**Problem Description**

Source Data to process

**1. order\_detail.csv**

page2image4400064

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Note** |
| order\_created\_timestamp | timestamp | format YYYY-MM-DD  HH:MM:SS |
| status | string |  |
| price | integer |  |
| discount | float |  |
| id | string |  |
| driver\_id | string |  |
| user\_id | string |  |
| restaurant\_id | string |  |

**2. restaurant\_detail.csv**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Name** | **Type** | **Note** |
| id | string |  |
| restaurant\_name | string |  |
| category | string |  |
| esimated\_cooking\_time | float |  |
| latitude | float |  |
| longitude | float |  |

Business Requirements  
● Create two tables in postgres database with the above given column

types.  
○ order\_detail table using **order\_detail.csv**

○ restaurant\_detail table using **restaurant\_detail.csv**

● Once we have these two tables in postgres DB, ETL the same tables to Hive with the same names and corresponding Hive data type using the below guidelines

* ○  Both the tables should be **external table**
* ○  Both the tables should have **parquet file format**
* ○  restaurant\_detail table should be partitioned by a column name

**dt** (type string) with a static value **latest**

* ○  order\_detail table should be partitioned by a column named **dt**

(type string) extracted from **order\_created\_timestamp** in the format **YYYYMMDD**

Example of dt column

● After creating the above tables in Hive, create two new tables \_\_order\_detail\_new\_\_ and \_\_restaurant\_detail\_new\_\_ with their respective columns and partitions and add one new column for each table as explained below.

page3image4479904

order\_created\_timestamp: "2019-06-08 17:31:57"

page3image4562032

dt: "20190608"

|  |  |  |
| --- | --- | --- |
| **Table Name** | **New Column Name** | **Logic** |
| order\_detail | discount\_no\_null | replace all the NULL values of discount  column with 0 |
| restaurant\_det  ail | cooking\_bin | using esimated\_cooking\_time column and the  below logic |

|  |  |
| --- | --- |
| **esimated\_cooking\_time** | **cooking\_bin** |
| 10-40 | 1 |
| 41-80 | 2 |
| 81-120 | 3 |
| greater than 120 | 4 |

Final column count of each table (including partition column): 1. order\_detail = 9  
2. restaurant\_detail = 7  
3. order\_detail\_new = 10

4. restaurant\_detail\_new = 8

**SQL requirements**

* ●  Get the average discount for each category
* ●  Row count per each cooking\_bin

**CSV output requirements**

Save the above query output to CSV files name discount.csv and cooking.csv.

**Technical Requirements**

* ●  Use Apache Spark, Apache Sqoop or any other big data frameworks
* ●  Use a scheduler tool to run the pipeline daily. Airflow is preferred
* ●  Include a README file that explains how we can deploy your code
* ●  (bonus) Use Docker or Kubernetes for up-and-running program

Question output

1. Source code
2. Docker, docker-compose, kubernetes files if possible.
3. README of how to test / run