Sistema BI sobre el clima de Navarra

Autor: Lucas Fortún.

NIA: 140006

Asignatura: Análisis de aplicaciones empresariales.

Índice

Proceso ETL

Lo primero indicar que he realizado este proceso mediante un programa ad-hoc en Python y que todo el código que he utilizado y al que voy hacer referencia esta adjuntado en esta entrega en: SistemaBI.ipynb.
Este proceso se divide en tres fases clave: extracción, transformación y carga.
Previamente a la extracción, he creado un diccionario con las estaciones meteorológicas automáticas, de las cuales voy a extraer los datos. En este diccionario, almaceno el ID de la estación meteorológica, el nombre del pueblo, la organización propietaria de la estación y la comarca en la que se encuentra. Posteriormente, he generado un archivo .txt, donde cada fila contiene información de una estación, separada por el delimitador ;. Este archivo me permitirá construir las URLs necesarias para descargar los datos de las estaciones desde la página web:
http://meteo.navarra.es/estaciones/mapadeestaciones.cfm

A continuación, explico cada una de estas fases de manera detallada.

- **Extracción (Extract)**

  La extracción es el primer paso del proceso ETL y se refiere a la obtención de datos desde diversas fuentes. En este caso, los datos iniciales sobre los pueblos de Navarra, como el nombre del pueblo, la estación meteorológica y la comarca, están almacenados en un archivo de texto que he comentado anteriormente.

  El proceso de extracción para este caso consiste en descargar los CSV con los datos de cada estación desde la página web mencionada. Para ello, los datos de los pueblos se extraen de un archivo pueblos_completos.txt, que contiene información sobre distintos pueblos y su estación meteorológica asociada. Este archivo de texto es leído línea por línea, extrayendo los campos de cada pueblo: ID, nombre, estación y comarca.

  Flujo de la extracción:

  1. **Lectura de archivo**: Abro el archivo pueblos_completos.txt en modo lectura. Cada línea contiene datos separados por punto y coma (**;**), y cada pueblo tiene una línea con su respectiva información.

  2. **Recopilación de datos**: Para cada pueblo, extraigo el ID, nombre, estación y comarca. Cada uno de estos valores se obtiene mediante funciones de lectura que identifican el delimitador **;** y extraen los valores entre ellos. Esta extracción de datos es crucial para poder acceder a la información necesaria en el siguiente paso del proceso.

  3. **Normalización de la estación**: El nombre de la estación meteorológica se ajusta según el año de interés. Si el año es menor a 2019 y la estación es "MAPA", el nombre de la estación se transforma a "mapama". De lo contrario, si el año es posterior a 2019, el nombre de la estación se ajusta a **"mapa"**. Esta transformación es necesaria y que a partir 2019 la web cambio el nombre de la organización "mapama" a "mapa", cambiando con ello el nombre de los csv.

4. **Generación de URL**: A partir de la información de cada pueblo (ID, nombre, estación y año), construyo una URL específica para descargar los archivos CSV con los datos meteorológicos. Este paso implica codificar correctamente los espacios en los nombres de los pueblos (reemplazándolos por %20) y asegurarme de que la URL apunte al archivo correcto.

5. **Descarga del archivo CSV**: Una vez que he construido la URL adecuada para cada estación meteorológica, procedo a descargar el archivo CSV que contiene los datos meteorológicos correspondientes al pueblo y al año determinado. Para realizar esta descarga, utilizo una solicitud HTTP con la librería requests. El archivo descargado se guarda en un directorio adecuado, dentro de una carpeta organizada por el nombre del pueblo y el año de los datos. Esta estructura jerárquica facilita la organización y el posterior acceso a los archivos.

6. **Verificación del archivo descargado**: Después de completar la descarga del archivo CSV, realizo una verificación para asegurarme de que el archivo descargado tiene el contenido esperado. Esto implica comprobar que el archivo contiene el encabezado adecuado y que los datos están correctamente estructurados. Si el archivo no cumple con los requisitos, como por ejemplo si está vacío o si no tiene los datos esperados, el archivo es eliminado para evitar que se almacenen datos incorrectos o incompletos en el sistema. Esto es útil porque algunas estaciones no tienen csv para ciertos años.

Este paso se completa cuando los archivos CSV se guardan correctamente en el sistema de archivos local, organizados según el nombre del pueblo y el año. En este caso, los archivos se almacenan en una carpeta llamada "datos" para asegurar que los datos sean fácilmente accesibles y puedan ser utilizados para su análisis posterior.

- **Transformación (Transform)**

  La fase de transformación implica adaptar y procesar los datos extraídos para que sean útiles en el análisis y estén listos para ser cargados en el sistema de destino, en este caso, una base de datos MySQL. He realizado diversas transformaciones con el objetivo de limpiar, normalizar y estructurar los datos meteorológicos. Entre las más importantes:

  - En caso de que alguno de los campos relevantes, como la temperatura máxima, mínima, media o precipitaciones, esté vacío o no aparezca en el archivo CSV, he asignado el valor None para permitir su inserción en la base de datos.

  - Al extraer las fechas, separo los días, meses y años, y luego los vuelvo a unir en el formato adecuado para su carga en la base de datos.

- **Carga (Load)**

  La fase de carga se refiere al proceso en el que se insertan los datos transformados en el sistema de destino, en este caso una base de datos mysql. Los pasos que sigo en la carga solo los siguientes:

  1. **Inicializo variables y establezco la conexión con la base de datos**

     Comienzo estableciendo la conexión con la base de datos y creo un cursor para realizar operaciones SQL. Abro el archivo pueblos_completos.txt, que contiene los datos de las estaciones meteorológicas. Defino variables auxiliares, como x para controlar la lectura del archivo y contador para contar los registros procesados. También establezco fin como una cadena vacía en formato de bytes para marcar el final de las filas.

  2. **Leo el archivo de pueblos**

     Recorro el archivo línea por línea. En cada iteración, utilizo funciones específicas para extraer información clave: leerId para obtener el identificador de la estación, leerPueblo para el nombre del pueblo, leerEstacion para la estación, y leerComarca para la comarca. Esta información me sirve para construir la ruta a los archivos de datos.

  3. **Verifico la existencia de archivos de datos**

     Genero la ruta al archivo correspondiente a cada año entre 2006 y 2024. Compruebo si el archivo existe utilizando os.path.exists. Si existe, procedo a abrirlo en modo binario.

  4. **Proceso los datos del archivo CSV**

     Dentro del archivo, salto la primera fila que contiene los encabezados con la función saltarFila. Luego, comienzo a recorrer las filas del archivo una a una:

     I. Leo la fecha y la hora: Extraigo la información de la fecha con leerFecha y omito la hora con saltarHora.
     II. Proceso las temperaturas: Leo las temperaturas máxima, mínima y media con la función leerTemperatura. En 2020 no se registraron las temperaturas medias, por ello si el año es 2020 no lee la temperatura media ya que estaría leyendo otro campo, así que asigno None a la temperatura media para 2020
     III. Salto la humedad: La humedad no me interesa asi que salto esos campos. En 2020 no se registró humedad media por ello no salto el campo ya que si no estaría saltando el campo precipitaciones que si me interesa.
     IV. Registro precipitación: Obtengo los valores correspondientes. En algunos casos, la precipitación es el último campo registrado. En esos casos x será fin de línea, y por ello no hará falta saltar los datos, ya que ya se avanzará de línea con la lectura.

5. **Busco el identificador de la fecha**

Para cada fila, ejecuto una consulta en la tabla LK_tiempo para encontrar el identificador correspondiente a la fecha (ID_tiempo).

6. **Inserto los datos en la base de datos**

Inserto los valores procesados en la tabla DT_temperaturas utilizando una consulta preparada que acepta valores nulos. Esto me permite reflejar correctamente los campos donde no hay datos disponibles, como he comentado anteriormente.

7. **Controlo el flujo del archivo y gestiono el final del procesamiento**

Avanzo en el archivo CSV utilizando read y verifico si he llegado al final del archivo. Si estoy en la última fila del último día del año (31 de diciembre), salgo del bucle.

8. **Finalizo el proceso**

Cierro tanto el archivo de pueblos como la conexión con la base de datos para liberar recursos. Finalmente, imprimo el valor del contador para verificar cuántos registros he procesado en total.

Data Warehouse

Mi objetivo ha sido estructurar y organizar la información de temperaturas y precipitaciones de manera que se facilite su consulta y análisis.

La base de datos sigue un esquema en estrella. Este diseño consta de una tabla central de hechos, donde se almacenan los datos numéricos clave, y varias tablas de dimensiones, que ofrecen contexto descriptivo para clasificar y detallar la información.

A continuación, detallo la estructura de la base de datos.

- **Tablas LK (Tablas de Dimensiones)**

  LK_comarcas

  - **ID_comarca**: Identificador único de la comarca. Generado automáticamente.
    Origen: Asignado al insertar una nueva comarca.
  - **DS_comarca**: Nombre descriptivo de la comarca.
    Origen: Obtenido de https://es.wikipedia.org/wiki/Comarcas_de_Navarra.

  LK_pueblos

  - **ID_pueblo**: Identificador único del pueblo. Generado automáticamente.
    Origen: Asignado al insertar un nuevo pueblo.
  - **DS_pueblo**: Nombre descriptivo del pueblo.
    Origen: Extraído del nombre de los csv extraídos de http://meteo.navarra.es/.
  - **ID_comarca**: Identificador único de la comarca a la que pertenece el pueblo.
    Origen: Relación con la tabla LK_comarcas, referenciando ID_comarca.

  LK_tiempo

  - **ID_tiempo**: Identificador único del tiempo. Generado automáticamente.
    Origen: Asignado al insertar un nuevo registro de tiempo.
  - **DS_fecha**: Fecha específica del registro (año-mes-día).
    Origen: Extraídas del campo fecha de los csv obtenidos en http://meteo.navarra.es/.
  - **DS_dia**: Día del mes asociado a la fecha.
    Origen: Derivado de DS_fecha.
  - **DS_mes**: Mes asociado a la fecha.
    Origen: Derivado de DS_fecha.
  - **DS_anio**: Año asociado a la fecha.
    Origen: Derivado de DS_fecha.
  - **DS_semana**: Semana asociada a la fecha.
    Origen: Derivado de DS_fecha.

- **Tablas DT (Tablas de Hechos)**

  DT_temperaturas

  - **ID_temperatura**: Identificador único del registro de temperatura. Generado automáticamente.
    Origen: Asignado al insertar un nuevo registro de temperatura.
  - **ID_pueblo**: Identificador del pueblo al que corresponde el registro de temperatura.
    Origen: Relación con la tabla LK_pueblos, referenciando ID_pueblo.
  - **ID_fecha**: Identificador de la fecha del registro de temperatura.
    Origen: Relación con la tabla LK_tiempo, referenciando ID_tiempo.
  - **IND_tmin**: Temperatura mínima registrada en el día.
    Origen: Obtenido de la columna Temperatura mínima ºC de los csv obtenidos en http://meteo.navarra.es/.
  - **IND_tmed**: Temperatura media registrada en el día.
    Origen: Obtenido de la columna Temperatura media ºC de los csv obtenidos en http://meteo.navarra.es/.
  - **IND_tmax**: Temperatura máxima registrada en el día.
    Origen: Obtenido de la columna Temperatura máxima ºC de los csv obtenidos en http://meteo.navarra.es/.
  - **IND_precipitacion**: Precipitación registrada en milímetros en el día.
    Origen: Obtenido de la columna Precipitación acumulada l/m^2 de los csv obtenidos en http://meteo.navarra.es/.

Informes

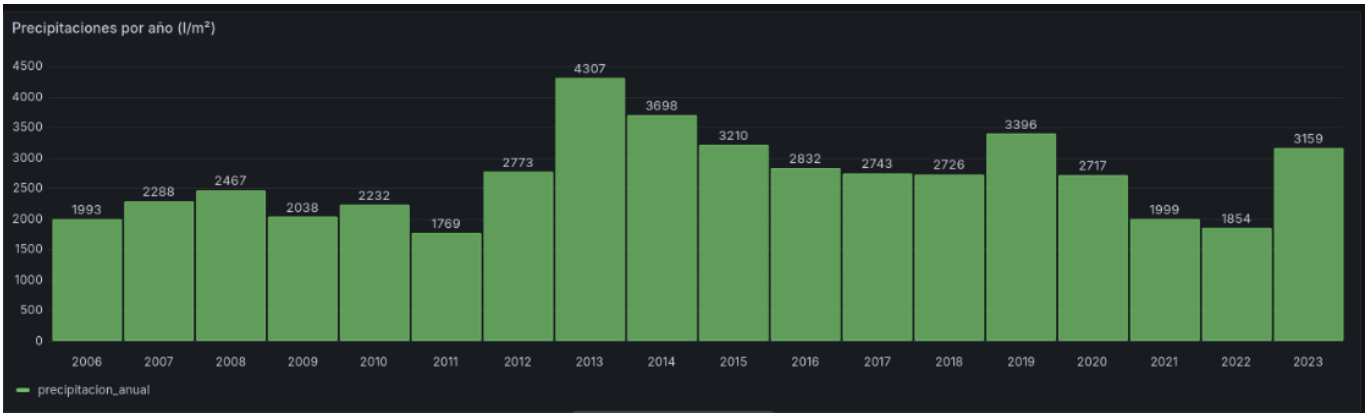
Para la realización del informe sobre la climatología de la Comunidad Foral de Navarra, he realizado dos conjuntos de gráficos. Uno para las precipitaciones por año, mes y semana, y otro para las temperaturas por año, mes y semana. He realizado esto para cada comarca, pero para no saturar todo él informe de imágenes, en la explicación de los conjuntos, mostrare solo imágenes para la comarca de Pamplona. Se puede acceder a las graficas del resto de comarcas a través del link al final de este informe.

- **Conjunto 1: Gráficos sobre la precipitación en la Comunidad Foral de Navarra**

  Este conjunto consta de 3 gráficos:

  - **Precipitaciones por año:**

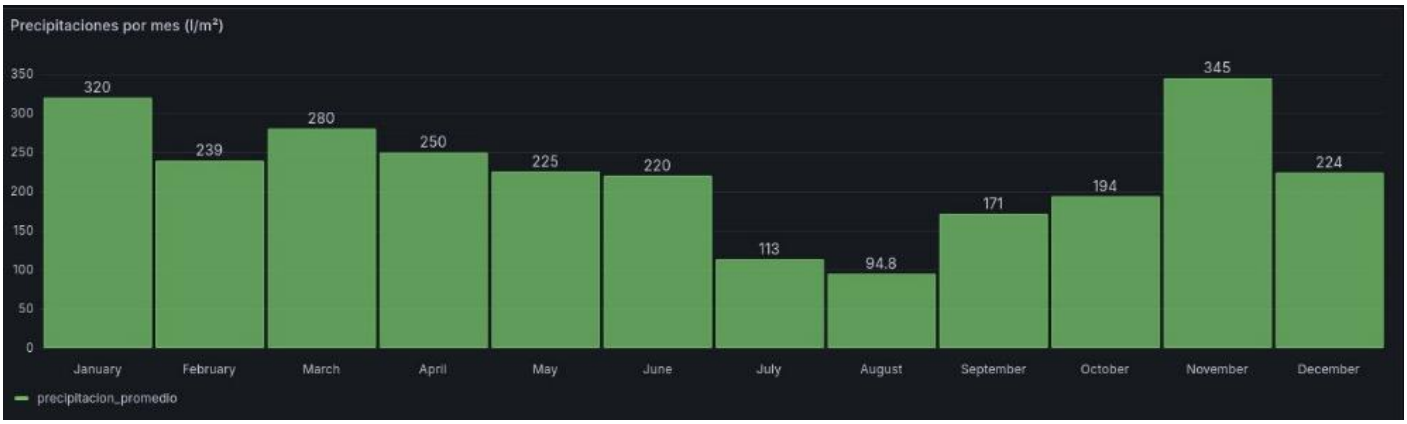
    Este gráfico muestra el total de las precipitaciones acumuladas en cada año para la comarca especificado en la consulta. Permite visualizar la variación anual en la cantidad de precipitaciones.




```sql
SELECT
    DS_fecha AS tiempo,
    SUM(IND_precipitacion) AS precipitacion_anual
FROM
    DT_temperaturas
INNER JOIN
    LK_pueblos ON DT_temperaturas.ID_pueblo = LK_pueblos.ID_pueblo
INNER JOIN
    LK_comarcas ON LK_pueblos.ID_comarca = LK_comarcas.ID_comarca
INNER JOIN
    LK_tiempo ON DT_temperaturas.ID_fecha = LK_tiempo.ID_tiempo
WHERE
    LK_comarcas.DS_comarca = 'Pamplona'
GROUP BY
    DS_anio
```

- **Precipitaciones por mes:**

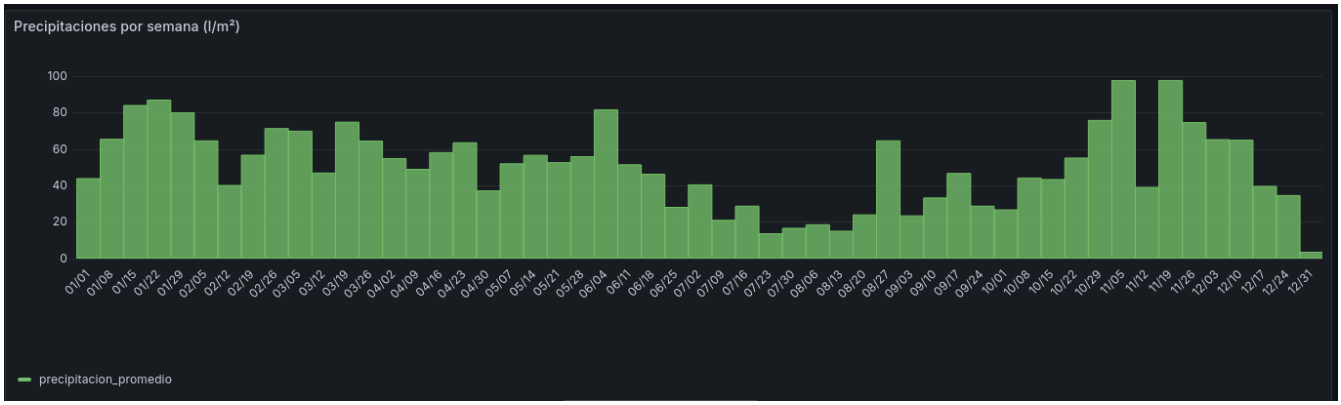
  Este gráfico calcula el promedio de las precipitaciones para cada mes, considerando todos los años. Por ejemplo, para enero, muestra la media de las precipitaciones de todos los meses de enero a lo largo de los años. Esto ofrece una visión general de las precipitaciones típicas de cada mes en la comarca indicada.




```sql
SELECT
    tiempo,
    MONTHNAME(tiempo) AS mes,
    AVG(precipitacion_mensual) AS precipitacion_promedio
FROM (
    SELECT
        DS_fecha AS tiempo,
        DS_anio AS anio,
        DS_mes,
        SUM(IND_precipitacion) AS precipitacion_mensual
    FROM
        DT_temperaturas
    INNER JOIN
        LK_pueblos ON DT_temperaturas.ID_pueblo = LK_pueblos.ID_pueblo
    INNER JOIN
        LK_comarcas ON LK_pueblos.ID_comarca = LK_comarcas.ID_comarca
    INNER JOIN
        LK_tiempo ON DT_temperaturas.ID_fecha = LK_tiempo.ID_tiempo
    WHERE
        LK_comarcas.DS_comarca = 'Pamplona'
    GROUP BY
        anio, DS_mes
) AS precipitaciones_por_mes
GROUP BY
    DS_mes
```

- **Precipitaciones por semana:**

  Este gráfico sigue el mismo enfoque que el anterior, pero en lugar de calcular la media por mes, lo hace por semana. Por ejemplo, para la semana 1, muestra el promedio de precipitaciones de todas las primeras semanas de cada año. Esto proporciona una perspectiva detallada de las precipitaciones promedio semanales a lo largo del tiempo para la comarca especificado en la consulta.



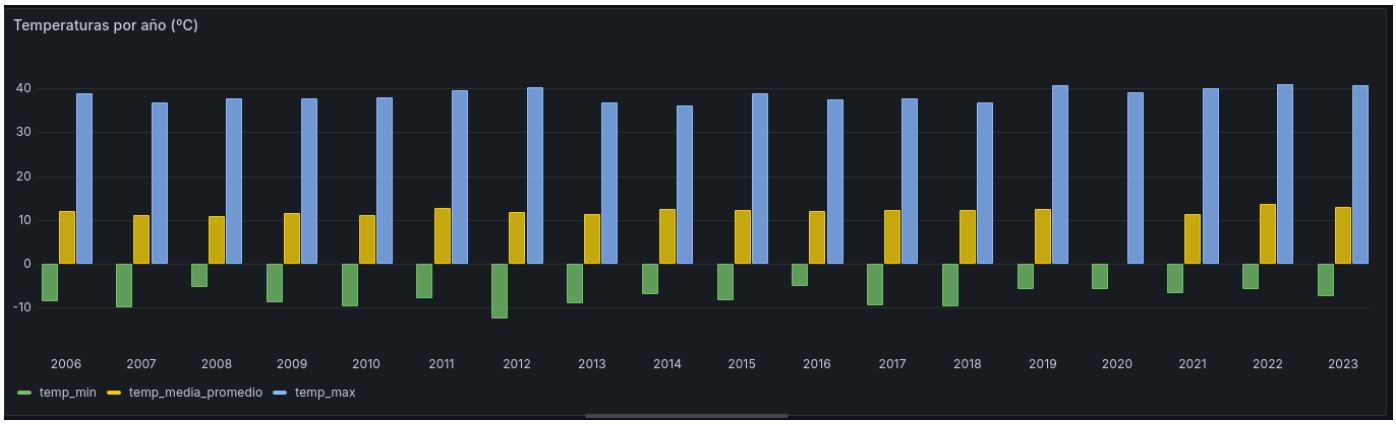

```sql
SELECT
    tiempo,
    AVG(precipitacion_semanal) AS precipitacion_promedio
FROM (
    SELECT
        DS_fecha AS tiempo,
        DS_semana AS semana,
        SUM(DT_temperaturas.IND_precipitacion) AS precipitacion_semanal
    FROM
        DT_temperaturas
    INNER JOIN
        LK_pueblos ON DT_temperaturas.ID_pueblo = LK_pueblos.ID_pueblo
    INNER JOIN
        LK_comarcas ON LK_pueblos.ID_comarca = LK_comarcas.ID_comarca
    INNER JOIN
        LK_tiempo ON DT_temperaturas.ID_fecha = LK_tiempo.ID_tiempo
    WHERE
        LK_comarcas.DS_comarca = 'Pamplona'
    GROUP BY
        DS_anio, semana
) AS precipitaciones_por_semana
GROUP BY
    semana
```

- **Conjunto 2: Gráficos sobre la temperatura en la Comunidad Foral de Navarra**

  Este conjunto consta de 3 gráficos:

  - **Temperaturas por año:**

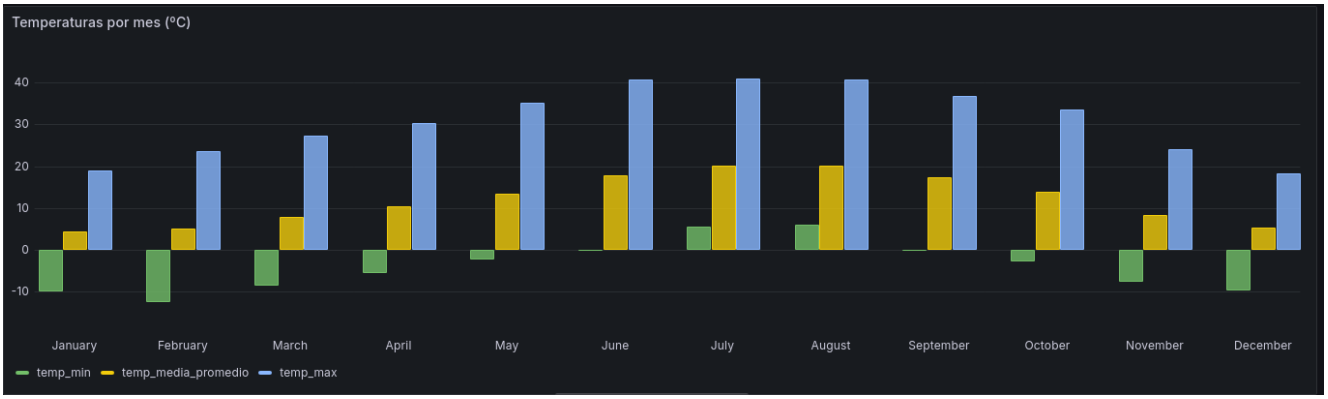
    Este gráfico muestra los valores de temperatura mínima, máxima y media promedio para cada año en la comarca especificada en la consulta. Las temperaturas mínima y máxima reflejan los extremos registrados en cada año, mientras que la temperatura media se calcula como el promedio de todas las temperaturas medias para ese año.




```sql
SELECT
    DS_fecha AS tiempo,
    MIN(IND_tmin) AS temp_min,
    AVG(IND_tmed) AS temp_media_promedio,
    MAX(IND_tmax) AS temp_max
FROM
    DT_temperaturas
INNER JOIN
    LK_pueblos ON DT_temperaturas.ID_pueblo = LK_pueblos.ID_pueblo
INNER JOIN
    LK_comarcas ON LK_pueblos.ID_comarca = LK_comarcas.ID_comarca
INNER JOIN
    LK_tiempo ON DT_temperaturas.ID_fecha = LK_tiempo.ID_tiempo
WHERE
    LK_comarcas.DS_comarca = 'Pamplona'
GROUP BY
    DS_anio
```

- **Temperaturas por mes:**

  Este gráfico agrupa las temperaturas mínima, máxima y media promedio por mes. Las temperaturas mínima y máxima reflejan los extremos registrados para todos los años, es decir, en enero se mostrará la temperatura más baja registrada en el mes de enero entre los años que hay, mientras que la temperatura media se calcula como el promedio de todas temperaturas medias para ese mes en todos los años.



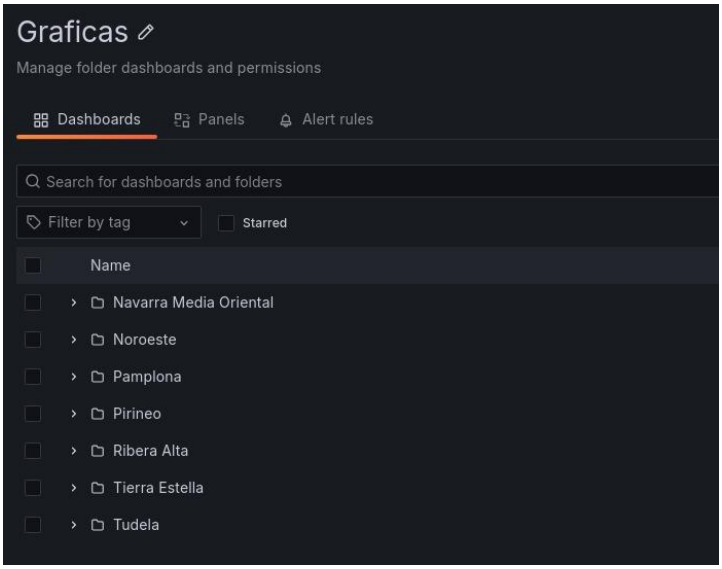

```sql
SELECT
    DS_fecha AS tiempo,
    MONTHNAME(DS_fecha) AS mes,
    MIN(IND_tmin) AS temp_min,
    AVG(IND_tmed) AS temp_media_promedio,
    MAX(IND_tmax) AS temp_max
FROM
    DT_temperaturas
INNER JOIN
    LK_pueblos ON DT_temperaturas.ID_pueblo = LK_pueblos.ID_pueblo
INNER JOIN
    LK_comarcas ON LK_pueblos.ID_comarca = LK_comarcas.ID_comarca
INNER JOIN
    LK_tiempo ON DT_temperaturas.ID_fecha = LK_tiempo.ID_tiempo
WHERE
    LK_comarcas.DS_comarca = 'Pamplona'
GROUP BY
    mes
ORDER BY
    tiempo
```

- **Temperaturas por semana:**

  Este gráfico, sigue el mismo enfoque que el anterior, pero esta vez agrupado por semanas. Las temperaturas mínima y máxima reflejan los extremos registrados para todos los años, es decir, en la semana 1 se mostrará la temperatura más baja registrada en la primera semana de enero entre los años que hay, mientras que la temperatura media se calcula como el promedio de todas temperaturas medias para la primera semana de enero de todos los años.


```sql
SELECT
    DS_fecha AS tiempo,
    MIN(IND_tmin) AS temp_min,
    AVG(IND_tmed) AS temp_media_promedio,
    MAX(IND_tmax) AS temp_max
FROM
    DT_temperaturas
INNER JOIN
    LK_pueblos ON DT_temperaturas.ID_pueblo = LK_pueblos.ID_pueblo
INNER JOIN
    LK_comarcas ON LK_pueblos.ID_comarca = LK_comarcas.ID_comarca
INNER JOIN
    LK_tiempo ON DT_temperaturas.ID_fecha = LK_tiempo.ID_tiempo
WHERE
    LK_comarcas.DS_comarca = 'Pamplona'
GROUP BY
    DS_semana
```

Como he comentado antes, he hecho esto para cada comarca, como se puede ver en la foto adjuntada:



Para acceder a las gráficas hay que utilizar este link, en mi MV:
http://localhost:3000/dashboards/f/ee5lpxfabsv7ka/

Rendimiento de las consultas

Para determinar el número de consultas que la base de datos puede atender por unidad de tiempo, considerando que los clientes están espaciados por 1 segundo y que no hay más de 100 usuarios concurrentes, inicialmente intenté utilizar JMeter. Sin embargo, durante la ejecución, la carga generada por esta herramienta saturó mi máquina virtual. A pesar de intentar reducir el número de hilos, simplificar la consulta y optimizar los parámetros, no logré solucionar este problema.

Como alternativa, he diseñado un código en Python para calcular el rendimiento de la base de datos bajo las condiciones requeridas. Este código me permite medir tanto el tiempo promedio de ejecución de una consulta individual como el rendimiento bajo carga simulada con múltiples usuarios concurrentes. El código se encuentra al final del archivo SistemaBI.ipynb.

Explicación del código

El código implementado en Python realiza las siguientes tareas:

1. **Configuración de la conexión**:
    - Se define un diccionario (db_config) con los parámetros de conexión a la base de datos, incluyendo el servidor, usuario, contraseña y nombre de la base de datos.

2. **Consulta SQL**:
    - Se utiliza una consulta específica que realiza un cálculo de la precipitación anual agrupado por año, filtrando únicamente para la comarca de Pamplona.

3. **Medición del tiempo de ejecución de una consulta**:
    - La función measure_query_performance(query) mide el tiempo necesario para ejecutar la consulta. Se conecta a la base de datos, ejecuta la consulta, obtiene los resultados y calcula la duración en segundos.

4. **Cálculo del rendimiento base**:
    - Se ejecuta la consulta individualmente para medir su tiempo promedio y calcular la cantidad de consultas que la base de datos puede manejar por segundo en condiciones normales.

5. **Simulación de concurrencia**:
    - Para simular 100 usuarios concurrentes:
        - Se crean 100 hilos (threads), cada uno ejecutando la consulta de forma independiente.
        - Los hilos están espaciados 1 segundo entre sí para emular la llegada progresiva de clientes.
        - Los tiempos de ejecución de todas las consultas son recolectados para calcular el tiempo promedio por consulta bajo concurrencia.
6. **Resultados**:
    - El tiempo promedio por consulta y las consultas por segundo son reportados tanto en condiciones individuales como con concurrencia.

Resultados obtenidos

```
Medición de rendimiento de la consulta:
Tiempo de ejecución: 0.0455 segundos

Consultas por segundo estimadas: 21.97

Simulación con 100 usuarios concurrentes:
Promedio de tiempo por consulta con concurrencia: 0.0453 segundos
Consultas por segundo estimadas con concurrencia: 22.10
```