Prueba Técnica.

Requerimientos:

* Instalar MySQL
* Instalar Python 3
* Correr en la terminal los siguientes comandos.
  + pip install fastapi
  + pip install uvicorn
  + pip install pandas
  + pip install mysql-connector-python

Para el entorno virtual:

* python3 -m venv tutorial-env
* tutorial-env\Scripts\activate.bat
  + pip install fastapi
  + pip install uvicorn

Sección1. Procesamiento y transferencia de datos.

* 1. *Carga de información:*



La carga de información se realizo a una base de datos de MySQL debido a que entre sus ventajas tiene la rapidez y la comodidad con la que se puede utilizar, además, otra de sus ventajas es que puede correr en diferentes sistemas operativos, Windows, Linux, Mac OS, etc.

También elegí esta base de datos debido al soporte técnico que tiene, es decir, ya que es de código abierto hace que exista mucha comunidad y las problemáticas se puedan resolver de una manera más sencilla.

Finalmente, y por la mayor razón que la elegí fue debido a que ya he tenido acercamiento con esta base de datos en proyectos escolares.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Fig1. El rectángulo muestra el botón para iniciar una base de datos.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Fig2. Se coloca el nombre de la nueva base de datos y se presiona el botón ‘Apply’

Además de utilizar el entorno de ‘ My SQL Workbench’ utilice un entorno gráfico llamado ‘HeidiSQL’ que a mi parecer me es más cómodo.

Interfaz de usuario gráfica, Tabla, Excel

Descripción generada automáticamente

Fig3. Entorno de Heidi SQL

Observamos en la imagen un rectángulo rojo, este nos señala nuestra base de datos y que tablas contiene. El rectángulo azul, muestra la primera tabla creada para cargar los datos del archivo ‘data\_prueba\_tecnica.csv’.

Finalmente, el recuadro amarillo muestra los datos que ya fueron cargados a la tabla con nombre ‘data\_prueba\_tecnica’.

* 1. *Extracción:*

Para esta parte utilice el lenguaje Python, debido a que cuenta con librerías de mucha ayuda para la extracción de información. En primer lugar había que realizar un conector entre Python y MySQL por lo que la librería ‘mysql.connector’ nos ayudaría en esta parte, además de esta utilice ‘Pandas’ para los dataFrame y ‘datetime’ para poder obtener el formato de horas.

El mayor reto es recorrer la base de datos y después almacenar sus valores, se utiliza el método fetchall() con el cual obtenemos las coincidencias.

Captura de pantalla de computadora

Descripción generada automáticamente

Fig4. Conexión de Python con MySQL además de recorrer la DB.

Un poco más abajo en el código nos apoyaremos de listas auxiliares para poder concatenar valores. Una vez hecho esto nos apoyaremos de un diccionario para almacenar todos los datos.

Finalmente ocupamos la librería ‘Pandas’ y con el, DataFrame() que nos ayudará a generar la tabla que estamos buscanso.

Como último paso convertimos este dataframe en un archivo .csv, especificando la ruta y el nombre de este.

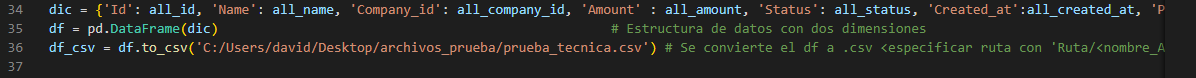


Fig5. Creación de diccionario y conversión a .CSV

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamenteFig6. Muestra el archivo creado en la ruta y el archivo generado abierto.

* 1. *Transformación:*

*Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja*

Fig7. Propuesto de esquema para la información.

El propósito en este paso era filtrar la información mediante Python, para que las variables, en este caso las columnas, cumplieran con dichos parámetros.

Esta filtración se realizo al mismo tiempo que se recorre la base de datos y se asignan a las listas para la importación del archivo .csv

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

Fig8. Muestra el código de Python para la filtración de los datos.

En esta parte las variables de tipo srt o chart no hubo mayor problema, solo se tomaron los primeros números de cada cadena, se da por ‘cadena[:# de elementos]’.

En la parte de los flotantes presento un reto debido a que solo había que tomar dos decimales según el esquema, la documentación de float menciona que: **‘ “{:.#decimalesf}”.format(amount)”** donde amount toma el lado izquierdo de los dos puntos ‘:’

Finalmente y el que mayor complejidad me presento fue el formato de la fecha, ya que había que pasarlo de una variable de tipo srt a datatime, además de que el formato se tenia que respetar. El .date() al final nos ayuda a quitar el formato de tiempo ya que si este se queda, el programa no encuentra coincidencias y nos lleva a un error.

* 1. *Dispersión de la información:*

En este paso básicamente regresamos al paso 1.1 ya que tenemos que crear dos tablas y tenemos que cargar los nuevos datos filtrados que realizamos con Python.

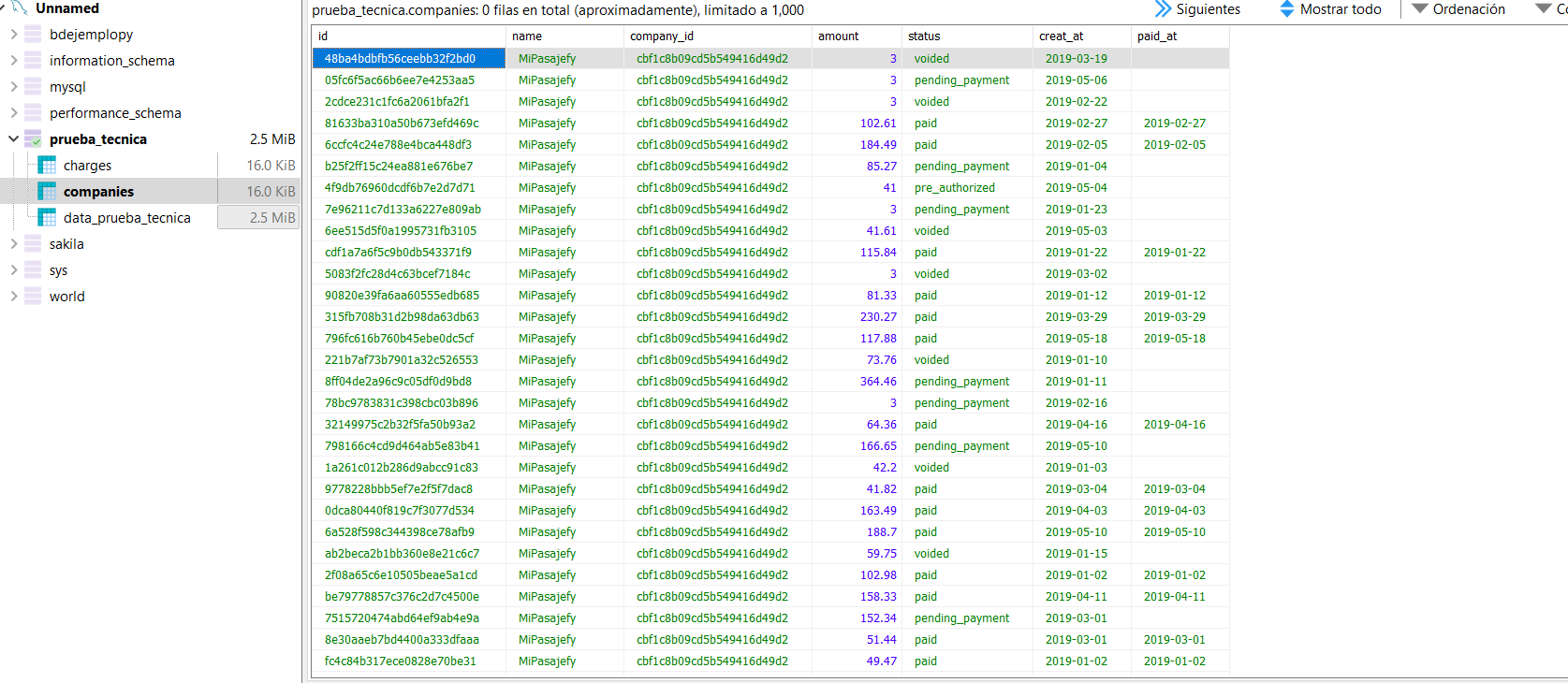


Fig9. Tablas solicitadas con carga de datos filtrados.

Las bases de datos relacionales tienen diversor ‘tipos de relacionales’ que podemos utilizar para vincular nuestras tablas. Este viculo dependerá de la cantidad de ocurrencias que tiene ‘ un registro’ de una tabla dentro de otra.

Sección 2. Creación de una API

Problema: Calcular el numero faltante de un conjunto de los primeros 100 números naturales del cual se extrajo uno.

Para poder correr esta API programa es necesario:

* Ir a la ruta de la carpeta que creamos para el entorno virtual: en este caso: *cd Desktop\FastApi*
* Accedemos al entorno: tutorial-env\Scripts\activate.bat
* Levantamos el servidor de FastApi: *uvicorn main:app –reload 🡪* Accedemos al url dado.