Proyecto para el Sprint 4: Ejercicio

Este proyecto consta de dos partes. Vas a trabajar en la consola con los logs y la base de datos para una aplicación de taxis.

Consola

Ejercicio 1

El equipo de desarrollo ha enviado una tarea: debes averiguar qué solicitudes han venido de una dirección IP. Una dirección IP consta de cuatro números separados por un punto. Necesitas direcciones que comiencen con "233.201".

Los logs están en un servidor remoto en `logs/2019/12`. No sabes qué día ocurrió el error.

Tu tarea es averiguar qué solicitudes fueron enviadas.

Esto es lo que debes adjuntar en la respuesta:

1. El comando que usaste para obtener los logs necesarios.
2. Los strings adecuados, por ejemplo: `184.79.247.161 - - [30/12/2019:21:38:13 +0000] "PUT /alerts HTTP/1.1" 400 3557`

Ejercicio 2

Se detectó un error en el sistema. Estuvo activo el 30.12.2019 y el 31.12.2019 de 9:30:00 p. m. a 9:39:59 p. m. A esa misma hora, se produjeron errores 400 y 500. Tu tarea es guardar los logs que se registraron durante este periodo en un archivo por separado.

Después, debes poner estos logs en archivos por separado con base en los errores. A continuación, te explicamos cómo hacerlo:

1. En el directorio de inicio del servidor remoto, crea un directorio llamado `bug1`.

2. Pon todas las solicitudes que ocurrieron durante este periodo específico en el archivo `main.txt` en el directorio `bug1`.
3. Dentro del directorio `bug1` , crea un directorio `events`.
4. Dentro del directorio `events`, crea archivos de error 400 y 500. Llama a estos archivos `400.txt` y `500.txt`, respectivamente. Identifica en ellos los logs utilizando el error correspondiente del archivo `main.txt`.

Esto es lo que debes adjuntar en la respuesta:

1. Los comandos que crean los directorios `bug1` y `events`.
2. El comando que usas para seleccionar las solicitudes para el periodo especificado. Estas son las solicitudes que usas para recopilar logs en el archivo `main.txt`.
3. Los comandos que usas para colocar los logs en los archivos `400.txt` y `500.txt` de `main.txt`.
4. Los archivos de texto `400.txt` y `500.txt`.

Base de datos

Descripción de los datos

Base de datos de los viajes en taxi de Chicago:

La tabla `neighborhoods`, con información sobre los barrios de la ciudad:

- `neighborhood_id`: código del vecindario.
- `name`: nombre del vecindario.

La tabla `cabs`, con información sobre los taxis:

- `cab_id`: código único del vehículo.
- `vehicle_id`: identificador técnico del vehículo.
- `company_name`: la compañía dueña del automóvil.

La tabla `trips`, con información sobre viajes:

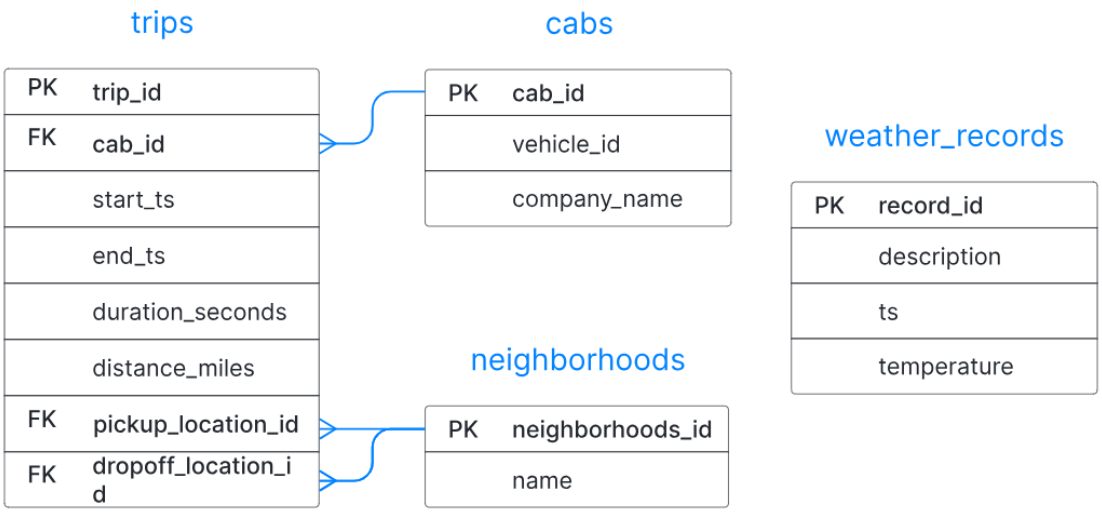
- `trip_id`: código de viaje.
- `cab_id`: el código del automóvil usado para el viaje.

- `start_ts`: fecha y hora del inicio del viaje (tiempo redondeado a la hora más próxima).
- `end_ts`: fecha y hora del fin del viaje (tiempo redondeado a la hora más próxima).
- `duration_seconds`: la duración del viaje en segundos.
- `distance_miles`: la distancia del viaje en millas.
- `pickup_location_id`: el código del vecindario donde inició el viaje.
- `dropoff_location_id`: el código del vecindario donde terminó el viaje.

La tabla `weather_records` (registros meteorológicos), con información sobre el clima:

- `record_id`: código del registro de observación meteorológica.
- `ts`: fecha y hora de la observación (tiempo redondeado a la hora más próxima).
- `temperature`: temperatura a la hora de la observación.
- `description`: una breve descripción de las condiciones meteorológicas (p. ej., lluvia ligera o nubes dispersas).

Disposición de la tabla



No existe un enlace directo entre las tablas `trips` y `weather_records` en la base de datos. Puedes vincular estas tablas por la hora de inicio

(`trips.start_ts`) y el momento de las observaciones meteorológicas (`weather_records.ts`).

Ejercicio 1

Tienes una base de datos con los viajes en taxi. El plan era tener 10 550 vehículos disponibles, lo que cubre la demanda del usuario; sin embargo, el equipo recibió muchas quejas de que no había vehículos suficientes. ¿Cuántos taxis hay actualmente en las calles? La información sobre todos los automóviles suficientes está en la tabla `cabs`.

1. Ve al servidor remoto.
2. Conéctate a la base de datos `chicago_taxi` con el nombre de usuario `morty` y la contraseña `smith`.
3. Cuenta el número total de automóviles en la tabla `cabs`. Recuerda que un automóvil podría pertenecer a distintas compañías.

Esto es lo que debes adjuntar en la respuesta:

1. Número de automóviles.
2. La solicitud que usaste para resolver el problema.

Ejercicio 2

En la tabla `cabs`, cuenta el número de automóviles de cada compañía. Ordena los valores en orden descendente. El equipo piensa que algunas compañías no tuvieron suficientes automóviles disponibles.

Devuelve las compañías con menos de 100 automóviles. Llama `cnt` (contados) al campo con el número de automóviles, y `company_name` al campo con el nombre de la compañía.

Para resolver el problema, utiliza el operador `HAVING`, una analogía de `WHERE` para las funciones agregadas. Estudia la documentación para aprender cómo funciona el operador:

(https://postgrespro.com/docs/postgrespro/11/queries-table-expressions *(materiales en inglés)*)

Esto es lo que debes adjuntar en la respuesta:

1. Una lista de compañías con menos de 100 automóviles.
2. La solicitud que usaste para resolver el problema.

Nota: la consola muestra una lista incompleta. Para verla en su totalidad, presiona Enter o usa las flechas de tu teclado.

Ejercicio 3

La aplicación de taxis calcula el coeficiente del costo del viaje. Si el clima es bueno, el valor del coeficiente es 1. Si llueve o hay tormentas en el exterior, el coeficiente aumenta a 2. El equipo tiene una hipótesis de que hay un error en el cálculo del coeficiente. Para revisar el cálculo del coeficiente, el equipo necesita una selección de datos: el área de desarrollo puede comparar el coeficiente con los datos en los logs y corregir el bug. Tu tarea es obtener una selección.

Para hacerlo:

1. Obtén una descripción de las condiciones meteorológicas de la tabla `weather_records` para cada hora.
2. Divide todas las horas en dos grupos a través del operador `CASE`: Está `Bad` ("mal") si el campo `description` contiene las palabras "rain" (lluvia) o "storm" (tormenta); `Good` ("bien"), para todas las demás horas.
3. Pon el nombre `weather_conditions` al campo resultante.

La tabla resultante debe tener dos campos: fecha y hora (`ts`) y `weather_conditions`.

Haz una selección para el periodo entre 11-05-2017 12:00 a. m. a 11-06-2017 12:00 a. m.

Esto es lo que debes adjuntar en la respuesta:

1. La tabla resultante con los datos para el periodo especificado.
2. La solicitud que ayudó a resolver el problema.

Ejercicio 4

Tras actualizar el software, la compañía de taxis comienza a reportar que la ganancia que reciben no coincide con los datos que proporciona la aplicación. El equipo de desarrollo sugiere que el problema puede estar en los datos sobre el número de viajes.

Para determinar si hay un bug, debes obtener la selección del número de viajes de cada compañía de taxi para los días 15 y 16 de noviembre de 2017.

1. Devuelve el campo `company_name`. Nombra `trips_amount` (cantidad de viajes) al campo con el número de viajes y devuélvelo.
2. Organiza en orden descendente los resultados obtenidos en el campo `trips_amount`.

Pista: para resolver el problema, conecta las tablas de taxis y viajes. Aplica funciones de agregación con agrupamiento. No olvides escribir la construcción con una condición.

Esto es lo que debes adjuntar en la respuesta:

1. La tabla resultante con los datos para el periodo especificado.
2. La solicitud que ayudó a resolver el problema.