**from pyspark.sql.functions import lit**

**lapd\_df1 = spark.read.option("header", "true").csv("s3://final-project-bucket-group-5/raw-data-ny-la-cpd/Crime\_Data\_from\_2010\_to\_2019.csv").withColumn("source", lit("lapd"))**

**lapd\_df2 = spark.read.option("header", "true").csv("s3://final-project-bucket-group-5/raw-data-ny-la-cpd/Crime\_Data\_from\_2020\_to\_Present\_20250730.csv").withColumn("source", lit("lapd"))**

**# One NYPD file**

**nypd\_df = spark.read.option("header", "true").csv("s3://final-project-bucket-group-5/raw-data-ny-la-cpd/NYPD\_Complaint\_Data\_Historic.csv").withColumn("source", lit("nypd"))**

**# One CPD file**

**cpd\_df = spark.read.option("header", "true").csv("s3://final-project-bucket-group-5/raw-data-ny-la-cpd/Crimes\_-\_2001\_to\_Present.csv").withColumn("source", lit("cpd"))**

**def map\_lapd(df):**

**return df.selectExpr(**

**"DR\_NO as case\_id",**

**"`Date Rptd` as report\_date",**

**"`DATE OCC` as occurrence\_start\_date",**

**"`TIME OCC` as occurrence\_start\_time",**

**"null as occurrence\_end\_date",**

**"null as occurrence\_end\_time",**

**"`Crm Cd` as crime\_code",**

**"`Crm Cd Desc` as crime\_description",**

**"`Status Desc` as crime\_completed",**

**"`Part 1-2` as offense\_severity",**

**"`Weapon Desc` as weapon\_description",**

**"Mocodes as mocodes",**

**"`Premis Desc` as location\_description",**

**"`Cross Street` as address\_block",**

**"LOCATION as location\_name",**

**"cast(LAT as double) as latitude",**

**"cast(LON as double) as longitude",**

**"null as x\_coord",**

**"null as y\_coord",**

**"`Rpt Dist No` as reporting\_district",**

**"`AREA NAME` as area\_name",**

**"null as jurisdiction",**

**"null as station\_name",**

**"null as fbi\_code",**

**"case when Status = 'AR' then 'true' else 'false' end as arrest\_made",**

**"null as domestic\_incident",**

**"cast(`Vict Age` as int) as victim\_age",**

**"`Vict Sex` as victim\_sex",**

**"`Vict Descent` as victim\_race",**

**"null as suspect\_age",**

**"null as suspect\_sex",**

**"null as suspect\_race",**

**"null as housing\_project\_name",**

**"null as park\_name",**

**"null as transit\_district",**

**"source"**

**)**

**def map\_nypd(df):**

**return df.selectExpr(**

**"CMPLNT\_NUM as case\_id",**

**"RPT\_DT as report\_date",**

**"CMPLNT\_FR\_DT as occurrence\_start\_date",**

**"CMPLNT\_FR\_TM as occurrence\_start\_time",**

**"CMPLNT\_TO\_DT as occurrence\_end\_date",**

**"CMPLNT\_TO\_TM as occurrence\_end\_time",**

**"KY\_CD as crime\_code",**

**"OFNS\_DESC as crime\_description",**

**"CRM\_ATPT\_CPTD\_CD as crime\_completed",**

**"LAW\_CAT\_CD as offense\_severity",**

**"null as weapon\_description",**

**"null as mocodes",**

**"PREM\_TYP\_DESC as location\_description",**

**"null as address\_block",**

**"Lat\_Lon as location\_name",**

**"cast(Latitude as double) as latitude",**

**"cast(Longitude as double) as longitude",**

**"cast(X\_COORD\_CD as double) as x\_coord",**

**"cast(Y\_COORD\_CD as double) as y\_coord",**

**"ADDR\_PCT\_CD as reporting\_district",**

**"BORO\_NM as area\_name",**

**"JURIS\_DESC as jurisdiction",**

**"STATION\_NAME as station\_name",**

**"null as fbi\_code",**

**"case when CRM\_ATPT\_CPTD\_CD = 'COMPLETED' then 'true' else 'false' end as arrest\_made",**

**"null as domestic\_incident",**

**"null as victim\_age",**

**"VIC\_SEX as victim\_sex",**

**"VIC\_RACE as victim\_race",**

**"SUSP\_AGE\_GROUP as suspect\_age",**

**"SUSP\_SEX as suspect\_sex",**

**"SUSP\_RACE as suspect\_race",**

**"HADEVELOPT as housing\_project\_name",**

**"PARKS\_NM as park\_name",**

**"TRANSIT\_DISTRICT as transit\_district",**

**"source"**

**)**

**def map\_cpd(df):**

**return df.selectExpr(**

**"`Case Number` as case\_id",**

**"`Updated On` as report\_date",**

**"Date as occurrence\_start\_date",**

**"null as occurrence\_start\_time",**

**"null as occurrence\_end\_date",**

**"null as occurrence\_end\_time",**

**"IUCR as crime\_code",**

**"`Primary Type` as crime\_description",**

**"case when Arrest = 'true' then 'true' else 'false' end as crime\_completed",**

**"null as offense\_severity",**

**"null as weapon\_description",**

**"null as mocodes",**

**"`Location Description` as location\_description",**

**"Block as address\_block",**

**"Location as location\_name",**

**"cast(Latitude as double) as latitude",**

**"cast(Longitude as double) as longitude",**

**"cast(`X Coordinate` as double) as x\_coord",**

**"cast(`Y Coordinate` as double) as y\_coord",**

**"District as reporting\_district",**

**"`Community Area` as area\_name",**

**"null as jurisdiction",**

**"null as station\_name",**

**"`FBI Code` as fbi\_code",**

**"Arrest as arrest\_made",**

**"Domestic as domestic\_incident",**

**"null as victim\_age",**

**"null as victim\_sex",**

**"null as victim\_race",**

**"null as suspect\_age",**

**"null as suspect\_sex",**

**"null as suspect\_race",**

**"null as housing\_project\_name",**

**"null as park\_name",**

**"null as transit\_district",**

**"source"**

**)**

**# Apply mappings**

**lapd\_df1\_mapped = map\_lapd(lapd\_df1)**

**lapd\_df2\_mapped = map\_lapd(lapd\_df2)**

**nypd\_df\_mapped  = map\_nypd(nypd\_df)**

**cpd\_df\_mapped   = map\_cpd(cpd\_df)**

**from functools import reduce**

**# All should have the same column order now**

**all\_dfs = [lapd\_df1\_mapped, lapd\_df2\_mapped, nypd\_df\_mapped, cpd\_df\_mapped]**

**master\_df = reduce(lambda df1, df2: df1.unionByName(df2), all\_dfs)**

**from pyspark.sql.functions import col, when, lpad, split, lit, concat\_ws**

**transform\_df = master\_df.withColumn(**

**"occurrence\_start\_time",**

**when(col("source") == "lapd",**

**concat\_ws(":",**

**lpad(col("occurrence\_start\_time"), 4, "0").substr(1, 2),**

**lpad(col("occurrence\_start\_time"), 4, "0").substr(3, 2),**

**lit("00")**

**)**

**)**

**.when(col("source") == "cpd",**

**split(col("occurrence\_start\_date"), " ").getItem(1)**

**)**

**.otherwise(col("occurrence\_start\_time"))**

**)**

**from pyspark.sql.functions import split, col**

**# Remove time part (anything after space) from report\_date**

**transform\_df = transform\_df.withColumn("report\_date", split(col("report\_date"), " ").getItem(0))**

**# Remove time part from occurrence\_start\_date**

**transform\_df = transform\_df.withColumn("occurrence\_start\_date", split(col("occurrence\_start\_date"), " ").getItem(0))**

**from pyspark.sql.functions import col, when**

**transform\_df = transform\_df.withColumn(**

**"crime\_category",**

**when(col("crime\_description").contains("ASSAULT"), "Assault & Battery")**

**.when(col("crime\_description").contains("BATTERY"), "Assault & Battery")**

**.when(col("crime\_description").contains("HOMICIDE"), "Homicide & Attempted Murder")**

**.when(col("crime\_description").contains("MURDER"), "Homicide & Attempted Murder")**

**.when(col("crime\_description").contains("THEFT"), "Theft & Burglary")**

**.when(col("crime\_description").contains("BURGLARY"), "Theft & Burglary")**

**.when(col("crime\_description").contains("LARCENY"), "Theft & Burglary")**

**.when(col("crime\_description").contains("ROBBERY"), "Robbery")**

**.when(col("crime\_description").contains("RAPE"), "Sex Crimes")**

**.when(col("crime\_description").contains("SEX"), "Sex Crimes")**

**.when(col("crime\_description").contains("DRUG"), "Drug-Related Offenses")**

**.when(col("crime\_description").contains("WEAPON"), "Weapons Offenses")**

**.when(col("crime\_description").contains("FIREARM"), "Weapons Offenses")**

**.when(col("crime\_description").contains("CHILD"), "Child-Related Crimes")**

**.when(col("crime\_description").contains("FRAUD"), "Fraud & Forgery")**

**.when(col("crime\_description").contains("FORGERY"), "Fraud & Forgery")**

**.when(col("crime\_description").contains("CREDIT"), "Fraud & Forgery")**

**.when(col("crime\_description").contains("VEHICLE"), "Traffic & Vehicle Offenses")**

**.when(col("crime\_description").contains("DRIVING"), "Traffic & Vehicle Offenses")**

**.when(col("crime\_description").contains("LOITERING"), "Public Disturbance & Nuisance")**

**.when(col("crime\_description").contains("DISORDERLY"), "Public Disturbance & Nuisance")**

**.when(col("crime\_description").contains("DOMESTIC"), "Domestic & Intimate Partner Violence")**

**.when(col("crime\_description").contains("PARTNER"), "Domestic & Intimate Partner Violence")**

**.when(col("crime\_description").contains("PROSTITUTION"), "Sex Work / Human Trafficking")**

**.when(col("crime\_description").contains("PIMPING"), "Sex Work / Human Trafficking")**

**.otherwise("Other / Miscellaneous")**

**)**

**from pyspark.sql.functions import col, when**

**transform\_df = transform\_df.withColumn(**

**"weapon\_category",**

**when(col("weapon\_description").contains("GUN"), "Firearm")**

**.when(col("weapon\_description").contains("RIFLE"), "Firearm")**

**.when(col("weapon\_description").contains("REVOLVER"), "Firearm")**

**.when(col("weapon\_description").contains("PISTOL"), "Firearm")**

**.when(col("weapon\_description").contains("FIREARM"), "Firearm")**

**.when(col("weapon\_description").contains("KNIFE"), "Knife/Sharp Object")**

**.when(col("weapon\_description").contains("BLADE"), "Knife/Sharp Object")**

**.when(col("weapon\_description").contains("RAZOR"), "Knife/Sharp Object")**

**.when(col("weapon\_description").contains("SWORD"), "Knife/Sharp Object")**

**.when(col("weapon\_description").contains("ICE PICK"), "Knife/Sharp Object")**

**.when(col("weapon\_description").contains("CLEAVER"), "Knife/Sharp Object")**

**.when(col("weapon\_description").contains("SCREWDRIVER"), "Knife/Sharp Object")**

**.when(col("weapon\_description").contains("PIPE"), "Blunt Object")**

**.when(col("weapon\_description").contains("BAT"), "Blunt Object")**

**.when(col("weapon\_description").contains("HAMMER"), "Blunt Object")**

**.when(col("weapon\_description").contains("BOARD"), "Blunt Object")**

**.when(col("weapon\_description").contains("IRON"), "Blunt Object")**

**.when(col("weapon\_description").contains("BRICK"), "Blunt Object")**

**.when(col("weapon\_description").contains("CHAIN"), "Blunt Object")**

**.when(col("weapon\_description").contains("BLUNT INSTRUMENT"), "Blunt Object")**

**.when(col("weapon\_description").contains("MACE"), "Chemical/Explosive")**

**.when(col("weapon\_description").contains("PEPPER SPRAY"), "Chemical/Explosive")**

**.when(col("weapon\_description").contains("CHEMICAL"), "Chemical/Explosive")**

**.when(col("weapon\_description").contains("EXPLOSIVE"), "Chemical/Explosive")**

**.when(col("weapon\_description").contains("SCALDING LIQUID"), "Chemical/Explosive")**

**.when(col("weapon\_description").contains("BOMB"), "Chemical/Explosive")**

**.when(col("weapon\_description").contains("SIMULATED"), "Simulated Weapon")**

**.when(col("weapon\_description").contains("TOY"), "Simulated Weapon")**

**.when(col("weapon\_description").contains("STARTER PISTOL"), "Simulated Weapon")**

**.when(col("weapon\_description").contains("DEMAND NOTE"), "Verbal/Threat")**

**.when(col("weapon\_description").contains("VERBAL THREAT"), "Verbal/Threat")**

**.when(col("weapon\_description").contains("DOG"), "Biological/Animal/Body Force")**

**.when(col("weapon\_description").contains("PHYSICAL"), "Biological/Animal/Body Force")**

**.when(col("weapon\_description").contains("STRONG-ARM"), "Biological/Animal/Body Force")**

**.when(col("weapon\_description").contains("LIQUOR"), "Biological/Animal/Body Force")**

**.otherwise("Other/Unknown")**

**)**

**from pyspark.sql.functions import when, col**

**transform\_df = transform\_df.withColumn(**

**"city",**

**when(col("source") == "nypd", "New York") \**

**.when(col("source") == "lapd", "Los Angeles") \**

**.when(col("source") == "cpd", "Chicago") \**

**.otherwise("Unknown")**

**)**

**transform\_df.createOrReplaceTempView("tdf")**

**transform\_df = spark.sql("""**

**SELECT \*,**

**CASE**

**WHEN location\_description IN (**

**'SINGLE FAMILY DWELLING', 'MULTI-UNIT DWELLING', 'APARTMENT',**

**'CONDOMINIUM/TOWNHOUSE', 'HOUSE', 'MOBILE HOME/TRAILERS'**

**) THEN 'Residential'**

**WHEN location\_description IN (**

**'GAS STATION', 'JEWELRY STORE', 'GROCERY STORE', 'LIQUOR STORE',**

**'CLOTHING STORE', 'SUPERMARKET', 'BAR/COCKTAIL/NIGHTCLUB',**

**'NAIL SALON', 'PHOTO/COPY', 'RESTAURANT/FAST FOOD'**

**) THEN 'Commercial'**

**WHEN location\_description LIKE 'MTA%' OR location\_description IN (**

**'BUS STOP', 'SUBWAY PLATFORM', 'TRAIN TRACKS', 'PARKING LOT',**

**'TAXI', 'AIRPORT TERMINAL'**

**) THEN 'Transit'**

**WHEN location\_description IN (**

**'HOSPITAL', 'DOCTOR/DENTIST OFFICE', 'NURSING HOME',**

**'CLINIC', 'VETERINARIAN', 'HOSPICE', 'MEDICAL MARIJUANA'**

**) THEN 'Medical'**

**WHEN location\_description IN (**

**'ELEMENTARY SCHOOL', 'HIGH SCHOOL', 'COLLEGE',**

**'UNIVERSITY', 'PRIVATE SCHOOL', 'TRADE SCHOOL'**

**) THEN 'Educational'**

**WHEN location\_description IN (**

**'GOVERNMENT FACILITY', 'FIRE STATION', 'POLICE FACILITY',**

**'POST OFFICE', 'COURTHOUSE', 'JAIL/DETENTION CENTER'**

**) THEN 'Government'**

**WHEN location\_description IN (**

**'CHURCH', 'MOSQUE', 'SYNAGOGUE', 'TEMPLE',**

**'PLACE OF WORSHIP'**

**) THEN 'Religious'**

**WHEN location\_description IN (**

**'PARK/PLAYGROUND', 'STREET', 'ALLEY', 'BEACH', 'LAKE',**

**'RIVER', 'YARD', 'OPEN LOT'**

**) THEN 'Outdoor'**

**ELSE 'Other'**

**END AS location\_category**

**FROM tdf**

**""")**

**transform\_df.createOrReplaceTempView("tdf1")**

**transform\_df = spark.sql("""**

**SELECT \*,**

**CASE**

**WHEN victim\_race IN ('WHITE') THEN 'White'**

**WHEN victim\_race IN ('BLACK') THEN 'Black or African American'**

**WHEN victim\_race IN ('WHITE HISPANIC') THEN 'Hispanic'**

**WHEN victim\_race IN ('BLACK HISPANIC') THEN 'Hispanic'**

**WHEN victim\_race IN ('ASIAN / PACIFIC ISLANDER') THEN 'Asian / Pacific Islander'**

**WHEN victim\_race IN ('AMERICAN INDIAN/ALASKAN NATIVE') THEN 'American Indian / Alaska Native'**

**WHEN victim\_race IN ('OTHER') THEN 'Other'**

**WHEN victim\_race IN ('UNKNOWN') THEN 'Unknown'**

**WHEN victim\_race IN ('W') THEN 'White'**

**WHEN victim\_race IN ('B') THEN 'Black or African American'**

**WHEN victim\_race IN ('A') THEN 'Asian'**

**WHEN victim\_race IN ('I') THEN 'American Indian / Alaska Native'**

**WHEN victim\_race IN ('H') THEN 'Hispanic'**

**WHEN victim\_race IN ('X') THEN 'Unknown'**

**WHEN victim\_race IN ('P', 'F', 'L', 'U', 'V', 'O', 'D', 'K', 'Z', 'C', 'S', 'J', 'G') THEN 'Other'**

**WHEN victim\_race IS NULL OR victim\_race = '-' OR victim\_race = '(null)' THEN 'Unknown'**

**ELSE 'Other'**

**END AS race\_group**

**FROM tdf1**

**""")**

**from pyspark.sql.functions import to\_date, row\_number**

**transform\_df1 = transform\_df.withColumn("report\_date", to\_date("report\_date", "MM/dd/yyyy")) \**

**.withColumn("occurrence\_start\_date", to\_date("occurrence\_start\_date", "MM/dd/yyyy")) \**

**.withColumn("occurrence\_end\_date", to\_date("occurrence\_end\_date", "MM/dd/yyyy"))**

**from pyspark.sql.window import Window**

**windowSpec = Window.orderBy("report\_date")**

**transform\_df1 = transform\_df1.withColumn("case\_num", row\_number().over(windowSpec))**

**transform\_df2 = transform\_df1.withColumnRenamed("occurrence\_start\_date", "occurred\_date") \**

**.withColumnRenamed("occurrence\_start\_time", "occurred\_time")**

**transform\_df2.createOrReplaceTempView("tdf2")**

**transform\_df2 = transform\_df2.withColumn("suspect\_race\_group",**

**when(col("suspect\_race").isin("WHITE HISPANIC", "BLACK HISPANIC", "HISPANIC"), "HISPANIC")**

**.otherwise(col("suspect\_race")))**

**from pyspark.sql.functions import when, col**

**transform\_df2 = transform\_df2.withColumn(**

**"suspect\_race\_group",**

**when(col("suspect\_race\_group").isNull(), "UNKNOWN")**

**.when(col("suspect\_race\_group") == "(null)", "UNKNOWN")**

**.otherwise(col("suspect\_race\_group"))**

**)**

**transform\_df2 = transform\_df2.withColumnRenamed("race\_group", "victim\_race\_group")**

**selected\_columns = [**

**'report\_date', 'occurred\_date', 'occurred\_time', 'crime\_code',**

**'latitude', 'longitude', 'jurisdiction', 'arrest\_made', 'domestic\_incident',**

**'victim\_age', 'victim\_sex', 'suspect\_age', 'suspect\_sex', 'source',**

**'crime\_category', 'weapon\_category', 'city', 'location\_category',**

**'victim\_race\_group', 'case\_num', 'suspect\_race\_group'**

**]**

**transform\_df2 = transform\_df2.select(\*selected\_columns)**

**transform\_df2.createOrReplaceTempView("tdf2")**

**from pyspark.sql.functions import abs**

**transform\_df2 = transform\_df2.withColumn("victim\_age", abs(transform\_df2["victim\_age"]))**

**from pyspark.sql.functions import when, col, trim**

**transform\_df2 = transform\_df2.withColumn(**

**"victim\_age\_group",**

**when(trim(col("victim\_age")) == "", "")  # preserve blank values**

**.when(col("victim\_age").cast("int") <= 18, "0-18")**

**.when(col("victim\_age").cast("int") <= 30, "19-30")**

**.when(col("victim\_age").cast("int") <= 45, "31-45")**

**.when(col("victim\_age").cast("int") <= 60, "46-60")**

**.when(col("victim\_age").cast("int") > 60, "60+")**

**)**

**transform\_df2.createOrReplaceTempView("tdf2")**

**transform\_df2 = transform\_df2.drop("suspect\_age")**

**transform\_df2 = transform\_df2.withColumn("victim\_Sex",**

**when(col("victim\_sex") == "F", "Female")**

**.when(col("victim\_sex") == "M", "Male")**

**.when(col("victim\_sex").isin("E", "D", "H", "L", "N", "U", "X", "-"), "Others")**

**.when(col("victim\_sex") == "(null)", "Unknown")**

**.when((col("victim\_sex").isNull()) | (col("victim\_sex") == ""), "Unknown")**

**.otherwise(col("victim\_sex"))  # Keep original if it doesn't match any case**

**)**

**from pyspark.sql.functions import when, col, trim**

**transform\_df2 = transform\_df2.withColumn(**

**"suspect\_sex",**

**when(trim(col("suspect\_sex")) == "F", "Female")**

**.when(trim(col("suspect\_sex")) == "M", "Male")**

**.when(trim(col("suspect\_sex")) == "U", "Unknown")**

**.when(trim(col("suspect\_sex")) == "(null)", "Unknown")**

**.when(col("suspect\_sex").isNull(), "Unknown")**

**.when(trim(col("suspect\_sex")) == "", "Unknown")**

**.otherwise("Unknown")**

**)**

**from pyspark.sql.functions import monotonically\_increasing\_id**

**# Dimension: Crime**

**dim\_crime = transform\_df2.select("crime\_code", "crime\_category", "weapon\_category", "source") \**

**.dropDuplicates() \**

**.withColumn("crime\_id", monotonically\_increasing\_id())**

**# Dimension: Victim**

**dim\_victim = transform\_df2.select("victim\_age", "Victim\_Sex", "victim\_race\_group", "victim\_age\_group") \**

**.dropDuplicates() \**

**.withColumn("victim\_id", monotonically\_increasing\_id())**

**# Dimension: Suspect**

**dim\_suspect = transform\_df2.select("Suspect\_Sex", "suspect\_race\_group") \**

**.dropDuplicates() \**

**.withColumn("suspect\_id", monotonically\_increasing\_id())**

**# Dimension: Location**

**dim\_location = transform\_df2.select("latitude", "longitude", "city", "location\_category") \**

**.dropDuplicates() \**

**.withColumn("location\_id", monotonically\_increasing\_id())**

**# Dimension: Jurisdiction**

**dim\_jurisdiction = transform\_df2.select("jurisdiction") \**

**.dropDuplicates() \**

**.withColumn("jurisdiction\_id", monotonically\_increasing\_id())**

**# Join all dimensions to create fact table**

**df\_fact = transform\_df2 \**

**.join(dim\_crime, on=["crime\_code", "crime\_category", "weapon\_category", "source"], how="left") \**

**.join(dim\_victim, on=["victim\_age", "Victim\_Sex", "victim\_race\_group", "victim\_age\_group"], how="left") \**

**.join(dim\_suspect, on=["Suspect\_Sex", "suspect\_race\_group"], how="left") \**

**.join(dim\_location, on=["latitude", "longitude", "city", "location\_category"], how="left") \**

**.join(dim\_jurisdiction, on=["jurisdiction"], how="left")**

**fact\_crime\_cases = df\_fact.select(**

**"case\_num",**

**"report\_date",**

**"occurred\_date",**

**"occurred\_time",**

**"arrest\_made",**

**"domestic\_incident",**

**"crime\_id",**

**"victim\_id",**

**"suspect\_id",**

**"location\_id",**

**"jurisdiction\_id"**

**)**

**from pyspark.sql.functions import monotonically\_increasing\_id**

**fact\_crime\_cases = fact\_crime\_cases.withColumn("fact\_id", monotonically\_increasing\_id())**

**fact\_crime\_cases.show()**

**dim\_suspect.show()**

**# Common settings**

**write\_mode = "overwrite"**

**file\_format = "csv"**

**# Write LAPD Raw Data**

**lapd\_df1.coalesce(1).write.mode(write\_mode).option("header", "true").csv("s3://raw-master-transformed-factdim-grp-5/raw-data/lapd-raw-1/")**

**lapd\_df2.coalesce(1).write.mode(write\_mode).option("header", "true").csv("s3://raw-master-transformed-factdim-grp-5/raw-data/lapd-raw-2/")**

**# Write NYPD Raw Data**

**nypd\_df.coalesce(1).write.mode(write\_mode).option("header", "true").csv("s3://raw-master-transformed-factdim-grp-5/raw-data/nypd-raw/")**

**# Write CPD Raw Data**

**cpd\_df.coalesce(1).write.mode(write\_mode).option("header", "true").csv("s3://raw-master-transformed-factdim-grp-5/raw-data/cpd-raw/")**

**# Write Transformed Data**

**transform\_df2.coalesce(1).write.mode(write\_mode).option("header", "true").csv("s3://raw-master-transformed-factdim-grp-5/transformed-data/")**

**# Write Fact Table**

**fact\_crime\_cases.coalesce(1).write.mode(write\_mode).option("header", "true").csv("s3://raw-master-transformed-factdim-grp-5/facts-dims/fact/")**

**# Write Dimension Tables**

**dim\_crime.coalesce(1).write.mode(write\_mode).option("header", "true").csv("s3://raw-master-transformed-factdim-grp-5/facts-dims/crime-dim/")**

**dim\_victim.coalesce(1).write.mode(write\_mode).option("header", "true").csv("s3://raw-master-transformed-factdim-grp-5/facts-dims/victim-dim/")**

**dim\_suspect.coalesce(1).write.mode(write\_mode).option("header", "true").csv("s3://raw-master-transformed-factdim-grp-5/facts-dims/suspect-dim/")**

**dim\_location.coalesce(1).write.mode(write\_mode).option("header", "true").csv("s3://raw-master-transformed-factdim-grp-5/facts-dims/location-dim/")**

**dim\_jurisdiction.coalesce(1).write.mode(write\_mode).option("header", "true").csv("s3://raw-master-transformed-factdim-grp-5/facts-dims/jurisdiction-dim/")**

**job.commit()**