

ProyectoBimestral

PROGRAMACION AVANZADA

FINISHED

PROYECTO BIMESTRAL

Integrantes: Lus Fernando Cordova Carrion, Diego Nicolay Jimenez Carrion, Juan Diego Villamagua Alvarado

Fecha: 25/07/2025

Took 0 sec. Last updated by anonymous at July 25 2025, 5:23:42 AM.

Lectura de la tabla detenidos, siendo la principal

FINISHED

Took 0 sec. Last updated by anonymous at July 25 2025, 5:24:09 AM.

```
val url = "jdbc:mysql://localhost:3306/ProyectoBimestral"
val table = "(SELECT * FROM detenidos_copy) AS tmp"
```

```
val properties = new java.util.Properties()
properties.setProperty("user", "root")
properties.setProperty("password", "LUIS")
properties.setProperty("driver", "com.mysql.cj.jdbc.Driver")
```

```
val dfDetenidos = spark.read.jdbc(url, table, properties)
```

```
// Muestra los primeros registros para verificar la conexión
dfDetenidos.show()
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
|codigo_iccs|      tipo|estado_civil|estatus_migratorio|edad|  sexo|  genero|nacionalidad|autoidentifica
cion_etnica|nivel_de_instruccion|      condicion|movilizacion|      tipo_arma|      arma|fecha_detenci
on_aprehension|hora_detencion_aprehension|      lugar|      tipo_lugar|codigo_distrito|codigo_circ
uito|codigo_subcircuito|nombre_zona| nombre_subzona| nombre_distrito| nombre_circuito| nombre_subcirc
uito|codigo_parroquia| presunta_infraccion|id_detenido|
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
| SIN_DATO|APREHENDIDO| SOLTERO/A| NO APLICA| 25|HOMBRE|MASCULINO| ECUATORIANO|
MESTIZO/A| SE DESCONOCE| SIN_DATO| LANCHAS| NINGUNA| NINGUNA|
2/25/2025| 00:00:00|00:00:00|00:00:00|00:00:00|00:00:00|00:00:00|00:00:00|
```

Took 23 sec. Last updated by anonymous at July 25 2025, 4:26:55 AM.

Lectura de DataFrame de Provincias

FINISHED

Took 0 sec. Last updated by anonymous at July 25 2025, 5:24:42 AM.

ProyectoBimestral

```
val url = "jdbc:mysql://localhost:3306/ProyectoBimestral"
val table = "(SELECT * FROM provincias_copy) AS tmp"

val properties = new java.util.Properties()
properties.setProperty("user", "root")
properties.setProperty("password", "LUIS")
properties.setProperty("driver", "com.mysql.cj.jdbc.Driver")

val dfProvincias = spark.read.jdbc(url, table, properties)

// Muestra los primeros registros para verificar la conexión
dfProvincias.show()
```

codigo_provincia	nombre_provincia	Poblacion	Area-km2	Densidad_poblacional-personas_por_km2
0	MAR TERRITORIAL	null	1,060,053	null
1	AZUAY	801,609	8,173	98
2	BOLIVAR	199,078	3,957	50
3	CANAR	227,578	3,647	62
4	CARCHI	172,828	3,783	46
5	COTOPAXI	470,210	6,188	76
6	CHIMBORAZO	471,933	6,116	77
7	EL ORO	714,592	5,870	122
8	ESMERALDAS	553,900	15,836	35
9	GUAYAS	4,391,923	15,900	276
10	IMBABURA	469,879	4,791	98
11	LOJA	485,421	11,064	44
12	LOS RIOS	898,652	7,238	124
13	MANABI	1,592,840	19,517	82

Took 0 sec. Last updated by anonymous at July 25 2025, 4:27:03 AM.

Lectura de DataFrame de Cantones

FINISHED

Took 0 sec. Last updated by anonymous at July 25 2025, 5:25:00 AM.

```
val url = "jdbc:mysql://localhost:3306/ProyectoBimestral"
val table = "(SELECT * FROM cantones_copy) AS tmp"

val properties = new java.util.Properties()
properties.setProperty("user", "root")
properties.setProperty("password", "LUIS")
properties.setProperty("driver", "com.mysql.cj.jdbc.Driver")

val dfCantones = spark.read.jdbc(url, table, properties)

// Muestra los primeros registros para verificar la conexión
dfCantones.show()
```

codigo_canton	codigo_provincia	nombre_canton	Poblacion	Area-km2	Densidad_poblacional-personas_por_km2
0	0	MAR TERRITORIAL	null	1,060,053	

187	101	1	CUENCA	596,101	3,195
5	102	1	GIRON	12,182	343
125	103	1	GUALACEO	43,188	346
23	104	1	NABON	14,776	632
	105	1	PAUTE	26,782	268

Took 0 sec. Last updated by anonymous at July 25 2025, 4:27:07 AM.

Lectura de DataFrame de Parroquias

FINISHED

Took 0 sec. Last updated by anonymous at July 25 2025, 5:25:16 AM.

```
val url = "jdbc:mysql://localhost:3306/ProyectoBimestral"
val table = "(SELECT * FROM parroquias_copy) AS tmp"

val properties = new java.util.Properties()
properties.setProperty("user", "root")
properties.setProperty("password", "LUIS")
properties.setProperty("driver", "com.mysql.cj.jdbc.Driver")

val dfParroquias = spark.read.jdbc(url, table, properties)

// Muestra los primeros registros para verificar la conexión
dfParroquias.show()
```

codigo_parroquia	codigo_canton	nombre_parroquia	Poblacion	Area-km2	Densidad_poblacional-personas_por_km2
0	0	MAR TERRITORIAL	null	1,060,053	
10150	101	CUENCA	361,524	72	
10151	101	BANOS	21,797	251	
10152	101	CUMBE	6,455	75	
10153	101	CHAUCHA	1,721	380	
10154	101	CHECA	3,204	111	

Took 0 sec. Last updated by anonymous at July 25 2025, 4:27:16 AM.

Consultas

FINISHED

- Análisis multidimensional: cruce entre nacionalidad, arma y nivel de instrucción

Consulta construida a partir de 3 columnas clave: nacionalidad, tipo_arma y nivel_de_instruccion. Permite identificar tendencias peligrosas según el perfil educativo y origen de los detenidos, útil para generar perfiles de riesgo y orientar políticas

preventivas.



Took 0 sec. Last updated by anonymous at July 25 2025, 5:25:28 AM.

ProyectoBimestral

```
z.show(dfDetenidos
  .groupBy("nacionalidad", "tipo_arma", "nivel_de_instruccion")
  .count()
  .orderBy($"count".desc))
```

SPARK JOB FINISHED





 settings ▼

nacionalidad ▼	tipo_arma	⋮
ECUATORIANO	NINGUNA	
ECUATORIANO	NINGUNA	
ECUATORIANO	NINGUNA	
ECUATORIANO	NINGUNA	
ECUATORIANO	ARMAS DE FUEGO	
ECUATORIANO	NINGUNA	
ECUATORIANO	ARMAS DE FUEGO	
VENEZOLANO	NINGUNA	
ECUATORIANO	NINGUNA	
ECUATORIANO	ARMAS DE FUEGO	
ECUATORIANO	ARMAS BLANCAS (OBJETO CORTAN	
COLOMBIANO	NINGUNA	
VENEZOLANO	NINGUNA	

Took 1 sec. Last updated by anonymous at July 25 2025, 4:27:23 AM.

• Distribución por estado civil, tipo y estatus migratorio

FINISHED



Se basa en 3 columnas principales: estado_civil, tipo y estatus_migratorio. Busca determinar si existe alguna relación entre la condición migratoria, el tipo de proceso y la situación personal de los detenidos, lo cual puede reflejar entornos de vulnerabilidad social o exclusión.

Took 0 sec. Last updated by anonymous at July 25 2025, 3:32:05 AM.

```
z.show(dfDetenidos
  .groupBy("estado_civil", "tipo", "estatus_migratorio")
  .count()
  .withColumnRenamed("count", "total")
  .orderBy('total.desc))
```

SPARK JOB FINISHED





 settings ▼

	⋮
--	---

estado_civil



tipo

ProyectoBimestral

Took 0 sec. Last updated by anonymous at July 25 2025, 4:27:28 AM.

- Análisis de la cantidad de detenidos/aprehendidos por provincia y cantón, FINISHED desglosado por tipo de arma, además mostrando el porcentaje que representa cada tipo de arma dentro de cada cantón.

Esto te ayuda a saber qué tipos de armas son más comunes en las detenciones o aprehensiones por cada cantón y provincia, y a ver la distribución relativa.

Took 0 sec. Last updated by anonymous at July 25 2025, 5:06:27 AM.

```
// Unir detenidos con parroquias
val df_det_parr = dfDetenidos.join(dfParroquias, Seq("codigo_parroquia"), "left")

// Unir con cantones
val df_det_cant = df_det_parr.join(dfCantones, Seq("codigo_canton"), "left")

// Unir con provincias
val df_det_cant_prov = df_det_cant.join(dfProvincias, Seq("codigo_provincia"), "left")

// Conteo total por provincia, cantón, tipo y tipo_arma
val conteo = df_det_cant_prov
  .groupBy("nombre_provincia", "nombre_canton", "tipo", "tipo_arma")
  .agg(count("id_detenido").alias("total_detenidos"))

// Conteo total por provincia, cantón y tipo (sin tipo_arma) para calcular porcentaje
val total_por_tipo = df_det_cant_prov
  .groupBy("nombre_provincia", "nombre_canton", "tipo")
  .agg(count("id_detenido").alias("total_tipo"))

// Unir para cálculo de porcentaje
val resultado = conteo.join(total_por_tipo, Seq("nombre_provincia", "nombre_canton", "tipo"), "left")
  .withColumn("porcentaje_tipo_arma", round(col("total_detenidos") / col("total_tipo") * 100, 2))
  .orderBy(desc("total_detenidos"))

// Mostrar resultado en Zeppelin
z.show(resultado)
```

SPARK JOB FINISHED

settings ▼

nombre_provincia	nombre_canton	tipo
PICHINCHA	QUITO	APREHENDIDO

ProyectoBimestral

GUAYAS	GUAYAQUIL	APREHENDIDO
PICHINCHA	QUITO	DETENIDO
GUAYAS	GUAYAQUIL	DETENIDO
GUAYAS	DURAN	APREHENDIDO
IMBABURA	IBARRA	APREHENDIDO
GUAYAS	GUAYAQUIL	APREHENDIDO
EL ORO	MACHALA	APREHENDIDO

```
df_det_parr: org.apache.spark.sql.DataFrame = [codigo_parroquia: int, codigo_iccs: string ... 32 more fields]
df_det_cant: org.apache.spark.sql.DataFrame = [codigo_canton: int, codigo_parroquia: int ... 37 more fields]
df_det_cant_prov: org.apache.spark.sql.DataFrame = [codigo_provincia: int, codigo_canton: int ... 41 more fields]
conteo: org.apache.spark.sql.DataFrame = [nombre_provincia: string, nombre_canton: string ... 3 more fields]
total_por_tipo: org.apache.spark.sql.DataFrame = [nombre_provincia: string, nombre_canton: string ... 2 more fields]
resultado: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [nombre_provin...
```

Took 1 sec. Last updated by anonymous at July 25 2025, 5:06:16 AM.

- ¿Cuál es la edad promedio de las personas detenidas o aprehendidas en cada parroquia, desglosada por tipo de lugar donde se produjo la detención?

Esta pregunta busca entender cómo varía la edad promedio según la ubicación geográfica específica (parroquia) y el contexto espacial (tipo de lugar), para identificar patrones demográficos y territoriales en las detenciones.

Took 0 sec. Last updated by anonymous at July 25 2025, 5:15:12 AM.

```
// Unir detenidos con parroquias para obtener nombre de parroquia
val df_det_parr = dfDetenidos.join(dfParroquias, Seq("codigo_parroquia"), "left")

// Calcular edad promedio por parroquia y tipo_lugar
val resultado = df_det_parr
  .groupBy("codigo_parroquia", "nombre_parroquia", "tipo_lugar", "tipo")
  .agg(round(avg("edad"), 2).alias("edad_promedio"))
  .orderBy(desc("tipo_lugar"))

// Mostrar resultado en Zeppelin
z.show(resultado)
```

SPARK JOB FINISHED



settings ▼

codigo_parroquia	nombre_parroquia	
170184	TUMBACO	Z
80550	SAN LORENZO	Z
170184	TUMBACO	Z

ProyectoBimestral

90150	GUAYAQUIL	Z
100150	SAN MIGUEL DE IBARRA	Z
280150	ALLURIQUIN	V
11550	CAMILO PONCE ENRIQUEZ	V
71050	PINAS	V

Output is truncated to 1000 rows. Learn more about **zeppelin.spark.maxResult**



df_det_parr: org.apache.spark.sql.DataFrame = [codigo_parroquia: int, codigo_iccs: string ... 32 more fields]

resultado: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [codigo_parroquia: int, nombre_parroquia: string ... 3 more fields]

Took 1 sec. Last updated by anonymous at July 25 2025, 5:14:32 AM. (outdated)

- ¿Cómo se distribuyen las personas detenidas y aprehendidas según su autoidentificación étnica, sexo y estado civil en cada provincia?

FINISHED

Esta pregunta permite entender el perfil demográfico y territorial de las personas involucradas, facilitando análisis sociológicos y la toma de decisiones focalizadas en políticas públicas o seguridad.

Took 0 sec. Last updated by anonymous at July 25 2025, 5:21:04 AM.

```
import org.apache.spark.sql.functions._
```

SPARK JOB FINISHED

```
// Unir detenidos con parroquias
```

```
val df_det_parr = dfDetenidos.join(dfParroquias, Seq("codigo_parroquia"), "left")
```

```
// Unir con cantones
```

```
val df_det_cant = df_det_parr.join(dfCantones, Seq("codigo_canton"), "left")
```

```
// Unir con provincias
```

```
val df_det_cant_prov = df_det_cant.join(dfProvincias, Seq("codigo_provincia"), "left")
```

```
// Agrupar con la información completa
```

```
val resultado = df_det_cant_prov.groupBy("nombre_provincia", "autoidentificacion_etnica", "sexo", "estado")
```

```
.agg(count("id_detenido").alias("total_personas"))
```

```
.orderBy(
```

```
  col("nombre_provincia").asc,
```

```
  col("autoidentificacion_etnica").asc,
```

```
  col("sexo").asc,
```

```
  col("estado_civil").asc,
```

```
  col("tipo").asc
```

```
)
```

```
z.show(resultado)
```



settings ▼

nombre_provincia	autoidentificacion_etnica	sexo
AZUAY	AFROECUATORIANO/A AFRODESCENDIENTE	HOMBRE

ProyectoBimestral

AZUAY	AFROECUATORIANO/A AFRODESCENDIENTE	HOMBRE
AZUAY	AFROECUATORIANO/A AFRODESCENDIENTE	HOMBRE
AZUAY	AFROECUATORIANO/A AFRODESCENDIENTE	HOMBRE
AZUAY	AFROECUATORIANO/A AFRODESCENDIENTE	HOMBRE

Output is truncated to 1000 rows. [Learn more about zeppelin.spark.maxResult](#) ✕

```
import org.apache.spark.sql.functions._
df_det_parr: org.apache.spark.sql.DataFrame = [codigo_parroquia: int, codigo_iccs: string ... 32 more fields]
df_det_cant: org.apache.spark.sql.DataFrame = [codigo_canton: int, codigo_parroquia: int ... 37 more fields]
df_det_cant_prov: org.apache.spark.sql.DataFrame = [codigo_provincia: int, codigo_canton: int ... 41 more fields]
resultado: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [nombre_provincia: string, autoidentificacion_etnica: string ... 4 more fields]
```

Took 1 sec. Last updated by anonymous at July 25 2025, 5:20:14 AM. (outdated)



READY