemover.loadDefaultStopWords("english"))
    The final value(testDescription) must be a string.
    Note: Please do not change original file/dataframe.

    Return a spark dataframe with following columns.
    |productLine|htmlDescription|image|textDescription|
    '''
```

Analytics:

```
Q10) def return_top_5_employees(spark :pyspark.sql.SparkSession) ->
     pyspark.sql.DataFrame:
    '''
    Return the top 5 employees(Sales Representatives) who made maximum no. of
    sales throughout the given timeperiod. You can consider all sales with all
     statuses.

    Return a spark dataframe with following columns.
    employeeNumber|firstName|lastName |TotalOrders
    '''

Q11) def report_cancelled_orders(spark :pyspark.sql.SparkSession) ->
     pyspark.sql.DataFrame:
    '''
    Return email adresses of your manager to send a report about the cancelled
    orders. Return
    Order details and a column to define if the cancelled order has been
    shipped already or not('Yes' or  'No').

    Return a spark dataframe with following columns.
    +-----------+---------+-------------------+-------------+--------------
    -------+---------+--------------------+
    |orderNumber|isShipped|comments|customerNumber|salesRepEmployeeNumber|repo
    rtsTo|email|
    +-----------+---------+-------------------+-------------+--------------
    -------+---------+--------------------+
    |10167|No|Customer called t...|  448| 1504| 1102|gbondur@classicmo...|
           ...
           ... so on
    '''

Q12) def return_top_5_big_spenders(spark :pyspark.sql.SparkSession) ->
    pyspark.sql.DataFrame:
    '''
    Return top 5 big spenders who had spent the most(highest order value).

    Return a spark dataframe with following columns.
```

```
    customerNumber| customerName| totalOrderValue|
    '''

Q13) def return_top_5_big_spend_countries(spark :pyspark.sql.SparkSession) ->
     pyspark.sql.DataFrame:
    '''
    Return top 5 big countries which had spent the most $(highest order
    value).

    Return a spark dataframe with following columns.
    |country|totalOrderValue|
    '''
```

Setting up the environment:

Login to the MySQL shell of Workbench and run below commands to create a mysql table. (Credentials for the same are available README.txt file on Desktop)

```
1. create database classicmodels;

2. use classicmodels;

3. create table orderdetails(orderNumber int,
productCode varchar(100),
quantityOrdered int,
priceEach decimal(10,2),
orderLineNumber int
);

# copy data/orderdetails.csv to 'C:/ProgramData/MySQL/MySQL
Server8.0/Uploads/'

4. LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server
8.0/Uploads/orderdetails.csv'
INTO TABLE orderdetails
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES

5. select * from orderdetails; -test
```

NOTE: mysql-connector-jar file is available at: "D:/…". Pls make sure you copy the same in C driver root.

Execution Steps to Follow:

1. All actions like build, compile, running application, running test cases will be through Command Terminal.

2. These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.
3. To setup environment:
   pip install -r requirements.txt --user
   (Import other libraries, if needed)
4. Add your code in src/DE_with_pyspark.ipynb:
5. To run Test cases:
   python -m pytest tests

-----x-----