UNIVERSIDAD TECNOLÓGICA DE CHIHUAHUA

DESARROLLO Y GESTIÓN DE SOFTWARE

1.4. REPORTE DE INVESTIGACIÓN DE LOS LENGUAJES Y BIBLIOTECAS PARA ANÁLISIS Y PROCESAMIENTO DE DATOS

EXTRACCIÓN DE CONOCIMIENTO EN BASES DE DATOS

PRESENTA:

KARLA ALEJANDRA DE LA CRUZ ZEA

DOCENTE:

ING. LUIS ENRIQUE MASCOTE CANO

22 de septiembre de 2025

Contenido

Introducción

En proyectos que son de IA, Machine Learning , Data Mining y de Big Data, la elección del lenguaje de programación y las bibliotecas está fuertemente relacionado a la productividad, al rendimiento, a la disponibilidad de herramientas y la escalabilidad de la solución.

Conocer los lenguajes más usados y sus librerías permite seleccionar el stack adecuado según el tipo de proyecto. Además de que facilita la integración con infraestructuras, como bases de datos analíticas y reduce riesgos en la etapa de despliegue. Mostrare un resumen de los lenguajes más relevantes, también describiré sus paradigmas y bibliotecas clave, presentare ejemplos sencillos para “cargar y ver” datos concluyendo con una serie de recomendaciones para distintos escenarios.

Objetivos del reporte

1. Documentación de las características de los lenguajes más relevantes para ciencia de datos y Big Data.
2. Resumen de las bibliotecas y frameworks clave por lenguaje.
3. Proveer ejemplos sencillos y prácticos de carga de CSV y vista inicial de datos.
4. Finalmente concluir con una comparación y recomendaciones para proyectos ligeros y proyectos de producción a gran escala.

Sección por lenguaje

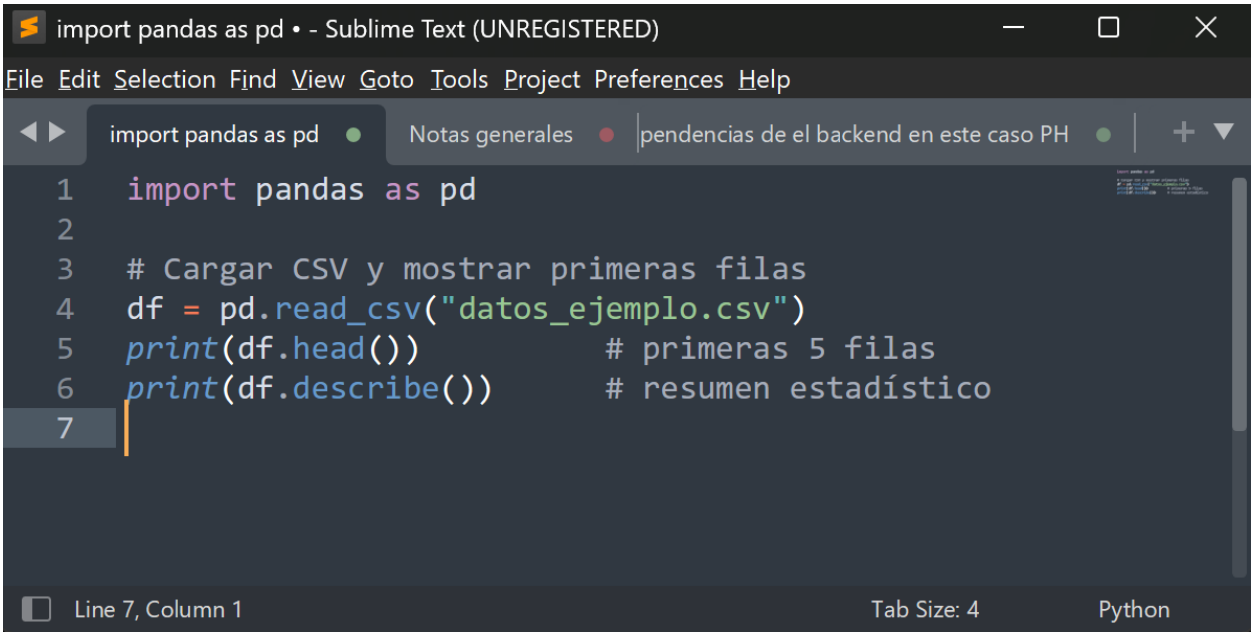
Python

Descripción general

- Paradigma: multiparadigma (imperativo, orientado a objetos, funcional en parte), interpretado.

- Tipado: dinámico, opcionalmente gradual (con type hints).
- Ámbito: data science, machine learning, prototipado rápido, APIs, ETL, integración con frameworks de producción. Python es el lenguaje dominante en ciencia de datos por su ecosistema maduro y la gran cantidad de librerías científicas. **Bibliotecas y frameworks clave**
- **pandas** — manipulación y análisis de datos tabulares (DataFrame). Caso de uso: limpieza, merge, agregaciones, preparación de features.
- **NumPy / SciPy** — operaciones numéricas y arrays eficientes; base para muchas librerías. Caso de uso: álgebra lineal, transformaciones numéricas.
- **scikit-learn** — algoritmos clásicos de ML (regresión, clasificación, clustering, preprocesado). Caso de uso: entrenar modelos supervisados y pipelines de features.
- **TensorFlow / PyTorch** — deep learning; PyTorch muy popular en investigación, TensorFlow en producción y modelos a escala.
- **Dask / PySpark** — procesamiento distribuido para DataFrames y tareas que exceden memoria local.

Ejemplo “Hola datos” (Python / pandas)



R

Descripción general

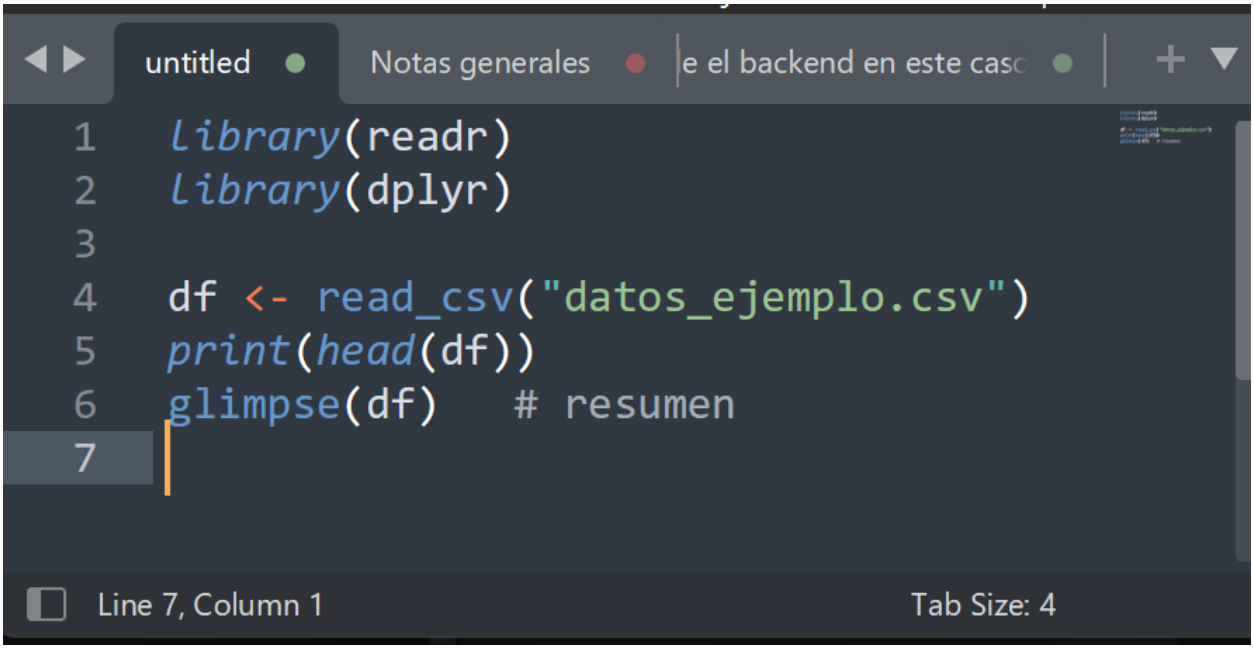
- Paradigma: interpretado; orientado a datos y funciones.
- Tipado: dinámico.
- Ámbito: estadística, visualización, análisis exploratorio y modelado (ampliamente usado en estadística académica y análisis reproducible). R ofrece un ecosistema centrado en análisis estadístico y visualización declarativa (tidyverse).

Bibliotecas y frameworks clave

- **tidyverse (dplyr, tidyr, ggplot2, readr)** — conjunto integrado para manipulación (dplyr), limpieza (tidyr), I/O (readr) y visualización (ggplot2). Caso de uso: ETL ligero, visualizaciones investigativas.

- **caret / tidymodels** — frameworks para modelado ML (preprocesado, tuning, validación).
- **data.table** — manipulación ultra-rápida de tablas (memory-efficient para grandes tablas en un solo nodo).

Ejemplo “Hola datos” (R / tidyverse)



Scala (con Apache Spark)

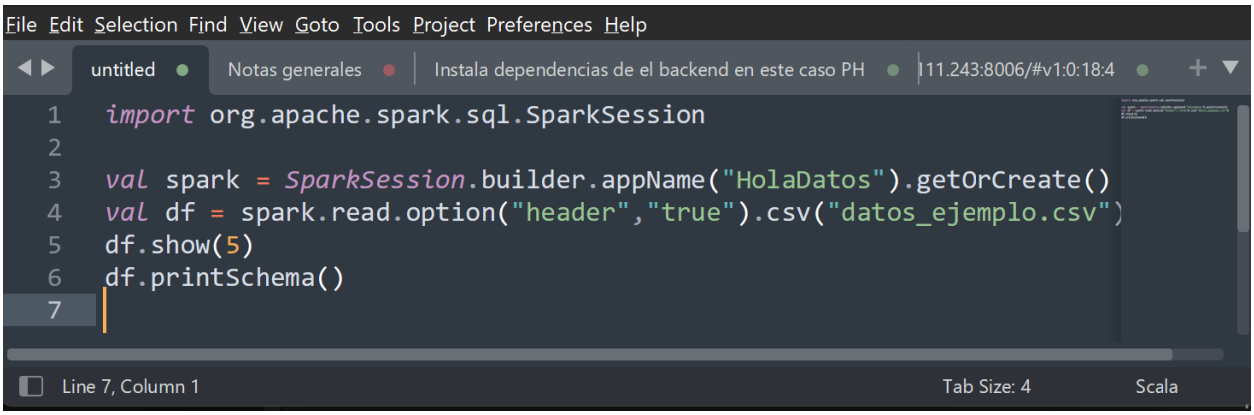
Descripción general

- Paradigma: compilado a JVM, multiparadigma (orientado a objetos + funcional).
- Tipado: estático (con tipo fuerte).
- Ámbito: procesamiento distribuido y Big Data (Spark), back-end a gran escala. Scala es el lenguaje nativo en el que se implementó Apache Spark y por tanto es extrapolado para aplicaciones de alto rendimiento en clusters.

Bibliotecas y frameworks clave

- **Apache Spark (Spark SQL, DataFrame API, Structured Streaming)** — procesamiento distribuido para ETL, consultas SQL a gran escala, streaming y MLlib para ML distribuido. Caso de uso: pipelines ETL en cluster, entrenamiento de modelos sobre grandes volúmenes (terabytes).
- **MLlib** — biblioteca ML distribuida de Spark (regresión, clasificación, clustering, pipelines).
- **Alpakka / Akka Streams** — integración y streaming asíncrono para microservicios si se requiere.

Ejemplo "Hola datos" (Scala + Spark — cargar CSV y mostrar filas)



SQL (lenguaje declarativo para datos)

Descripción general

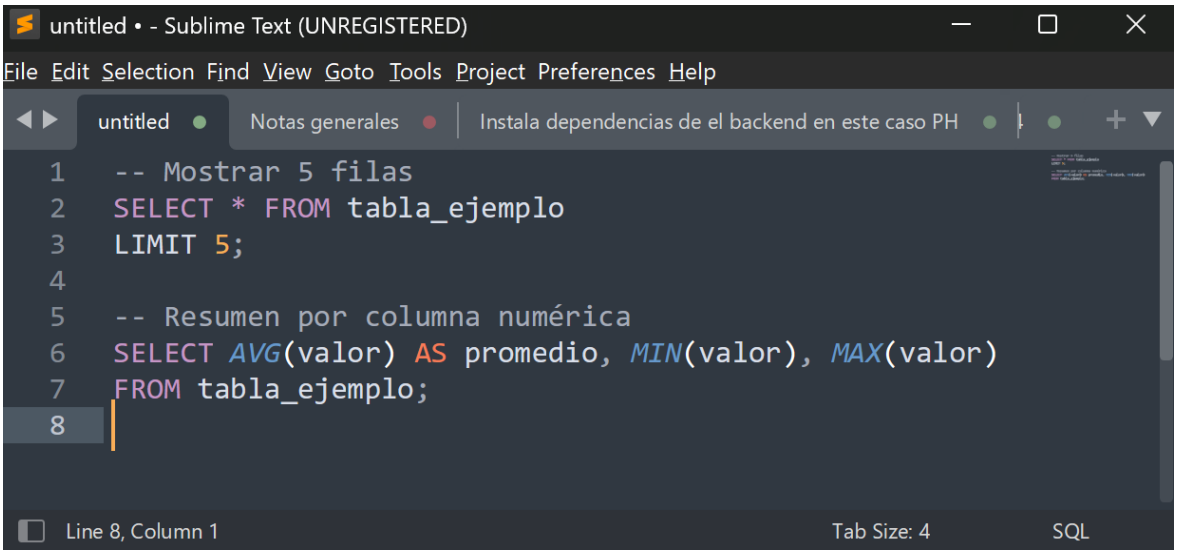
- Paradigma: declarativo (consulta), típicamente interpretado por un motor RDBMS o analítico.
- Tipado: dependiente del motor (tipado estático en columnas).
- Ámbito: consulta y transformación de datos, agregaciones, join a gran escala; usado en OLAP y data warehouses (BigQuery, Snowflake, Redshift, PostgreSQL). SQL es el lenguaje estándar para consultar datos tabulares y es crucial en pipelines de BI y analytics.

Sistemas / motores relevantes

- **PostgreSQL / MySQL / SQL Server** — bases relacionales tradicionales para OLTP/OLAP moderado.
- **BigQuery / Snowflake / Redshift** — almacenes analíticos escalables para petabytes; soportan SQL extendido y funciones analíticas. Caso de uso: análisis ad-hoc y reporting a escala.

Ejemplo "Hola datos" (SQL — lectura y primeras filas)

(Ejemplo SQL genérico — en un cliente p. ej. psql, BigQuery UI, o cualquier connector)



Julia

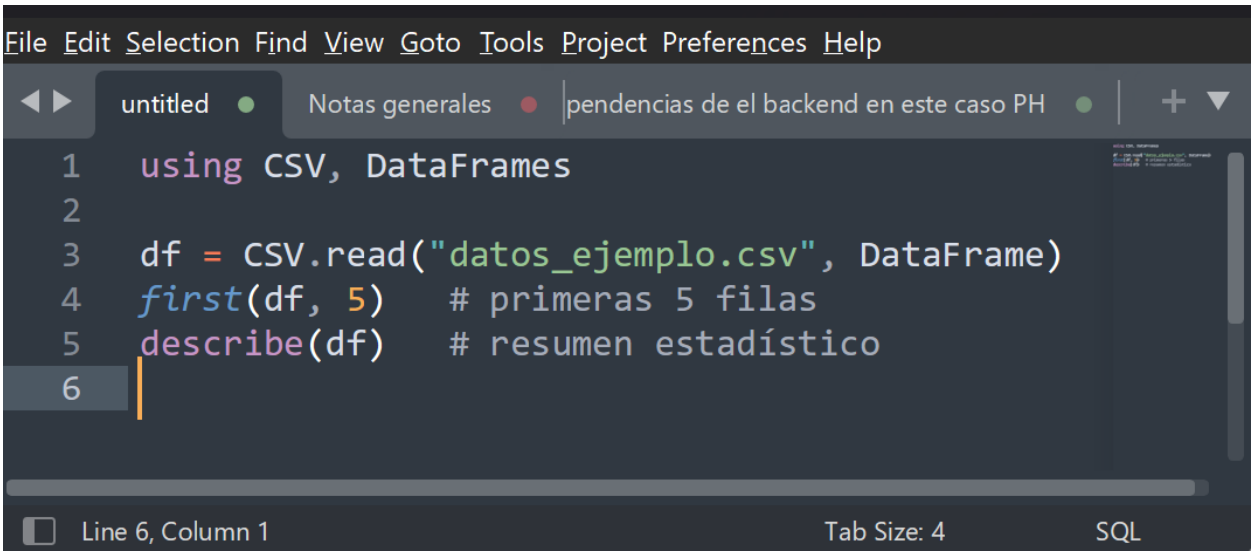
Descripción general

- Paradigma: compilado JIT (Just-In-Time) mediante LLVM; multiparadigma (procedural, funcional, orientado a objetos ligero).
- Tipado: dinámico con tipado opcional y alto rendimiento cercano a lenguajes compilados en tareas numéricas.
- Ámbito: cómputo numérico de alto rendimiento, investigación, algunos proyectos de ML y modelado científico; ecosistema en crecimiento para data science. La comunidad de Julia ha desarrollado paquetes competentes para DataFrames, CSV y ML.

Bibliotecas y frameworks clave

- **DataFrames.jl** — manipulación de datos tabulares (similar a pandas / dplyr).
- **CSV.jl** — lectura/escritura eficiente de CSVs (multithread).
- **Flux.jl / MLJ.jl** — frameworks para deep learning y machine learning en Julia. Flux para redes neuronales; MLJ ofrece interoperabilidad de modelos.

Ejemplo “Hola datos” (Julia / DataFrames + CSV)



Recomendaciones prácticas

- **Proyectos de análisis ligero / exploratorio / investigación:** Python (pandas, scikit-learn, matplotlib/seaborn) o R (tidyverse, ggplot2) por facilidad y rapidez.
- **Prototipos que luego deben ponerse en producción a gran escala:** prototipar en Python/R y, si hace falta, portar tareas pesadas a Spark (Scala/Python) o servicios en contenedores (modelo servido con TensorFlow Serving, Seldon, etc.).
- **Procesamiento distribuido / Big Data (ETL en cluster):** Apache Spark (preferiblemente con Scala o PySpark) o motores cloud-managed (BigQuery, Snowflake).
- **Cómputo numérico de alto rendimiento / investigación matemática:** Julia es una excelente opción si se espera código matemático intensivo que requiere alto rendimiento y menor brecha entre prototipo y producción numérica.

Conclusión

Al investigar los principales lenguajes y bibliotecas para análisis y procesamiento de datos se ve que cada uno tiene sus propias ventajas. Python y R son los más usados cuando se necesita analizar datos de manera rápida y hacer prototipos porque son fáciles de aprender y tienen muchas herramientas listas para usar, como pandas, scikit-learn o tidyverse. Julia también es una opción interesante cuando se requiere mucho cálculo numérico porque es rápida y sencilla, aunque todavía no tiene tantas librerías como Python o R.

Cuando se trata de manejar grandes volúmenes de datos o trabajar en sistemas más complejos, entran en juego lenguajes como Scala y SQL. Scala, junto con Apache Spark, permite procesar datos distribuidos en varios servidores, y SQL sigue siendo básico para consultar y organizar información en bases de datos. Esto muestra que, en la práctica, no basta con conocer un solo lenguaje, sino que es común combinar varios para cubrir todo el ciclo de trabajo con los datos: desde que se obtienen hasta que se analizan y presentan los resultados.

Referencias

Pandas development team. (s.f.). pandas.read_csv — pandas documentation. PyData. Recuperado de https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html pandas.pydata.org

Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (s.f.). scikit-learn: machine learning in Python. scikit-learn.org. Recuperado de https://scikit-learn.org/ scikit-learn.org

Wickham, H., & tidyverse team. (s.f.). The tidyverse — packages including dplyr and ggplot2. Tidyverse. Recuperado de https://www.tidyverse.org/packages/ tidyverse.org

Apache Software Foundation. (s.f.). Apache Spark — unified analytics engine (Spark SQL, MLlib, Structured Streaming). Recuperado de https://spark.apache.org/docs/latest/ Apache Spark

The Julia Language. (s.f.). Julia — data ecosystem (DataFrames.jl, CSV.jl, Flux.jl). Recuperado de https://julialang.org/ Julia

Google Cloud. (s.f.). BigQuery documentation. Recuperado de https://cloud.google.com/bigquery/docs Google Cloud

DataCamp. (2022). Top Python libraries for data science. Recuperado de https://www.datacamp.com/blog/top-python-libraries-for-data-science DataCamp

DataFrames.jl Developers. (s.f.). DataFrames.jl — documentation. Recuperado de https://juliadata.github.io/DataFrames.jl/stable/ JuliaData