```
containerName = "ecommercecontainer"
storageAccountName = "ecommercedataset"
accountkey =
"QH5MFe3NP8AMpzSRABR44l4/f0VUoxVMX6eqarbW7czDWBVzFAxerxFfQDlE99+seDD8m
3QpVnuB+ASthfLbRw==" #Copied from Accesskeys
config = "fs.azure.sas." + containerName+ "." + storageAccountName +
".blob.core.windows.net"
spark.conf.set("fs.azure.account.key.
{storage}.dfs.core.windows.net".format(storage=storageAccountName),
accountkey)
PATH_TEMPLATE = "abfss://{container}@{storage}.dfs.core.windows.net"
RAW_PATH = PATH_TEMPLATE.format(container=containerName,
storage=storageAccountName)
RAW_FOLDER_PATH = '/olist_public_dataset.csv'
PATH=RAW_PATH+RAW_FOLDER_PATH
print(RAW_PATH+RAW_FOLDER_PATH)
```

```
abfss://ecommercecontainer@ecommercedataset.dfs.core.windows.net/
olist_public_dataset.csv
```

```
df=spark.read.csv(PATH,sep=',',inferSchema=True,header=True)
display(df)
```

```
#Create a new directory in HDFS and copy the data from DF into DBFS
%fs mkdirs /ecommerce


%fs ls ecommerce/
```

```
#writing data into DBFS
df.write.csv('/ecommerce/raw2_source/')
```

```
%fs ls ecommerce/raw2_source/
```

```
#Lets create table using data in DBFS
%sql
create table ecommerce_table
(id int, order_status string, order_products_value float,
order_freight_value float, order_items_qty int,
 customer_city string, customer_state string,customer_zip_code_prefix
int, product_name_lenght int,
 product_description_lenght int, product_photos_qty int, review_score
int, order_purchase_timestamp string,
 order_aproved_at string, order_delivered_customer_date string) USING
csv OPTIONS (PATH "dbfs:/ecommerce/raw2_source/")
```

```
#query the data
%sql
SELECT *
FROM default.ecommerce_table
LIMIT 10
```

```
from pyspark.sql.functions import
col,to_timestamp,dayofmonth,weekofyear,to_date,coalesce

df = spark.table("ecommerce_table")


# Convert timestamp string to date type
df =
df.withColumn("order_purchase_timestamp",to_timestamp(col("order_purch
ase_timestamp"), 'dd/MM/yy H:mm'))

df=df.withColumn("day", dayofmonth("order_purchase_timestamp"))

df=df.withColumn("week", F.weekofyear("order_purchase_timestamp"))

df=df.createOrReplaceTempView("ecomview")

spark.sql("select id,
order_status,sum(order_products_value),sum(order_freight_value),custom
er_city,day from ecomview group by id,
order_status,customer_city,day").write.csv(RAW_PATH+'/proceessed_data1
')

spark.sql("select id,
order_status,sum(order_products_value),sum(order_freight_value),custom
er_city,week from ecomview group by id,
order_status,customer_city,week").write.csv(RAW_PATH+'/proceessed_data
2')

#Total sales and order distribution per day and week for each state
spark.sql('select id,
order_status,sum(order_products_value),sum(order_freight_value),custom
er_state,week from ecomview group by id,
order_status,customer_state,week').write.csv(RAW_PATH+'/proceessed_dat
a3')

spark.sql("select id,
order_status,sum(order_products_value),sum(order_freight_value),custom
er_state,day from ecomview group by id,
order_status,customer_state,day").write.csv(RAW_PATH+'/proceessed_data
4')

# Average review score, average freight value, average order approval,
and delivery time
spark.sql('select avg(review_score), avg(order_freight_value),
avg(order_products_value),order_delivered_customer_date from ecomview
group by
review_score,order_freight_value,order_products_value,order_delivered_
customer_date').write.csv(RAW_PATH+'/proceessed_data5')

#The freight charges per city and total freight charges
spark.sql('select order_freight_value,customer_city from ecomview
group by
order_freight_value,customer_city').write.csv(RAW_PATH+'/proceessed_da
ta6')

#Total freight charg
spark.sql("select sum(order_freight_value) from
ecomm_view").write.csv(RAW_PATH+'/proceessed_data7')
```