UNIVERSIDAD NACIONAL EXPERIMENTAL DE GUAYANA
DEPARTAMENTO DE CIENCIA Y TECNOLOGÍA
COORDINACIÓN DE INGENIERÍA INFORMÁTICA
SISTEMAS DE BASES DE DATOS II
SEMESTRE: 2025-II
PROFA. CLINIA CORDERO

PROYECTO N° 2

Implementación Práctica de un Data Pipeline: Carga NoSQL con Apache Cassandra y Procesamiento Paralelo con Apache Spark.

Especificaciones del Proyecto: "Data Pipeline Escalable y Analítico"

Objetivo general

Diseñar, implementar y validar un prototipo funcional de arquitectura de datos escalable que simula la carga de grandes volúmenes de datos (NoSQL), su procesamiento distribuido (Paralelo) y su posterior almacenamiento analítico (Data Warehouse).

Requisitos Técnicos Obligatorios

Requisito Solicitado	Tecnología Implementada	Rol en el Pipeline
Base de Datos NoSQL / Escalabilidad	**Apache Cassandra**	Capa de Ingesta (OLTP). Almacena los datos crudos de las transacciones.
Tecnologías de Escalabilidad (Paralelas)	**Apache Spark (PySpark)**	Capa de Transformación (ELT). Procesa, limpia y agrega los datos en paralelo.
Componente de almacén de datos (Data Warehouse)	**ClickHouse**	Capa Analítica (OLAP). Almacena los datos agregados para la generación de informes.
Plataforma de Despliegue	**Docker Compose**	Entorno de desarrollo unificado y de bajos recursos.

Fases del Proyecto y Entregables

El proyecto se divide en 4 fases secuenciales. La entrega final debe incluir el código fuente y la documentación.

Fase 1: Configuración del Entorno y Arquitectura (Docker)

Objetivo: Levantar la infraestructura necesaria utilizando el archivo `docker-compose.yml` optimizado.

Tarea	Descripción y comando clave
1.1. Configuración local	Instalar Docker Desktop y crear la carpeta de trabajo (`proyecto_bigdata`).
1.2. Configuración YAML	Implementar el archivo `docker-compose.yml` proporcionado para levantar los 3 servicios (Cassandra, ClickHouse, Jupyter/Spark).
1.3. Despliegue	Iniciar el entorno: `docker-compose up -d`.
1.4. Validación de Conectividad	Acceda a Jupyter (http://localhost:8888) y ejecute el "Hola Mundo" de PySpark para confirmar que el conector de Cassandra funciona.
1.5. Configuración de esquemas	Ejecutar los scripts CQL (Cassandra) y SQL (ClickHouse) para crear las tablas `ventas_crudas` y `ventas_resumen`.

Fase 2: Ingesta Masiva de Datos (NoSQL - Cassandra)

Objetivo: Simular un proceso de ingesta de datos transaccionales en la base de datos NoSQL.

Tarea	Descripción del Entregable
2.1. Generador de datos	Cree un script en Python (dentro del contenedor Jupyter) que genere **100.000 registros de ventas** ficticias. Las columnas deben incluir: `id_venta`, `fecha_venta`, `id_producto`, `categoria`(ej. Electrónica, Ropa), `monto_total`, e `id_cliente`.
2.2. Inserción Masiva	Utilice el controlador de Python de Cassandra (ej. `cassandra-driver`) para insertar los 100.000 registros en la tabla `ventas_crudas` de Cassandra.
2.3. Validación de Ingesta	Conéctese vía `cqlsh` y verifique con una consulta `SELECT COUNT(*) FROM ventas_crudas;` que los 100,000 registros fueron guardados.

Fase 3: Procesamiento Paralelo y Transformación (Spark)

Objetivo: Implementar la lógica de negocio usando PySpark para transformar los datos crudos en datos analíticos.

Tarea	Descripción del Entregable
3.1. Lectura distribuida	Cree un script PySpark que lea los datos de la tabla `ventas_crudas` en Cassandra, especificando la `Partition Key( fecha_venta)` para optimizar la lectura paralela.
3.2. Lógica de Transformación	Implementar las siguientes transformaciones utilizando *DataFrames* de Spark:
	a) Agrupar los datos por `fecha_venta` y `categoria`.
	b) Calcular la **suma** del `monto_total` (Ventas Totales por Categoría).
	c) Calcular el **conteo** de `id_venta` (Cantidad de Transacciones).
3.3. Limpieza de datos	(Opcional, pero recomendado): Aplique una transformación de limpieza simple, como eliminar registros con `monto_total` nulo o negativo.

Fase 4: Carga y Consulta Analítica (Data Warehouse)

Objetivo: Cargar los datos transformados al Data Warehouse (ClickHouse) y demostrar el potencial analítico.

Tarea	Descripción del Entregable
4.1. Carga (ELT)	Desde el script de PySpark, escriba el DataFrame resultante de la Fase 3 en la tabla `dw_analitico.ventas_resumen` de ClickHouse.
4.2. Validación analítica	Conectarse al cliente de ClickHouse y ejecutar dos (2) consultas SQL analíticas que no serían posibles o serían muy lentas en Cassandra:
	a) Consulta que muestre las **10 categorías con mayor volumen de ventas** en todo el período.

	b) Consulta que calcule el **promedio de ventas diarias** por categoría.
4.3. Informe final	Documentar el tiempo de ejecución de la ingesta (Fase 2) versus el tiempo de ejecución de la consulta analítica (Fase 4.2).

Entrega del Proyecto

Cada equipo (3 integrantes) debe consignar a través de Ungevirtual un único documento (PDF) con la siguiente estructura:

1. **Diagrama de Arquitectura:** Un diagrama de bloques que muestre el flujo de datos: **Cassandra$\rightarrow$Chispa$\rightarrow$ClickHouse** .
2. **Código fuente:** Adjuntar el archivo `docker-compose.yml` y los scripts de Python (generador, Spark ETL) y SQL/CQL utilizados.
3. **Resultados y Análisis:**
    - Captura de pantalla de la validación de la conectividad (PySpark, Fase 1.4).
    - Captura de pantalla del contenido de 100.000 registros en Cassandra (Fase 2.3).
    - El código de las dos consultas analíticas de ClickHouse (Fase 4.2).
    - Breve análisis de **por qué Cassandra fue la elección correcta** para la ingesta y **por qué ClickHouse es superior a Cassandra** para las consultas analíticas.