UTPL

UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

PROYECTO INTEGRADOR DE SABERES 1.2.

COMPONENTE DE BASE DE DATOS AVANZADA

INTEGRANTES:

- FLORES SERRANO DANIEL EDUARDO
- JIMENEZ ROBLES CRISTIAN MIGUEL
- PANAMITO RUEDA JEAN PAUL
- RODRIGUEZ SALCEDO CRISTIAN DAVID

Índice

1. Introducción 4
2. Repositorio GitHub: 5
3. Planificación del trabajo 5
   3.1 Posibles análisis a realizar 5
4. Fuentes de datos 6
   4.1 Datos Base 6
   4.2 Datos complementarios 10
5. Componente de base de datos 12
   5.1 Explicación de preprocesamiento de datos realizados 12
   5.2 Diseño conceptual 12
      5.2.1 Diccionario de datos 12
   5.3 Modelo Lógico Relacional 18
      5.3.1 Diccionario Modelo lógico 19
   5.4 Implementación y carga de base de datos 20
   5.5 Script SQL 21
6. Componente Programación 21
   6.1 Herramientas utilizadas: 22
      1. Apache Zeppelin: 22
      2. MySQL Workbench: 22
      3. AngularJS: 22
      4. CSS (Cascading Style Sheets): 22
      5. Python: 23
      6. HTML (HyperText Markup Language): 23
      7. JDK 1.8 (Java Development Kit 1.8): 23
      8. Scala 2.12: 23
      9. Ubuntu 20.04: 23
      10. Apache Spark 3.2.1 Hadoop 3.2: 23
   6.2 Explicación de comandos y sentencias usadas para consultar y visualizar datos tanto a nivel del archivo fuente. tanto a nivel de archivo fuente, como a nivel de base de datos MySQL 24
      6.1.1 Exploración de la data desde el archivo .csv 24
7. Resultados Obtenidos 52
8. Referencias 55

- Script SQL: Enlace: DDL_PR_ENEMDU_VIVIENDAS.sql en el repositorio GitHub del proyecto. 56
- Repositorio GitHub: Enlace: cdrodriguez7/Proyecto-Integrado-1.2 56
- Implementación y carga: El script de implementación y carga se divide en dos partes: 56

- Parte 1: complementarios_1.py 56
- Parte 2: principales_2.py 56

1. Introducción

Este informe final del Proyecto Integrador de Saberes Prácticum 1.2 se centra en la obtención, normalización e implementación de una base de datos MySQL con datos sobre las características de vivienda en Ecuador. Se recopilaron datos primarios de la encuesta ENEMDU realizada por el INEC en el primer trimestre de 2023.

Además, se buscaron y obtuvieron datos complementarios para enriquecer el conjunto de datos.
El proyecto se dividió en etapas que incluyeron el estudio y comprensión de los datos, el diseño conceptual, lógico y físico de la base de datos, siguiendo las mejores prácticas en el diseño de bases de datos relacionales. Se implementó y cargó la base de datos en MySQL, garantizando la integridad y consistencia de los datos.

Esta base de datos resultante es fundamental para el análisis de las características de vivienda en Ecuador y facilita la toma de decisiones en políticas públicas y planificación urbana. El informe detalla los procesos realizados, presenta los resultados y conclusiones, y destaca la importancia de contar con una base de datos confiable y estructurada.

El informe se organiza en secciones que abarcan la obtención de datos primarios, la obtención de datos complementarios, el estudio y comprensión de los datos, el diseño conceptual, lógico y físico de la base de datos, y los resultados de la implementación y carga. Se espera que este informe contribuya al conocimiento de las características de vivienda en Ecuador y sea una herramienta útil para futuras investigaciones y estudios en este ámbito.

2. Repositorio GitHub:

cdrodriguez7/Proyecto-Integrado-1.2 (github.com)

3. Planificación del trabajo

3.1 Posibles análisis a realizar

A continuación, presentamos una lluvia de ideas sobre los posibles análisis que podrían llevarse a cabo utilizando los datos base y complementarios recopilados en el proyecto de características de vivienda en Ecuador. Estos análisis nos permitirán obtener una visión más profunda y completa de la situación del acceso a la vivienda en el país, considerando diferentes variables y dimensiones. A continuación, detallamos algunos de los análisis que nos gustaría realizar:

1. Análisis de la distribución geográfica de las características de vivienda: Mediante el uso de datos geográficos y la incorporación de variables relacionadas con subdivisiones geográficas, como provincias o cantones, podríamos realizar un análisis para identificar patrones y diferencias regionales en las características de vivienda. Esto nos permitiría comprender mejor las disparidades en el acceso a vivienda en distintas áreas del país y orientar acciones de política pública en función de las necesidades específicas de cada región.

2. Análisis de correlación entre características de vivienda y variables demográficas: Al combinar los datos de características de vivienda con información demográfica, como la edad, el género o el nivel educativo de los hogares, podríamos explorar posibles relaciones y correlaciones entre estos factores. Por ejemplo, podríamos investigar si existe una asociación entre la calidad de la vivienda y la edad de los ocupantes o si hay diferencias en las condiciones de vivienda entre hogares encabezados por hombres y mujeres.

3. Análisis de costo de vida y acceso a vivienda: Mediante la incorporación de datos sobre el costo de vida, como el salario promedio en diferentes regiones o el índice de precios al consumidor, podríamos realizar un análisis para evaluar la relación entre el acceso a vivienda y el nivel de vida en distintas áreas geográficas. Esto nos ayudaría a comprender si existen desafíos particulares relacionados con el costo de vida que afectan el acceso a vivienda en determinadas regiones.

4. Análisis de características de vivienda y sostenibilidad ambiental: Considerando los datos complementarios relacionados con el medio ambiente, como la calidad del aire o la disponibilidad de espacios verdes en diferentes áreas, podríamos explorar la relación entre las características de vivienda y la sostenibilidad ambiental. Esto nos permitiría identificar oportunidades para promover viviendas más sostenibles desde el punto de vista ambiental y mejorar la calidad de vida de los residentes.

4. Fuentes de datos

4.1 Datos Base

Base de datos cruda en la cual nos basamos principalmente solo para el análisis de los datos, tipos de variables y valores que tomarían las mismas

Nro.	Colum_Original	Columna	Tipo de Dato	Etiquetas	Propósito	Observación
1	area	area	Entero	Urbana=1 Rural=2	Indicar el área	
2	ciudad	cod_parroquia	VARCHAR2		Indicar la ciudad	
3	conglomerado	conglomerado	Entero		Indicar el conglomerado	
4	panelm	panelm	Entero	Panel D21 = "025", Panel A21 = "037", Panel B21 = "028", Panel C21 = "029"	Indicar el panel	Los valores no coinciden con las etiquetas, no se lo utilizará
5	nro_vivienda	nro_vivienda	Entero	Vivienda Diez (reemplazo) = "10", Vivienda Uno = "1", Vivienda Dos = "2", Vivienda Tres = "3", Vivienda Cuatro = "4", Vivienda Cinco = "5", Vivienda Seis = "6", Vivienda Siete = "7", Vivienda Ocho (reemplazo) = "8", Vivienda Nueve (reemplazo) = "9"	Indicar la vivienda	
6	nro_hogar	nro_hogar	Entero	Hogar Uno = "1", Hogar Dos = "2", Hogar Tres = "3", Hogar Cuatro = "4", Hogar Cinco = "5"	Indicar el hogar	
7	vi01	acceso_principal	Entero	Carretera, calle pavimentada = 1, Empedrado = 2, Lastrado, calle de tierra = 3, Sendero = 4, Río, mar = 5, Otro = 6	Indicar el acceso principal	
8	vi02	tipo_vivienda	Entero	Casa o villa = 1,	Indicar el tipo de	

				Departamento = 2, Cuartos en casa de inquilinato = 3, Mediagua = 4, Rancho, covacha = 5, Choza = 6, Otra = 7	vivienda	
9	vi03a	material_techo	Entero	Hormigón (losa, cemento) = 1, Fibrocemento, asbesto (eternit, eurolit) = 2, Zinc, Aluminio = 3, Teja = 4, Palma, paja u hoja = 5, Otro Material = 6	Indicar el material del techo	
10	vi03b	estado_techo	Entero	Bueno = 1, Regular = 2, Malo = 3	Indicar el estado del techo	
11	vi04a	piso_material	Entero	Bueno = 1, Regular = 2, Malo = 3	Indicar el material del piso	
12	vi04b	estado_piso	Entero	Bueno = 1, Regular = 2, Malo = 3	Indicar el estado del piso	
13	vi05a	material _pared	Entero	Hormigón/Ladrillo o Bloque = 1, Asbesto/Cemento (Fibrolit) = 2, Adobe o Tapia = 3, Madera = 4, Caña revestida o bahareque = 5, Caña no revestida o estera = 6, Otra Material = 7	Indicar el material de las paredes	
14	vi05b	estado_pared	Entero	Bueno = 1, Regular = 2, Malo = 3	Indicar el estado de las paredes	
15	vi06	nro_cuartos	Entero	No informa = 99	Indicar el número de cuartos	
16	vi07	nro_dormitorios	Entero	No informa = 99	Indicar el número de dormitorios	
17	vi07a	nro_cuartos_negocio	Entero	No informa = 99	Indicar el número de cuartos para negocio	
18	vi07b	cocina_cuarto	Entero	Si = 1, NO= 2	Indicar si la cocina es un cuarto	
19	vi08	combustible_cocina	Entero	Gas = 1, Leña, carbón = 2, Electricidad = 3, Otro = 4	Indicar el material de la cocina	
20	vi09	servicio_higienico	Entero	Excusado y alcantarillado = 1, Excusado y pozo	Indicar el tipo de servicio higiénico	

				séptico = 2, Excusado y pozo ciego = 3, Letrina = 4, No tiene = 5		
21	vi09a	sshh_alternativa	Entero	Descarga directa al mar, río, lago o quebrada = 1. Van al monte, campo, bota la basura en paquete =2; Usan una instalación sanitaria cercana y/o prestada =3	Indicar la alternativa no higiénica	
22	vi09b	instalacion_sanitaria	Entero	Excusado y alcantarillado = 1, Excusado y pozo séptico = 2, Excusado y pozo ciego = 3, Letrina = 4	Indicar el tipo de instalación sanitaria	
23	vi10	agua_obtencion	Entero	Red pública = 1, Pila o llave pública = 2, Otra fuente por tubería = 3, Carro repartidor, triciclo = 4, Pozo = 5, Río, vertiente, acequia = 6, Otro = 7	Indicar la forma de obtención de agua	
24	vi101	agua_medidor	Entero	Si = 1, NO = 2	Indicar si hay medidor de agua en la vivienda	
25	vi102	agua_junta	Entero	Si = 1, NO = 2	Indicar si hay junta de agua en la vivienda	
26	vi10a	agua_tuberia	Entero	Por tubería dentro de la vivienda = 1, Por tubería fuera de la vivienda, pero en el lote = 2, Por tubería fuera de la vivienda, lote o terreno = 3, No recibe agua por tubería sino por otros medios = 4	Indicar el tipo de tubería utilizado	
27	vi11	ducha	Entero	Exclusivo del hogar = 1, Compartido con otros hogares = 2, No tiene = 3	Indicar si hay ducha en la vivienda	
28	vi12	tipo_alumbrado	VARCHAR2	Empresa eléctrica	Indicar el tipo de	

				pública, Planta eléctrica privada, Vela, candil, mechero, gas, Ninguno	alumbrado de la vivienda	
29	vi13	eliminacion_basura	Entero	Contratan el servicio = 1, Servicio municipal = 2, Botan a la calle, quebrada, río = 3, La queman, entierran = 4, Otra = 5	Indicar el método de eliminación de basura	
30	vi14	tipo_tenencia	Entero	En arriendo = 1, Anticresis y/o arriendo = 2, Propia y la está pagando = 3, Propia y totalmente pagada = 4, Cedida = 5, Recibida por servicios = 6, Otra = 7	Indicar la tenencia de la vivienda	
31	vi141	valor_arriendo	Entero	No informa = 99	Indicar el valor de arriendo de la vivienda	
32	vi143	agua_inlcuida	Entero	Si = 1, NO = 2	Indicar si el agua está incluida en el arriendo	
33	vi143	luz_incluida	Entero	Si = 1, NO = 2	Indicar si la luz está incluida en el arriendo	
34	vi144	parentesco_propietario	Entero	Si = 1, NO = 2	Indicar el parentesco del propietario de la vivienda	
35	vi1511	posesion_vehiculos	Entero	Si = 1, NO = 2	Indicar si se poseen vehículos	*
36	vi1521	cant_vehiculos	Entero		Indicar la cantidad de vehículos	
37	vi1512	posesion_motos	Entero	Si = 1, NO = 2	Indicar si se poseen motos	
38	vi1522	cant_motos	Entero		Indicar la cantidad de motos	
39	vi1531	abastecimiento_super	Entero	Si = 1, NO = 2	Indicar el abastecimiento en gasolina super	
40	vi1541	gasto_super	Entero	No informa = 999	Indicar el gasto en gasolina super	
41	vi1532	abastecimiento_extra	Entero	Si = 1, NO = 2	Indicar el abastecimiento en	

					gasolina extra	
42	vi1542	gasto_extra	Entero	No informa = 999	Indicar el gasto en tienda extra	
43	vi1533	abastecimiento_diesel	Entero	Si = 1, NO = 2	Indicar el abastecimiento de diesel	
44	vi1543	gasto_diesel	Entero	No informa = 999	Indicar el gasto en diesel	
45	vi1534	abastecimiento_eco	Entero	Si = 1, NO = 2	Indicar el abastecimiento en tienda ecológica	
46	vi1544	gasto_eco	Entero	No informa = 999	Indicar el gasto en tienda ecológica	
47	vi1535	abastecimiento_elect	Entero	Si = 1, NO = 2	Indicar el abastecimiento eléctrico	
48	vi1545	gasto_elect	Entero	No informa = 999	Indicar el gasto en electricidad	
49	vi1536	abastecimiento_gas	Entero	Si = 1, NO = 2	Indicar el abastecimiento de gas	
50	vi1546	gasto_gas	Entero	No informa = 999	Indicar el gasto en gas	
51	estrato	estrato	VARCHAR2		Indicar el estrato de la vivienda	
52	fexp	fexp	VARCHAR2		Indicar el factor de expansión	
53	upm	upm	VARCHAR2		Indicar la Unidad Primaria de Muestreo	
54	id_vivienda	id_vivienda	VARCHAR2		Identificador de la vivienda	
55	id_hogar	id_hogar	VARCHAR2		Identificador del hogar	
56	periodo	periodo	VARCHAR2	ene-23 =202301; feb-23= 202302; mar-23= 202303	Indicar el periodo de la encuesta	
57	mes	mes	VARCHAR2		Indicar el mes de la encuesta	

4.2 Datos complementarios

Nro.	Columna	Tipo de Dato	Etiquetas	Propósito	Observación
58	nombre_parroquia	VARCHAR2		Indicar el nombre parroquia	
59	cod_canton	VARCHAR2		Identificador cantón	Se asigna desde el cod_parroquia
60	nombre_canton	VARCHAR2		Indicar el nombre cantón por provincia	
61	cod_provincia	VARCHAR2		Identificador provincia	Se asigna desde el

				por provincia	cod_canton
62	nombre_provincia	VARCHAR2		Indicar el nombre provincia por provincia	
63	tasa_desempleo	VARCHAR2		Porcentaje de desempleo por provincia	
64	empleo_no_remunerado	VARCHAR2		Porcentaje empleo no remunerado por provincia	
65	empleo_bruto	VARCHAR2		Porcentaje empleo total por provincia	
66	asistencia_edu_basica	VARCHAR2		Porcentaje personas que tuvieron educación básica por provincia	
67	tasa_analfabetismo	VARCHAR2		Porcentaje de desempleo por provincia	

Tuvimos la iniciativa de tomar tasas de información con valores en porcentajes por provincia por el hecho de que eran parámetros de valor que complementaban la información con la que principalmente vamos a trabajar. Conjunto a estos datos vamos a incluir más información sobre la localización en la que se hizo la encuesta tanto para encontrar el cantón y la provincia basándonos netamente en el dato recibido en el csv "ciudad" al que vinimos a trabajar como "cod_parroquia" que, al ser un código postal, poseía la información para encontrar los datos previamente mencionados.
Para lograr realizar tanto la carga de datos complementarios como para identificar los códigos dados realizamos una búsqueda dentro de páginas oficiales para tener información veraz y de valor basándonos principalmente en información publicada dentro de la página del INEC. Al no estar todos los archivos en formato csv, ni que estén estructurados de una manera legible para una lectura, optamos por separar los archivos sin alterar la información recuperada para facilitar la lectura y la carga de la misma.

Fuente de recuperación de datos complementarios: https://www.ecuadorencifras.gob.ec/enemdu-anual-2022/

Fuente de recuperación de catálogo de códigos (código postal): https://n9.cl/xnbk4

5. Componente de base de datos

5.1 Explicación de preprocesamiento de datos realizados

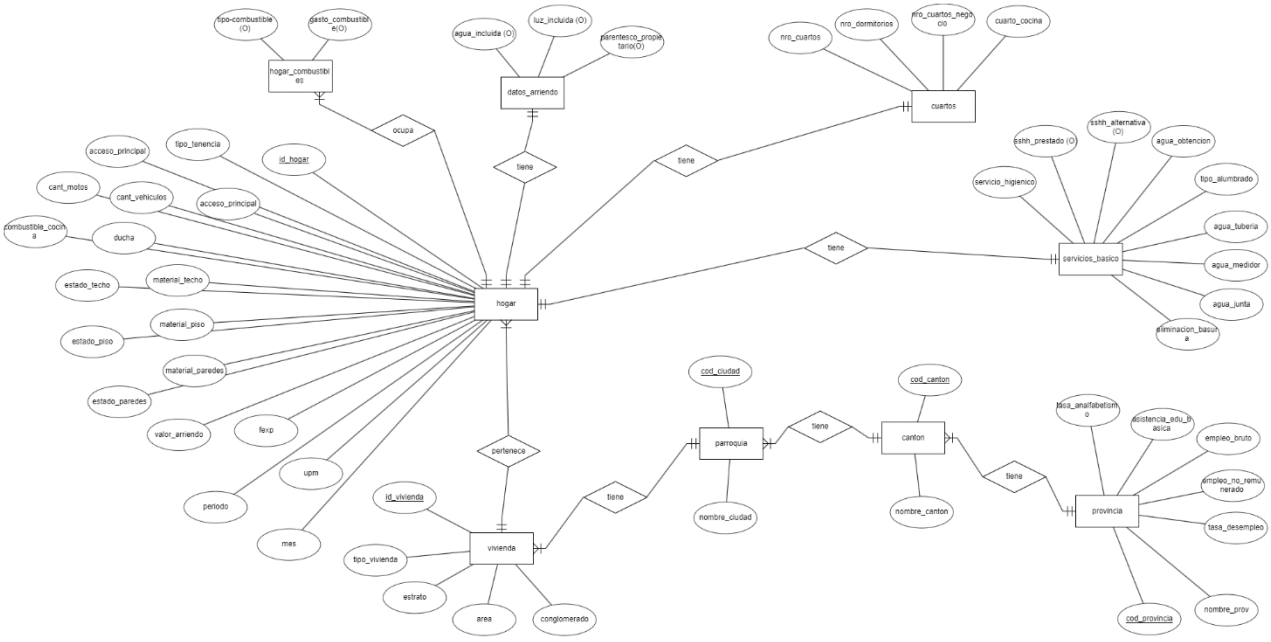
Decidimos utilizar el lenguaje de programación Python para esta etapa del desarrollo del proyecto. La elección de esta herramienta se basó en la comodidad y eficiencia que los dataframes de Pandas ofrecen para el manejo de datos.

Inicialmente, los archivos CSV fueron leídos y procesados a través de scripts de Python. En la lectura cambiamos los nombres de las columnas que se obtenían de la data en bruto, esto con el objetivo de obtener un esquema que en pasos siguientes permita desarrollar los modelos de bases de datos. Posteriormente, estos datos se almacenaron en distintos dataframes, agrupándolos de acuerdo con las relaciones existentes entre ellos. Por ejemplo, los datos relacionados con los combustibles y los vehículos se consolidaron en un único dataframe.

Una vez organizados los datos, procedimos a la etapa de limpieza y transformación. Utilizamos estructuras condicionales if-else para reemplazar los valores numéricos presentes en la data en bruto. Este reemplazo se realizó con base en las correspondencias establecidas en nuestro diccionario de datos, permitiendo así una interpretación más clara y precisa de la información.

En resumen, este proceso de preprocesamiento de datos nos permitió optimizar la organización, limpieza y transformación de la información, facilitando su posterior análisis y utilización en el proyecto.

5.2 Diseño conceptual



5.2.1 Diccionario de datos

Nro	Entidad	Atributo	Tipo de dato	Obligatori	Descripción	Valore

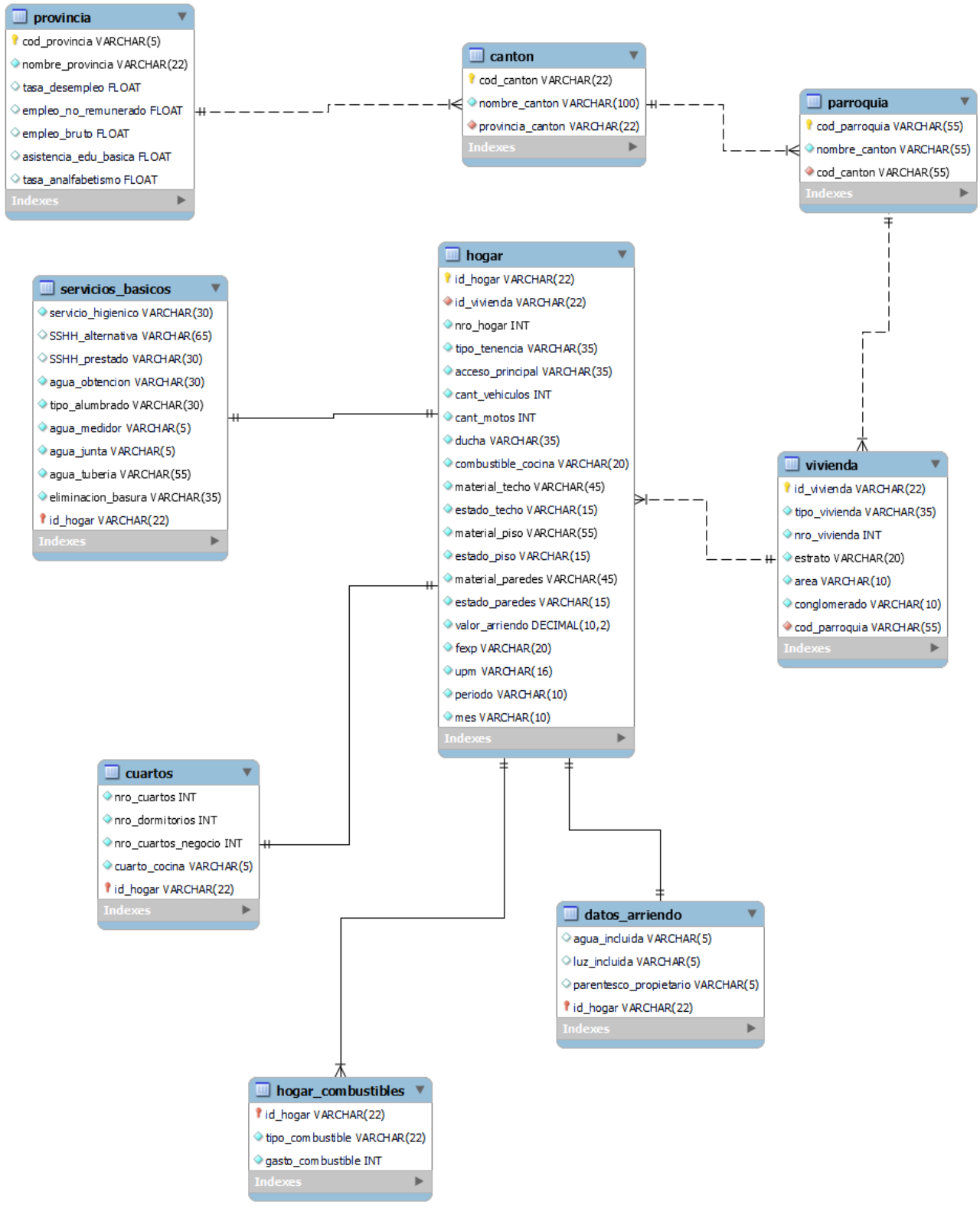
				o		s por Defecto
1	vivienda	id_vivienda	VARCHAR	x	Identificador único de la vivienda FK	-
2	vivienda	nro_vivienda	INT	x	Estrato de la vivienda	-
3	vivienda	estrato	VARCHAR	x	Factor de expansión	-
4	vivienda	id_hogar	VARCHAR	x	Tipo de vivienda	-
5	vivienda	area	VARCHAR	x	area de la vivienda	-
6	vivienda	conglomerado	VARCHAR	x	Información sobre el conglomerado al que pertenece	-
7	vivienda	cod_parroquia	VARCHAR	x	Código de la parroquia a la que pertenece la vivienda	-
8	vivienda	parroquia_nombre_parroquia	VARCHAR	x	Nombre de la parroquia a la que pertenece la vivienda	-
9	parroquia	nombre_parroquia	VARCHAR	x	Nombre de la parroquia	-
10	parroquia	cod_canton	VARCHAR	x	Código del cantón al que pertenece la parroquia	-
11	parroquia	cod_parroquia	VARCHAR	x	Código de la parroquia	-
12	parroquia	canton_cod_canton	VARCHAR	x	Código del cantón al que pertenece la parroquia	-
13	canton	cod_canton	VARCHAR	x	identificador unico de canton	-
14	canton	nombre_canton	VARCHAR	x	nombre del canton	-
15	canton	provincia_canton	VARCHAR	x	identificador unico de provincia	-
16	canton	provincia_cod_provincia	VARCHAR	x	Fk	-
17	provincia	cod_provincia	VARCHAR	x	identificador	-

					unico de provincia	
18	provincia	nombre_provincia	VARCHAR	x	nombre de provincia	-
19	provincia	tasa_desempleo	VARCHAR DOUBLE	x	Porcentaje de desempleo	-
20	provincia	empleo_no_remunerado	VARCHAR DOUBLE	x	porcentaje empleo no remunerado	-
21	provincia	empleo_bruto	VARCHAR DOUBLE	x	porcentaje empleo bruo	-
22	provincia	asistencia_edu_basica	VARCHAR DOUBLE	x	asistencia educacion basica	-
23	provincia	tasa_analfabetismo	VARCHAR DOUBLE	x	tasa analfabetismo	-
24	hogar	id_hogar	INT	x	Identificador único del hogar	-
25	hogar	nro_hogar	INT	x	Número de hogar	-
26	hogar	tipo_tenencia	INT	x	tipo de pertenencia de la propiedad	-
27	hogar	acceso_principal	VARCHAR	x	Acceso principal a la vivienda	-
28	hogar	cant_vehiculos	INT		Cantidad de motos	-
29	hogar	cant_motos	INT		Gasto en combustible	-
30	hogar	material_techo	VARCHAR	x	Material del techo	-
31	hogar	material_piso	VARCHAR	x	Material del piso	-
32	hogar	estado_techo	VARCHAR	x	Estado del techo	-
33	hogar	estado_piso	VARCHAR	x	Estado del piso	-
34	hogar	pared_material	VARCHAR		Indicar el material de las paredes	-
35	hogar	estado_paredes	VARCHAR	x	Estado de las paredes	-
36	hogar	periodo	VARCHAR	x	Período de la encuesta	-
37	hogar	mes	VARCHAR	x	Mes de la encuesta	-
38	hogar	upm	VARCHAR	x	Unidad primaria de muestreo	-

39	hogar	fexp	VARCHAR	x	Identificador único de la encuesta	-
40	hogar	ducha	VARCHAR	x	Tipo de ducha	-
41	hogar	combustible_cocina	VARCHAR		Combustible utilizado en la cocina	-
42	vivienda	id_vivienda	INT	x	Identificador de la vivienda asociada	-
43	ubicacion	id_ubicacion	INT	x	Identificador único de la ubicación	-
44	ubicacion	area	VARCHAR		Área de ubicación	-
45	ubicacion	ciudad	VARCHAR		Ciudad de ubicación	-
46	ubicacion	conglomerado	VARCHAR		Conglomerado de ubicación	-
47	ubicacion	panelm	VARCHAR		Panel M de ubicación	-
48	ubicacion	id_vivienda	INT	x	Identificador de la vivienda asociada	-
49	servicios_basico	id_servicios	INT	x	Identificador único de los servicios básicos.	-
50	servicios_basico	servicio_higienico	VARCHAR	x	Tipo de servicio higiénico.	-
51	servicios_basico	sshh_prestado	VARCHAR		Servicio higiénico prestado.	‘’
52	servicios_basico	sshh_alternativa	VARCHAR		Alternativa de servicio higiénico.	Vacio
53	servicios_basico	agua_obtencion	VARCHAR	x	Fuente de obtención de agua.	-
54	servicios_basico	tipo_alumbrado	VARCHAR	x	Tipo de alumbrado utilizado.	-
55	servicios_basico	agua_medidor	VARCHAR	x	Medidor de agua instalado.	-
56	servicios_basico	agua_junta	VARCHAR	x	Agua suministrada por junta de vecinos.	-

57	servicios_basico	agua_tuberia	VARCHAR	x	Agua suministrada a través de tubería.	-
58	servicios_basico	eliminacion_basura	VARCHAR	x	Tipo de eliminación de basura.	-
59	servicios_basico	id_hogar	INT	x	Identificador del hogar asociado	-
60	hogar_combustibles	gasto_combustible	INT	x	Gasto en combustibles en Enero	0
61	hogar_combustibles	tipo_combustible	VARCHAR	x	Tipo de combustible	-
62	hogar_combustibles id_hogar		VARCHAR	-	Identificador único de hogar	-
63	hogar_combustibles	hogar_id_hogar	VARCHAR		ID de hogar	-
64	cuartos	id_cuarto	INT	x	Identificador único del cuarto	-
65	cuartos	nro_cuartos	INT		Número de cuartos	0
66	cuartos	nro_dormitorios	INT		Número de dormitorios	0
67	cuartos	nro_cuartos_negocio	INT		Número de cuartos utilizados para negocios	0
68	cuartos	cuarto_cocina	VARCHAR	x	Cuarto utilizado como cocina	
69	cuartos	id_hogar	INT	x	Identificador del hogar asociado	-
70	datos_arriendo	valor_arriendo	INT		Valor del arriendo.	0
71	datos_arriendo	agua_incluida			Indicador de si el agua está incluida.	Vacio
72	datos_arriendo	luz_incluida	INT		Indicador de si la luz está incluida.	Vacio
73	datos_arriendo	parentesco_propietario	INT		Parentesco con el propietario.	Vacio
74	datos_arriendo	id_hogar	VARCHAR	x	Identificador del hogar.	-
						-

5.3 Modelo Lógico Relacional



5.3.1 Diccionario Modelo lógico

Entidad	Atributo	Restricción de Dominio
vivienda	tipo_vivienda	Casa o villa = 1, Departamento = 2, Cuartos en casa de inquilinato = 3, Mediagua = 4, Rancho, covacha = 5, Choza = 6, Otra = 7
vivienda	area	Urbana=1 Rural=2
hogar	acceso_principal	Carretera, calle pavimentada = 1, Empedrado = 2, Lastrado, calle de tierra = 3, Sendero = 4, Río, mar = 5, Otro = 6
hogar	cant_vehiculos	No informa = 99
hogar	cant_motos	No informa = 99
hogar	ducha	Exclusivo del hogar = 1, Compartido con otros hogares = 2, No tiene = 3
hogar	combustible_cocina	No informa = 99
hogar	material_techo	Hormigón (losa, cemento) = 1, Fibrocemento, asbesto (eternit, eurolit) = 2, Zinc, Aluminio = 3, Teja = 4, Palma, paja u hoja = 5, Otro Material = 6
hogar	estado_techo	Bueno = 1, Regular = 2, Malo = 3
hogar	material_piso	Duela, parquet, tablón tratado o piso flotante= 1; Cerámica, baldosa, vinil o porcelanato =2 Mármol o marmetón = 3 ;Ladrillo o cemento =4
hogar	estado_piso	;Tabla / tablón no tratado = 5 ; Caña = 6 ; Tierra = 7; Otro Material = 8 "
hogar	material_paredes	Hormigón/Ladrillo o Bloque = 1, Asbesto/Cemento (Fibrolit) = 2, Adobe o Tapia = 3, Madera = 4, Caña revestida o bahareque = 5, Caña no revestida o estera = 6, Otra Material = 7
hogar	estado_paredes	Bueno = 1, Regular = 2, Malo = 3
hogar	valor_arriendo	DECIMAL(10,2) NOT NULL DEFAULT 0
hogar	fexp	VARCHAR(20) NOT NULL
hogar	upm	VARCHAR(16) NOT NULL
hogar	periodo	ene-23 =202301; feb-23= 202302; mar-23= 202303
hogar	mes	
cuartos	nro_cuartos	No informa = 99
cuartos	nro_dormitorios	No informa = 99
cuartos	nro_cuartos_negocio	No informa = 99
cuartos	cuarto_cocina	Si = 1, NO = 2
cuartos	id_hogar	
datos_arriendo	agua_incluida	Si = 1, NO = 2
datos_arriendo	luz_incluida	Si = 1, NO = 2
datos_arriendo	parentesco_propietario	Si = 1, NO = 2
servicios_basicos	servicio_higienico	VARCHAR(30) NOT NULL

servicios_basicos	sshh_alternativa	Excusado y alcantarillado = 1, Excusado y pozo séptico = 2, Excusado y pozo ciego = 3, Letrina = 4, No tiene = 5
servicios_basicos	sshh_prestado	Excusado y alcantarillado = 1, Excusado y pozo séptico = 2, Excusado y pozo ciego = 3, Letrina = 4, No tiene = 5
servicios_basicos	agua_obtencion	Red pública = 1, Pila o llave pública = 2, Otra fuente por tubería = 3, Carro repartidor, triciclo = 4, Pozo = 5, Río, vertiente, acequia = 6, Otro = 7
servicios_basicos	tipo_alumbrado	Empresa eléctrica pública, Planta eléctrica privada, Vela, candil, mechero, gas, Ninguno
servicios_basicos	agua_medidor	Si = 1, NO = 2
servicios_basicos	agua_junta	Si = 1, NO = 2
servicios_basicos	agua_tuberia	Por tubería dentro de la vivienda = 1, Por tubería fuera de la vivienda, pero en el lote = 2, Por tubería fuera de la vivienda, lote o terreno = 3, No recibe agua por tubería sino por otros medios = 4
servicios_basicos	eliminacion_basura	Contratan el servicio = 1, Servicio municipal = 2, Botan a la calle, quebrada, río = 3, La queman, entierran = 4, Otra = 5
servicios_basicos	id_hogar	

5.4 Implementación y carga de base de datos

Una vez realizado el modelo lógico relacional en la herramienta de MySQL Workbench, se generó el script DDL para su implementación.

Y para la población de los datos se realizó con un script de Python compuesto por dos partes, la primera "*complementarios_1.py*" para realizar la carga de datos complementarios hacia las tablas "Provincia", "Canton", "Parroquia"; y la segunda parte "*principales_2.py*" el cual debe ser ejecutado después de complementarios para así asegurar que los *inserts* en tablas con llaves foráneas se realizarán correctamente.

El código de Python utiliza cursores para iterar a través de columnas de un *dataframe* de Pandas que contiene la información del archivo csv.

Este script se encuentra en: [Proyecto-Integrado-1.2/Base de Datos/Carga de Datos at main · cdrodriguez7/Proyecto-Integrado-1.2 · GitHub]

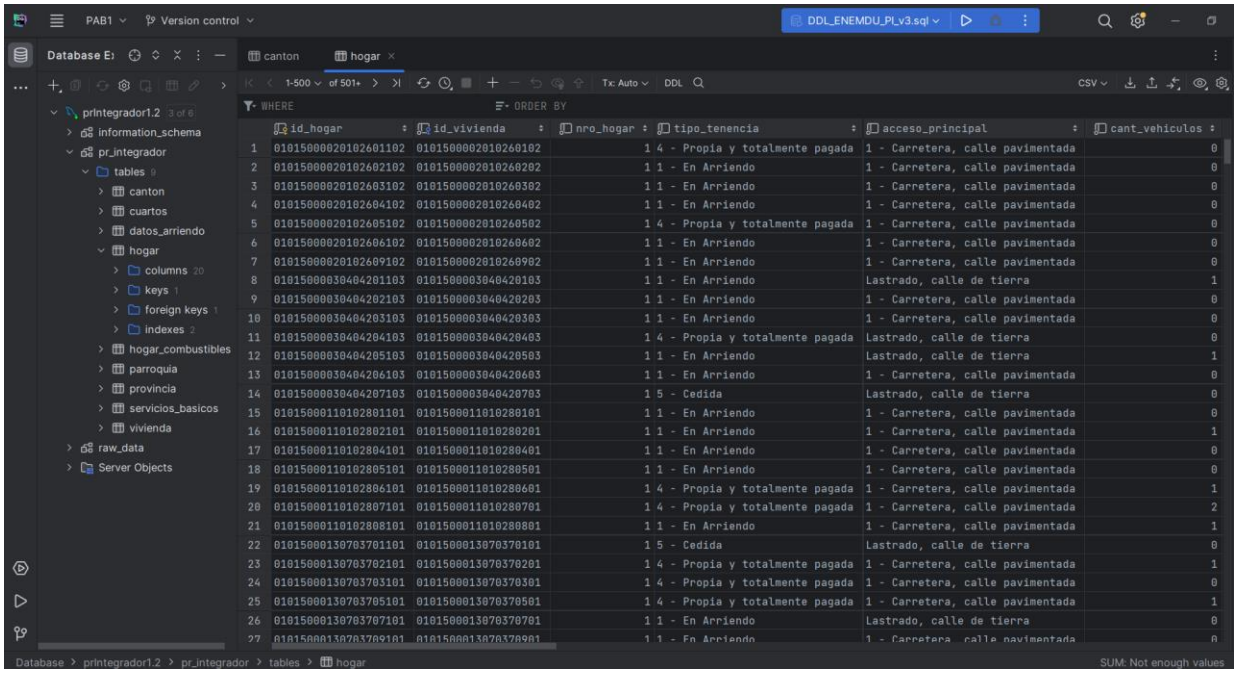


Ilustración 1 Evidencia de implementación y carga de la tabla "hogar"

```python
cursor08 = cnx.cursor()
for index, row in datos_arriendo.iterrows():
    agua_incluida = row['agua_incluida']
    luz_incluida = row['luz_incluida']
    parentesco_propietario = row['parentesco']
    id_hogar = row['id_hogar']

    # Perform any necessary transformations or formatting on the data
    if agua_incluida == '1':
        agua_incluida = '1-Si'
    else:
        agua_incluida = '2-No'

    if luz_incluida == '1':
        luz_incluida = '1-Si'
    else:
        luz_incluida = '2-No'

    if parentesco_propietario == '1':
        parentesco_propietario = '1-Si'
    else:
        parentesco_propietario = '2-No'
    # Construct the INSERT query
    query = "INSERT INTO datos_arriendo (agua_incluida, luz_incluida, parentesco_propietario, id_hogar) " \
            "VALUES (%s, %s, %s, %s)"
    values = (agua_incluida, luz_incluida, parentesco_propietario, id_hogar)
    cursor08.execute(query, values)
# Commit the changes to the database
cnx.commit()
cursor08.close()
```

Ilustración 2 Evidencia de script Python para carga de la tabla "datos_arriendo"

5.5 Script SQL

Enlace: [Proyecto-Integrado-1.2/Base_de_Datos/Modelos/DDL_PR_ENEMDU_VIVIENDAS.sql at main · cdrodriguez7/Proyecto-Integrado-1.2 · GitHub]

6. Componente Programación

En el marco del Proyecto Integrador de Saberes coordinado desde la asignatura Prácticum 1.2, se destaca el componente de Programación como una pieza fundamental en el desarrollo de una solución informática

innovadora. Este componente se enfoca en aplicar las capacidades y competencias adquiridas durante los primeros niveles de la carrera de Ciencias de la Computación.

En este apartado, abordaremos la importancia de las herramientas utilizadas en el proyecto, las cuales desempeñan un papel crucial en el análisis de datos sobre información de acceso y condiciones de vivienda en el Ecuador. Asimismo, se explicarán los comandos y sentencias que fueron empleados tanto para consultar como para visualizar datos, tanto a nivel del archivo fuente como en la base de datos MySQL, permitiendo así un estudio exhaustivo y detallado de la información recolectada.

La utilización de bases de datos MySQL y entornos interactivos de análisis de datos, como Apache Zeppelin, ha permitido a nuestro equipo de trabajo realizar una exploración profunda y rigurosa de los datos reales relacionados con problemas de interés social en el país.

En el desarrollo de este componente, se han generado resultados significativos, cuyo valor radica en la capacidad de presentar visualizaciones claras y concisas, acompañadas de un análisis e interpretación detallada. Estas visualizaciones son una herramienta poderosa para comprender mejor la situación del acceso y las condiciones de vivienda en el Ecuador, aportando información y conocimiento relevante a la sociedad y a las instituciones involucradas en la toma de decisiones.

6.1 Herramientas utilizadas:

En el desarrollo de este proyecto, se emplearon diversas herramientas tecnológicas que desempeñaron un papel fundamental en el éxito y la eficiencia del trabajo realizado. A continuación, se explicará cada una de las herramientas utilizadas y su importancia en el proyecto:

1. Apache Zeppelin: Apache Zeppelin es un entorno interactivo de análisis de datos que permite la creación de cuadernos colaborativos. Nos sirvió para realizar exploraciones y visualizaciones de datos de manera interactiva, facilitando la presentación y compartición de resultados con el equipo de trabajo.

2. MySQL Workbench: MySQL Workbench es una herramienta gráfica para el diseño, desarrollo y administración de bases de datos MySQL. Fue fundamental para la gestión y manipulación de datos almacenados en la base de datos MySQL utilizada en el proyecto.

3. AngularJS: AngularJS es un framework de JavaScript que se empleó para desarrollar la interfaz de usuario del sistema. Su enfoque en la creación de aplicaciones web de una sola página (SPA) permitió una experiencia de usuario fluida y dinámica.

4. CSS (Cascading Style Sheets): CSS se utilizó para el diseño y la presentación visual del sitio web del proyecto. Proporcionó una manera de definir estilos, formatos y disposición de elementos HTML

para lograr una apariencia estética y coherente.

5. Python: Python es un lenguaje de programación ampliamente utilizado en análisis de datos y desarrollo de aplicaciones. Su uso en el proyecto nos permitió implementar algoritmos para el análisis y procesamiento de datos relacionados con problemas de interés social.

6. HTML (HyperText Markup Language): HTML es el lenguaje estándar para crear páginas web. Se empleó para estructurar y organizar el contenido de la aplicación web, definiendo los elementos y su disposición en la interfaz.

7. JDK 1.8 (Java Development Kit 1.8): JDK 1.8 es un conjunto de herramientas necesarias para el desarrollo de aplicaciones en Java. Su uso en el proyecto estuvo vinculado a la compatibilidad con ciertas librerías y el entorno de ejecución necesario para el correcto funcionamiento de algunas partes del sistema.

8. Scala 2.12: Scala es un lenguaje de programación que se ejecuta en la JVM (Java Virtual Machine) y es altamente compatible con Apache Spark. Se utilizó para desarrollar funciones y algoritmos avanzados en el procesamiento de datos.

9. Ubuntu 20.04: Ubuntu es un sistema operativo basado en Linux. Las versiones 18.04 y 20.04 se utilizaron como entornos de desarrollo y despliegue para el proyecto debido a su estabilidad y amplia compatibilidad con las herramientas y tecnologías empleadas.

10. Apache Spark 3.2.1 Hadoop 3.2: Apache Spark es un potente motor de procesamiento de datos distribuido, compatible con Hadoop, que facilita el análisis de grandes volúmenes de información de manera eficiente. Su implementación nos permitió trabajar con datos masivos y realizar operaciones de procesamiento en paralelo.

En resumen, cada una de las herramientas mencionadas cumplió un rol específico y complementario en el desarrollo del proyecto. Desde la exploración y visualización de datos con Apache Zeppelin, hasta la administración de la base de datos con MySQL Workbench, y el desarrollo de la interfaz de usuario con AngularJS y CSS. Además, Python y Scala proporcionaron las capacidades de programación necesarias para el análisis de datos, mientras que las versiones de Ubuntu y el conjunto de herramientas JDK garantizaron la compatibilidad y el ambiente de ejecución requerido. Todo ello se potenció con Apache Spark, que permitió el procesamiento eficiente de grandes conjuntos de datos. La elección y utilización de estas herramientas fue fundamental para lograr un enfoque integral y exitoso en el desarrollo del proyecto integrador.

6.2 Explicación de comandos y sentencias usadas para consultar y visualizar datos tanto a nivel del archivo fuente. tanto a nivel de archivo fuente, como a nivel de base de datos MySQL

6.1.1 Exploración de la data desde el archivo .csv

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

val mySchema = StructType(
    Array(
        StructField("area", IntegerType, false),
        StructField("ciudad", IntegerType, false),
        StructField("conglomerado", IntegerType, false),
        StructField("panelm", IntegerType, false),
        StructField("vivienda", IntegerType, false),
        StructField("hogar", IntegerType, false),
        StructField("acceso_principal", IntegerType, false),
        StructField("tipo_vivienda", IntegerType, false),
        StructField("techo_material", IntegerType, false),
        StructField("estado_techo", IntegerType, false),
        StructField("piso_material", IntegerType, false),
        StructField("estado_piso", IntegerType, false),
        StructField("pared_material", IntegerType, false),
        StructField("estado_pared", IntegerType, false),
        StructField("nro_cuartos", IntegerType, false),
        StructField("nro_dormitorios", IntegerType, false),
        StructField("nro_cuartos_negocio", IntegerType, false),
        StructField("cocina_cuarto", IntegerType, false),
        StructField("cocinar_material", IntegerType, false),
        StructField("tipo_servicio_higienico", IntegerType, false),
        StructField("alternativa_no_higienico", IntegerType, true),
        StructField("tipo_instalacion_sanitaria", IntegerType, true),
        StructField("obtencion_agua", IntegerType, false),
        StructField("medidor_agua", IntegerType, true),
        StructField("junta_agua", IntegerType, true),
        StructField("tipo_tuberia", IntegerType, false),
        StructField("ducha", IntegerType, false),
        StructField("tipo_alumbrado", StringType, false),
        StructField("eliminacion_basura", IntegerType, false),
        StructField("tenencia_vivienda", IntegerType, false),
        StructField("valor_arriendo", IntegerType, true),
        StructField("inarriendo_agua", IntegerType, true),
        StructField("inarriendo_luz", IntegerType, true),
        StructField("parentesco_propietario", IntegerType, true),
        StructField("posesion_vehiculos", IntegerType, false),
        StructField("cantidad_vehiculos", IntegerType, true),
        StructField("posesion_motos", IntegerType, true),
        StructField("cantidad_motos", IntegerType, true),
        StructField("abastecimiento_super", IntegerType, true),
        StructField("gasto_super", IntegerType, true),
        StructField("abastecimiento_extra", IntegerType, true),
        StructField("gasto_extra", IntegerType, true),
        StructField("abastecimiento_diesel", IntegerType, true),
        StructField("gasto_diesel", IntegerType, true),
        StructField("abastecimiento_eco", IntegerType, true),
```

```
StructField("gasto_eco", IntegerType, true),
StructField("abastecimiento_elect", IntegerType, true),
StructField("gasto_elect", IntegerType, true),
StructField("abastecimiento_gas", IntegerType, true),
StructField("gasto_gas", IntegerType, true),
StructField("estrato", IntegerType, false),
StructField("fexp", StringType, false),
StructField("upm", StringType, false),
StructField("id_vivienda", StringType, false),
StructField("id_hogar", StringType, false),
StructField("periodo", IntegerType, false),
StructField("mes", StringType, false)
));
```

Esta sentencia de código en Scala nos sirvió para crear un esquema de datos que se utilizará en el contexto del proyecto de análisis.

En este caso específico, el esquema se define mediante la variable `mySchema`, que es de tipo `StructType`, que es un tipo de datos proporcionado por la biblioteca de Apache Spark para definir esquemas de datos estructurados.

La definición del esquema `mySchema` consta de varios `StructField`, cada uno representando un campo en el conjunto de datos que se va a analizar. Cada `StructField` está compuesto por tres parámetros:

1. Nombre del campo: Es el nombre que se asigna a ese campo específico en el conjunto de datos.
2. Tipo de datos: Indica el tipo de datos que almacena el campo. Por ejemplo, `IntegerType` representa un campo de números enteros, y `StringType` representa un campo de texto (cadena de caracteres).
3. Booleano: El tercer parámetro es un valor booleano que indica si el campo permite valores nulos (`true`) o no (`false`). Un valor nulo significa que el campo puede estar vacío o sin datos.

El esquema `mySchema` parece representar información sobre viviendas, hogares y gastos asociados, ya que contiene campos como "area", "ciudad", "conglomerado", "vivienda", "gasto_super", etc. Cada uno de estos campos se espera que tenga un valor entero o de cadena de caracteres, y algunos de ellos permiten valores nulos.

Este esquema es especialmente útil en el contexto de Apache Spark, un motor de procesamiento de datos distribuido, ya que ayuda a optimizar la ejecución de operaciones y permite realizar análisis más eficientes y coherentes de grandes volúmenes de datos.

6.1.2 Lectura del csv a Dataframe de Spark

```
val data = spark
```

```
    .read
    .schema(mySchema)
    .option("header", "true")
    .option("delimiter", ";")
    .csv("/workspace/zeppelin-
paavanzada/enemdu_vivienda_hogar_2023_I_trimestre.csv")
```

En esta sentencia de código se realizó la lectura de un archivo CSV en Spark y su transformación en un DataFrame utilizando el esquema previamente definido `mySchema`. Vamos a explicar cada parte de la sentencia y su función:

1. val data: Es la variable que almacena el DataFrame resultante después de leer y procesar el archivo CSV. En este caso, se ha llamado "data", pero podría tener cualquier otro nombre válido.

2. spark: Se refiere a la sesión o contexto de Spark, que es el punto de entrada principal para interactuar con el entorno de Spark y realizar operaciones de procesamiento de datos.

3. read: Es el método utilizado para iniciar el proceso de lectura de datos en Spark.

4. schema(mySchema): Aquí se especifica el esquema que se utilizará para estructurar los datos leídos desde el archivo CSV. Se emplea el esquema previamente definido `mySchema`, lo que garantiza que los datos se organizarán de acuerdo con los campos y tipos de datos especificados.

5. option("header", "true"): Esta opción le indica a Spark que el archivo CSV contiene una primera fila con los nombres de las columnas (encabezados). Al establecer esta opción como "true", Spark utilizará esa fila como los nombres de las columnas en el DataFrame resultante.

6. option("delimiter", ";"): Esta opción define el delimitador utilizado en el archivo CSV. En este caso, se ha especificado que el delimitador es ";". Esto asegura que Spark pueda interpretar correctamente la estructura del archivo y separar cada campo correctamente.

7.csv("/workspace/zeppelin-paavanzada/enemdu_vivienda_hogar_2023_I_trimestre.csv"):
Finalmente, esta parte de la sentencia indica la ubicación del archivo CSV que se desea leer. En este caso, el archivo se encuentra en el directorio "/workspace/zeppelin-paavanzada/" y se llama "enemdu_vivienda_hogar_2023_I_trimestre.csv".

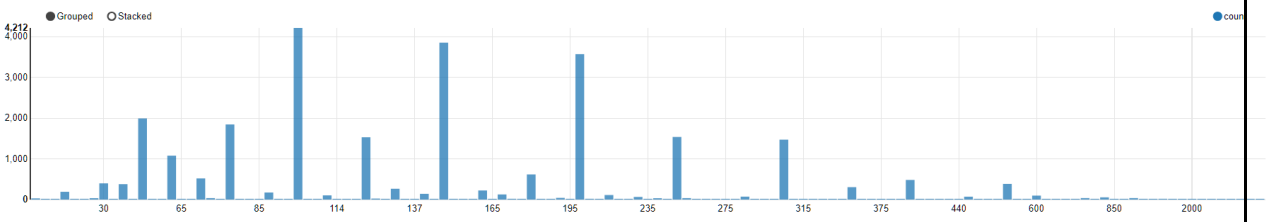
6.1.3 Columnas Cuantitativas

Una vez que se ha creado el DataFrame de Spark, este contiene toda la información del archivo CSV estructurada en columnas y filas. Para realizar la exploración de datos por columnas cuantitativas, lo primero que realizamos es el identificar qué columnas del DataFrame contienen datos numéricos o valores cuantitativos, como números enteros o decimales. Estas columnas son de interés para realizar

análisis estadísticos y visualizaciones relacionadas con sus distribuciones y tendencias

- **Columna Valor Arriendo:** Esta columna posee los valores que las personas pagan o pagarían por el arriendo de su hogar

```
1       val dfArriendoUnic =
2       data.select("valor_arriendo").groupBy("valor_arriendo").count
3
4       dfArriendoUnic.count
5
        z.show(dfArriendoUnic)
```



Esta secuencia de código en Spark está destinada a realizar un análisis y exploración de datos en relación con la columna "valor_arriendo" del DataFrame `data`. Vamos a explicar cada parte de la secuencia y su función:

1. `val dfArriendoUnic = data.select("valor_arriendo").groupBy("valor_arriendo").count`: En esta línea, se selecciona la columna "valor_arriendo" del DataFrame `data`. Luego, se agrupan los datos en función de los valores únicos que aparecen en la columna "valor_arriendo" mediante la operación `groupBy`. Finalmente, se cuenta la cantidad de registros para cada valor único usando la función `count`. El resultado es un nuevo DataFrame llamado `dfArriendoUnic` que contiene dos columnas: "valor_arriendo" y "count", donde "valor_arriendo" representa los valores únicos y "count" representa la cantidad de ocurrencias de cada valor.

2. `z.show(dfArriendoUnic)`: Finalmente, se muestra el contenido del DataFrame `dfArriendoUnic` en un formato tabular. Los valores únicos de la columna "valor_arriendo" se presentan en la columna "valor_arriendo", y en la columna "count" se muestra la cantidad de ocurrencias para cada valor único.

Este análisis proporciona una visión general de la distribución de los valores en la columna "valor_arriendo". La conclusión de esta visualización es que los datos no siguen una distribución normal.

1. Identificación de valores únicos: El informe muestra los valores únicos presentes en la columna "valor_arriendo", lo que es útil para conocer los diferentes montos de arriendo que aparecen en los datos.

2. Análisis de frecuencia: La columna "count" muestra cuántas veces aparece cada valor único en la columna "valor_arriendo". Esto permite comprender cuántos registros tienen montos de arriendo iguales y cuántos registros tienen montos diferentes.

3. Detección de valores faltantes: Si hay algún valor con "null" o sin información en la columna "valor_arriendo", es posible que aparezca un valor con una etiqueta "null" o "-1" en la tabla de resultados.

En resumen, esta secuencia de código y su informe proporcionan información sobre la distribución de los valores en la columna "valor_arriendo", lo que permite entender mejor la composición de los datos en esa columna específica y detectar posibles patrones o valores faltantes.

6.1.4 Fórmula para eliminar Valores Atípicos

Análisis Descriptivo de la columna

```
data.select("valor_arriendo").summary().show()
```

```
+-------+-----------------+
|summary|    valor_arriendo|
+-------+-----------------+
|  count|            26580|
|   mean|692.5930398796087|
| stddev|22949.32374617818|
|    min|               10|
|    25%|               90|
|    50%|              150|
|    75%|              200|
|    max|           999999|
+-------+-----------------+
```

Media

```
val avg = data.select(mean("valor_arriendo"))
        .first()(0)//fila 0 columna 0
        .asInstanceOf[Double]
```

```
avg: Double = 692.5930398796087
```

Desviación Estándar

```
val stdDesv = data.select(stddev("valor_arriendo"))
        .first()(0)//fila 0 columna 0
        .asInstanceOf[Double]//Traelo como instacia de tipo Double
```

```
stdDesv: Double = 22949.32374617818
```

Calcular límites superior e inferior

```
val inferior = avg - 3 * stdDesv //Si es negativo no tendremos ningun valor
val superior = avg + 3 * stdDesv
```

```
inferior: Double = -68155.37819865493
superior: Double = 69540.56427841414
```

Valores por debajo del límite inferior

SPARK JOB FINISHED

```
val valoresMenoresInferior =  data.select("valor_arriendo").where($"valor_arriendo" < inferior)
valoresMenoresInferior.describe().show
```

```
+-------+--------------+
|summary|valor_arriendo|
+-------+--------------+
|  count|             0|
|   mean|          null|
| stddev|          null|
|    min|          null|
|    max|          null|
+-------+--------------+
```

Resumen de Valor_Arriendo sin Outliers

```
val arriendoSinOutliers = data.select("valor_arriendo").where($"valor_arriendo" < superior)

arriendoSinOutliers.summary().show()
```

```
+-------+------------------+
|summary|    valor_arriendo|
+-------+------------------+
|  count|             26565|
|   mean|162.21110483719178|
| stddev|134.42121128571992|
|    min|                10|
|    25%|                90|
|    50%|               150|
|    75%|               200|
|    max|              5000|
+-------+------------------+
```

Agrupación de valores únicos en valor_arriendo

```
val dataSinOutliersUnic = arriendoSinOutliers.select("valor_arriendo").groupBy("valor_arriendo").count

dataSinOutliersUnic.count
```

```
dataSinOutliersUnic: org.apache.spark.sql.DataFrame = [valor_arriendo: int, count: bigint]
res15: Long = 125
```

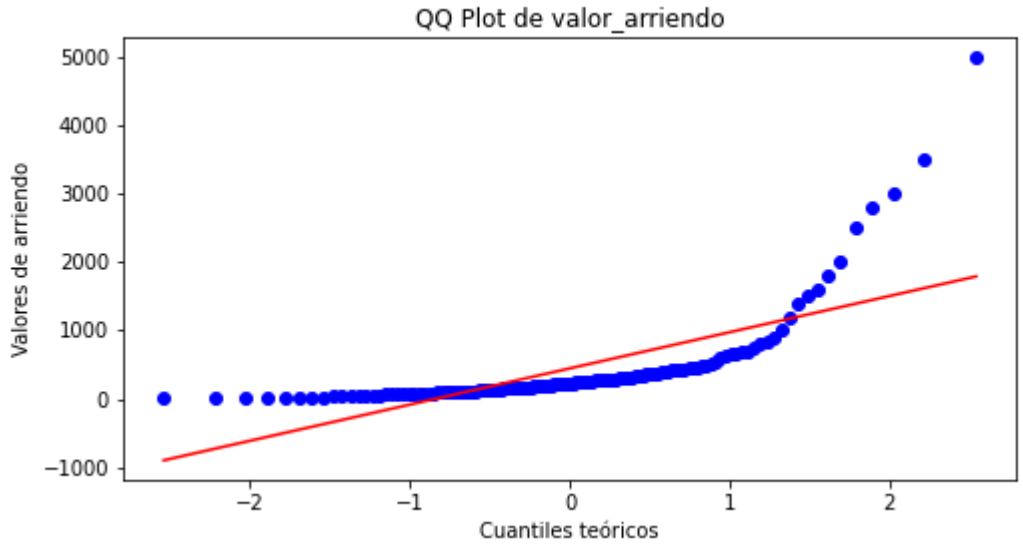
Resultados despúes de eliminar valores atípicos

Cantidad de Valores Únicos en valor_arriendo = 125

Cantidad de Valores Atípicos eliminados = 15

QQ PLOT para valores únicos y sin outliers de valor_arriendo vs Distribución Normal

```
%pyspark
from pyspark.sql import DataFrame
from pyspark.sql.functions import col
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats

# Obtener el DataFrame de dfCuantitativo
dfValorArriendo = DataFrame(z.get("dfSinOutliers"), sqlContext)

# Convertir la columna "valor_arriendo" a tipo Double
dfValorArriendo = dfValorArriendo.withColumn("valor_arriendo", col("valor_arriendo").cast("double"))

# Obtener los datos como una lista
valor_arriendo_data = dfValorArriendo.select("valor_arriendo").rdd.flatMap(lambda x: x).collect()

# Ordenar los datos de menor a mayor
valor_arriendo_data_sorted = sorted(valor_arriendo_data)

# Convertir los datos a un array de numpy
valor_arriendo_array = np.array(valor_arriendo_data_sorted)

# Crear el QQ plot con los datos
plt.figure(figsize=(8, 4))
stats.probplot(valor_arriendo_array, plot=plt, dist='norm')

plt.title('QQ Plot de valor_arriendo')
plt.xlabel('Cuantiles teóricos')
plt.ylabel('Valores de arriendo')
plt.show()
```

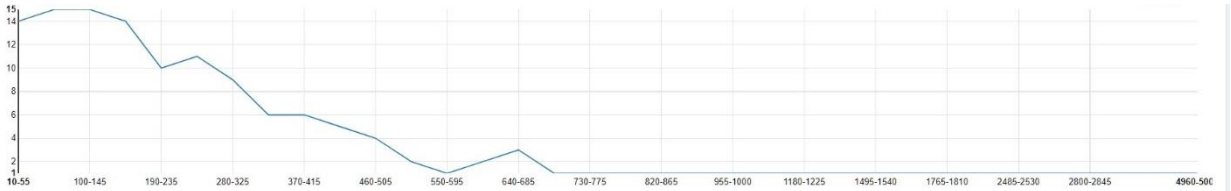


Generación de intervalos de Valor_Arriendo

```
val rowValues = dataSinOutliersUnic.select(min($"valor_arriendo"), max($"valor_arriendo")).first
val minValue = rowValues(0).asInstanceOf[Int]
val maxValue = rowValues(1).asInstanceOf[Int]
val bins = math.sqrt(dataSinOutliersUnic.count).toInt
val range = (maxValue / minValue) / bins
var minCounter = minValue
var maxCounter = range
```

```
val intervalCounterValorArriendo = (minValue to maxValue by range).map{ i => (i, i + range) }
.map(r => (r._1, r._2, dataSinOutliersUnic.where($"valor_arriendo".between(r._1, r._2)).count))
```

```
val valorArrienoIntervals = spark
.createDataFrame(intervalCounterValorArriendo.toSeq)
.toDF("low", "high", "count")
.withColumn("interval", concat($"low", lit("-"), $"high"))
```



1. Cálculo de límites superior e inferior para eliminar valores atípicos:

En esta sección, se realiza un análisis descriptivo de la columna "valor_arriendo" para obtener la media y la desviación estándar. Luego, se calculan los límites inferior y superior utilizando la fórmula de ±3 veces la desviación estándar alrededor de la media. Estos límites se utilizarán para identificar y eliminar los valores atípicos en la columna "valor_arriendo".

2. Valores por debajo del límite inferior y por encima del límite superior:

Se identifican los valores que están por debajo del límite inferior y por encima del límite superior, ya que se consideran valores atípicos. Se muestra un resumen de estos valores para su revisión.

3. Resumen de valor_arriendo sin outliers:

Se crea un nuevo DataFrame llamado "arriendoSinOutliers" que excluye los valores atípicos identificados previamente. Se muestra un resumen del DataFrame resultante que proporciona estadísticas descriptivas para la columna "valor_arriendo" sin los valores atípicos.

4. Agrupación de valores únicos en valor_arriendo:

Se crea un nuevo DataFrame llamado "dataSinOutliersUnic" que contiene los valores únicos en la columna "valor_arriendo" después de eliminar los valores atípicos. Se realiza un conteo de los valores únicos y se muestra la cantidad total de valores después de la eliminación de outliers.

5. QQ PLOT para valores únicos y sin outliers de valor_arriendo vs Distribución Normal:

En esta parte, se realiza un QQ Plot (Quantile-Quantile Plot) para visualizar si los valores únicos en la columna "valor_arriendo" sin los outliers siguen una distribución normal. El QQ Plot compara los cuantiles teóricos de una distribución normal con los cuantiles de los datos reales. Si los puntos en el

gráfico se alinean aproximadamente con una línea recta, sugiere que los datos siguen una distribución normal. La gráfica generada muestra cómo se ajustan los datos reales a una distribución normal.

Este conjunto de sentencias de código y su tipo de informe se enfocan en el análisis y eliminación de valores atípicos en la columna "valor_arriendo". Además, se realiza un análisis para determinar si los valores únicos después de la eliminación de outliers siguen una distribución normal.

El informe proporciona información valiosa sobre la distribución de los datos en la columna "valor_arriendo" y cómo afecta la eliminación de valores atípicos a la distribución de los datos. Mediante la visualización (QQ Plot) para evaluar si los datos sin outliers siguen una distribución normal, con lo que al ver que no tienen una aproximación a la línea en rojo deducimos que no posee una distribución normal.

6.1.5 Columnas Cuantitativas

Crear un DataFrame con columnas no numericas

```
val dfCualitativo = data.select("area",
"ciudad",
"tipo_vivienda",
"techo_material",
"estado_techo",
"piso_material",
"estado_piso",
"pared_material",
"estado_pared",
"tipo_servicio_higienico",
"alternativa_no_higienico",
"tipo_instalacion_sanitaria",
"tipo_tuberia",
"tenencia_vivienda",
"valor_arriendo",
"parentesco_propietario")
```

```
dfCualitativo: org.apache.spark.sql.DataFrame = [area: int, ciudad: int ... 14 more fields]
```

Reemplazo de Valores en la Columna 'area' y Conteo Agrupado por 'area_etiqueta'

```
// Definir una función para realizar el reemplazo de valores en la columna "area"
val reemplazarArea = udf((valor: Int) => valor match {
  case 1 => "Urbana"
  case 2 => "Rural"
})

// Crear una nueva columna "area_etiqueta" con los valores reemplazados
val data_with_etiquetas = dfCualitativo.withColumn("area", reemplazarArea(col("area")))

// Agrupar y contar por "area_etiqueta"
data_with_etiquetas.groupBy("area").count().sort("area").show()
```

```
+------+-----+
|  area|count|
+------+-----+
| Rural| 6884|
|Urbana|19696|
```

Definición de Funciones de Reemplazo para Variables Categóricas

```
// Definir una función para realizar el reemplazo de valores en las columnas
val reemplazarEstadoTecho = udf((valor: Int) => valor match {
  case 1 => "Bueno"
  case 2 => "Regular"
  case 3 => "Malo"

})

val reemplazarPisoMaterial = udf((valor: Int) => valor match {
  case 1 => "Duela, parquet, tablón tratado o piso flotante"
  case 2 => "Cerámica, baldosa, vinil o porcelanato "
  case 3 => "Mármol o marmetón"
  case 4 => "Ladrillo o cemento"
  case 5=> "Tabla / tablón no tratado"
  case 6=> "Caña"
  case 7 => "Tierra"
  case 8=> "Otro Material"
})

val reemplazarEstadoPiso = udf((valor: Int) => valor match {
  case 1 => "Bueno"
  case 2 => "Regular"
  case 3 => "Malo"

})

val reemplazarParedMaterial = udf((valor: Int) => valor match {
  case 1 => "Hormigón/Ladrillo o Bloque"
  case 2 => "Asbesto/Cemento (Fibrolit)"
  case 3 => "Adobe o Tapia"
  case 4 => "Madera"
  case 5 => "Caña revestida o bahareque"
  case 6 => "Caña no revestida o estera"
  case 7 => "Otra Material"

})

val reemplazarEstadoPared = udf((valor: Int) => valor match {
  case 1 => "Bueno"
  case 2 => "Regular"
  case 3 => "Malo"

})


// Definir una función para realizar el reemplazo de valores en la columna "tipo_servicio_higienico"
val reemplazarTipoServicioHigienico = udf((valor: Int) => valor match {
  case 1 => "Excusado y alcantarillado"
  case 2 => "Excusado y pozo séptico"
  case 3 => "Excusado y pozo ciego"
  case 4 => "Letrina"
  case 5 => "No tiene"

})
```

Reemplazo de Valores en Múltiples Columnas del DataFrame dfCualitativo

FINISHED

```
// Crear un nuevo dataFrame con las columnas que han sido cambiadas
val data_with_etiquetas = dfCualitativo
  .withColumn("area", reemplazarArea(col("area")))
  .withColumn("tipo_vivienda", reemplazarTipoVivienda(col("tipo_vivienda")).cast(StringType))
  .withColumn("techo_material", reemplazarTechoMaterial(col("techo_material")).cast(StringType))
  .withColumn("estado_techo", reemplazarEstadoTecho(col("estado_techo")).cast(StringType))
  .withColumn("piso_material", reemplazarPisoMaterial(col("piso_material")).cast(StringType))
  .withColumn("estado_piso", reemplazarEstadoPiso(col("estado_piso")).cast(StringType))
  .withColumn("pared_material", reemplazarParedMaterial(col("pared_material")).cast(StringType))
  .withColumn("estado_pared", reemplazarEstadoPared(col("estado_pared")).cast(StringType))
   .withColumn("tipo_servicio_higienico", reemplazarTipoServicioHigienico(col("tipo_servicio_higienico")).cast(StringType))
  .withColumn("alternativa_no_higienico", reemplazarAlternativaNoHigienico(col("alternativa_no_higienico")).cast(StringType)
  .withColumn("tipo_instalacion_sanitaria", reemplazarTipoInstalacionSanitaria(col("tipo_instalacion_sanitaria")).cast(Strin
      ))
  .withColumn("tenencia_vivienda", reemplazarTenenciaVivienda(col("tenencia_vivienda")).cast(StringType))
  .withColumn("tipo_tuberia", reemplazarTipoTuberia(col("tipo_tuberia")).cast(StringType))
 .withColumn("parentesco_propietario ",reemplazarParentescoPropietario(col("parentesco_propietario")).cast(StringType))
```

Resumen Estadístico por Categorías de Variables Cualitativas

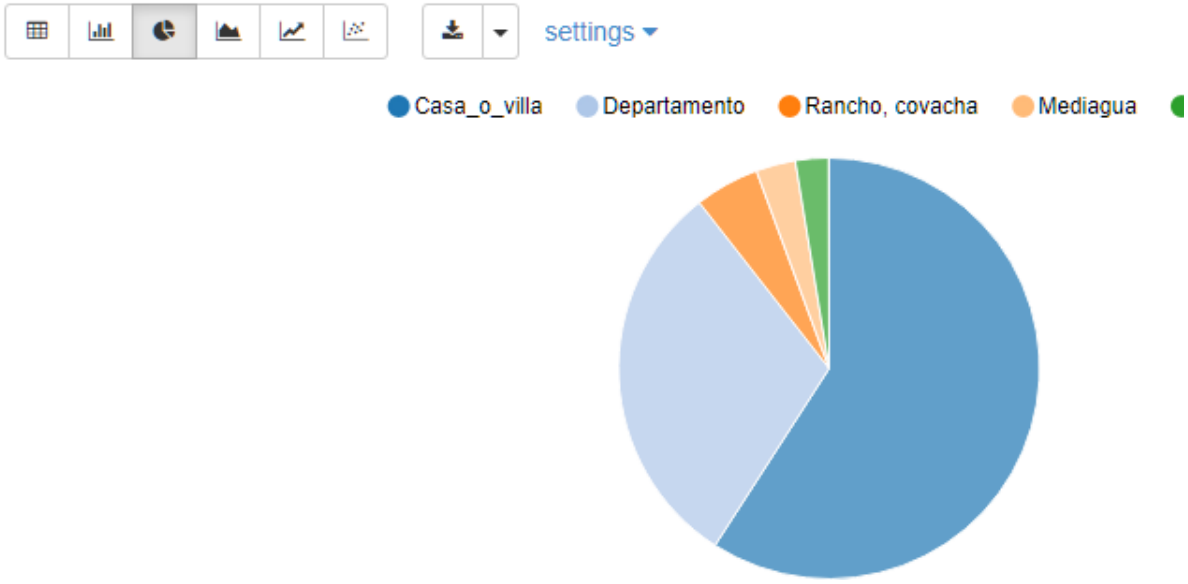
SPARK JOB FINISHE

```
data_with_etiquetas.groupBy("tipo_instalacion_sanitaria").count().sort("tipo_instalacion_sanitaria").show()
data_with_etiquetas.groupBy("tenencia_vivienda").count().sort("tenencia_vivienda").show()
data_with_etiquetas.groupBy("tipo_tuberia").count().sort("tipo_tuberia").show()
data_with_etiquetas.groupBy("tipo_servicio_higienico").count().sort("tipo_servicio_higienico").show()
data_with_etiquetas.groupBy("alternativa_no_higienico").count().sort("alternativa_no_higienico").show()
data_with_etiquetas.groupBy("estado_techo").count().sort("estado_techo").show()
data_with_etiquetas.groupBy("piso_material").count().sort("piso_material").show()
data_with_etiquetas.groupBy("estado_piso").count().sort("estado_piso").show()
data_with_etiquetas.groupBy("pared_material").count().sort("pared_material").show()
data_with_etiquetas.groupBy("estado_pared").count().sort("estado_pared").show()
```

```
|      Excusado y pozo s...|   82|
|                   Letrina|    2|
+--------------------------+-----+

+--------------------+-----+
|   tenencia_vivienda|count|
+--------------------+-----+
|Anticresis y/o ar...|   15|
|              Cedida| 4134|
|         En arriendo| 4907|
|               Otra|    8|
|Propia y la está ...|  580|
|Propia y totalmen...|16680|
|Recibida por serv...|  256|
+--------------------+-----+

+--------------------+-----+
```

Calcular la cantidad de viviendas por cada tipo de vivienda eliminada

```
+--------------------+-----+
|       tipo_vivienda|count|
+--------------------+-----+
|        Casa_o_villa|15714|
|        Departamento| 8057|
|     Rancho, covacha| 1317|
|            Mediagua|  808|
|Cuartos_en_casa_d...|  678|
|               Choza|    6|
+--------------------+-----+
```



Filtrar por tipo de vivienda "Departamento"

SPARK

```
val dfFiltradoTipoVivienda = data_with_etiquetas.filter($"tipo_vivienda" === "Departamento")

// Filtrar por estado del techo "Regular"
val dfFiltradoEstadoTecho = data_with_etiquetas.filter($"estado_techo" === "Regular")

// Filtrar por tipo de vivienda "Departamento" y estado del techo "Regular"
val dfFiltrado = data_with_etiquetas.filter($"tipo_vivienda" === "Departamento" && $"estado_techo" === "Re

dfFiltrado.count()
```

```
dfFiltradoTipoVivienda: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [area: string, ciudad: int
dfFiltradoEstadoTecho: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [area: string, ciudad: int
dfFiltrado: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [area: string, ciudad: int ... 15 more
res31: Long = 2726
```

Took 0 sec. Last updated by anonymous at July 26 2023, 4:35:03 PM.

Cálculo promedio del valor de arriendo por tipo de vivienda

SPARK

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

// Convertir la columna "valor_arriendo" a tipo Double
val dfConValorArriendo = data_with_etiquetas.withColumn("valor_arriendo", $"valor_arriendo".cast("Double")

// Calcular el promedio del valor arriendo por tipo de vivienda
val promedioArriendoPorTipo = dfConValorArriendo.groupBy("tipo_vivienda")
  .agg(functions.avg($"valor_arriendo").alias("promedio_arriendo"))
  .sort(desc("promedio_arriendo"))

// Mostrar el resultado
promedioArriendoPorTipo.show()
```

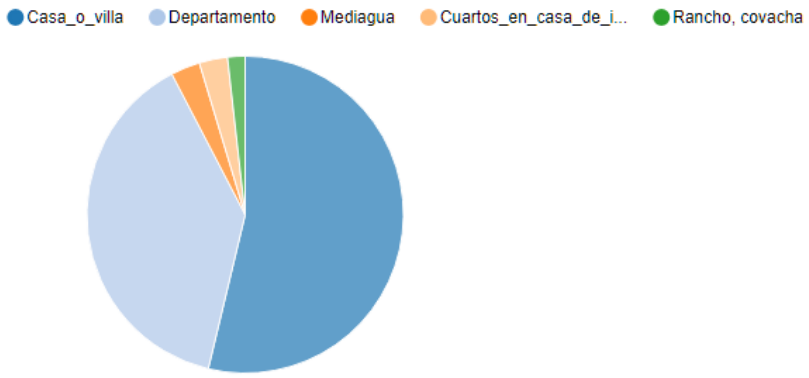
```
+--------------------+------------------+
|       tipo_vivienda| promedio_arriendo|
+--------------------+------------------+
|     Rancho, covacha|1582.7547456340167|
|            Mediagua|1321.9913366336634|
|        Casa_o_villa| 677.0442917143948|
|        Departamento| 566.3598113441727|
|Cuartos_en_casa_d...| 79.72123893805309|
|               Choza|              30.0|
+--------------------+------------------+
```

Contar la cantidad de viviendas por cada tipo de vivienda urbana existente:

SPARK JOB FINISHED

```
val viviendasPorTipo = data_with_etiquetas.filter($"area" === "Urbana")
  .groupBy("tipo_vivienda")
  .count()
  .sort(desc("count"))
z.show(viviendasPorTipo)
```

settings



6.1.6 . Creación del DataFrame `dfCualitativo`:

En esta sección, se seleccionan las columnas no numéricas del DataFrame `data`, que representan variables cualitativas. Estas columnas incluyen información sobre el área, ciudad, tipo de vivienda, material del techo, estado del techo, material del piso, estado del piso, material de las paredes, estado de las paredes, tipo de servicio higiénico, alternativa no higiénica, tipo de instalación sanitaria, tipo de tubería, tenencia de vivienda, valor de arriendo y parentesco del propietario.

6.1.7 . Cantidad de valores distintos en la columna "area":

En esta parte, se muestra la cantidad de valores distintos en la columna "area" utilizando la función

`distinct()` y `count()`. Se utiliza la función `groupBy()` y `count()` para agrupar los valores únicos y contar cuántos registros corresponden a cada valor.

3. Reemplazo de valores en la columna 'area' y conteo agrupado por 'area_etiqueta':
Se define una función de reemplazo (`reemplazarArea`) que convierte los valores numéricos de la columna "area" en etiquetas más descriptivas ("Urbana" y "Rural"). Luego, se crea una nueva columna llamada "area_etiqueta" con los valores reemplazados. Finalmente, se agrupa y cuenta los registros por la nueva columna "area_etiqueta" para mostrar la cantidad de viviendas en áreas urbanas y rurales.

4. Definición de funciones de reemplazo para variables categóricas:
En esta sección, se definen varias funciones de reemplazo (por ejemplo, `reemplazarEstadoTecho`, `reemplazarPisoMaterial`, etc.) que convierten los valores numéricos de algunas columnas cualitativas en etiquetas más descriptivas. Por ejemplo, se convierten los códigos numéricos en valores como "Bueno", "Regular" o "Malo" para describir el estado del techo, estado del piso, etc.

5. Reemplazo de valores en múltiples columnas del DataFrame dfCualitativo:
Se aplica el reemplazo de valores utilizando las funciones definidas anteriormente en las columnas cualitativas del DataFrame `dfCualitativo`. Los valores numéricos son reemplazados por etiquetas más descriptivas para facilitar el análisis y la interpretación.

6. Resumen estadístico por categorías de variables cualitativas:
En esta sección, se muestra un resumen estadístico para cada columna cualitativa después de aplicar el reemplazo de valores. Se utiliza `groupBy()` y `count()` para agrupar y contar los registros por cada categoría en las columnas cualitativas. El resultado muestra la cantidad de viviendas en cada categoría, lo que permite obtener información descriptiva sobre las características de las viviendas en cada categoría.

7. Cálculo promedio del valor de arriendo por tipo de vivienda:
En este paso, se convierte la columna "valor_arriendo" a tipo Double y luego se calcula el promedio del valor de arriendo agrupado por el tipo de vivienda. El resultado muestra el promedio del valor de arriendo para cada tipo de vivienda, lo que proporciona una idea de cómo varían los precios de arriendo según el tipo de vivienda.

Este conjunto de sentencias de código y su tipo de informe se centran en el análisis y exploración de datos para las columnas cualitativas del DataFrame. Se realiza el reemplazo de valores numéricos por etiquetas descriptivas para facilitar la interpretación y el análisis de los datos. Además, se presenta

información descriptiva sobre la cantidad de viviendas en cada categoría y se calcula el promedio del valor de arriendo por tipo de vivienda. El informe proporciona una visión general de las características de las viviendas y los precios de arriendo según diferentes categorías, lo que puede ser útil para comprender mejor los datos y tomar decisiones informadas en el análisis del conjunto de datos.

6. Gráficas realizadas con Conexión a la Base de Datos

Conversión a Dataframe de Spark

```
val dfFromMysqlAng = z.get("consulAngular01")

val arr_a = dfFromMysqlAng.toString.split("\n")
val colsName_a = arr_a(0).split("\t")
val dataArr_a = arr_a.drop(1)
val data_a = dataArr_a.map(row=> row.split("\t")).map(row=> (row(0), row(1), row(2).t

val dfAux_a = spark
 .createDataFrame(data_a)
 .toDF(colsName_a(0), colsName_a(1), colsName_a(2))
```

```
dfFromMysqlAng: Object =
"nombre_provincia       tipo_vivienda   promedio_arriendo
 Zonas No Delimitadas   Departamento    142.307692
 Zonas No Delimitadas   Casa o Villa    103.906250
 Zonas No Delimitadas   Cuartos en casa de inquilinato  100.000000
 Zonas No Delimitadas   Rancho, covacha 63.333333
 Zamora Chinchipe       Cuartos en casa de inquilinato  69.545455
 Zamora Chinchipe       Mediagua        73.333333
 Zamora Chinchipe       Rancho, covacha 57.500000
 Zamora Chinchipe       Departamento    165.047170
 Zamora Chinchipe       Casa o Villa    110.801187
 Tungurahua     Cuartos en casa de inquilinato  80.900000
 Tungurahua     Departamento    173.679920
 Tungurahua     Casa o Villa    181.202237
 Tungurahua     Mediagua        100.444444
 Santo Domingo  Casa o Villa    131.213873
 Santo Domingo  Departamento    134.176991
 Santo Domingo  Rancho, covacha 49...
```

Función Pivot para hacer colocar el dataframe en la forma en que necesitamos

SPARK JOB FINISHED

```
val auxDF_a = dfAux_a.groupBy("nombre_provincia").pivot("tipo_vivienda").avg("promedio_arriendo").orderBy("nombre_provin
    .fill(0)
auxDF_a.show
```

```
+--------------------+------------+-----+----------------------------+------------+----------+---------------+
|    nombre_provincia|Casa o Villa|Choza|Cuartos en casa de inquilinato|Departamento|  Mediagua|Rancho, covacha|
+--------------------+------------+-----+----------------------------+------------+----------+---------------+
|             Bolivar|  122.106017|  0.0|                         0.0|       130.0| 52.727273|      59.090909|
|          Chimborazo|  109.027397| 60.0|                   70.454545|  153.089888| 65.294118|           60.0|
|            Cotopaxi|  133.416149|  0.0|                   65.555556|  148.173913| 66.785714|           56.0|
|           Galápagos|  296.741935|  0.0|                       175.0|   241.88125|     182.5|            0.0|
|              Guayas|  216.758291|  0.0|                   64.285714|  200.426421|100.108333|      63.461538|
|           Los Rios|  112.286136|  0.0|                        70.0|  127.731959| 91.428571|           59.0|
|             Manabi|  130.198251|  0.0|                         0.0|  151.115702|      57.5|      75.714286|
|     Morona Santiago|  105.683761| 20.0|                   75.714286|  159.878049| 74.166667|      48.833333|
|                Napo|  122.802691|  0.0|                   76.730769|       136.5|      80.0|      60.592105|
|            Orellana|   135.02008|  0.0|                   75.217391|  134.181818|105.833333|      78.531469|
|            Pastaza|  158.847656|  0.0|                        65.0|   149.52381|    99.375|      68.444444|
|         Santa Elena|  141.084337|  0.0|                        50.0|  142.231405| 91.666667|      69.333333|
|       Santo Domingo|  131.213873|  0.0|                   50.833333|  134.176991| 87.142857|       49.83871|
|          Tungurahua|  181.202237|  0.0|                        80.9|   173.67992|100.444444|            0.0|
|Zamora Chinchipe|  110.801187|  0.0|                   69.545455|  165.047170|  73.333333|           57.5|
```

SPARK JOB FINISHED

```
val json_a = auxDF_a.toJSON.map(row => row.mkString).collectAsList

z.angularBind("dataAsJSON_aa", json_a)
```

```
json_a: java.util.List[String] = [{"nombre_provincia":" Bolivar","Casa o Villa":122.106017,"Choza":0.0,"Cuartos en casa de inquilinato":0.0,"Departamento":130.0,"Mediagua":52.727273,"Rancho, covacha":59.090909}, {"nombre_provincia":" Chimborazo","Casa o Villa":109.027397,"Choza":60.0,"Cuartos en casa de inquilinato":70.454545,"Departamento":153.089888,"Mediagua":65.294118,"Rancho, covacha":60.0}, {"nombre_provincia":" Cotopaxi","Casa o Villa":133.416149,"Choza":0.0,"Cuartos en casa de inquilinato":65.555556,"Departamento":148.173913,"Mediagua":66.785714,"Rancho, covacha":56.0}, {"nombre_provincia":" Galápagos","Casa o Villa":296.741935,"Choza":0.0,"Cuartos en casa de inquilinato":175.0,"Departamento":241.88125,"Mediagua":182.5,"Rancho, covacha":0.0}, {"nombre_p...
```

```
%angular
<input type="text" id="data4Spark_a" value={{dataAsJSON_aa}}>
```

[{"nombre_provincia":" Boliv

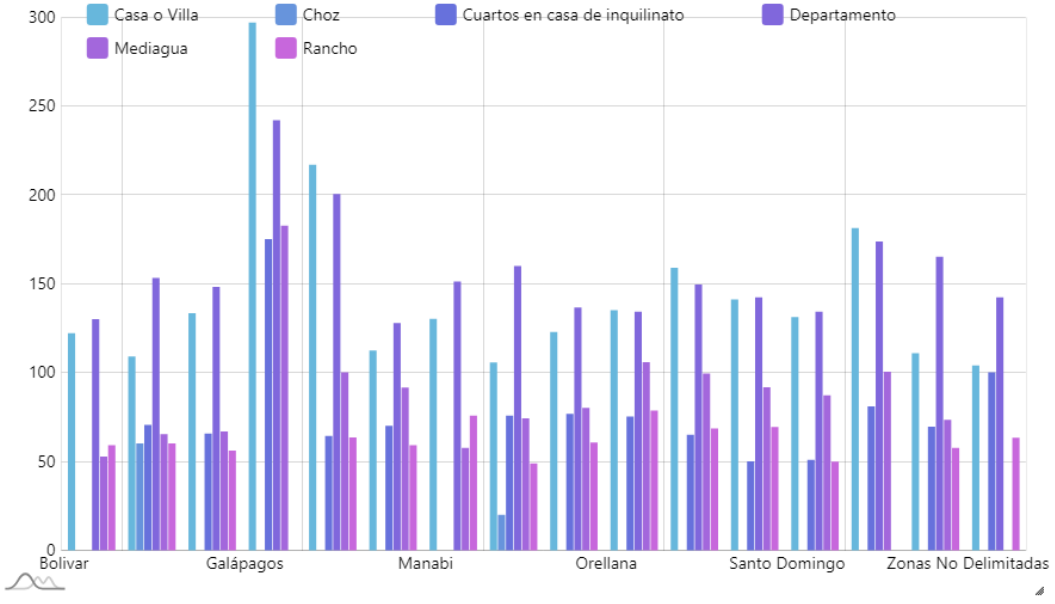
Took 0 sec. Last updated by anonymous at July 29 2023, 10:00:02 PM.

```
%angular
<script>
var cleanText = $('#data4Spark_a').val().replaceAll("\\", "").replaceAll("\"{", "{").replaceAll("}\"",
"}");
console.log(cleanText);
$('#data4Spark_a').val(cleanText);
</script>
```

Took 0 sec. Last updated by anonymous at July 29 2023, 10:00:04 PM.

Gráfica Interactiva con Angular

FINISHED



Took 0 sec. Last updated by anonymous at July 29 2023, 10:08:45 PM. (outdated)

Grafica de pastel del porcentaje de piso que tienen las “Casas o Villas”

Took 0 sec. Last updated by anonymous at July 26 2023, 3:42:15 PM.

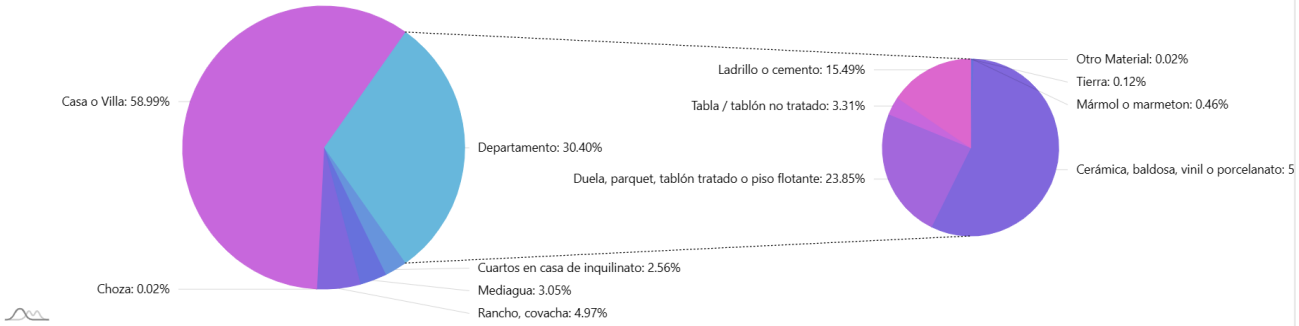
Consulta de porcetaje del piso que tienen las "Casa o Villas"

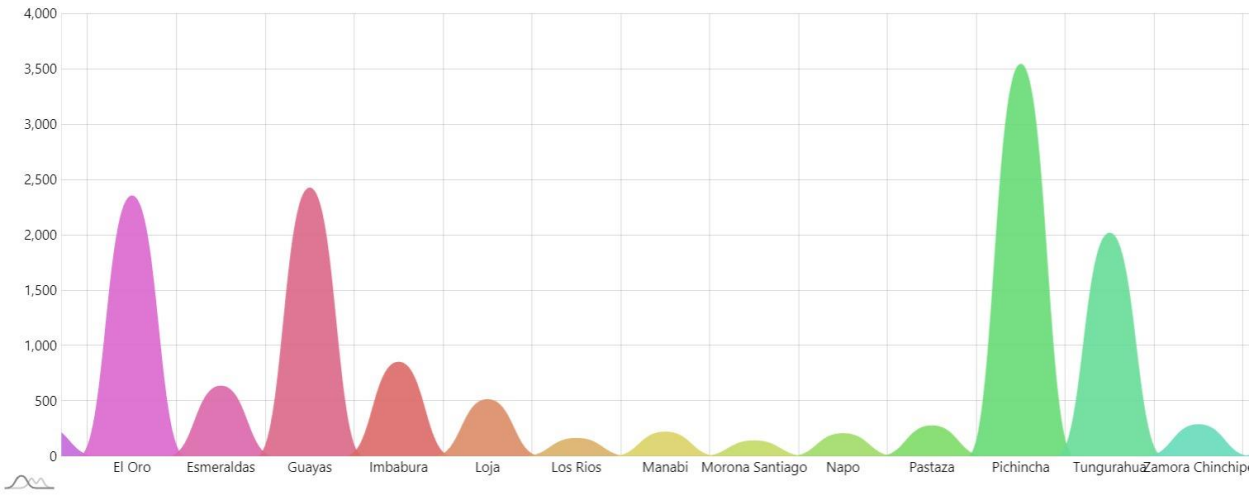
```
%mySQL(saveAs=casa_porcen)
SELECT
    h.material_piso,
    (COUNT(v.id_vivienda) / (SELECT COUNT(*) FROM vivienda) * 100) AS porcentaje
FROM
    vivienda v
JOIN
    hogar h ON v.id_vivienda = h.id_vivienda
GROUP BY
    h.material_piso
ORDER BY
    h.material_piso ASC;
```

settings

material_piso	porcentaje
Caña	0.0378
Otro Material	0.0416
Mármol o marmeton	0.3518
Duela, parquet, tablón tratado o piso flotante	15.5866
Tierra	2.3645
Ladrillo o cemento	30.2992
Cerámica, baldosa, vinil o porcelanato	42.9614
Tabla / tablón no tratado	8.9131

Diagrama de Pastel porcentaje de tipos de piso en relacion con el porcentaje de tipo vivienda





Proporción del estado del hogar según el material

FINISHED

Estado de paredes, piso y techo según su material

Took 0 sec. Last updated by anonymous at July 26 2023, 3:32:42 PM.

FINISHED

```
val query = """
    (SELECT estado_techo, estado_piso, estado_paredes, material_piso, material_techo, material_paredes
     FROM hogar) as qryData
    """

val tempDF = spark.read
    .format("jdbc")
    .option("url", "jdbc:mysql://localhost:3306/mydb_integrador") //"jdbc:mysql://hostname:port/dbname"
    .option("driver", "com.mysql.jdbc.Driver")
    .option("user", "root")
    .option("password", "")
    .option("dbtable", query)
    .load()
```

```
query: String =
"
    (SELECT estado_techo, estado_piso, estado_paredes, material_piso, material_techo, material_paredes
     FROM hogar) as qryData
    "
tempDF: org.apache.spark.sql.DataFrame = [estado_techo: string, estado_piso: string ... 4 more fields]
```

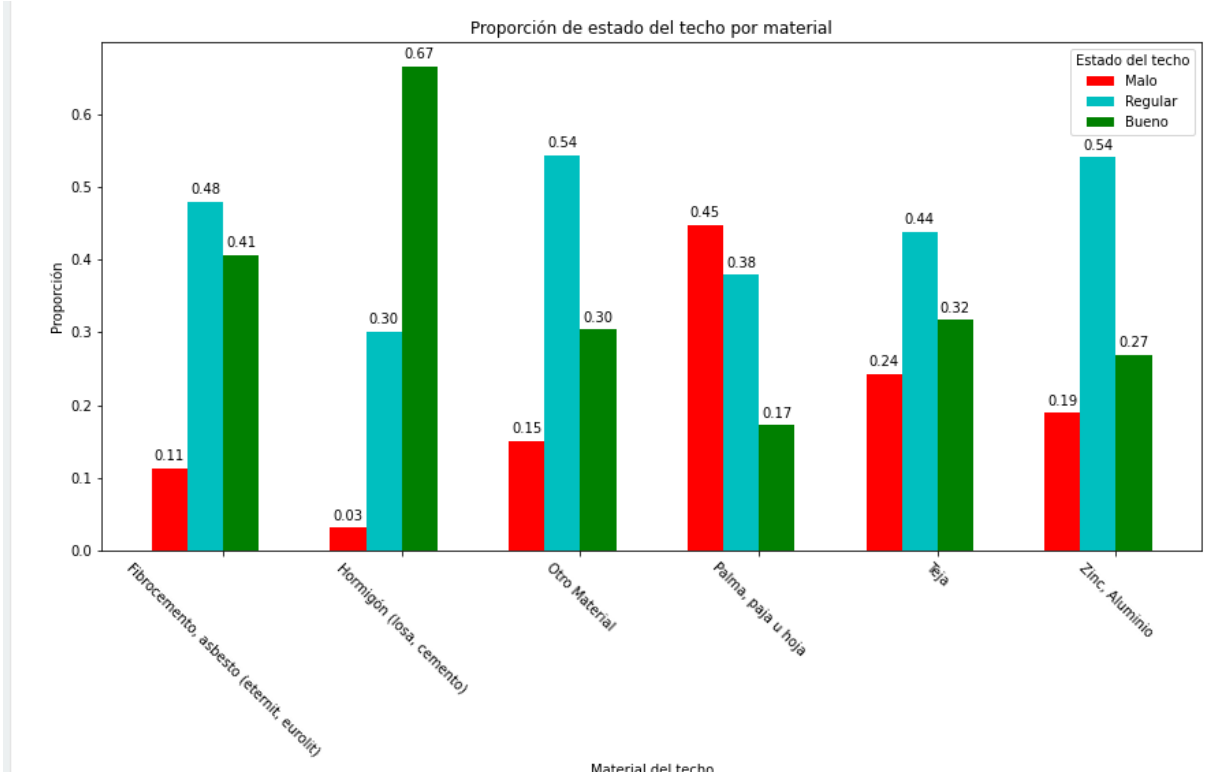
Took 1 sec. Last updated by anonymous at July 29 2023, 10:04:25 PM.

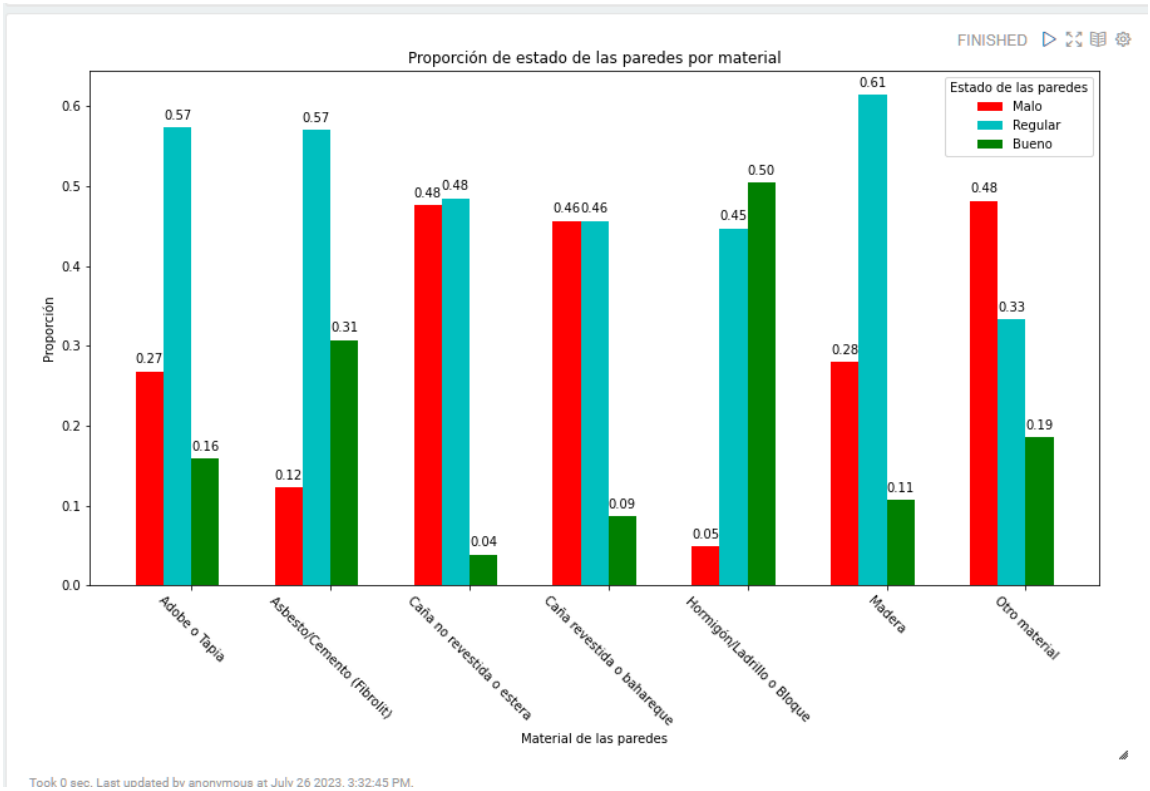
SPARK JOB FINISHED

```
z.show(tempDF.limit(5))
```

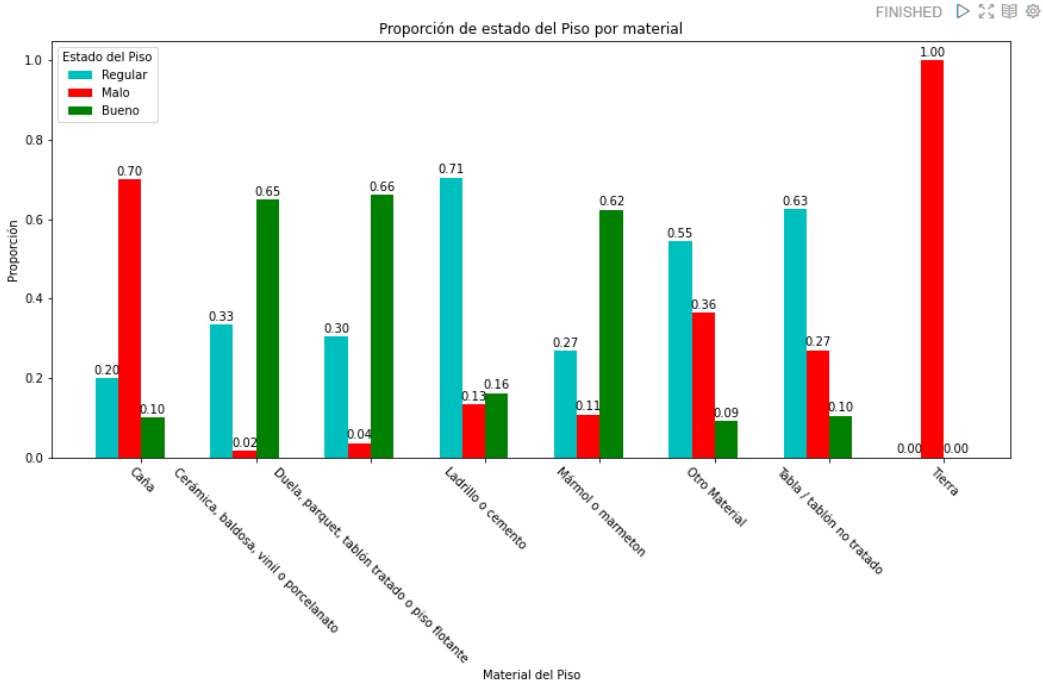
settings

estado_techo	estado_piso	estado_paredes	material_piso	material_techo	material_pared
Malo	Regular	Regular	Ladrillo o cemento	Fibrocemento, asbesto (eternit, eurolit)	Hormigón/Ladrillo Bloque
Bueno	Bueno	Bueno	Cerámica, baldosa, vinil o porcelanato	Otro Material	Hormigón/Ladrillo Bloque
Bueno	Bueno	Bueno	Duela, parquet, tablón tratado o piso flotante	Hormigón (losa, cemento)	Hormigón/Ladrillo Bloque
Malo	Bueno	Bueno	Duela, parquet, tablón tratado o piso flotante	Otro Material	Adobe o Tapia


Proporción de estado del techo por material
Estado del techo
Malo
Regular
Bueno
Proporción
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.11
0.48
0.41
0.03
0.30
0.67
0.15
0.54
0.30
0.45
0.38
0.17
0.24
0.44
0.32
0.19
0.54
0.27
Fibrocemento, asbesto (eternit, eurolit)
Hormigón (losa, cemento)
Otro Material
Palma, paja u hoja
Teja
Zinc, Aluminio
Material del techo


FINISHED
Proporción de estado de las paredes por material
Estado de las paredes
Malo
Regular
Bueno
Proporción
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.27
0.57
0.16
0.12
0.57
0.31
0.48
0.48
0.04
0.46
0.46
0.09
0.05
0.45
0.50
0.28
0.61
0.11
0.48
0.33
0.19
Adobe o Tapia
Asbesto/Cemento (Fibrolit)
Caña no revestida o estera
Caña revestida o bahareque
Hormigón/Ladrillo o Bloque
Madera
Otro material
Material de las paredes
Took 0 sec. Last updated by anonymous at July 26 2023, 3:32:45 PM.

FINISHED



FINISHED

Grafico de dispersion

tasa de desempleo - gasto de combustibles

este grafico permite identificar comportamiento atipico entre las provincias en relacion son su tasa de desempleo promedio y la relacion que tienen con el gato de combustibel.

Se encontro que Tungurahua presentan un comportamiento atipo a las demas indicando que a pesar que la tasa de desempleo es alta su gato de combustible tambien lo es a diferencia de las otras.

Esto podria indicar que tiene potencial para proyectos de inversion orientado a combustibles, ademas se podria deducir que albergan industrias estrategicas para la economia.

Ademas se infiere que las provincias con un comportamiento inusual al resto podrian ser areas de mayor crecimiento economico.

Took 0 sec. Last updated by anonymous at July 26 2023, 3:32:46 PM.

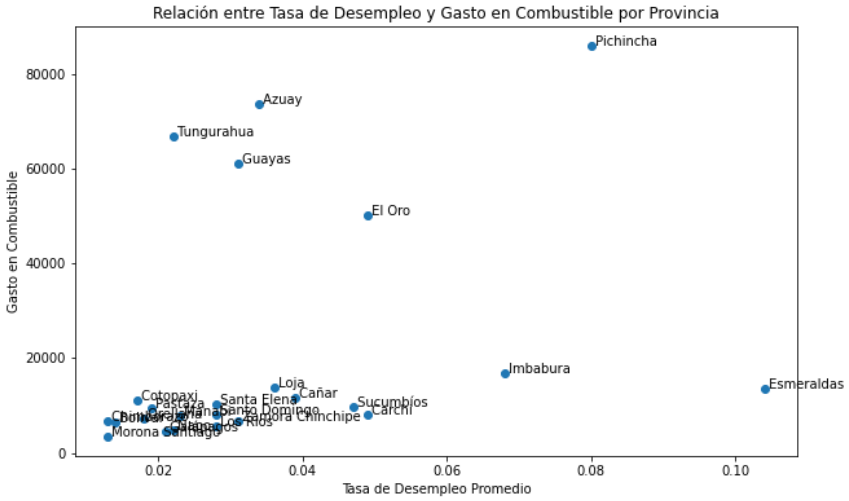
FINISHED

```
%mySQL(saveAs= result2)

SELECT
    p.nombre_provincia,
    AVG(p.tasa_desempleo) AS tasa_desempleo_promedio,
    SUM(hc.gasto_combustible) AS total_gasto_combustible
FROM
    provincia p
JOIN
    canton c ON p.cod_provincia = c.provincia_canton
JOIN
    parroquia pa ON c.cod_canton = pa.cod_canton
JOIN
    vivienda v ON pa.cod_parroquia = v.cod_parroquia
JOIN
    hogar h ON v.id_vivienda = h.id_vivienda
JOIN
    hogar_combustibles hc ON h.id_hogar = hc.id_hogar
WHERE p.nombre_provincia != ' Zonas No Delimitadas'
GROUP BY
    p.nombre_provincia;
```

settings

FINIS

```
%pyspark
import matplotlib.pyplot as plt

# Crear el gráfico de dispersión
plt.figure(figsize=(10, 6))
plt.scatter(dfdemo2['tasa_desempleo_promedio'], dfdemo2['total_gasto_combustible'])

# Etiquetas y título del gráfico
plt.xlabel("Tasa de Desempleo Promedio")
plt.ylabel("Gasto en Combustible")
plt.title("Relación entre Tasa de Desempleo y Gasto en Combustible por Provincia")

# Etiquetas para cada punto (provincia)
for i, row in dfdemo2.iterrows():
    plt.text(row['tasa_desempleo_promedio'], row['total_gasto_combustible'], row['nombre_provincia'])

# Mostrar el gráfico
plt.show()
```



Porcentaje de viviendas con servicios básicos ideales

Took 0 sec. Last updated by anonymous at July 26 2023, 3:32:46 PM.

```
%mySQL(saveAs=viviendas)
SELECT p.nombre_provincia,
       COUNT(*) AS cantidad_hogares,
       COUNT(*) * 100.0 / total_hogares_provincia.total AS porcentaje
FROM provincia p
INNER JOIN canton c ON p.cod_provincia = c.provincia_canton
INNER JOIN parroquia pa ON c.cod_canton = pa.cod_canton
INNER JOIN vivienda v ON pa.cod_parroquia = v.cod_parroquia
INNER JOIN hogar h ON v.id_vivienda = h.id_vivienda
INNER JOIN servicios_basicos sb ON h.id_hogar = sb.id_hogar
INNER JOIN (
  SELECT p.cod_provincia, COUNT(*) AS total
  FROM provincia p
  INNER JOIN canton c ON p.cod_provincia = c.provincia_canton
  INNER JOIN parroquia pa ON c.cod_canton = pa.cod_canton
  INNER JOIN vivienda v ON pa.cod_parroquia = v.cod_parroquia
  INNER JOIN hogar h ON v.id_vivienda = h.id_vivienda
  INNER JOIN servicios_basicos sb ON h.id_hogar = sb.id_hogar
  GROUP BY p.cod_provincia
) AS total_hogares_provincia ON p.cod_provincia = total_hogares_provincia.cod_provincia
WHERE sb.agua_tuberia = 'Por tubería dentro de la vivienda' AND sb.servicio_higienico = 'Excusado y alcantaril
GROUP BY p.nombre_provincia, total_hogares_provincia.total;
```

settings

nombre_provincia	cantidad_hogares	porcentaje
Azuay	2148	88.87050
Bolivar	301	61.42857
Cañar	350	71.28310
Carchi	418	71.69811
Cotopaxi	301	62.31884
Chimborazo	274	56.61157
El Oro	2356	82.98697
Esmeraldas	637	40.75496

Distribucion de gasto combustible Super

Took 0 sec. Last updated by anonymous at July 26 2023, 3:32:48 PM.

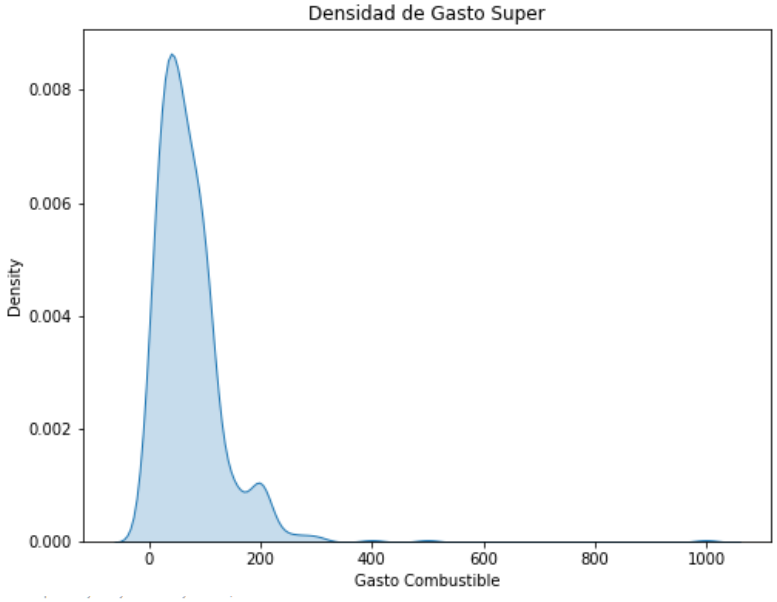
FINISHED

```
%mySQL(saveAs=gasolina)

SELECT *
FROM `mydb_integrador`.`hogar_combustibles`
WHERE `tipo_combustible` = 'super';
```

settings

id_hogar	tipo_combustible	gasto_combustible
01015000110102802101	Súper	50
01015000320302807101	Súper	25
01015000620203703101	Súper	120
01015000620203707101	Súper	50
01015000930302802101	Súper	40
01015000930302805101	Súper	30
01015001470203702101	Súper	50
01015001640203207102	Súper	40

```pyspark
%pyspark
from pyspark.sql import DataFrame
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df = DataFrame(z.get("dfAuxGasolina"), sqlContext)
pdf = df.toPandas()
# Gráfico de densidad para valor promedio de arriendo
plt.figure(figsize=(8, 6))
sns.kdeplot(data=pdf, x='gasto_combustible', fill=True)
plt.xlabel('Gasto Combustible')
plt.title('Densidad de Gasto Super')
plt.show()
```



FINISHED

Consulta de Datos de Viviendas y Hogares en la Provincia de Loja cantón Loja.

Took 0 sec. Last updated by anonymous at July 26 2023, 3:32:49 PM.

FINISHED

```mysql
%mySQL(saveAs=loja)

SELECT c.nombre_canton, p.tasa_desempleo, v.tipo_vivienda, AVG(h.valor_arriendo) AS valor_arriendo_promedio, MAX(da.luz_incluida)
    AS luz_incluida, pa.nombre_parroquia FROM provincia p INNER JOIN canton c ON p.cod_provincia = c.provincia_canton INNER JOIN
    parroquia pa ON c.cod_canton = pa.cod_canton INNER JOIN vivienda v ON pa.cod_parroquia = v.cod_parroquia INNER JOIN hogar h ON
    v.id_vivienda = h.id_vivienda LEFT JOIN datos_arriendo da ON h.id_hogar = da.id_hogar WHERE c.nombre_canton = 'Loja' GROUP BY c
    .nombre_canton, p.tasa_desempleo, v.tipo_vivienda, pa.nombre_parroquia;
```

settings

nombre_canton	tasa_desempleo	tipo_vivienda	valor_arriendo_promedio	luz_incluida	no
Loja	0.036	Mediagua	72.307692	No	Loja
Loja	0.036	Rancho, covacha	50.000000	No	Loja
Loja	0.036	Casa o Villa	104.285714	No	Jim
Loja	0.036	Casa o Villa	64.000000	No	Mal
Loja	0.036	Mediagua	30.000000	No	Mal
Loja	0.036	Cuartos en casa de inquilinato	30.000000	No	Mal
Loja	0.036	Casa o Villa	52.142857	No	Taq

Gráfico de pastel para la proporción de viviendas por tipo de vivienda"

```
%pyspark
from pyspark.sql import DataFrame
import pandas as pd
import matplotlib.pyplot as plt

# Gráfico de pastel para tipo de vivienda
plt.figure(figsize=(8, 8))
pdf['tipo_vivienda'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.axis('equal')
plt.title('Proporción de Viviendas por Tipo de Vivienda')
plt.show()
```

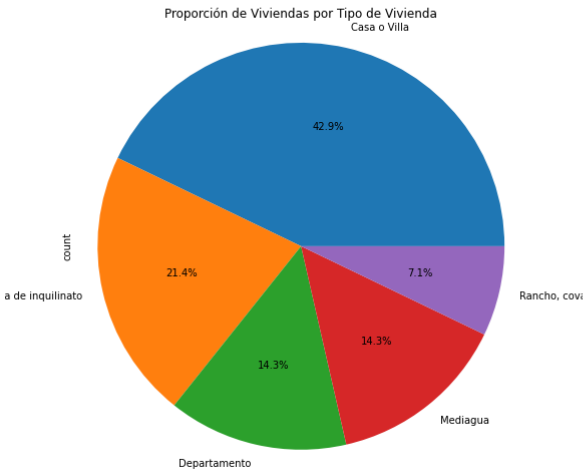
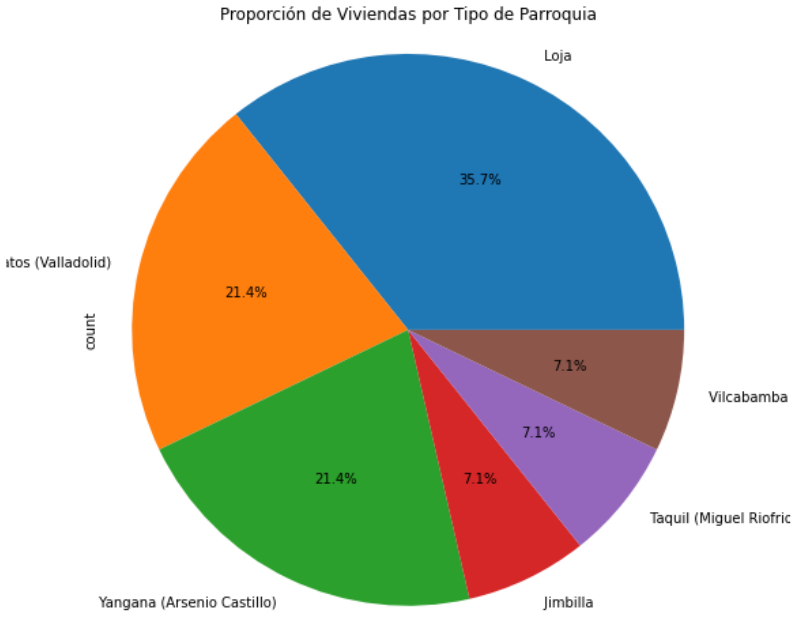


Gráfico de pastel para la proporción de viviendas por tipo de parroquia"

```
%pyspark
# Gráfico de pastel para tipo de parroquia
plt.figure(figsize=(8, 8))
pdf['nombre_parroquia'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.axis('equal')
plt.title('Proporción de Viviendas por Tipo de Parroquia')
plt.show()
```



Distribucion de los datos en valores Arriendos por parroquias de la ciudad de Loja

```
%pyspark
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt



# Gráfico de densidad para valor promedio de arriendo
plt.figure(figsize=(8, 6))
sns.kdeplot(data=pdf, x='valor_arriendo_promedio', fill=True)
plt.xlabel('Valor Arriendo Promedio')
plt.title('Densidad de Valor Arriendo Promedio')
plt.show()
```

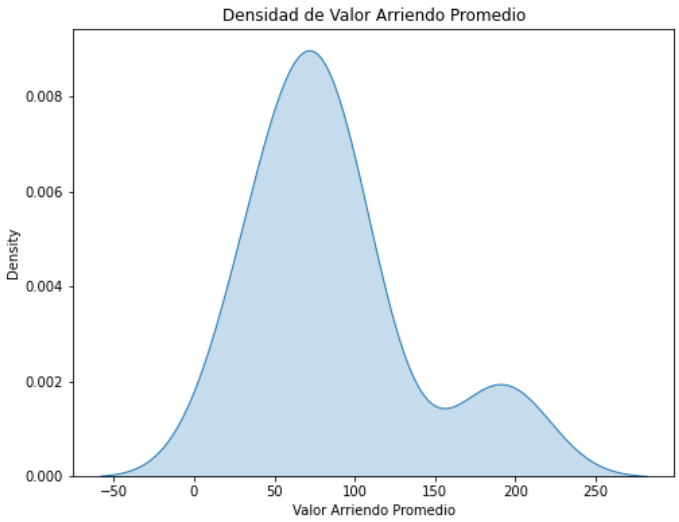
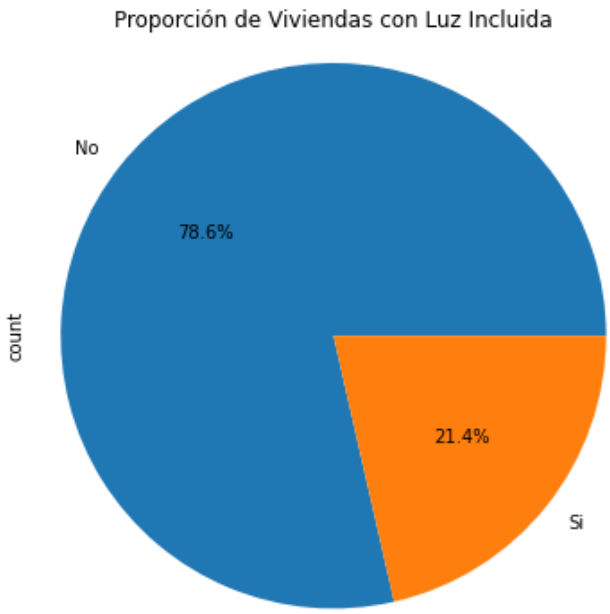
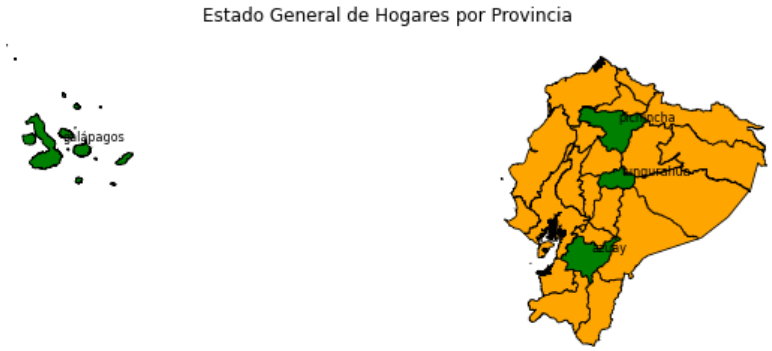


Gráfico de pastel para la proporción de viviendas que su arriendo incluye luz

```
%pyspark
# Gráfico de pastel para luz incluida o no
plt.figure(figsize=(6, 6))
pdf['luz_incluida'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.axis('equal')
plt.title('Proporción de Viviendas con Luz Incluida')
plt.show()
```

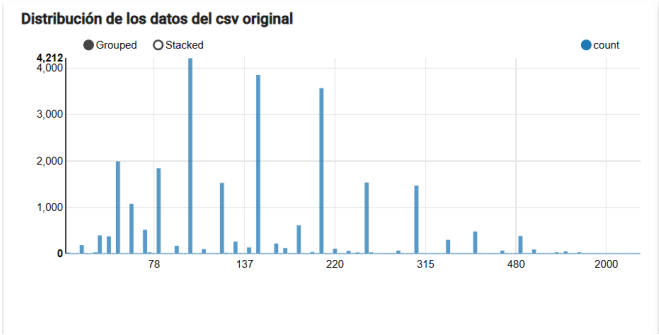


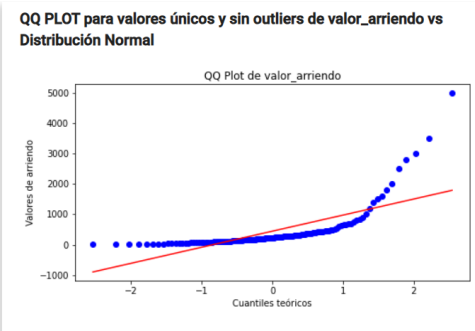
```python
def asignar_color(row):
    if row['hogares_malos'] > row['hogares_buenos'] and row['hogares_malos'] > row['hogares_regulares']:
        return 'red'     # Color rojo para 'malo'
    elif row['hogares_buenos'] > row['hogares_malos'] and row['hogares_buenos'] > row['hogares_regulares']:
        return 'green'   # Color verde para 'bueno'
    else:
        return 'orange'  # Color naranja para 'regular'

# Crear una nueva columna para asignar el color según el estado general
data_mapa['color_estado'] = data_mapa.apply(asignar_color, axis=1)

# Crear el gráfico de mapa
fig, ax = plt.subplots(1, 1, figsize=(10, 6))

# Dibujar las provincias con los colores correspondientes
data_mapa.plot(ax=ax, linewidth=0.8, edgecolor='k', facecolor=data_mapa['color_estado'], legend=True)

# Dibujar las etiquetas solo para las provincias con color verde (bueno)
for x, y, label in zip(data_mapa.geometry.centroid.x, data_mapa.geometry.centroid.y, data_mapa['provincia']):
    if data_mapa.loc[data_mapa['provincia'] == label, 'color_estado'].values[0] == 'green':
        ax.annotate(label, xy=(x, y), xytext=(3, 3), textcoords="offset points", color='black', fontsize=8)

# Personalizar el gráfico
ax.set_title('Estado General de Hogares por Provincia')
ax.set_axis_off()

# Mostrar el gráfico
plt.show()
```



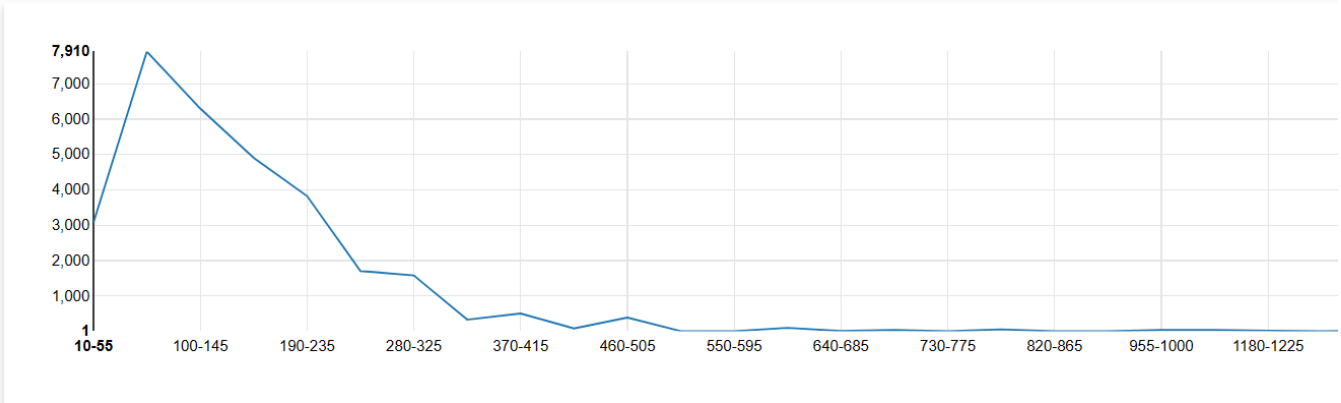
7. Resultados Obtenidos

Los datos de valor arriendo serán una de las columnas más relevantes para nuestro análisis, en estás gráficas podemos concluir que no siguen una distribución normal, además de valores relevantes como el mínimo y máximo valor presente en estos datos.

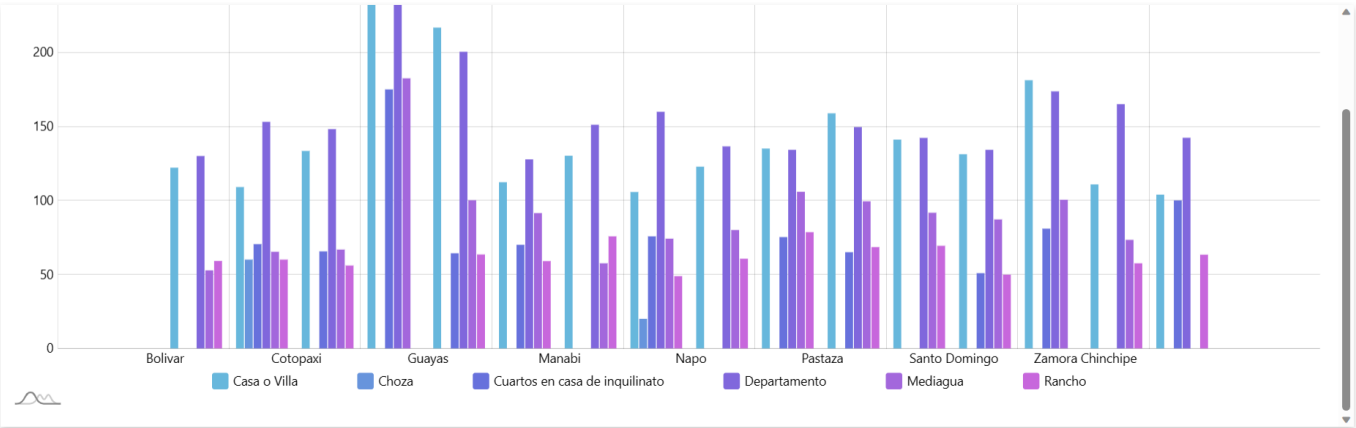




Lo más común para precios en arriendos se encuentra en el rango de $55 a $100
A partir de 100$ va disminuyendo el mercado para los arriendos.

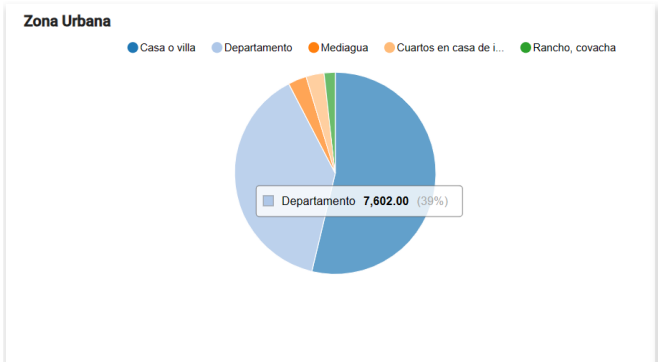


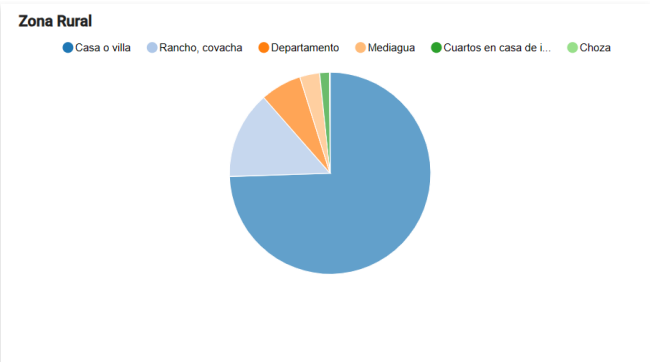
Promedio de arriendo según el tipo de vivienda en cada provincia. Podemos observar los promedios de arriendo según el tipo de vivienda (Casa, Departamento, Choza, etc)
sobre las provincias con menor desempleo en comparación al promedio de desempleo a nivel nacional.
la visualización de estos datos se encuentra en una gráfica interactiva de amcharts.



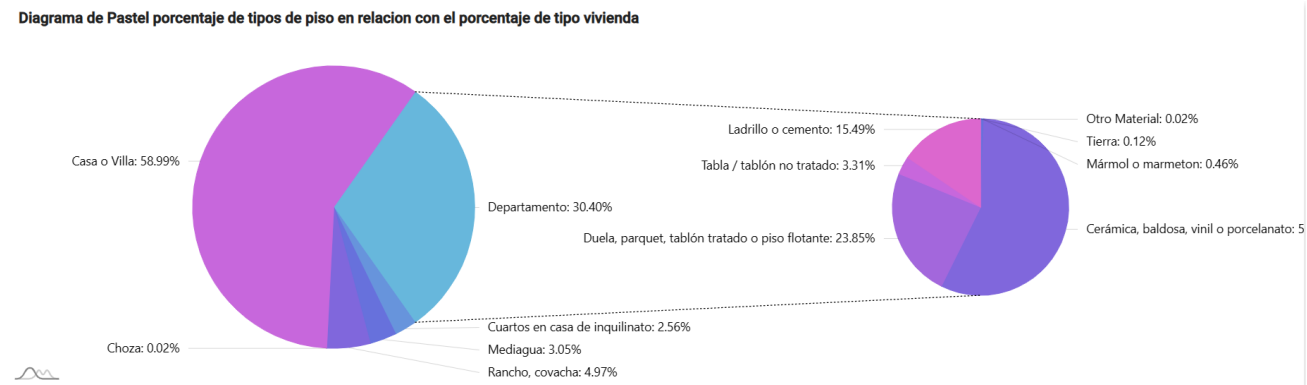
Distribución de los tipos de vivienda por zona urbana y rural.

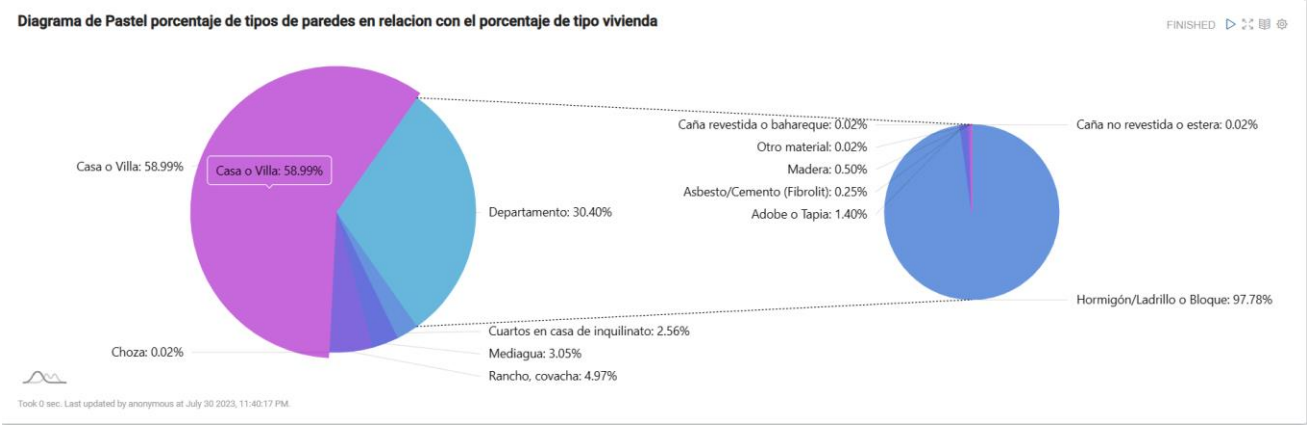
La mayor diferencia se encuentra en el porcentaje de departamentos para la zona urbana. Siendo este el segundo mayor después de las Casas. A diferencia de el rancho o covacha para la zona rural.

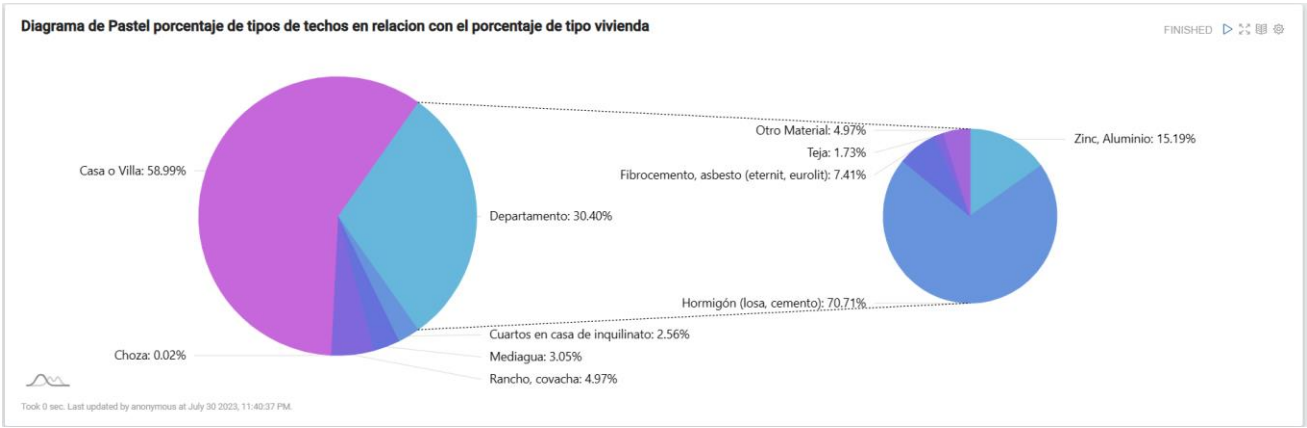




La siguiente gráfica de pastel muestra porcentajes de los materiales usados dependiendo de la vivienda en una gráfica interactiva.



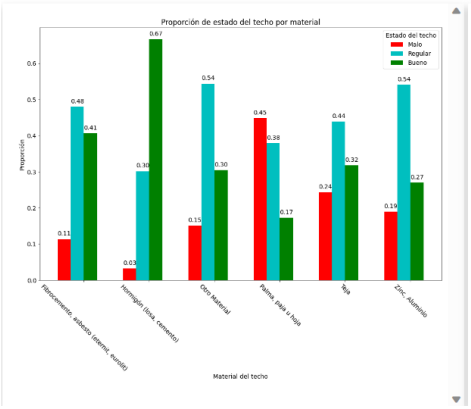


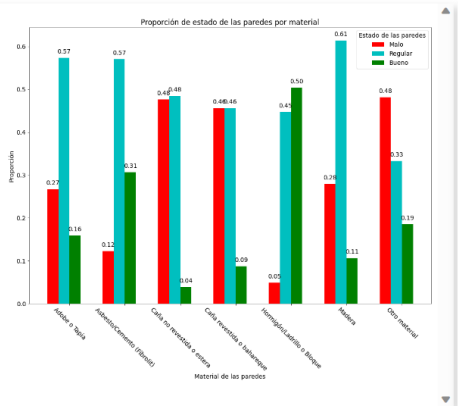


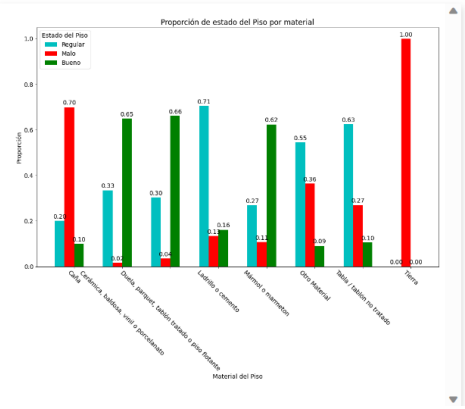
Cada material tiene una calificación de: Bueno, Regular, Malo.

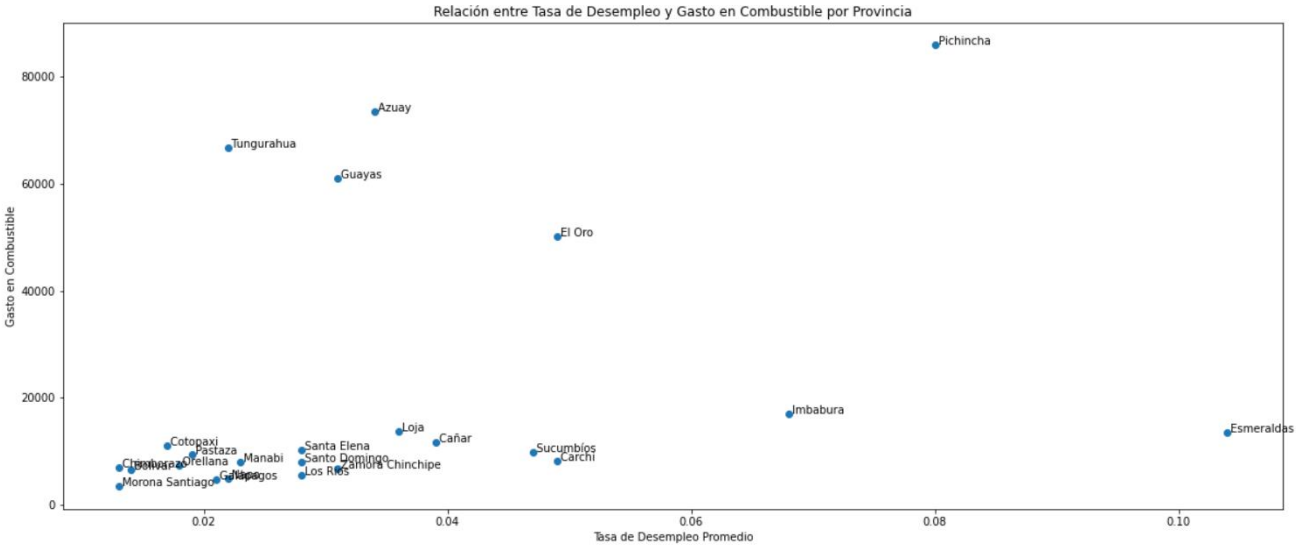
Hemos obtenido una proporción de como se encuentran los materiales en base a su cantidad a nivel nacional.

Esta gráfica ayuda a tomar decisiones sobre que material preferir para cada parte de los hogares.









8. Referencias

- Script SQL: Enlace: DDL_PR_ENEMDU_VIVIENDAS.sql en el repositorio GitHub del proyecto.
- Repositorio GitHub: Enlace: cdrodriguez7/Proyecto-Integrado-1.2
- Implementación y carga: El script de implementación y carga se divide en dos partes:
- Parte 1: complementarios_1.py
- Parte 2: principales_2.py