# ETLAeropuertosInfoExt

November 28, 2021

[1]:
```
from pyspark.sql import SparkSession
from pyspark import SparkContext, SparkConf, SQLContext
from pyspark.sql import functions as f
from pyspark.sql.functions import monotonically_increasing_id as mid
import mysql.connector
```

[2]:
```
host='localhost'
user='root'
passwd=''
```

[3]:
```
mydb = mysql.connector.connect(
  host=host,
  user=user,
  passwd=passwd,
)
```

[4]:
```
mycursor = mydb.cursor()
```

[5]:
```
#se crea la base de dato usando comandos de sql en pyspark, en este momento se␣
 ↪encuentra como comentario
#query = 'CREATE DATABASE proyectoaeropuertosinfoext'
#mycursor.execute(query)
```

[6]:
```
#Verificar que la base de datos existe
databases = ("show databases")
mycursor.execute(databases)
for i in mycursor:
     print(i)
```

```
('Application',)
('Sales',)
('WWImportersDWH',)
('WWImportersTransactional',)
('Warehouse',)
('information_schema',)
('mysql',)
('performance_schema',)
('proyectoaeropuertosV2',)
```

```
('proyectoaeropuertosconhistoria',)
('proyectoaeropuertosinfoext',)
('sys',)
```

[7]:
```
definedb="USE proyectoaeropuertosinfoext"
mycursor.execute(definedb)
```

[71]:
```
# Create tables
creacion_tablas='\
CREATE TABLE IF NOT EXISTS DimFecha\
(\
    IDFecha INT NOT NULL,\
    Año CHARACTER(4),\
    Mes CHARACTER(2),\
    PRIMARY KEY(IDFecha)\
);\
\
CREATE TABLE IF NOT EXISTS DimTipo_Equipo\
(\
    IDEquipo INT NOT NULL,\
    NombreEquipo CHARACTER(4),\
    PRIMARY KEY(IDEquipo)\
);\
\
CREATE TABLE IF NOT EXISTS DimTipoVuelo\
(\
    IDTipoVuelo INT NOT NULL,\
    CodigoVuelo CHARACTER(1),\
    TipoVuelo CHARACTER(10),\
    PRIMARY KEY(IDTipoVuelo)\
);\
\
CREATE TABLE IF NOT EXISTS DimTipo_Trafico\
(\
    IDTipoTrafico INT NOT NULL,\
    Codigo_Trafico CHARACTER(1),\
    Descripcion CHARACTER(10),\
    PRIMARY KEY(IDTipoTrafico)\
);\
\
CREATE TABLE IF NOT EXISTS DimEmpresaTransportadora\
(\
    IDEmpresa INT NOT NULL,\
    NombreEmpresa CHARACTER(50),\
    PRIMARY KEY(IDEmpresa)\
);\
\
```

```
CREATE TABLE IF NOT EXISTS FactVuelos\
(\
    ID INT NOT NULL,\
    IDFecha INT,\
    IDEquipo INT,\
    IDAeropuertoOrigen INT,\
    IDAeropuertoDestino INT,\
    IDTipoVuelo INT,\
    IDTipoTrafico INT,\
    IDEmpresa INT,\
    Vuelos INT,\
    Pasajeros INT,\
    CargaBordo INT,\
    TotalSillas INT,\
    TotalCarga INT,\
    PRIMARY KEY(ID)\
);\
\
CREATE TABLE IF NOT EXISTS DimAeropuertoHistoria\
(\
    IDAeropuerto INT NOT NULL,\
    Sigla CHARACTER(3),\
    IATA CHARACTER(3),\
    NombreAeropuerto CHARACTER(50),\
    Municipio CHARACTER(50),\
    Departamento CHARACTER(50),\
    Categoria CHARACTER(50),\
    Latitud DOUBLE,\
    Longitud DOUBLE,\
    idMunicipio INT,\
    Propietario CHARACTER(50),\
    Explotador CHARACTER(50),\
    LongitudPista INT,\
    AnchoPista INT,\
    PBMO INT,\
    Elevacion INT,\
    Resolucion INT,\
    Clase CHARACTER(50),\
    Tipo CHARACTER(50),\
    GCD_Municipio CHARACTER(50),\
    GCD_Departamento CHARACTER(50),\
    FechaInicioVigencia DATE,\
    FechaFinVigencia DATE,\
    VersionDelRegistro CHARACTER(1),\
    Anio INT,\
    IDPIB INT,\
    PRIMARY KEY(IDAeropuerto)\
```

```
);\
\
CREATE TABLE IF NOT EXISTS DimInformacionPIB\
(\
    IDPIB INT NOT NULL,\
    PIB INT,\
    FechaInicioVigencia DATE,\
    FechaFinVigencia DATE,\
    VersionRegistro CHARACTER(1),\
    PRIMARY KEY(IDPIB)\
);\
'
mycursor.execute(creacion_tablas)

# reestablecer_historia='DROP TABLE proyectoaeropuertosinfoext.
 ↪DimInformacionPIB;'
# mycursor.execute(reestablecer_historia)
```

[72]:
```
# se muestran las tablas
showtables="SHOW TABLES FROM proyectoaeropuertosinfoext"
mycursor.execute(showtables)

for table_name in mycursor:
    print(table_name)
```

```
('DimAeropuertoHistoria',)
('DimEmpresaTransportadora',)
('DimFecha',)
('DimInformacionPIB',)
('DimTipoVuelo',)
('DimTipo_Equipo',)
('DimTipo_Trafico',)
('FactVuelos',)
```

[9]:
```
db_multidimensional_connection_string = 'jdbc:mysql://localhost:3306/
 ↪proyectoaeropuertosinfoext'
```

[10]:
```
import os
os.environ['PYSPARK_SUBMIT_ARGS'] = '--jars /usr/share/java/
 ↪mariadb-java-client-2.5.3.jar pyspark-shell'
```

[11]:
```
spark_context = SparkContext()
sql_context = SQLContext(spark_context)
spark = sql_context.sparkSession
```

[29]:
```
dfvuelosnew = spark.read.format("csv").load("vuelos.csv",format="csv",sep=",",
                                            inferSchema='true',header='true')
```

```
dfaeropuertosconhistoria =spark.read.format("csv").load("aeropuertos.
 ↪csv",format="csv",sep=",",
                                        inferSchema='true',header='true')
dfPIB=spark.read.format("csv").load("InformacionPIB.csv",format="csv",sep=",",
                                        inferSchema='true',header='true')
```

[13]:
```
dfvuelosnew.show(5)
```

```
+----+---+------+-------+----------+---------+-------+----------+------+-----
-+-------------+---------+-----------+
| ano|mes|origen|destino|tipo_equipo|tipo_vuelo|trafico|
empresa|vuelos|sillas|carga_ofrecida|pasajeros|carga_bordo|
+----+---+------+-------+----------+---------+-------+----------+------+-----
-+-------------+---------+-----------+
|2010|  1|   7NS|    VVC|       B190|         T|      N|SEARCA S.A.|     3|
48|           460|       48|        460|
|2010|  1|   7NT|    BOG|       B190|         T|      N|SEARCA S.A.|     6|
65|           521|       65|        521|
|2010|  1|   7NT|    VVC|       B190|         T|      N|SEARCA S.A.|     1|
14|             0|       14|          0|
|2010|  1|   9AI|    EYP|       C172|         T|      N| AERO APOYO|     2|
3|             0|        3|          0|
|2010|  1|   9AI|    EYP|       C182|         T|      N| AERO APOYO|     2|
5|            30|        5|         30|
+----+---+------+-------+----------+---------+-------+----------+------+-----
-+-------------+---------+-----------+
only showing top 5 rows
```

[14]:
```
dfaeropuertosconhistoria.show(5)
```

```
+-----+--------------+-----------+------------+-----------+--------+-------
+--------+-------------------+-------------+-------------+----------+----+--
-------+----------+-----------------+-------------+-----+----------+----------
----------+---------------+-------------+----+
|sigla|          iata|      nombre|
municipio|departamento|categoria|latitud|longitud|         propietario|    explo
tador|longitud_pista|ancho_pista|pbmo|elevacion|resolucion|fecha_construccion|fe
cha_vigencia|clase|
tipo|numero_vuelos_origen|gcd_departamento|gcd_municipio| Ano|
+-----+--------------+-----------+------------+-----------+--------+-------
+--------+-------------------+-------------+-------------+----------+----+--
-------+----------+-----------------+-------------+-----+----------+----------
----------+---------------+-------------+----+
|  7FO|no-especificado|     LA ISLA|Puerto Gaitán|       Meta|Aeródromo|
4.4211|-71.6271|LA ISLA Y EL ROSA…| LA CEIBA S.A.|          1079|
19|3000|      538| 1,325,000|        06/05/2015|    06/11/2018|   1A|Fumigación|
```

```
2|              50|        50568|2015|
|  7FO|no-especificado|     LA ISLA|Puerto Gaitán|        Meta|Aeródromo|
4.4211|-71.6271|LA ISLA Y EL ROSA…| LA CEIBA S.A.|          1321|
38|3000|      538| 1,325,000|        06/05/2015|    06/11/2018|   1A|Fumigación|
2|              50|        50568|2016|
|  7FU|no-especificado|LA ESCONDIDA|Puerto Gaitán|        Meta|Aeródromo|
4.6108|-71.1935|PALMAS SICARARE S…|COSARGO S.A.S.|          2141|
17|5000|      564| 1,843,000|        04/26/2013|    05/07/2016|   1A| Aerocivil|
24|              50|        50568|2014|
|  7FU|no-especificado|LA ESCONDIDA|Puerto Gaitán|        Meta|Aeródromo|
4.6108|-71.1935|PALMAS SICARARE S…|COSARGO S.A.S.|          3853|
5|5000|      564| 1,843,000|        04/26/2013|    05/07/2016|   1A| Aerocivil|
24|              50|        50568|2015|
|  7FU|no-especificado|LA ESCONDIDA|Puerto Gaitán|        Meta|Aeródromo|
4.6108|-71.1935|PALMAS SICARARE S…|COSARGO S.A.S.|          6667|
5|5000|      564| 1,843,000|        04/26/2013|    05/07/2016|   1A|   Público|
12|              50|        50568|2016|
+-----+---------------+-----------+-------------+-----------+---------+-------
+--------+--------------------+-------------+-------------+-----------+----+--
-------+----------+-----------------+--------------+-----+----------+----------
----------+----------------+-------------+----+
only showing top 5 rows
```

[30]:
```
dfPIB.show(5)
```

```
+------+-------------+-----+-----+-----+-----+-----+-----+-----+
|Codigo|DEPARTAMENTOS| 2014| 2015| 2016| 2017| 2018| 2019| 2020|
+------+-------------+-----+-----+-----+-----+-----+-----+-----+
|    91|     Amazonas|  157|  178|  199|  213|  224|  238|  174|
|     5|    Antioquia|16705|18734|21462|22797|24232|26405|23337|
|    81|       Arauca|  411|  457|  517|  514|  542|  580|  519|
|     8|    Atlántico| 6041| 7001| 7899| 8294| 8886| 9652| 8806|
|    11|  BOGOTÁ D.C.|35338|39760|44469|47413|50966|55824|48915|
+------+-------------+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

[16]:
```
from pyspark.sql.functions import sequence, to_date, explode,␣
 ↪col,when,lit,expr,substring,regexp_replace
from pyspark.sql import functions as sf
```

[87]:
```
# Creacion tabla DimFechaMes
df_fechames=dfvuelosnew.selectExpr('ano as Anio', 'mes as Mes')
df_fechames=df_fechames.dropDuplicates()
df_fechames=df_fechames.sort(col("Anio"),col('Mes'))
df_fechames = df_fechames.coalesce(1).withColumn("IDFecha", mid())
```

```
df_fechames.toPandas().to_csv('ModelExt/DimFechaMes.csv')
df_fechames.select('*').write.format('jdbc') \
          .mode('overwrite') \
          .option('url', db_multidimensional_connection_string) \
          .option('dbtable', 'DimFechaMes') \
          .option('user', user) \
          .option('password', passwd) \
          .option('driver', 'org.mariadb.jdbc.Driver') \
          .save()
df_fechames.show(5)
```

```
+----+---+-------+
|Anio|Mes|IDFecha|
+----+---+-------+
|2010|  1|      0|
|2010|  2|      1|
|2010|  3|      2|
|2010|  4|      3|
|2010|  5|      4|
+----+---+-------+
only showing top 5 rows
```

[88]:
```
# Creacion tabla DimTipoVuelo
df_tipo_vuelo=dfvuelosnew.selectExpr('tipo_vuelo as CodigoVuelo')
df_tipo_vuelo=df_tipo_vuelo.dropDuplicates()
df_tipo_vuelo=df_tipo_vuelo.withColumn("TipoVuelo", \
   when((df_tipo_vuelo.CodigoVuelo =="A"), "Adicionales") \
   .when((df_tipo_vuelo.CodigoVuelo =="C"),"Charter") \
   .when((df_tipo_vuelo.CodigoVuelo =="R"),"Regular") \
   .when((df_tipo_vuelo.CodigoVuelo =="T"),"Taxi") \
   .otherwise("nan") \
  )
nandf = spark.createDataFrame([('nan','nan')], df_tipo_vuelo.columns)
df_tipo_vuelo = df_tipo_vuelo.union(nandf)
df_tipo_vuelo=df_tipo_vuelo.sort(col("CodigoVuelo"))
df_tipo_vuelo = df_tipo_vuelo.coalesce(1).withColumn("IDTipoVuelo", mid())
df_tipo_vuelo.toPandas().to_csv('ModelExt/DimTipoVuelo.csv')
df_tipo_vuelo.select('*').write.format('jdbc') \
          .mode('overwrite') \
          .option('url', db_multidimensional_connection_string) \
          .option('dbtable', 'DimTipoVuelo') \
          .option('user', user) \
          .option('password', passwd) \
          .option('driver', 'org.mariadb.jdbc.Driver') \
          .save()
df_tipo_vuelo.show(5)
```

```
+-----------+-----------+-----------+
|CodigoVuelo|  TipoVuelo|IDTipoVuelo|
+-----------+-----------+-----------+
|          A|Adicionales|          0|
|          C|    Charter|          1|
|          R|    Regular|          2|
|          T|       Taxi|          3|
|        nan|        nan|          4|
+-----------+-----------+-----------+
```

[89]:
```
# Creacion tabla DimTipo_Trafico
df_trafico=dfvuelosnew.selectExpr('trafico as Codigo_Trafico')
df_trafico=df_trafico.dropDuplicates()
df_trafico=df_trafico.withColumn("Descripcion", \
   when((df_trafico.Codigo_Trafico =="I"), "Internacional") \
   .when((df_trafico.Codigo_Trafico =="N"),"Nacional") \
   .when((df_trafico.Codigo_Trafico =="E"),"Externo") \
   .otherwise("nan") \
  )
df_trafico=df_trafico.sort(col("Codigo_Trafico"))
df_trafico = df_trafico.coalesce(1).withColumn("IDTipoTrafico", mid())
df_trafico.toPandas().to_csv('ModelExt/DimTipoTrafico.csv')
df_trafico.select('*').write.format('jdbc') \
          .mode('overwrite') \
          .option('url', db_multidimensional_connection_string) \
          .option('dbtable', 'DimTipo_Trafico') \
          .option('user', user) \
          .option('password', passwd) \
          .option('driver', 'org.mariadb.jdbc.Driver') \
          .save()
df_trafico.show(5)
```

```
+--------------+-----------+-------------+
|Codigo_Trafico|Descripcion|IDTipoTrafico|
+--------------+-----------+-------------+
|             N|   Nacional|            0|
+--------------+-----------+-------------+
```

[90]:
```
# Creacion tabla DimEmpresaTransportadora
df_empresatrans1=dfvuelosnew.selectExpr('empresa as NombreEmpresa')
df_empresatrans1=df_empresatrans1.dropDuplicates()
df_empresatrans1=df_empresatrans1.sort(col("NombreEmpresa"))
nandf = spark.createDataFrame([('nan', )], df_empresatrans1.columns)
df_empresatrans1 = df_empresatrans1.union(nandf)
df_empresatrans1 = df_empresatrans1.coalesce(1).withColumn("IDEmpresa", mid())
```

```
df_empresatrans1.toPandas().to_csv('ModelExt/DimEmpresaTransportadora.csv')
df_empresatrans1.select('*').write.format('jdbc') \
          .mode('overwrite') \
          .option('url', db_multidimensional_connection_string) \
          .option('dbtable', 'DimEmpresaTransportadora') \
          .option('user', user) \
          .option('password', passwd) \
          .option('driver', 'org.mariadb.jdbc.Driver') \
          .save()
df_empresatrans1.show(5)
```

```
+--------------------+---------+
|       NombreEmpresa|IDEmpresa|
+--------------------+---------+
| AER CARIBE LIMITADA|        0|
|          AERO APOYO|        1|
|AERO SERVICIOS ES…|        2|
|AERO TAXI GUAYMAR…|        3|
|AEROCHARTER ANDIN…|        4|
+--------------------+---------+
only showing top 5 rows
```

[91]:
```
# Creacion tabla DimTipo_Equipo
df_tipoequipo=dfvuelosnew.selectExpr('tipo_equipo as NombreEquipo')
df_tipoequipo=df_tipoequipo.dropDuplicates()
df_tipoequipo=df_tipoequipo.sort(col("NombreEquipo"))
nandf = spark.createDataFrame([('nan', )], df_tipoequipo.columns)
df_tipoequipo = df_tipoequipo.union(nandf)
df_tipoequipo = df_tipoequipo.coalesce(1).withColumn("IDEquipo", mid())
df_tipoequipo.toPandas().to_csv('ModelExt/DimTipo_Equipo.csv')
df_tipoequipo.select('*').write.format('jdbc') \
          .mode('overwrite') \
          .option('url', db_multidimensional_connection_string) \
          .option('dbtable', 'DimTipo_Equipo') \
          .option('user', user) \
          .option('password', passwd) \
          .option('driver', 'org.mariadb.jdbc.Driver') \
          .save()
df_tipoequipo.show(5)
```

```
+------------+--------+
|NombreEquipo|IDEquipo|
+------------+--------+
|         318|       0|
|         319|       1|
|         330|       2|
|         332|       3|
```

```
|         727|       4|
+------------+--------+
only showing top 5 rows
```

[31]:
```
#TransformacionInformacionPIB
dfPIBUnpivot=dfPIB.withColumnRenamed("DEPARTAMENTOS","Departamento").
 ↪withColumnRenamed("2014","Ano2014").withColumnRenamed("2015","Ano2015").
 ↪withColumnRenamed("2016","Ano2016")\
             .withColumnRenamed("2017","Ano2017").
 ↪withColumnRenamed("2018","Ano2018").withColumnRenamed("2019","Ano2019")\
             .withColumnRenamed("2020","Ano2020")
unpivotExpr = "stack(7, 'Ano2014',Ano2014, 'Ano2015', Ano2015, 'Ano2016',␣
 ↪Ano2016,'Ano2017', Ano2017,'Ano2018', Ano2018,'Ano2019', Ano2019,'Ano2020',␣
 ↪Ano2020) as (Anio,PIB)"
dfPIBUnpivot = dfPIBUnpivot.selectExpr("Codigo","Departamento", unpivotExpr).
 ↪where("PIB is not null")
dfPIBUnpivot=dfPIBUnpivot.withColumn('Anio', substring('Anio', 4,4))
dfPIBUnpivot=dfPIBUnpivot.withColumn('Departamento',
    when(dfPIBUnpivot.Departamento.startswith('San Andrés'),'San Andrés islas')␣
 ↪\
   .when(dfPIBUnpivot.Departamento.startswith('BOG'),'Bogotá, D.C.') \
   .otherwise(dfPIBUnpivot.Departamento))
dfPIBUnpivot.show()
```

```
+------+------------+----+-----+
|Codigo|Departamento|Anio|  PIB|
+------+------------+----+-----+
|    91|    Amazonas|2014|  157|
|    91|    Amazonas|2015|  178|
|    91|    Amazonas|2016|  199|
|    91|    Amazonas|2017|  213|
|    91|    Amazonas|2018|  224|
|    91|    Amazonas|2019|  238|
|    91|    Amazonas|2020|  174|
|     5|   Antioquia|2014|16705|
|     5|   Antioquia|2015|18734|
|     5|   Antioquia|2016|21462|
|     5|   Antioquia|2017|22797|
|     5|   Antioquia|2018|24232|
|     5|   Antioquia|2019|26405|
|     5|   Antioquia|2020|23337|
|    81|      Arauca|2014|  411|
|    81|      Arauca|2015|  457|
|    81|      Arauca|2016|  517|
|    81|      Arauca|2017|  514|
|    81|      Arauca|2018|  542|
```

```
|    81|      Arauca|2019|  580|
+------+------------+----+-----+
only showing top 20 rows
```

[73]:
```
#Para el ejemplo del manejo de historia, se divide el dataframme de PIB por␣
 ↪año,
#con el fin de crear una base, y actualizaciones
dfBIPparte1 = dfPIBUnpivot.filter(dfPIBUnpivot.Anio=="2014")
dfBIPparte2 = dfPIBUnpivot.filter(dfPIBUnpivot.Anio=="2015")
dfBIPparte3 = dfPIBUnpivot.filter(dfPIBUnpivot.Anio=="2016")
dfBIPparte4 = dfPIBUnpivot.filter(dfPIBUnpivot.Anio=="2017")
dfBIPparte5 = dfPIBUnpivot.filter(dfPIBUnpivot.Anio=="2018")
```

[74]:
```
# Actualización tabla Dim InformacionPIB
from pyspark.sql.types import DateType
def actualizar_historia_PIB(df_a_cargar):

    tablaDWH=spark.read.format('jdbc')\
    .option('url', db_multidimensional_connection_string) \
    .option('dbtable', '''(SELECT *
    FROM  DimInformacionPIB) AS Temp_DimInformacionPIB''') \
    .option('user', user) \
    .option('password', passwd) \
    .option('driver', 'org.mariadb.jdbc.Driver') \
    .load()

    df=df_a_cargar.selectExpr('Departamento as Departamento', 'Anio as Anio',␣
 ↪'PIB as PIB')
    if tablaDWH.count()==0:

        df=df.withColumn("FechaInicioVigencia",lit("1900-01-01"))
        df=df.withColumn("FechaFinVigencia",lit("2300-01-01"))
        df=df.withColumn("VersionDelRegistro",lit("S"))
        df=df.dropDuplicates()
        df=df.sort(col("Departamento"))
        df = df.coalesce(1).withColumn("IDPIB", mid())
        df.createOrReplaceTempView("df")
        df = spark.sql("SELECT INT(IDPIB),STRING(Departamento),INT(PIB),␣
 ↪DATE(FechaInicioVigencia),\
                        DATE(FechaFinVigencia),␣
 ↪STRING(VersionDelRegistro),INT(Anio) from df")

        df.toPandas().to_csv('ModelExt/DimInformacionPIB.csv')
        x=df.count()
        df.select('*').write.format('jdbc') \
          .mode('overwrite') \
```

```
          .option('url', db_multidimensional_connection_string) \
          .option('dbtable', 'DimInformacionPIB') \
          .option('user', user) \
          .option('password', passwd) \
          .option('driver', 'org.mariadb.jdbc.Driver') \
          .save()
   else:
       tablaDWH.persist()
       df_fechas_antiguas=tablaDWH.selectExpr('Departamento','Departamento as␣
↪DepartamentoAnt')
       df_fechas_antiguas=df_fechas_antiguas.dropDuplicates()
       df_nuevo=df.selectExpr('Departamento','Anio as AnioNuevo')
       df_nuevo=df_nuevo.dropDuplicates()
       df_temp=tablaDWH.join(df_nuevo, how = 'left', on = 'Departamento')
       #registros que no tienen actualizacion
       df_temp.persist()
       df_mantener=df_temp.filter(df_temp.AnioNuevo.isNull())
       df_mantener=df_mantener.drop('AnioNuevo')
       df_mantener=df_mantener.withColumn("Origen",lit("Mantener"))
       #registros que si tienen actualización
       df_actualizar=df_temp.filter(df_temp.AnioNuevo.isNotNull())
       #registros viejos que no van a cambiar
       df_actualizar.persist()
       df_actualizar_registrosviejos=df_actualizar.filter(df_actualizar.
↪VersionDelRegistro=="N")
       df_actualizar_registrosviejos=df_actualizar_registrosviejos.
↪drop('AnioNuevo')
       #actualización de registros que eran vigentes
       df_actualizar_registrosvigentes=df_actualizar.filter(df_actualizar.
↪VersionDelRegistro=="S")
       df_actualizar_registrosvigentes=df_actualizar_registrosvigentes.
↪withColumn("VersionDelRegistro",lit("N"))
       df_actualizar_registrosvigentes=df_actualizar_registrosvigentes.
↪withColumn("FechaFinVigencia",
                                                                        ␣
↪    sf.concat(sf.col('Anio'),sf.lit('-12-31'))
                                                                        ␣
↪   )
       df_actualizar_registrosvigentes=df_actualizar_registrosvigentes.
↪drop('AnioNuevo')
       #registros nuevos para ingresar a la base
       #Encontar llave máxima
       max_key = tablaDWH.agg({"IDPIB": "max"}).collect()[0][0]
       df_nuevos_registros=df.alias('df_nuevos_registros')
       df_nuevos_registros=df_nuevos_registros.join(df_fechas_antiguas,how =␣
↪'left', on = 'Departamento')
```

```
        df_nuevos_registros=df_nuevos_registros.
↪withColumn("FechaInicioVigencia",when(df_nuevos_registros.DepartamentoAnt.
↪isNull(),\
                                                                          ␣
↪     '1900-01-01')\
                                                              .otherwise(sf.
↪concat(sf.col('Anio'),sf.lit('-01-01'))))
        df_nuevos_registros=df_nuevos_registros.
↪withColumn("FechaFinVigencia",lit("2300-01-01"))
        df_nuevos_registros=df_nuevos_registros.
↪withColumn("VersionDelRegistro",lit("S"))
        df_nuevos_registros = df_nuevos_registros.withColumn('IDPIB',  mid() +␣
↪max_key+1)
        #unir en un solo dataframe
        df_mantener.createOrReplaceTempView("df_mantener")
        df_mantener = spark.sql("SELECT INT(IDPIB)␣
↪,STRING(Departamento),INT(PIB), DATE(FechaInicioVigencia),\
                        DATE(FechaFinVigencia),␣
↪STRING(VersionDelRegistro),INT(Anio) from df_mantener")

        df_actualizar_registrosviejos.
↪createOrReplaceTempView("df_actualizar_registrosviejos")
        df_actualizar_registrosviejos = spark.sql("SELECT INT(IDPIB)␣
↪,STRING(Departamento),INT(PIB), DATE(FechaInicioVigencia),\
                        DATE(FechaFinVigencia),␣
↪STRING(VersionDelRegistro),INT(Anio) from df_actualizar_registrosviejos")

        df_actualizar_registrosvigentes.
↪createOrReplaceTempView("df_actualizar_registrosvigentes")
        df_actualizar_registrosvigentes = spark.sql("SELECT INT(IDPIB)␣
↪,STRING(Departamento),INT(PIB), DATE(FechaInicioVigencia),\
                        DATE(FechaFinVigencia),␣
↪STRING(VersionDelRegistro),INT(Anio) from df_actualizar_registrosvigentes")

        df_nuevos_registros.createOrReplaceTempView("df_nuevos_registros")
        df_nuevos_registros = spark.sql("SELECT INT(IDPIB)␣
↪,STRING(Departamento),INT(PIB), DATE(FechaInicioVigencia),\
                        DATE(FechaFinVigencia),␣
↪STRING(VersionDelRegistro),INT(Anio) from df_nuevos_registros")

        df2 = df_nuevos_registros.union(df_mantener)
        df2 = df2.union(df_actualizar_registrosviejos)
        df2 = df2.union(df_actualizar_registrosvigentes)
        df2.toPandas().to_csv('ModelExt/DimInformacionPIB.csv')
        x=df2.count()
        df2.select('*').write.format('jdbc') \
```

```
            .mode('overwrite') \
            .option('url', db_multidimensional_connection_string) \
            .option('dbtable', 'DimInformacionPIB') \
            .option('user', user) \
            .option('password', passwd) \
            .option('driver', 'org.mariadb.jdbc.Driver') \
            .save()
    return x
```

[75]:
```
#Validación proceso de carga PIB con historia
DimPIBHistoria=spark.read.format('jdbc')\
    .option('url', db_multidimensional_connection_string) \
    .option('dbtable', '''(SELECT *
    FROM  DimInformacionPIB) AS Temp_DimInformacionPIB''') \
    .option('user', user) \
    .option('password', passwd) \
    .option('driver', 'org.mariadb.jdbc.Driver') \
    .load()
print("conteo inicial base de datos : {0}".format(DimPIBHistoria.count()))

x=actualizar_historia_PIB(dfBIPparte1)
print("conteo primera carga dataframe : {0}".format(x))

DimPIBHistoria=spark.read.format('jdbc')\
    .option('url', db_multidimensional_connection_string) \
    .option('dbtable', '''(SELECT *
    FROM  DimInformacionPIB) AS Temp_DimInformacionPIB''') \
    .option('user', user) \
    .option('password', passwd) \
    .option('driver', 'org.mariadb.jdbc.Driver') \
    .load()
DimPIBHistoria.count()
print("conteo primera carga base de datos : {0}".format(DimPIBHistoria.count()))

x=actualizar_historia_PIB(dfBIPparte2)
print("conteo segunda carga dataframe : {0}".format(x))

DimPIBHistoria=spark.read.format('jdbc')\
    .option('url', db_multidimensional_connection_string) \
    .option('dbtable', '''(SELECT *
    FROM  DimInformacionPIB) AS Temp_DimInformacionPIB''') \
    .option('user', user) \
    .option('password', passwd) \
    .option('driver', 'org.mariadb.jdbc.Driver') \
    .load()
DimPIBHistoria.count()
print("conteo segunda carga base de datos : {0}".format(DimPIBHistoria.count()))
```

```
x=actualizar_historia_PIB(dfBIPparte3)
print("conteo tercera carga dataframe : {0}".format(x))

DimPIBHistoria=spark.read.format('jdbc')\
    .option('url', db_multidimensional_connection_string) \
    .option('dbtable', '''(SELECT *
    FROM  DimInformacionPIB) AS Temp_DimInformacionPIB''') \
    .option('user', user) \
    .option('password', passwd) \
    .option('driver', 'org.mariadb.jdbc.Driver') \
    .load()
DimPIBHistoria.count()
print("conteo tercera carga base de datos : {0}".format(DimPIBHistoria.count()))

x=actualizar_historia_PIB(dfBIPparte4)
print("conteo tercera carga dataframe : {0}".format(x))

DimPIBHistoria=spark.read.format('jdbc')\
    .option('url', db_multidimensional_connection_string) \
    .option('dbtable', '''(SELECT *
    FROM  DimInformacionPIB) AS Temp_DimInformacionPIB''') \
    .option('user', user) \
    .option('password', passwd) \
    .option('driver', 'org.mariadb.jdbc.Driver') \
    .load()
DimPIBHistoria.count()
print("conteo tercera carga base de datos : {0}".format(DimPIBHistoria.count()))


x=actualizar_historia_PIB(dfBIPparte5)
print("conteo tercera carga dataframe : {0}".format(x))

DimPIBHistoria=spark.read.format('jdbc')\
    .option('url', db_multidimensional_connection_string) \
    .option('dbtable', '''(SELECT *
    FROM  DimInformacionPIB) AS Temp_DimInformacionPIB''') \
    .option('user', user) \
    .option('password', passwd) \
    .option('driver', 'org.mariadb.jdbc.Driver') \
    .load()
DimPIBHistoria.count()
print("conteo tercera carga base de datos : {0}".format(DimPIBHistoria.count()))
```

```
conteo inicial base de datos : 0
conteo primera carga dataframe : 33
conteo primera carga base de datos : 33
```

```
conteo segunda carga dataframe : 66
conteo segunda carga base de datos : 66
conteo tercera carga dataframe : 99
conteo tercera carga base de datos : 99
conteo tercera carga dataframe : 132
conteo tercera carga base de datos : 132
```

[76]:
```
#Para el ejemplo del manejo de historia, se divide el dataframme de aeropuertos
 ↪en 3,
#con el fin de crear una base, y dos actualizaciones
dfaeropuertosconhistoriaparte1 = dfaeropuertosconhistoria.
 ↪filter(dfaeropuertosconhistoria.Ano=="2014")
dfaeropuertosconhistoriaparte2 = dfaeropuertosconhistoria.
 ↪filter(dfaeropuertosconhistoria.Ano=="2015")
dfaeropuertosconhistoriaparte3 = dfaeropuertosconhistoria.
 ↪filter(dfaeropuertosconhistoria.Ano=="2016")
```

[77]:
```
InfoPIB=spark.read.format('jdbc')\
.option('url', db_multidimensional_connection_string) \
.option('dbtable', '''(SELECT *
FROM  DimInformacionPIB) AS Temp_DimInformacionPIB''') \
.option('user', user) \
.option('password', passwd) \
.option('driver', 'org.mariadb.jdbc.Driver') \
.load()

InfoPIB.show(5)
```

```
+-----+------------+-----+------------------+---------------+-----------------
-+----+
|IDPIB|Departamento|
PIB|FechaInicioVigencia|FechaFinVigencia|VersionDelRegistro|Anio|
+-----+------------+-----+------------------+---------------+-----------------
-+----+
|    0|    Amazonas|  157|         1900-01-01|      2014-12-31|
N|2014|
|    1|   Antioquia|16705|         1900-01-01|      2014-12-31|
N|2014|
|    2|      Arauca|  411|         1900-01-01|      2014-12-31|
N|2014|
|    3|   Atlántico| 6041|         1900-01-01|      2014-12-31|
N|2014|
|    4|Bogotá, D.C.|35338|         1900-01-01|      2014-12-31|
N|2014|
+-----+------------+-----+------------------+---------------+-----------------
-+----+
only showing top 5 rows
```

[84]:
```
# Actualización tabla DimAeropuertoHistoria
from pyspark.sql.types import DateType
def actualizar_historia_aeropuertos(df_a_cargar):

    tablaDWH=spark.read.format('jdbc')\
    .option('url', db_multidimensional_connection_string) \
    .option('dbtable', '''(SELECT *
    FROM  DimAeropuertoHistoria) AS Temp_DimAeropuertoHistoria''') \
    .option('user', user) \
    .option('password', passwd) \
    .option('driver', 'org.mariadb.jdbc.Driver') \
    .load()


    InfoPIB=spark.read.format('jdbc')\
    .option('url', db_multidimensional_connection_string) \
    .option('dbtable', '''(SELECT *
    FROM  DimInformacionPIB) AS Temp_DimInformacionPIB''') \
    .option('user', user) \
    .option('password', passwd) \
    .option('driver', 'org.mariadb.jdbc.Driver') \
    .load()
    InfoPIB.createOrReplaceTempView("InfoPIB")

    df=df_a_cargar.selectExpr('sigla as Sigla', 'iata as IATA', 'nombre as␣
↪NombreAeropuerto',
                                          'municipio as␣
↪Municipio','departamento as Departamento', 'categoria as Categoria',
                                           'latitud as Latitud','longitud as␣
↪Longitud', 'propietario as Propietario',
                                           'explotador as␣
↪Explotador','longitud_pista as LongitudPista',
                                           'ancho_pista as AnchoPista', 'pbmo␣
↪as PBMO', 'elevacion as Elevacion',
                                           'resolucion as Resolucion','clase as␣
↪Clase', 'tipo as Tipo',
                                           'gcd_municipio as GCD_Municipio',␣
↪'gcd_departamento as GCD_Departamento',
                                           'Ano as Anio')
    if tablaDWH.count()==0:

        df=df.withColumn("FechaInicioVigencia",lit("1900-01-01"))
        df=df.withColumn("FechaFinVigencia",lit("2300-01-01"))
        df=df.withColumn("VersionDelRegistro",lit("S"))
        df=df.dropDuplicates()
```

```
        df=df.sort(col("Sigla"))
        df = df.coalesce(1).withColumn("IDAeropuerto", mid())
        df.createOrReplaceTempView("df")
        df = spark.sql("SELECT INT(IDAeropuerto), STRING(Sigla), STRING(IATA),␣
↪STRING(NombreAeropuerto),\
                        STRING(Categoria),DOUBLE(Latitud),␣
↪DOUBLE(Longitud),STRING(Municipio), STRING(df.Departamento), \
                        ␣
↪STRING(Propietario),STRING(Explotador),INT(LongitudPista), INT(AnchoPista),\
                        STRING(PBMO),INT(Elevacion), STRING(Resolucion),␣
↪STRING(Clase),\
                        STRING(Tipo),STRING(GCD_Municipio),␣
↪STRING(GCD_Departamento), DATE(df.FechaInicioVigencia),\
                       DATE(df.FechaFinVigencia), STRING(df.
↪VersionDelRegistro),INT(df.Anio), INT(InfoPIB.IDPIB) from df\
                       left join InfoPIB on InfoPIB.Departamento= df.
↪Departamento and\
                            InfoPIB.Anio= df.Anio")

        df.toPandas().to_csv('ModelExt/DimAeropuertoHistoria.csv')
        x=df.count()
        df.select('*').write.format('jdbc') \
          .mode('overwrite') \
          .option('url', db_multidimensional_connection_string) \
          .option('dbtable', 'DimAeropuertoHistoria') \
          .option('user', user) \
          .option('password', passwd) \
          .option('driver', 'org.mariadb.jdbc.Driver') \
          .save()
    else:
        tablaDWH.persist()
        df_fechas_antiguas=tablaDWH.selectExpr('Sigla','Sigla as SiglaAnt')
        df_fechas_antiguas=df_fechas_antiguas.dropDuplicates()
        df_nuevo=df.selectExpr('Sigla','Anio as AnioNuevo')
        df_nuevo=df_nuevo.dropDuplicates()
        df_temp=tablaDWH.join(df_nuevo, how = 'left', on = 'Sigla')
        #registros que no tienen actualizacion
        df_temp.persist()
        df_mantener=df_temp.filter(df_temp.AnioNuevo.isNull())
        df_mantener=df_mantener.drop('AnioNuevo')
        df_mantener=df_mantener.withColumn("Origen",lit("Mantener"))
        #registros que si tienen actualización
        df_actualizar=df_temp.filter(df_temp.AnioNuevo.isNotNull())
        #registros viejos que no van a cambiar
        df_actualizar.persist()
```

```
      df_actualizar_registrosviejos=df_actualizar.filter(df_actualizar.
↪VersionDelRegistro=="N")
      df_actualizar_registrosviejos=df_actualizar_registrosviejos.
↪drop('AnioNuevo')
      #actualización de registros que eran vigentes
      df_actualizar_registrosvigentes=df_actualizar.filter(df_actualizar.
↪VersionDelRegistro=="S")
      df_actualizar_registrosvigentes=df_actualizar_registrosvigentes.
↪withColumn("VersionDelRegistro",lit("N"))
      df_actualizar_registrosvigentes=df_actualizar_registrosvigentes.
↪withColumn("FechaFinVigencia",
                                                                        ␣
↪    sf.concat(sf.col('AnioNuevo')-1,sf.lit('-12-31'))
                                                                        ␣
↪   )
      df_actualizar_registrosvigentes=df_actualizar_registrosvigentes.
↪drop('AnioNuevo')
      #registros nuevos para ingresar a la base
      #Encontar llave máxima
      max_key = tablaDWH.agg({"IDAeropuerto": "max"}).collect()[0][0]
      df_nuevos_registros=df.alias('df_nuevos_registros')
      df_nuevos_registros=df_nuevos_registros.join(df_fechas_antiguas,how =␣
↪'left', on = 'Sigla')
      df_nuevos_registros=df_nuevos_registros.
↪withColumn("FechaInicioVigencia",when(df_nuevos_registros.SiglaAnt.isNull(),\
                                                                        ␣
↪     '1900-01-01')\
                                                    .otherwise(sf.
↪concat(sf.col('Anio'),sf.lit('-01-01'))))
      df_nuevos_registros=df_nuevos_registros.
↪withColumn("FechaFinVigencia",lit("2300-01-01"))
      df_nuevos_registros=df_nuevos_registros.
↪withColumn("VersionDelRegistro",lit("S"))
      df_nuevos_registros = df_nuevos_registros.withColumn('IDAeropuerto', ␣
↪mid() + max_key+1)
      #unir en un solo dataframe
      df_mantener.createOrReplaceTempView("df_mantener")
      df_mantener = spark.sql("SELECT INT(IDAeropuerto), STRING(Sigla),␣
↪STRING(IATA), STRING(NombreAeropuerto),\
                        STRING(Categoria),DOUBLE(Latitud),␣
↪DOUBLE(Longitud),STRING(Municipio), STRING(df_mantener.Departamento), \
                        ␣
↪STRING(Propietario),STRING(Explotador),INT(LongitudPista), INT(AnchoPista),\
                        STRING(PBMO),INT(Elevacion), STRING(Resolucion),␣
↪STRING(Clase),\
```

```
                        STRING(Tipo),STRING(GCD_Municipio),␣
↪STRING(GCD_Departamento), DATE(df_mantener.FechaInicioVigencia),\
                       DATE(df_mantener.FechaFinVigencia), STRING(df_mantener.
↪VersionDelRegistro),INT(df_mantener.Anio), INT(InfoPIB.IDPIB) from␣
↪df_mantener\
                       left join InfoPIB on InfoPIB.Departamento= df_mantener.
↪Departamento and\
                            InfoPIB.Anio= df_mantener.Anio")

      df_actualizar_registrosviejos.
↪createOrReplaceTempView("df_actualizar_registrosviejos")
      df_actualizar_registrosviejos = spark.sql("SELECT INT(IDAeropuerto),␣
↪STRING(Sigla), STRING(IATA), STRING(NombreAeropuerto),\
                        STRING(Categoria),DOUBLE(Latitud),␣
↪DOUBLE(Longitud),STRING(Municipio), STRING(df_actualizar_registrosviejos.
↪Departamento), \
                        ␣
↪STRING(Propietario),STRING(Explotador),INT(LongitudPista), INT(AnchoPista),\
                        STRING(PBMO),INT(Elevacion), STRING(Resolucion),␣
↪STRING(Clase),\
                        STRING(Tipo),STRING(GCD_Municipio),␣
↪STRING(GCD_Departamento), DATE(df_actualizar_registrosviejos.
↪FechaInicioVigencia),\
                       DATE(df_actualizar_registrosviejos.FechaFinVigencia),␣
↪STRING(df_actualizar_registrosviejos.
↪VersionDelRegistro),INT(df_actualizar_registrosviejos.Anio), INT(InfoPIB.
↪IDPIB) from df_actualizar_registrosviejos\
                       left join InfoPIB on InfoPIB.Departamento=␣
↪df_actualizar_registrosviejos.Departamento and\
                            InfoPIB.Anio= df_actualizar_registrosviejos.Anio")

      df_actualizar_registrosvigentes.
↪createOrReplaceTempView("df_actualizar_registrosvigentes")
      df_actualizar_registrosvigentes = spark.sql("SELECT INT(IDAeropuerto),␣
↪STRING(Sigla), STRING(IATA), STRING(NombreAeropuerto),\
                        STRING(Categoria),DOUBLE(Latitud),␣
↪DOUBLE(Longitud),STRING(Municipio), STRING(df_actualizar_registrosvigentes.
↪Departamento), \
                        ␣
↪STRING(Propietario),STRING(Explotador),INT(LongitudPista), INT(AnchoPista),\
                        STRING(PBMO),INT(Elevacion), STRING(Resolucion),␣
↪STRING(Clase),\
                        STRING(Tipo),STRING(GCD_Municipio),␣
↪STRING(GCD_Departamento), DATE(df_actualizar_registrosvigentes.
↪FechaInicioVigencia),\
```

```
                      DATE(df_actualizar_registrosvigentes.FechaFinVigencia),␣
↪STRING(df_actualizar_registrosvigentes.
↪VersionDelRegistro),INT(df_actualizar_registrosvigentes.Anio), INT(InfoPIB.
↪IDPIB) from df_actualizar_registrosvigentes\
                      left join InfoPIB on InfoPIB.Departamento=␣
↪df_actualizar_registrosvigentes.Departamento and\
                           InfoPIB.Anio= df_actualizar_registrosvigentes.
↪Anio")

      df_nuevos_registros.createOrReplaceTempView("df_nuevos_registros")
      df_nuevos_registros = spark.sql("SELECT INT(IDAeropuerto),␣
↪STRING(Sigla), STRING(IATA), STRING(NombreAeropuerto),\
                       STRING(Categoria),DOUBLE(Latitud),␣
↪DOUBLE(Longitud),STRING(Municipio), STRING(df_nuevos_registros.
↪Departamento), \
                       ␣
↪STRING(Propietario),STRING(Explotador),INT(LongitudPista), INT(AnchoPista),\
                       STRING(PBMO),INT(Elevacion), STRING(Resolucion),␣
↪STRING(Clase),\
                       STRING(Tipo),STRING(GCD_Municipio),␣
↪STRING(GCD_Departamento), DATE(df_nuevos_registros.FechaInicioVigencia),\
                      DATE(df_nuevos_registros.FechaFinVigencia),␣
↪STRING(df_nuevos_registros.VersionDelRegistro),INT(df_nuevos_registros.
↪Anio), INT(InfoPIB.IDPIB) from df_nuevos_registros\
                      left join InfoPIB on InfoPIB.Departamento=␣
↪df_nuevos_registros.Departamento and\
                           InfoPIB.Anio= df_nuevos_registros.Anio")

      df2 = df_nuevos_registros.union(df_mantener)
      df2 = df2.union(df_actualizar_registrosviejos)
      df2 = df2.union(df_actualizar_registrosvigentes)

      df2.toPandas().to_csv('ModelExt/DimAeropuertoHistoria.csv')
      x=df2.count()
      df2.select('*').write.format('jdbc') \
        .mode('overwrite') \
        .option('url', db_multidimensional_connection_string) \
        .option('dbtable', 'DimAeropuertoHistoria') \
        .option('user', user) \
        .option('password', passwd) \
        .option('driver', 'org.mariadb.jdbc.Driver') \
        .save()
   return x
```

[85]:
```
#Validación proceso de carga con historia
DimAeropuertoHistoria=spark.read.format('jdbc')\
```

```
    .option('url', db_multidimensional_connection_string) \
    .option('dbtable', '''(SELECT *
    FROM  DimAeropuertoHistoria) AS Temp_DimAeropuertoHistoria''') \
    .option('user', user) \
    .option('password', passwd) \
    .option('driver', 'org.mariadb.jdbc.Driver') \
    .load()
print("conteo inicial base de datos : {0}".format(DimAeropuertoHistoria.
 ↪count()))

x=actualizar_historia_aeropuertos(dfaeropuertosconhistoriaparte1)
print("conteo primera carga dataframe : {0}".format(x))

DimAeropuertoHistoria=spark.read.format('jdbc')\
    .option('url', db_multidimensional_connection_string) \
    .option('dbtable', '''(SELECT *
    FROM  DimAeropuertoHistoria) AS Temp_DimAeropuertoHistoria''') \
    .option('user', user) \
    .option('password', passwd) \
    .option('driver', 'org.mariadb.jdbc.Driver') \
    .load()
DimAeropuertoHistoria.count()
print("conteo primera carga base de datos : {0}".format(DimAeropuertoHistoria.
 ↪count()))

x=actualizar_historia_aeropuertos(dfaeropuertosconhistoriaparte2)
print("conteo segunda carga dataframe : {0}".format(x))

DimAeropuertoHistoria=spark.read.format('jdbc')\
    .option('url', db_multidimensional_connection_string) \
    .option('dbtable', '''(SELECT *
    FROM  DimAeropuertoHistoria) AS Temp_DimAeropuertoHistoria''') \
    .option('user', user) \
    .option('password', passwd) \
    .option('driver', 'org.mariadb.jdbc.Driver') \
    .load()
DimAeropuertoHistoria.count()
print("conteo segunda carga base de datos : {0}".format(DimAeropuertoHistoria.
 ↪count()))

x=actualizar_historia_aeropuertos(dfaeropuertosconhistoriaparte3)
print("conteo tercera carga dataframe : {0}".format(x))

DimAeropuertoHistoria=spark.read.format('jdbc')\
    .option('url', db_multidimensional_connection_string) \
    .option('dbtable', '''(SELECT *
    FROM  DimAeropuertoHistoria) AS Temp_DimAeropuertoHistoria''') \
```

```
    .option('user', user) \
    .option('password', passwd) \
    .option('driver', 'org.mariadb.jdbc.Driver') \
    .load()
DimAeropuertoHistoria.count()
print("conteo tercera carga base de datos : {0}".format(DimAeropuertoHistoria.
 ↪count()))
```

```
conteo inicial base de datos : 0
conteo primera carga dataframe : 264
conteo primera carga base de datos : 264
conteo segunda carga dataframe : 562
conteo segunda carga base de datos : 562
conteo tercera carga dataframe : 860
conteo tercera carga base de datos : 860
```

[86]:
```
dfaeropuertosconhistoria.count()
```

[86]: 860

[92]:
```
DimAeropuertoHistoria.show(5)
```

```
+------------+-----+--------------+--------------------+---------+-------+-----
---+----------+------------+--------------------+-----------+-------------+-----
-----+-----+---------+----------+-----+----------+-------------+---------------
+-------------------+----------------+------------------+----+-----+
|IDAeropuerto|Sigla|           IATA|
NombreAeropuerto|Categoria|Latitud|Longitud| Municipio|Departamento|
Propietario| Explotador|LongitudPista|AnchoPista|
PBMO|Elevacion|Resolucion|Clase|      Tipo|GCD_Municipio|GCD_Departamento|FechaI
nicioVigencia|FechaFinVigencia|VersionDelRegistro|Anio|IDPIB|
+------------+-----+--------------+--------------------+---------+-------+-----
---+----------+------------+--------------------+-----------+-------------+-----
-----+-----+---------+----------+-----+----------+-------------+---------------
+-------------------+----------------+------------------+----+-----+
|         587|  7GY|no-especificado|         LAS ACACIAS|Aeródromo|
5.0949|-73.9054|   Nemocón|Cundinamarca|JUAN BERNANDO SAN…|   EL MISMO|
1289|        14|10000|     8431|   439,000|   3A|   Público|        25486|
25|         2016-01-01|      2300-01-01|                 S|2016|   80|
|         689|  9DI|no-especificado|          SAN FELIPE|Aeródromo|
1.915|-67.0776|San Felipe|     Guainía|         GOBERNACION|GOBERNACION|
3701|        59|25000|      312|   545,000|   1A| Aerocivil|        94883|
94|         2016-01-01|      2300-01-01|                 S|2016|   81|
|         593|  7HE|no-especificado|GUACHARACAS (COLO…|Aeródromo|
4.7371|-74.8021|   Beltrán|Cundinamarca| EMPRESA GUACHARACAS|  CCA. LTDA|
3|         0| 3000|      797| 5,920,000|   1A| Aerocivil|        25086|
25|         2016-01-01|      2300-01-01|                 S|2016|   80|
|         603|  7HQ|no-especificado|GENERAL ADELMO RU…|Aeródromo|
```

```
4.35|-74.7511|  Ricaurte|Cundinamarca|  EDUARDO A. VIVEROS|       ACAF|
69|        71|  500|      987|    46,000|   1A| Aerocivil|        25612|
25|         2016-01-01|      2300-01-01|                 S|2016|   80|
|         750|  A01|no-especificado|        CAMPO ALEGRE|Aeródromo|
1.8653|-69.0101|   Inírida|     Guainía|          GOBERNACION|GOBERNACION|
2341|        50| 2000|      577| 1,333,000|   1A|Fumigación|        94001|
94|         2016-01-01|      2300-01-01|                 S|2016|   81|
+------------+-----+--------------+--------------------+---------+-------+-----
---+----------+------------+--------------------+-----------+-------------+-----
-----+-----+---------+----------+-----+----------+-------------+---------------
+------------------+---------------+------------------+----+-----+
only showing top 5 rows
```

[95]:
```
#Creacion Tabla de Hechos Vuelos
import pyspark.sql.functions as F
df_aeropuertoorigen=DimAeropuertoHistoria.selectExpr('Sigla as␣
 ↪origen','Anio','FechaInicioVigencia',
                                                      ␣
 ↪'FechaFinVigencia','IDAeropuerto as IDAeropuertoOrigen')
df_aeropuertodestino=DimAeropuertoHistoria.selectExpr('Sigla as␣
 ↪destino','Anio','FechaInicioVigencia',
                                                     ␣
 ↪'FechaFinVigencia','IDAeropuerto as IDAeropuertoDestino')

df_hechos_vuelos=dfvuelosnew.alias('df_hechos_vuelos')
columns = ['vuelos', 'sillas','pasajeros','carga_bordo','carga_ofrecida']
for column in columns:
    df_hechos_vuelos = df_hechos_vuelos.withColumn(column,F.when(F.isnan(F.
 ↪col(column)),0).otherwise(F.col(column)))
df_hechos_vuelos= df_hechos_vuelos.withColumn("relativedate",sf.concat(sf.
 ↪col('ano'),lit("-"),sf.col('mes'),sf.lit('-01')))
df_hechos_vuelos= df_hechos_vuelos.withColumn("vuelos",df_hechos_vuelos.vuelos.
 ↪cast('int'))
df_hechos_vuelos= df_hechos_vuelos.withColumn("sillas",df_hechos_vuelos.sillas.
 ↪cast('int'))
df_hechos_vuelos= df_hechos_vuelos.withColumn("carga_ofrecida",df_hechos_vuelos.
 ↪carga_ofrecida.cast('int'))
df_hechos_vuelos= df_hechos_vuelos.withColumn("carga_bordo",df_hechos_vuelos.
 ↪carga_bordo.cast('int'))
df_hechos_vuelos= df_hechos_vuelos.withColumn("pasajeros",df_hechos_vuelos.
 ↪pasajeros.cast('int'))
df_hechos_vuelos=df_hechos_vuelos.
 ↪groupBy("relativedate","ano","mes","origen","destino","tipo_equipo","tipo_vuelo","trafico",
 ↪\
    .sum("vuelos","pasajeros","carga_bordo","sillas","carga_ofrecida")
```

```
df_hechos_vuelos=df_hechos_vuelos.withColumnRenamed("sum(vuelos)", "Vuelos")
df_hechos_vuelos=df_hechos_vuelos.withColumnRenamed("sum(pasajeros)",␣
↪"Pasajeros")
df_hechos_vuelos=df_hechos_vuelos.withColumnRenamed("sum(carga_bordo)",␣
↪"CargaBordo")
df_hechos_vuelos=df_hechos_vuelos.withColumnRenamed("sum(sillas)",␣
↪"TotalSillas")
df_hechos_vuelos=df_hechos_vuelos.withColumnRenamed("sum(carga_ofrecida)",␣
↪"TotalCarga")
df_hechos_vuelos.createOrReplaceTempView("df_hechos_vuelos")
df_aeropuertoorigen.createOrReplaceTempView("df_aeropuertoorigen")
df_aeropuertodestino.createOrReplaceTempView("df_aeropuertodestino")
df_fechames.createOrReplaceTempView("df_fechames")
df_tipo_vuelo.createOrReplaceTempView("df_tipo_vuelo")
df_empresatrans1.createOrReplaceTempView("df_empresatrans1")
df_trafico.createOrReplaceTempView("df_trafico")
df_tipoequipo.createOrReplaceTempView("df_tipoequipo")
df_hechos_vuelos = spark.sql("select ␣
↪IDFecha,IDTipoVuelo,IDTipoTrafico,IDEmpresa,IDEquipo,IDAeropuertoOrigen,\
                              ␣
↪IDAeropuertoDestino,Vuelos,Pasajeros,CargaBordo,TotalSillas,TotalCarga from␣
↪df_hechos_vuelos\
                              left join df_fechames on df_fechames.Anio=␣
↪df_hechos_vuelos.ano and\
                              df_fechames.Mes= df_hechos_vuelos.mes\
                              left join df_tipo_vuelo on df_tipo_vuelo.
↪CodigoVuelo= df_hechos_vuelos.tipo_vuelo\
                              left join df_trafico on df_trafico.Codigo_Trafico=␣
↪df_hechos_vuelos.trafico\
                              left join df_empresatrans1 on df_empresatrans1.
↪NombreEmpresa= df_hechos_vuelos.empresa\
                              left join df_tipoequipo on df_tipoequipo.
↪NombreEquipo= df_hechos_vuelos.tipo_equipo\
                              left join df_aeropuertoorigen on␣
↪df_aeropuertoorigen.origen= df_hechos_vuelos.origen and\
                              (df_hechos_vuelos.relativedate BETWEEN␣
↪df_aeropuertoorigen.FechaInicioVigencia\
                              AND df_aeropuertoorigen.FechaFinVigencia)\
                              left join df_aeropuertodestino on␣
↪df_aeropuertodestino.destino= df_hechos_vuelos.destino and\
                              (df_hechos_vuelos.relativedate BETWEEN␣
↪df_aeropuertodestino.FechaInicioVigencia\
                              AND df_aeropuertodestino.FechaFinVigencia)")
df_hechos_vuelos = df_hechos_vuelos.coalesce(1).withColumn("ID", mid())
df_hechos_vuelos.toPandas().to_csv('ModelExt/FactVuelos.csv')
df_hechos_vuelos.select('*').write.format('jdbc') \
```

```
          .mode('overwrite') \
          .option('url', db_multidimensional_connection_string) \
          .option('dbtable', 'FactVuelos') \
          .option('user', user) \
          .option('password', passwd) \
          .option('driver', 'org.mariadb.jdbc.Driver') \
          .save()
df_hechos_vuelos.show(5)
```

```
+-------+-----------+-------------+---------+--------+-----------------+-------
------------+------+---------+----------+-----------+----------+---+
|IDFecha|IDTipoVuelo|IDTipoTrafico|IDEmpresa|IDEquipo|IDAeropuertoOrigen|IDAerop
uertoDestino|Vuelos|Pasajeros|CargaBordo|TotalSillas|TotalCarga| ID|
+-------+-----------+-------------+---------+--------+-----------------+-------
------------+------+---------+----------+-----------+----------+---+
|     76|          3|            0|       50|      42|              828|
674|     1|        0|         0|          0|         0|  0|
|     17|          3|            0|       13|      50|              183|
97|     1|        2|       200|          2|       200|  1|
|     44|          3|            0|       62|      52|              211|
25|     1|        5|        40|          5|        40|  2|
|     24|          3|            0|       69|      50|              183|
53|     1|        2|         0|          2|         0|  3|
|     38|          3|            0|        1|      50|              211|
53|     1|        1|         0|          1|         0|  4|
+-------+-----------+-------------+---------+--------+-----------------+-------
------------+------+---------+----------+-----------+----------+---+
only showing top 5 rows
```

[94]:
```
print('DimFechaMes')
table=spark.read.format('jdbc')\
    .option('url', db_multidimensional_connection_string) \
    .option('dbtable', '''(SELECT * FROM DimFechaMes ) AS Temp_DimFechaMes''') \
    .option('user', user) \
    .option('password', passwd) \
    .option('driver', 'org.mariadb.jdbc.Driver') \
    .load()
print("columnas de la tabla : {0}".format(len(table.columns)))
print("filas de la tabla : {0}".format(table.count()))
print(table.show(5))
print('DimTipoVuelo')
table=spark.read.format('jdbc')\
    .option('url', db_multidimensional_connection_string) \
    .option('dbtable', '''(SELECT * FROM DimTipoVuelo ) AS␣
 ↪Temp_DimTipoVuelo''') \
    .option('user', user) \
```

```
    .option('password', passwd) \
    .option('driver', 'org.mariadb.jdbc.Driver') \
    .load()
print("columnas de la tabla : {0}".format(len(table.columns)))
print("filas de la tabla : {0}".format(table.count()))
print(table.show(5))
print('DimTipo_Trafico')
table=spark.read.format('jdbc')\
    .option('url', db_multidimensional_connection_string) \
    .option('dbtable', '''(SELECT * FROM DimTipo_Trafico ) AS␣
 ↪Temp_DimTipo_Trafico''') \
    .option('user', user) \
    .option('password', passwd) \
    .option('driver', 'org.mariadb.jdbc.Driver') \
    .load()
print("columnas de la tabla : {0}".format(len(table.columns)))
print("filas de la tabla : {0}".format(table.count()))
print(table.show(5))
print('DimEmpresaTransportadora')
table=spark.read.format('jdbc')\
    .option('url', db_multidimensional_connection_string) \
    .option('dbtable', '''(SELECT * FROM DimEmpresaTransportadora ) AS␣
 ↪Temp_DimTipo_Trafico''') \
    .option('user', user) \
    .option('password', passwd) \
    .option('driver', 'org.mariadb.jdbc.Driver') \
    .load()
print("columnas de la tabla : {0}".format(len(table.columns)))
print("filas de la tabla : {0}".format(table.count()))
print(table.show(5))
print('DimTipo_Equipo')
table=spark.read.format('jdbc')\
    .option('url', db_multidimensional_connection_string) \
    .option('dbtable', '''(SELECT * FROM DimTipo_Equipo ) AS␣
 ↪Temp_DimTipo_Equipo''') \
    .option('user', user) \
    .option('password', passwd) \
    .option('driver', 'org.mariadb.jdbc.Driver') \
    .load()
print("columnas de la tabla : {0}".format(len(table.columns)))
print("filas de la tabla : {0}".format(table.count()))
print(table.show(5))
print('DimInformacionPIB')
table=spark.read.format('jdbc')\
    .option('url', db_multidimensional_connection_string) \
    .option('dbtable', '''(SELECT * FROM DimInformacionPIB ) AS␣
 ↪Temp_DimInformacionPIB''') \
```

```
    .option('user', user) \
    .option('password', passwd) \
    .option('driver', 'org.mariadb.jdbc.Driver') \
    .load()
print("columnas de la tabla : {0}".format(len(table.columns)))
print("filas de la tabla : {0}".format(table.count()))
print(table.show(5))
print('DimAeropuertoHistoria')
table=spark.read.format('jdbc')\
    .option('url', db_multidimensional_connection_string) \
    .option('dbtable', '''(SELECT * FROM DimAeropuertoHistoria ) AS␣
 ↪Temp_DimAeropuertoHistoria''') \
    .option('user', user) \
    .option('password', passwd) \
    .option('driver', 'org.mariadb.jdbc.Driver') \
    .load()
print("columnas de la tabla : {0}".format(len(table.columns)))
print("filas de la tabla : {0}".format(table.count()))
print(table.show(5))
print('FactVuelos')
table=spark.read.format('jdbc')\
    .option('url', db_multidimensional_connection_string) \
    .option('dbtable', '''(SELECT * FROM FactVuelos ) AS Temp_DimFactVuelos''')␣
 ↪\
    .option('user', user) \
    .option('password', passwd) \
    .option('driver', 'org.mariadb.jdbc.Driver') \
    .load()
print("columnas de la tabla : {0}".format(len(table.columns)))
print("filas de la tabla : {0}".format(table.count()))
print(table.show(5))
```

```
DimFechaMes
columnas de la tabla : 3
filas de la tabla : 84
+----+---+-------+
|Anio|Mes|IDFecha|
+----+---+-------+
|2010|  1|      0|
|2010|  2|      1|
|2010|  3|      2|
|2010|  4|      3|
|2010|  5|      4|
+----+---+-------+
only showing top 5 rows

None
```

```
DimTipoVuelo
columnas de la tabla : 3
filas de la tabla : 5
+-----------+-----------+-----------+
|CodigoVuelo|  TipoVuelo|IDTipoVuelo|
+-----------+-----------+-----------+
|          A|Adicionales|          0|
|          C|    Charter|          1|
|          R|    Regular|          2|
|          T|       Taxi|          3|
|        nan|        nan|          4|
+-----------+-----------+-----------+

None
DimTipo_Trafico
columnas de la tabla : 3
filas de la tabla : 1
+--------------+-----------+-------------+
|Codigo_Trafico|Descripcion|IDTipoTrafico|
+--------------+-----------+-------------+
|             N|   Nacional|            0|
+--------------+-----------+-------------+

None
DimEmpresaTransportadora
columnas de la tabla : 2
filas de la tabla : 87
+--------------------+---------+
|       NombreEmpresa|IDEmpresa|
+--------------------+---------+
| AER CARIBE LIMITADA|        0|
|          AERO APOYO|        1|
|AERO SERVICIOS ES…|        2|
|AERO TAXI GUAYMAR…|        3|
|AEROCHARTER ANDIN…|        4|
+--------------------+---------+
only showing top 5 rows

None
DimTipo_Equipo
columnas de la tabla : 2
filas de la tabla : 112
+------------+--------+
|NombreEquipo|IDEquipo|
+------------+--------+
|         318|       0|
|         319|       1|
|         330|       2|
```

```
|         332|       3|
|         727|       4|
+------------+--------+
only showing top 5 rows

None
DimInformacionPIB
columnas de la tabla : 7
filas de la tabla : 132
+-----+------------+-----+------------------+---------------+-----------------
-+----+
|IDPIB|Departamento|
PIB|FechaInicioVigencia|FechaFinVigencia|VersionDelRegistro|Anio|
+-----+------------+-----+------------------+---------------+-----------------
-+----+
|    0|    Amazonas|  157|         1900-01-01|      2014-12-31|
N|2014|
|    1|   Antioquia|16705|         1900-01-01|      2014-12-31|
N|2014|
|    2|      Arauca|  411|         1900-01-01|      2014-12-31|
N|2014|
|    3|   Atlántico| 6041|         1900-01-01|      2014-12-31|
N|2014|
|    4|Bogotá, D.C.|35338|         1900-01-01|      2014-12-31|
N|2014|
+-----+------------+-----+------------------+---------------+-----------------
-+----+
only showing top 5 rows

None
DimAeropuertoHistoria
columnas de la tabla : 25
filas de la tabla : 860
+------------+-----+--------------+--------------------+---------+-------+-----
---+----------+------------+--------------------+-----------+-------------+-----
-----+-----+---------+----------+-----+----------+-------------+---------------
+-------------------+---------------+------------------+----+-----+
|IDAeropuerto|Sigla|           IATA|
NombreAeropuerto|Categoria|Latitud|Longitud| Municipio|Departamento|
Propietario| Explotador|LongitudPista|AnchoPista|
PBMO|Elevacion|Resolucion|Clase|      Tipo|GCD_Municipio|GCD_Departamento|FechaI
nicioVigencia|FechaFinVigencia|VersionDelRegistro|Anio|IDPIB|
+------------+-----+--------------+--------------------+---------+-------+-----
---+----------+------------+--------------------+-----------+-------------+-----
-----+-----+---------+----------+-----+----------+-------------+---------------
+-------------------+---------------+------------------+----+-----+
|         587|  7GY|no-especificado|         LAS ACACIAS|Aeródromo|
5.0949|-73.9054|   Nemocón|Cundinamarca|JUAN BERNANDO SAN…|   EL MISMO|
```

```
1289|        14|10000|     8431|   439,000|   3A|   Público|        25486|
25|         2016-01-01|      2300-01-01|                S|2016|   80|
|         689|  9DI|no-especificado|          SAN FELIPE|Aeródromo|
1.915|-67.0776|San Felipe|     Guainía|         GOBERNACION|GOBERNACION|
3701|        59|25000|      312|   545,000|   1A| Aerocivil|        94883|
94|         2016-01-01|      2300-01-01|                S|2016|   81|
|         593|  7HE|no-especificado|GUACHARACAS (COLO…|Aeródromo|
4.7371|-74.8021|   Beltrán|Cundinamarca| EMPRESA GUACHARACAS|  CCA. LTDA|
3|         0| 3000|      797| 5,920,000|   1A| Aerocivil|        25086|
25|         2016-01-01|      2300-01-01|                S|2016|   80|
|         603|  7HQ|no-especificado|GENERAL ADELMO RU…|Aeródromo|
4.35|-74.7511|  Ricaurte|Cundinamarca|  EDUARDO A. VIVEROS|       ACAF|
69|        71|  500|      987|    46,000|   1A| Aerocivil|        25612|
25|         2016-01-01|      2300-01-01|                S|2016|   80|
|         750|  A01|no-especificado|        CAMPO ALEGRE|Aeródromo|
1.8653|-69.0101|   Inírida|     Guainía|         GOBERNACION|GOBERNACION|
2341|        50| 2000|      577| 1,333,000|   1A|Fumigación|        94001|
94|         2016-01-01|      2300-01-01|                S|2016|   81|
+------------+-----+--------------+--------------------+---------+-------+-----
---+----------+------------+--------------------+-----------+-------------+-----
-----+-----+---------+----------+-----+----------+-------------+---------------
+-------------------+---------------+------------------+----+-----+
only showing top 5 rows

None
FactVuelos
columnas de la tabla : 13
filas de la tabla : 82386
+-------+-----------+-------------+---------+--------+-----------------+-------
------------+------+---------+----------+-----------+----------+---+
|IDFecha|IDTipoVuelo|IDTipoTrafico|IDEmpresa|IDEquipo|IDAeropuertoOrigen|IDAerop
uertoDestino|Vuelos|Pasajeros|CargaBordo|TotalSillas|TotalCarga| ID|
+-------+-----------+-------------+---------+--------+-----------------+-------
------------+------+---------+----------+-----------+----------+---+
|     76|          3|            0|       50|      42|               828|
674|     1|        0|         0|          0|         0|  0|
|     17|          3|            0|       13|      50|               183|
97|     1|        2|       200|          2|       200|  1|
|     44|          3|            0|       62|      52|               211|
25|     1|        5|        40|          5|        40|  2|
|     24|          3|            0|       69|      50|               183|
53|     1|        2|         0|          2|         0|  3|
|     38|          3|            0|        1|      50|               211|
53|     1|        1|         0|          1|         0|  4|
+-------+-----------+-------------+---------+--------+-----------------+-------
------------+------+---------+----------+-----------+----------+---+
only showing top 5 rows
```

```
None
```

[ ]: