**Enunciado:** Cargar un data set, realizar la carga y depuración del archivo OFEI1204.txt.

**Objetivo:** Entregar una tabla con las siguientes columnas:

Agente

Planta

Hora\_1

Hora\_2

Hora\_3

...

Hora\_24

**Restricción:** Solo se deben procesar los registros de tipo "D".

**Requisitos:**

Enviar la tabla resultante junto con el código utilizado.

Explicar el paso a paso del algoritmo en un archivo de texto (.doc o .pdf).

**Investigación de contexto:**

**¿Qué es un archivo OFEI?**

Es un archivo que contiene información sobre la Oferta Fija Estacional Interdiaria (OFEI). En teoría, estos archivos contienen información sobre las ofertas de energía eléctrica en un mercado mayorista. Las empresas comercializadoras de energía los utilizan para tomar decisiones sobre la compra y venta de energía.

**ANÁLISIS DEL PROBLEMA:**

1. **Complejidad del archivo:** Aunque el archivo OFEI1204.txt tiene un tamaño relativamente pequeño (66 KB), su complejidad radica en su estructura:
   1. No tiene un formato estándar.
   2. Combina diferentes tipos de estructuras.
   3. Contiene información inconsistente y valores vacíos.
2. **Procesamiento específico:** Se requiere un algoritmo que lea y procese el archivo línea por línea, separando por comas y discerniendo entre diferentes tipos de registros.

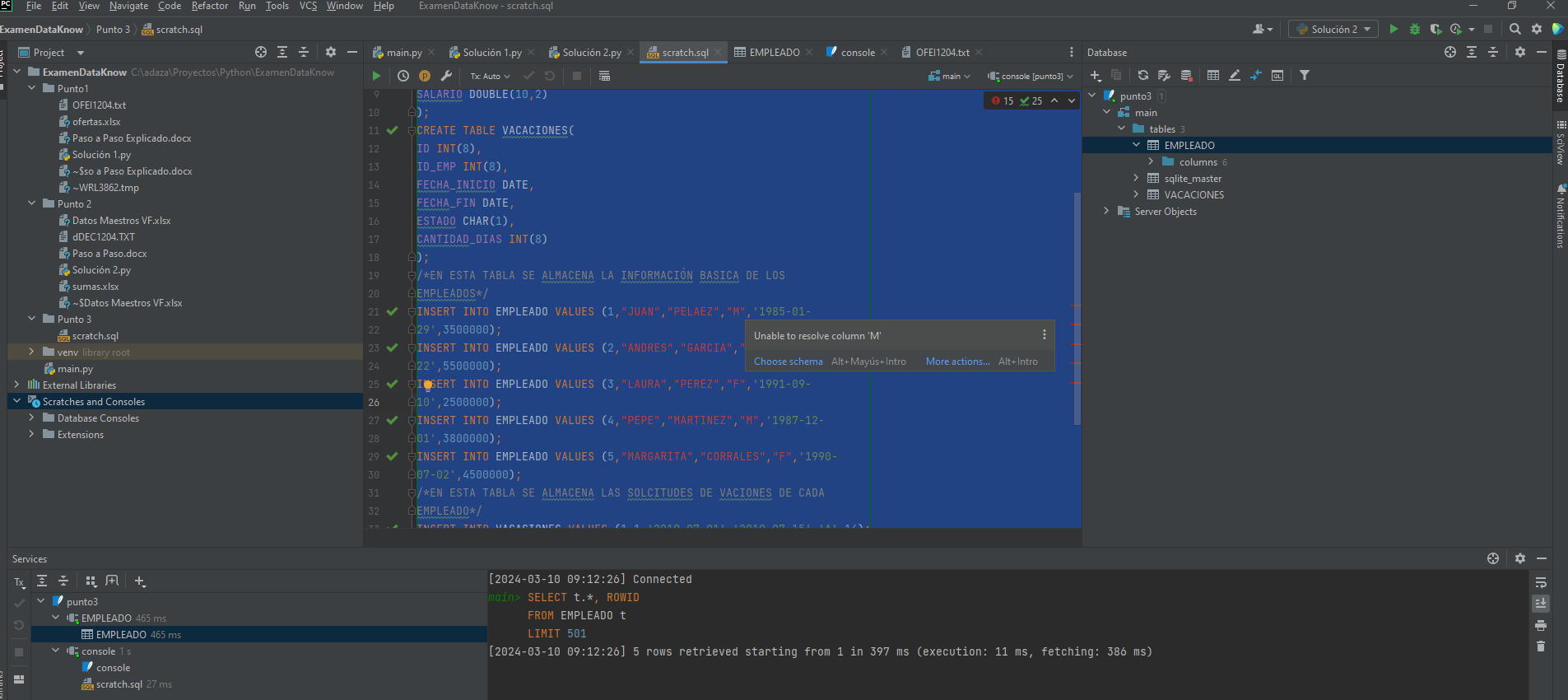
**SOLUCIÓN PROPUESTA**

Implementación de un algoritmo en Python con las siguientes características:

* Lectura y procesamiento línea por línea: El algoritmo leerá el archivo línea por línea, separando los campos por comas.
* Detección y procesamiento de registros Tipo D: El algoritmo identificará los registros de tipo "D", **que contienen las ofertas de energía**.
* Limpieza y transformación de datos: Se eliminarán valores inconsistentes y vacíos, normalizando los datos para su análisis.
* Creación de la tabla: Se generará una tabla con las columnas: Agente, Planta, Hora\_1 a Hora\_24, con las ofertas de energía por hora.
* Validación del contenido: Se verificará la integridad y consistencia de la información en la tabla.
* Se documentará el código con comentarios explicativos para facilitar su comprensión.

**EXPLICACIÓN DETALLADA Punto 3**

**Todo ha sido en Jetbrains así que vamos sqlite 😉**

****

/\* 1. Seleccione nombre, apellido y salario de todos los empleados.\*/  
SELECT ID, NOMBRE, APELLIDO, SALARIO FROM EMPLEADO;  
  
/\* 2. Seleccione nombre, apellido y salario de todos los empleados que ganen más de 4 millones.\*/  
SELECT ID, NOMBRE, APELLIDO, SALARIO FROM EMPLEADO WHERE SALARIO > 4000000;  
  
/\* 3. Cuente los empleados por sexo.\*/  
SELECT ID, SEXO, *COUNT*(\*) AS CANTIDAD FROM EMPLEADO GROUP BY SEXO;  
  
/\* 4. Seleccione los empleados que no han hecho solicitud de vacaciones.\*/  
SELECT E.ID, E.NOMBRE, E.APELLIDO FROM EMPLEADO E WHERE NOT *EXISTS*( SELECT 1 FROM VACACIONES V WHERE E.ID = V.ID\_EMP);  
  
/\* 4.1 En caso que no exista la funcionalidad NOT EXISTS\*/  
SELECT E.ID, E.NOMBRE, E.APELLIDO FROM EMPLEADO E LEFT JOIN VACACIONES V ON E.ID = V.ID\_EMP WHERE V.ID\_EMP IS NULL;  
  
  
  
/\* 5. Seleccione los empleados que tengan más de una solicitud de vacaciones y muestre cuantas solicitudes tienen los que cumplen.\*/  
SELECT E.ID, E.NOMBRE, E.APELLIDO, *COUNT*(V.ID) AS NUM\_SOLICITUDES  
FROM EMPLEADO E  
JOIN VACACIONES V ON E.ID = V.ID\_EMP  
GROUP BY E.ID  
HAVING *COUNT*(V.ID) > 1;  
  
/\* 6. Determine el salario promedio de los empleados.\*/  
SELECT *AVG*(SALARIO) AS SALARIO\_PROMEDIO FROM EMPLEADO;  
  
/\* 7. Determine la cantidad de días promedio solicitados de vacaciones por cada empleado.\*/  
/\* Existen diferentes variantes para los diferentes motores  
 https://www.w3schools.com/sql/sql\_isnull.asp \*/  
SELECT E.ID, E.NOMBRE, E.APELLIDO, *coalesce*(*AVG*(V.CANTIDAD\_DIAS), 0) AS PROMEDIO\_DIAS  
FROM EMPLEADO E  
LEFT JOIN VACACIONES V ON E.ID = V.ID\_EMP  
GROUP BY E.ID;  
  
/\* 8. Seleccione el empleado que mayor cantidad de días de vacaciones ha solicitado, muestre el nombre, apellido y cantidad de días totales solicitados.\*/  
SELECT E.NOMBRE, E.APELLIDO, *SUM*(V.CANTIDAD\_DIAS) AS TOTAL\_DIAS  
FROM EMPLEADO E  
JOIN VACACIONES V ON E.ID = V.ID\_EMP  
GROUP BY E.ID  
ORDER BY *SUM*(V.CANTIDAD\_DIAS) DESC  
LIMIT 1;  
  
/\* Si no existe un LIMIT por ejemplo con Oracle y rownum ;) hago tambien este que deberia funcionar en todas las BD \*/  
SELECT E.NOMBRE, E.APELLIDO, *SUM*(V.CANTIDAD\_DIAS) AS TOTAL\_DIAS  
FROM EMPLEADO E  
JOIN VACACIONES V ON E.ID = V.ID\_EMP  
GROUP BY E.ID  
HAVING *SUM*(V.CANTIDAD\_DIAS) = (  
 SELECT *MAX*(TOTAL\_DIAS)  
 FROM (  
 SELECT *SUM*(V.CANTIDAD\_DIAS) AS TOTAL\_DIAS  
 FROM VACACIONES V  
 GROUP BY V.ID\_EMP  
 ) AS SUBQUERY  
);  
  
/\* 9. Consulte la cantidad de días aprobados y rechazados por cada empleado, en caso de no tener solicitudes mostrar 0.\*/  
SELECT E.NOMBRE, E.APELLIDO,  
*COALESCE*(*SUM*(CASE WHEN V.ESTADO = 'A' THEN V.CANTIDAD\_DIAS ELSE 0 END), 0) AS DIAS\_APROBADOS,  
*COALESCE*(*SUM*(CASE WHEN V.ESTADO = 'R' THEN V.CANTIDAD\_DIAS ELSE 0 END), 0) AS DIAS\_RECHAZADOS  
FROM EMPLEADO E  
LEFT JOIN VACACIONES V ON E.ID = V.ID\_EMP  
GROUP BY E.ID;

main> SELECT ID, NOMBRE, APELLIDO, SALARIO FROM EMPLEADO

[2024-03-10 10:18:57] 5 rows retrieved starting from 1 in 174 ms (execution: 16 ms, fetching: 158 ms)

main> SELECT ID, NOMBRE, APELLIDO, SALARIO FROM EMPLEADO WHERE SALARIO > 4000000

[2024-03-10 10:18:58] 2 rows retrieved starting from 1 in 207 ms (execution: 4 ms, fetching: 203 ms)

main> SELECT ID, SEXO, COUNT(\*) AS CANTIDAD FROM EMPLEADO GROUP BY SEXO

[2024-03-10 10:18:58] 2 rows retrieved starting from 1 in 212 ms (execution: 34 ms, fetching: 178 ms)

main> SELECT E.ID, E.NOMBRE, E.APELLIDO FROM EMPLEADO E WHERE NOT EXISTS( SELECT 1 FROM VACACIONES V WHERE E.ID = V.ID\_EMP)

[2024-03-10 10:18:58] 2 rows retrieved starting from 1 in 201 ms (execution: 27 ms, fetching: 174 ms)

main> SELECT E.ID, E.NOMBRE, E.APELLIDO FROM EMPLEADO E LEFT JOIN VACACIONES V ON E.ID = V.ID\_EMP WHERE V.ID\_EMP IS NULL

[2024-03-10 10:18:58] 2 rows retrieved starting from 1 in 206 ms (execution: 46 ms, fetching: 160 ms)

main> SELECT E.ID, E.NOMBRE, E.APELLIDO, COUNT(V.ID) AS NUM\_SOLICITUDES

FROM EMPLEADO E

JOIN VACACIONES V ON E.ID = V.ID\_EMP

GROUP BY E.ID

HAVING COUNT(V.ID) > 1

[2024-03-10 10:18:58] 2 rows retrieved starting from 1 in 141 ms (execution: 15 ms, fetching: 126 ms)

main> SELECT AVG(SALARIO) AS SALARIO\_PROMEDIO FROM EMPLEADO

[2024-03-10 10:18:59] 1 row retrieved starting from 1 in 190 ms (execution: 30 ms, fetching: 160 ms)

main> SELECT E.ID, E.NOMBRE, E.APELLIDO, coalesce(AVG(V.CANTIDAD\_DIAS), 0) AS PROMEDIO\_DIAS

FROM EMPLEADO E

LEFT JOIN VACACIONES V ON E.ID = V.ID\_EMP

GROUP BY E.ID

[2024-03-10 10:18:59] 5 rows retrieved starting from 1 in 221 ms (execution: 0 ms, fetching: 221 ms)

main> SELECT E.NOMBRE, E.APELLIDO, SUM(V.CANTIDAD\_DIAS) AS TOTAL\_DIAS

FROM EMPLEADO E

JOIN VACACIONES V ON E.ID = V.ID\_EMP

GROUP BY E.ID

ORDER BY SUM(V.CANTIDAD\_DIAS) DESC

LIMIT 1

[2024-03-10 10:18:59] 1 row retrieved starting from 1 in 186 ms (execution: 6 ms, fetching: 180 ms)

main> SELECT E.NOMBRE, E.APELLIDO, SUM(V.CANTIDAD\_DIAS) AS TOTAL\_DIAS

FROM EMPLEADO E

JOIN VACACIONES V ON E.ID = V.ID\_EMP

GROUP BY E.ID

HAVING SUM(V.CANTIDAD\_DIAS) = (

SELECT MAX(TOTAL\_DIAS)

FROM (

SELECT SUM(V.CANTIDAD\_DIAS) AS TOTAL\_DIAS

FROM VACACIONES V

GROUP BY V.ID\_EMP

) AS SUBQUERY

)

[2024-03-10 10:19:00] 1 row retrieved starting from 1 in 233 ms (execution: 24 ms, fetching: 209 ms)

main> SELECT E.NOMBRE, E.APELLIDO,

COALESCE(SUM(CASE WHEN V.ESTADO = 'A' THEN V.CANTIDAD\_DIAS ELSE 0 END), 0) AS DIAS\_APROBADOS,

COALESCE(SUM(CASE WHEN V.ESTADO = 'R' THEN V.CANTIDAD\_DIAS ELSE 0 END), 0) AS DIAS\_RECHAZADOS

FROM EMPLEADO E

LEFT JOIN VACACIONES V ON E.ID = V.ID\_EMP

GROUP BY E.ID

[2024-03-10 10:19:00] 5 rows retrieved starting from 1 in 157 ms (execution: 14 ms, fetching: 143 ms)