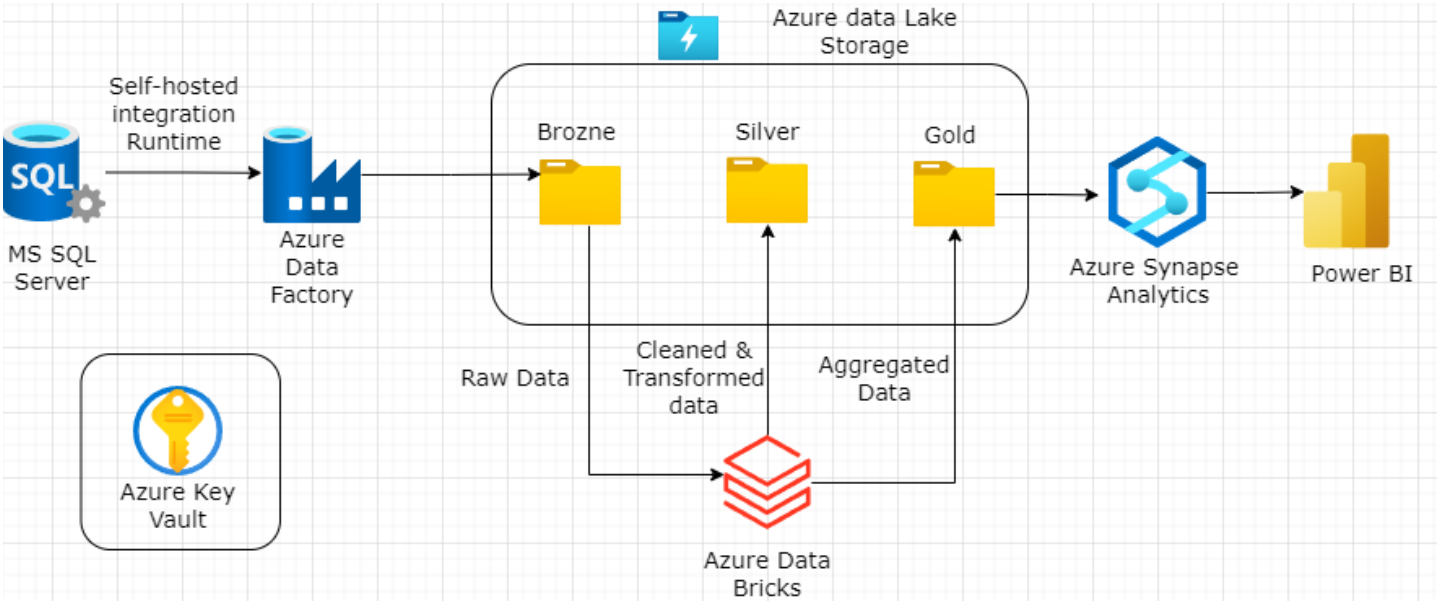
Azure data Lake Storage
Self-hosted integration Runtime
SQL
MS SQL Server
Azure Data Factory
Brozne
Silver
Gold
Azure Synapse Analytics
Power BI
Raw Data
Cleaned & Transformed data
Aggregated Data
Azure Key Vault
Azure Data Bricks

Mount ADLS

In [0]:

```
dbutils.fs.unmount("/mnt/adventureworks")
```

```
/mnt/adventureworks has been unmounted.
Out[1]: True
```

In [0]:

```
configs = {"fs.azure.account.auth.type": "OAuth",
          "fs.azure.account.oauth.provider.type": "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",
          "fs.azure.account.oauth2.client.id": "<application-id>",
          "fs.azure.account.oauth2.client.secret": "",
          "fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/<directory-id>/oauth2/token"}


dbutils.fs.mount(
  source = "abfss://<container-name>@<storage-account-name>.dfs.core.windows.net/",
  mount_point = "/mnt/<mount-name>",
  extra_configs = configs)
```

Import funtction and types

In [0]:

```
from pyspark.sql.functions import from_utc_timestamp,date_format,col,to_date,month,year,concat,substring,lit,sum,round,count,countDistinct,lag,lead,coalesce,datediff,min,max,avg,ceil,when
from pyspark.sql.types import TimestampType
from pyspark.sql.window import Window
```

list of tables

In [0]:

```
tablenames=[]
for i in dbutils.fs.ls("dbfs:/mnt/adventureworks/Bronze/"):
    tablenames.append(i.name.split('/')[-1])
tablenames
```

```
Out[4]: ['CreditCard',
 'Customer',
 'SalesOrderDetail',
 'SalesOrderHeader',
 'SalesPerson',
 'SalesTerritory',
 'SpecialOfferProduct',
 'Store']
```

table headers

In [0]:

```
sales_tables_columns = {
    "SalesOrderHeader": [
        "SalesOrderID", "RevisionNumber", "OrderDate", "DueDate", "ShipDate", "Status", "OnlineOrderFlag",
        "SalesOrderNumber", "PurchaseOrderNumber", "AccountNumber", "CustomerID", "SalesPersonID",
        "TerritoryID", "BillToAddressID", "ShipToAddressID", "ShipMethodID", "CreditCardID",
```

```
        "CreditCardApprovalCode", "CurrencyRateID", "SubTotal", "TaxAmt", "Freight", "To
talDue",
        "Comment", "rowguid", "ModifiedDate"
    ],
    "SalesOrderDetail": [
        "SalesOrderID", "SalesOrderDetailID", "CarrierTrackingNumber", "OrderQty", "Prod
uctID",
        "SpecialOfferID", "UnitPrice", "UnitPriceDiscount", "LineTotal", "rowguid", "Mod
ifiedDate"
    ],
    "Customer": [
        "CustomerID", "PersonID", "StoreID", "TerritoryID", "AccountNumber", "rowguid",
"ModifiedDate"
    ],
    "SalesPerson": [
        "SalesPersonID", "TerritoryID", "SalesQuota", "Bonus", "CommissionPct", "SalesYT
D",
        "SalesLastYear", "rowguid", "ModifiedDate"
    ],
    "SpecialOfferProduct": [
        "SpecialOfferID", "ProductID", "rowguid", "ModifiedDate"
    ],
    "Store": [
        "BusinessEntityID", "Name", "SalesPersonID", "Demographics", "rowguid", "Modifie
dDate"
    ],
    "SalesTerritory": [
        "TerritoryID", "Name", "CountryRegionCode", "Group", "SalesYTD", "SalesLastYear"
,
        "CostYTD", "CostLastYear", "rowguid", "ModifiedDate"
    ],
    "CreditCard": [
        "CreditCardID", "CardType", "CardNumber", "ExpMonth", "ExpYear", "ModifiedDate"
    ]
}
```

In [0]:

```
%fs ls "dbfs:/mnt/adventureworks/"
```

path	name	size	modificationTime
dbfs:/mnt/adventureworks/Bronze/	Bronze/	0	1727801721000
dbfs:/mnt/adventureworks/Gold/	Gold/	0	1727801741000
dbfs:/mnt/adventureworks/Silver/	Silver/	0	1727801731000

Create dataframe

In [0]:

```
for table in tablenames:
    columns=sales_tables_columns[table]
    df=spark.read.format("csv").\
        option("header","false").\
            option("inferSchema","true").\
                load("dbfs:/mnt/adventureworks/Bronze/"+table)

    for i, col in enumerate(columns):
        if "Date" in col or "date" in col:
            df = df.withColumnRenamed(df.columns[i], col) \
               .withColumn(col, date_format(from_utc_timestamp(col, "UTC").cast(Timestam
pType()), "yyyy-MM-dd"))\
                    .withColumn(col,to_date(col,"yyyy-MM-dd"))

        else:
        # Just rename the column
            df = df.withColumnRenamed(df.columns[i], col)
```

```
    locals()[f"{table}_df"]=df
```

data validation remove dublicates,null values

In [0]:

```
Customer_df=Customer_df.dropna(subset=["customerid"])
SalesPerson_df=SalesPerson_df.dropna(subset=["SalesPersonID"])\
    .dropDuplicates(["salesPersonid"])
SalesOrderHeader_df=SalesOrderHeader_df.\
    dropna(subset=["orderdate"])\
        .withColumn("Ordermonth",month("orderdate"))\
            .withColumn("orderYear",year("orderdate"))
SalesOrderDetail_df = SalesOrderDetail_df.filter((col("OrderQty") != 0) | (col("UnitPric
e") > 0))
```

mask card number

In [0]:

```
CreditCard_df=CreditCard_df.withColumn(
    "maskedcardnumber",
    concat(lit("xxxx-xxxx-xxxx-"),substring(col("cardnumber"),-4,4))
    ).drop(col("cardnumber"))
```

save transformed file in silver layer

In [0]:

```
for i in tablenames:
    df_name=f'{i}_df'
    df=globals()[df_name]
    df.coalesce(1).\
        write.mode('overwrite')\
            .format('parquet')\
                .option('header','true')\
                    .save(f'dbfs:/mnt/adventureworks/Silver/{i}')
```

In [0]:

```
for i in dbutils.fs.ls('dbfs:/mnt/adventureworks/Silver/'):
    filename=i.path.split('/')[-2]
    df=spark.read.format('parquet').load(f'dbfs:/mnt/adventureworks/Silver/{filename}/')
    locals()[f's_{filename}_df']=df
```

total sales per month how do they compare year-over-year

In [0]:

```
salesjoined=s_SalesOrderHeader_df.join(
    s_SalesOrderDetail_df,
    s_SalesOrderDetail_df["salesorderid"]==s_SalesOrderHeader_df["salesorderid"],
    "inner"
).drop(s_SalesOrderDetail_df["salesorderid"])
salestotal=salesjoined.withColumn("orderMonth",month("orderdate"))\
    .withColumn("orderyear",year("orderdate"))\
        .withColumn("discountprice",col("unitprice")*col("unitpricediscount"))\
            .groupBy(["orderyear","ordermonth"]).agg(round(sum(col("orderqty")*(col("uni
tprice")-col("discountprice"))),2).alias("totalSales"))\
                .orderBy(col("orderyear"),col("ordermonth"))
salestotal.show()
```

```
+---------+----------+----------+
|orderyear|ordermonth|totalSales|
+---------+----------+----------+
|     2011|         5| 503805.92|
|     2011|         6| 458910.82|
|     2011|         7| 2044600.0|
|     2011|         8|2495816.73|
```

```
|      2011|          9| 502073.85|
|      2011|         10|4588761.82|
|      2011|         11| 737839.82|
|      2011|         12|1309863.25|
|      2012|          1|3970627.28|
|      2012|          2|1475426.91|
|      2012|          3|2975748.24|
|      2012|          4| 1634600.8|
|      2012|          5|3074602.81|
|      2012|          6|4099354.36|
|      2012|          7|3417953.87|
|      2012|          8|2175637.22|
|      2012|          9|3454151.94|
|      2012|         10|2544091.11|
|      2012|         11|1872701.98|
|      2012|         12|2829404.82|
+---------+----------+----------+
only showing top 20 rows
```

In [0]:

```
salestotal.write.format("parquet").mode("overwrite").option("header","true").save("dbfs:/mnt/adventureworks/Gold/salestotal")
```

sales percentage compared with previous year

In [0]:

```
salespercentageLastyear=salestotal.alias("c").join(
    salestotal.alias("p"),
    (col("c.ordermonth")==col("p.ordermonth")) & (col("c.orderyear")==col("p.orderyear")+1)
    ,"left"
)\
    .withColumn("percentageIncrease",round((col("c.totalsales")-col("p.totalsales"))*100/col("p.totalsales"),2))\
        .select([
            col("c.orderyear"),
            col("c.ordermonth"),
            col("c.totalsales").alias("current month sales"),
            col("p.totalsales").alias("previous year sale"),
            col("percentageIncrease")
        ])\
            .orderBy(col("c.orderyear"),col("c.ordermonth"))
salespercentageLastyear.show()
```

```
+---------+----------+-------------------+------------------+------------------+
|orderyear|ordermonth|current month sales|previous year sale|percentageIncrease|
+---------+----------+-------------------+------------------+------------------+
|      2011|          5|          503805.92|              null|              null|
|      2011|          6|          458910.82|              null|              null|
|      2011|          7|           2044600.0|              null|              null|
|      2011|          8|         2495816.73|              null|              null|
|      2011|          9|          502073.85|              null|              null|
|      2011|         10|         4588761.82|              null|              null|
|      2011|         11|          737839.82|              null|              null|
|      2011|         12|         1309863.25|              null|              null|
|      2012|          1|         3970627.28|              null|              null|
|      2012|          2|         1475426.91|              null|              null|
|      2012|          3|         2975748.24|              null|              null|
|      2012|          4|          1634600.8|              null|              null|
|      2012|          5|         3074602.81|         503805.92|            510.28|
|      2012|          6|         4099354.36|         458910.82|            793.28|
|      2012|          7|         3417953.87|          2044600.0|             67.17|
|      2012|          8|         2175637.22|        2495816.73|            -12.83|
|      2012|          9|         3454151.94|         502073.85|            587.98|
|      2012|         10|         2544091.11|        4588761.82|            -44.56|
|      2012|         11|         1872701.98|         737839.82|            153.81|
|      2012|         12|         2829404.82|        1309863.25|            116.01|
+---------+----------+-------------------+------------------+------------------+
```

```
only showing top 20 rows
```

In [0]:

```
salespercentageLastyear.write.format("parquet").mode("overwrite").option("header","true")
.save("dbfs:/mnt/adventureworks/Gold/salespercentageLastyear")
```

High-Value Customers: Customers who consistently make high-value purchases and contribute significantly to revenue.

In [0]:

```
windowspec=Window.orderBy(col("totalsales").desc())

customer_sales=s_Customer_df.alias("c").join(
    salesjoined.alias("s"),
    col("c.customerid")==col("s.customerid"),
    "left"
).drop(col("s.customerid"))
high_value_customers=customer_sales.withColumn("discountprice",col("s.unitprice")*col("s.
unitpricediscount"))\
    .groupBy("c.customerid").agg(round(sum(col("orderqty")*(col("unitprice")-col("discoun
tprice"))),2).alias
          ("totalSales"),
          countDistinct(col("salesorderid")).alias("no od orders"))\
    .orderBy(col("totalsales").desc())

high_value_customers.show()
```

```
+----------+----------+------------+
|customerid|totalSales|no od orders|
+----------+----------+------------+
|     29818| 877107.19|          12|
|     29715| 853849.18|          12|
|     29722| 841908.77|          12|
|     30117| 816755.58|          12|
|     29614|  799277.9|          12|
|     29639| 787773.04|          12|
|     29701| 746317.53|           8|
|     29617| 740985.83|          12|
|     29994| 730798.71|          12|
|     29646| 727272.65|          12|
|     29580| 724299.64|          12|
|     29827| 711864.76|          12|
|     29497| 700803.79|          12|
|     29716| 693502.49|          12|
|     29913| 671618.03|           8|
|     30103|  643745.9|           8|
|     29957| 636226.47|           8|
|     29523| 618616.13|          12|
|     29616| 617340.46|           8|
|     30048| 602559.89|           8|
+----------+----------+------------+
only showing top 20 rows
```

In [0]:

```
totalrevenue=high_value_customers.agg(round(sum("totalsales"),2).alias("total")).collect
()[0]["total"]
```

In [0]:

```
high_value_customers.write.format("parquet").mode("overwrite").option("header","true").s
ave("dbfs:/mnt/adventureworks/Gold/high_value_customers")
```

first and last purchase of customer and days in between

In [0]:

```
windowspec1=Window.partitionBy("customerid").orderBy(col("orderdate"))
s_Customer_df.join(
    s_SalesOrderHeader_df,
    s_Customer_df["customerid"] == s_SalesOrderHeader_df["customerid"],
    "inner"
).drop(s_SalesOrderHeader_df["customerid"])\
  .select("customerid", "orderdate")\
  .groupBy("customerid").agg(min(col("orderdate").alias("first purchase")),max(col("order
date").alias("last purchase")))\
  .show()
```

```
+----------+----------------------------------+----------------------------------+
|customerid|min(orderdate AS `first purchase`)|max(orderdate AS `last purchase`)|
+----------+----------------------------------+----------------------------------+
|     11033|                        2011-07-18|                        2014-04-07|
|     29834|                        2011-08-01|                        2014-05-01|
|     28088|                        2011-08-06|                        2011-08-06|
|     28170|                        2011-08-18|                        2011-08-18|
|     28664|                        2011-10-01|                        2014-03-12|
|     14832|                        2011-11-14|                        2014-04-12|
|     22373|                        2011-11-20|                        2014-01-31|
|     29285|                        2012-01-10|                        2012-01-10|
|     11317|                        2012-02-01|                        2013-07-21|
|     15619|                        2012-03-14|                        2014-05-20|
|     12027|                        2012-04-11|                        2013-12-26|
|     29744|                        2012-05-30|                        2014-03-01|
|     29601|                        2012-06-30|                        2014-03-31|
|     29993|                        2012-06-30|                        2013-03-30|
|     28836|                        2012-07-17|                        2012-07-17|
|     13840|                        2012-07-27|                        2013-12-28|
|     29719|                        2012-09-30|                        2014-03-31|
|     26623|                        2012-11-20|                        2014-01-07|
|     26708|                        2012-11-30|                        2014-02-10|
|     17753|                        2012-12-06|                        2013-10-21|
+----------+----------------------------------+----------------------------------+
only showing top 20 rows
```

Order Fulfillment Analysis:

What is the average order fulfillment time, and how does it vary across different products or regions?

In [0]:

```
salesjoined\
  .select("productid","TerritoryID","orderdate","shipdate")\
  .withColumn("Diff",datediff(col("shipdate"),col("orderdate")))\
  .groupBy("productid","TerritoryID").agg(ceil(avg("diff")).alias("avg"))\
  .orderBy(ceil(col("avg").alias("avg delivered day")).desc())\
  .show()
```

```
+---------+-----------+---+
|productid|TerritoryID|avg|
+---------+-----------+---+
|      861|          8|  8|
|      765|          8|  8|
|      863|          8|  8|
|      761|          8|  8|
|      763|          8|  8|
|      852|          8|  8|
|      730|          8|  8|
|      712|          8|  8|
|      792|          8|  8|
|      855|          8|  8|
|      715|          8|  8|
|      760|          8|  8|
|      726|          8|  8|
|      762|          8|  8|
|      862|          8|  8|
|      856|          8|  8|
```

```
|      711|          8|  8|
|      836|          8|  8|
|      854|          8|  8|
|      835|          8|  8|
+---------+-----------+---+
only showing top 20 rows
```

Credit Card Usage Trends:

How does the usage of different credit card types vary across different customer segments and purchase amounts?

In [0]:

```
sales_creditcard_df=s_SalesOrderHeader_df\
    .select("salesorderid","customerid","creditcardid","totaldue")\
    .alias("s").join(
        s_CreditCard_df.alias("c"),
        col("s.creditcardid")==col("c.creditcardid"),
        "inner"
    )
```

In [0]:

```
credit_card_trent=sales_creditcard_df\
    .withColumn("category",
                when(col("totaldue") < 5000, "Budget Spender")
                .when((col("totaldue")>=5000) & (col("totaldue")<50000)  ,"Value Seeker"
)
                .when((col("totaldue")>=50000) & (col("totaldue")<100000) ,"Premium Shop
per")
                .otherwise("Luxury Spender")
                )\
    .groupBy("category","cardtype").agg(count("salesorderid").alias("count of category")
)\
    .orderBy(col("category"),col("cardtype"))
credit_card_trent.show()
```

```
+---------------+-------------+-----------------+
|       category|     cardtype|count of category|
+---------------+-------------+-----------------+
| Budget Spender|ColonialVoice|             6982|
| Budget Spender|  Distinguish|             7129|
| Budget Spender| SuperiorCard|             7120|
| Budget Spender|        Vista|             6836|
| Luxury Spender|ColonialVoice|               34|
| Luxury Spender|  Distinguish|               23|
| Luxury Spender| SuperiorCard|               27|
| Luxury Spender|        Vista|               24|
|Premium Shopper|ColonialVoice|              164|
|Premium Shopper|  Distinguish|               87|
|Premium Shopper| SuperiorCard|              113|
|Premium Shopper|        Vista|              149|
|   Value Seeker|ColonialVoice|              498|
|   Value Seeker|  Distinguish|              351|
|   Value Seeker| SuperiorCard|              374|
|   Value Seeker|        Vista|              423|
+---------------+-------------+-----------------+
```

In [0]:

```
credit_card_trent.write.format("parquet").mode("overwrite").option("header","true").save
("dbfs:/mnt/adventureworks/Gold/credit_card_trent")
```

Special Offers Effectiveness:

What is the impact of special offers on sales volume and revenue? Are certain offers more effective than others?

In [0]:

```
offer_sales_df=s_SalesOrderDetail_df.alias("s")\
    .join(s_SpecialOfferProduct_df.alias("d"),
          (col("s.productid")==col("d.productid")) & (col("s.SpecialOfferID")==col("d.SpecialOfferID")),
          "left"
          )\
    .withColumn("offer product",when(col("d.rowguid").isNotNull(),"yes").otherwise("No"))\
    .drop(col("d.SpecialOfferID"),col("d.productid"),col("d.modifieddate"),col("d.rowguid"))


offer_effectiveness=offer_sales_df.select("s.SalesOrderID","s.SalesOrderDetailID","s.OrderQty","s.linetotal","s.ProductID","s.SpecialOfferID")\
    .filter(col("d.rowguid").isNotNull())\
    .groupBy("s.specialofferid").agg(round(sum("s.linetotal"),2).alias("total revenue"),count(col("s.productid")).alias("total product sold"))\
    .orderBy(col("total revenue").desc(),col("total product sold").desc())
offer_effectiveness.show()
```

```
+--------------+--------------+------------------+
|specialofferid| total revenue|total product sold|
+--------------+--------------+------------------+
|             1|1.0237262226E8|            115884|
|             2|    4896451.91|              3428|
|             3|    1037643.33|               606|
|            14|     612324.54|               244|
|            13|     458091.19|               524|
|             7|      250927.7|               137|
|             4|     124148.53|                80|
|             9|      49986.08|                61|
|            16|      25899.14|               169|
|            11|        9100.9|                84|
|             8|       7448.83|                98|
|             5|       1736.99|                 2|
+--------------+--------------+------------------+
```

In [0]:

```
offer_effectiveness.write.format("parquet").mode("overwrite").option("header","true").save("dbfs:/mnt/adventureworks/Gold/Offers_Effectiveness")
```

How do special offers influence customer retention and repeat purchases?

In [0]:

```
windowspec2=Window.partitionBy("s.customerid").orderBy("s.orderdate")
fact_sales=s_SalesOrderHeader_df.alias("s")\
    .join(
        offer_sales_df.alias("o"),
        col("s.SalesOrderID")==col("o.SalesOrderID")
        ,"inner"
        )\
    .withColumn("first purchase",min(col("s.orderdate")).over(windowspec2))\
    .drop(col("o.SalesOrderID"),col("s.modifieddate"),col("s.rowguid"))
customer_retension=fact_sales.filter(col("s.orderdate")>col("first purchase"))\
        .groupBy("s.customerid").agg(count("s.SalesOrderID").alias("no of orders")
                                     ,countDistinct("o.ProductID").alias("no of products")
                                     ,round(sum("s.TotalDue"),2).alias("revenue"))
customer_retension.show()
```

```
+----------+------------+--------------+--------+
|customerid|no of orders|no of products| revenue|
+----------+------------+--------------+--------+
|     11000|           7|             7|19027.09|
|     11001|          10|             9|18647.34|
|     11002|           3|             3| 7882.09|
|     11003|           8|             8|20947.71|
```

```
|     11004|           5|             5|13225.74|
|     11005|           5|             5|13075.63|
|     11006|           4|             4|10374.82|
|     11007|           7|             7|18561.65|
|     11008|           6|             6|15566.31|
|     11009|           4|             4|10423.11|
|     11010|           3|             3| 7824.63|
|     11011|           3|             3| 7924.08|
|     11012|           2|             2|   13.88|
|     11013|           3|             3|  248.56|
|     11014|           4|             4|  439.66|
|     11017|           3|             3|  5940.8|
|     11018|           6|             6| 8673.39|
|     11019|          30|            20| 1925.91|
|     11023|           2|             2|   13.88|
|     11024|           3|             3|   47.17|
+----------+------------+--------------+--------+
only showing top 20 rows
```

customer order after the first special offer purchase

In [0]:

```
customer_retension\
    .withColumn("category",when(col("no of orders")>1,"repeat customer").otherwise("one t
ime buyer"))\
    .groupBy(col("category")).agg(count("customerid").alias("no of customers"))\
    .show()
```

```
+---------------+---------------+
|       category|no of customers|
+---------------+---------------+
| one time buyer|           1003|
|repeat customer|           6462|
+---------------+---------------+
```

In [0]:

```
customer_retension.write.format("parquet").mode("overwrite").option("header","true").save
("dbfs:/mnt/adventureworks/Gold/customer_retension")
```

In [0]:

```
fact_sales.write.format("parquet").mode("overwrite").option("header","true").save("dbfs:
/mnt/adventureworks/Gold/Tables/fact_sales")
```

In [0]:

```
dim_salesperson=s_SalesPerson_df.alias("sa").join(
    s_Store_df.alias("st"),
    col("sa.SalesPersonID")==col("st.SalesPersonID")
    ,"left"
).drop(col("st.SalesPersonID"),col("st.modifieddate"),col("st.rowguid"))
```

In [0]:

```
dim_salesperson.write.format("parquet").mode("overwrite").option("header","true").save("
dbfs:/mnt/adventureworks/Gold/Tables/dim_salesperson")
```

In [0]:

```
s_SalesTerritory_df.write.format("parquet").mode("overwrite").option("header","true").sav
e("dbfs:/mnt/adventureworks/Gold/Tables/dim_SalesTerritory")
```

In [0]:

```
customers=s_Customer_df.join(
    s_SalesOrderHeader_df,
```

```
    s_SalesOrderHeader_df["customerid"]==s_Customer_df["customerid"],
    "left"
).select(s_Customer_df["*"],s_SalesOrderHeader_df["CreditCardID"],s_SalesOrderHeader_df["
CreditCardApprovalCode"],)
dim_customer=customers.alias("c").join(
    s_CreditCard_df.alias("cr"),
    col("c.creditcardid")==col("cr.creditcardid")
    ,"left"
).drop(col("cr.ModifiedDate"),col("cr.creditcardid"))
```

In [0]:

```
dim_customer.write.format("parquet").mode("overwrite").option("header","true").save("dbf
s:/mnt/adventureworks/Gold/Tables/dim_customer")
```