Descripción del proyecto 7

Estás trabajando como analista para Zuber, una nueva empresa de viajes compartidos que se está lanzando en Chicago. Tu tarea es encontrar patrones en la información disponible. Quieres comprender las preferencias de los pasajeros y el impacto de los factores externos en los viajes.

Al trabajar con una base de datos, analizarás los datos de los competidores y probarás una hipótesis sobre el impacto del clima en la frecuencia de los viajes.

Descripción de los datos

Una base de datos con información sobre viajes en taxi en Chicago:

tabla neighborhoods: datos sobre los barrios de la ciudad

name: nombre del barrio

neighborhood\_id: código del barrio

tabla cabs: datos sobre los taxis

cab\_id: código del vehículo

vehicle\_id: ID técnico del vehículo

company\_name: la empresa propietaria del vehículo

tabla trips: datos sobre los viajes

trip\_id: código del viaje

cab\_id: código del vehículo que opera el viaje

start\_ts: fecha y hora del inicio del viaje (tiempo redondeado a la hora)

end\_ts: fecha y hora de finalización del viaje (tiempo redondeado a la hora)

duration\_seconds: duración del viaje en segundos

distance\_miles: distancia del viaje en millas

pickup\_location\_id: código del barrio de recogida

dropoff\_location\_id: código del barrio de finalización

tabla weather\_records: datos sobre el clima

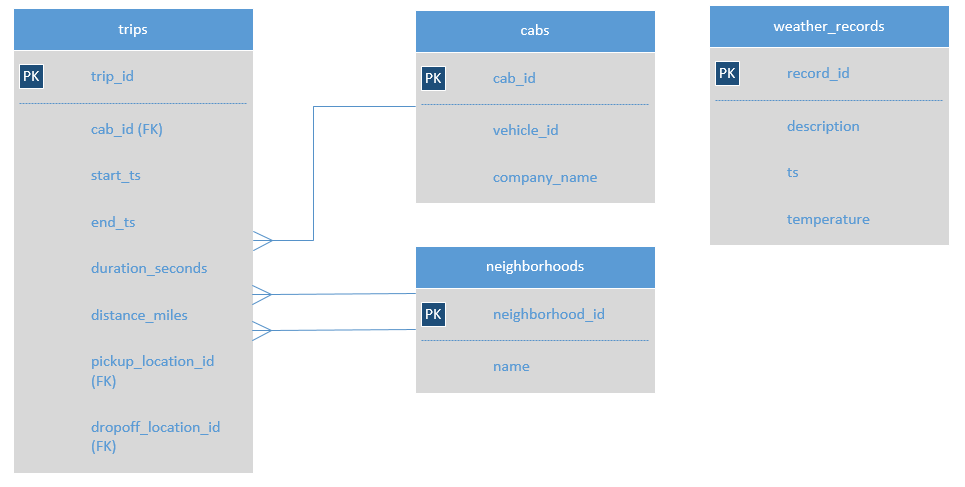
record\_id: código del registro meteorológico

ts: fecha y hora del registro (tiempo redondeado a la hora)

temperature: temperatura cuando se tomó el registro

description: breve descripción de las condiciones meteorológicas, por ejemplo, "lluvia ligera" o "nubes dispersas"

Esquema de la tabla

image

Nota: no existe una conexión directa entre las tablas trips y weather\_records en la base de datos. Pero aún puedes usar JOIN y vincularlas usando la hora en la que comenzó el viaje (trips.start\_ts) y la hora en la que se tomó el registro meteorológico (weather\_records.ts).

Instrucciones para completar el proyecto

|  |
| --- |
| Paso 1. Escribe un código para analizar los datos sobre el clima en Chicago en noviembre de 2017 desde el sitio web:  https://practicum-content.s3.us-west-1.amazonaws.com/data-analyst-eng/moved\_chicago\_weather\_2017.html  Paso 2. Análisis exploratorio de datos  2.1 Encuentra el número de viajes en taxi para cada empresa de taxis del 15 al 16 de noviembre de 2017.  2.2 Nombra el campo resultante trips\_amount y muéstralo junto con el campo company\_name.  2.3 Ordena los resultados por el campo trips\_amount en orden descendente.  2.4 Encuentra la cantidad de viajes para cada empresa de taxis cuyo nombre contenga las palabras "Yellow" o "Blue" del 1 al 7 de noviembre de 2017.  2.5 Nombra la variable resultante trips\_amount. Agrupa los resultados por el campo company\_name.  2.6 En noviembre de 2017 las empresas de taxis más populares fueron Flash Cab y Taxi Affiliation Services. Encuentra el número de viajes de estas dos empresas y asigna a la variable resultante el nombre trips\_amount.  2.7 Junta los viajes de todas las demás empresas en el grupo "Other".  Agrupa los datos por nombres de empresas de taxis. Nombra el campo con nombres de empresas de taxis company. Ordena el resultado en orden descendente por trips\_amount.  Paso 3. Prueba la hipótesis de que la duración de los viajes desde el Loop hasta el Aeropuerto Internacional O'Hare cambia los sábados lluviosos.  Recupera los identificadores de los barrios de O'Hare y Loop de la tabla neighborhoods.  Para cada hora recupera los registros de condiciones meteorológicas de la tabla weather\_records. Usando el operador CASE, divide todas las horas en dos grupos: "Bad" si el campo description contiene las palabras "rain" o "storm" y "Good" para los demás. Nombra el campo resultante weather\_conditions. La tabla final debe incluir dos campos: fecha y hora (ts) y weather\_conditions.  Recupera de la tabla trips todos los viajes que comenzaron en el Loop (neighborhood\_id: 50) y finalizaron en O'Hare (neighborhood\_id: 63) un sábado. Obtén las condiciones climáticas para cada viaje. Utiliza el método que aplicaste en la tarea anterior. Recupera también la duración de cada viaje.  Ignora los viajes para los que no hay datos disponibles sobre las condiciones climáticas. |
| Paso 4. Análisis exploratorio de datos (Python)  Además de los datos que recuperaste en las tareas anteriores te han dado un segundo archivo. Ahora tienes estos dos CSV:  project\_sql\_result\_01.csv. Contiene los siguientes datos:  company\_name: nombre de la empresa de taxis  trips\_amount: el número de viajes de cada compañía de taxis el 15 y 16 de noviembre de 2017.  project\_sql\_result\_04.csv. Contiene los siguientes datos:  dropoff\_location\_name: barrios de Chicago donde finalizaron los viajes average\_trips: el promedio de viajes que terminaron en cada barrio en noviembre de 2017.  Para estos dos datasets ahora necesitas:  importar los archivos  estudiar los datos que contienen  asegurarte de que los tipos de datos sean correctos  identificar los 10 principales barrios en términos de finalización  hacer gráficos: empresas de taxis y número de viajes, los 10 barrios principales por número de finalizaciones  sacar conclusiones basadas en cada gráfico y explicar los resultados  Paso 5. Prueba de hipótesis (Python)  project\_sql\_result\_07.csv: el resultado de la última consulta. Contiene datos sobre viajes desde el Loop hasta el Aeropuerto Internacional O'Hare. Recuerda, estos son los valores de campo de la tabla:  start\_ts: fecha y hora de recogida  weather\_conditions: condiciones climáticas en el momento en el que comenzó el viaje  duration\_seconds: duración del viaje en segundos  Prueba la hipótesis:  "La duración promedio de los viajes desde el Loop hasta el Aeropuerto Internacional O'Hare cambia los sábados lluviosos".  Establece el valor del nivel de significación (alfa) por tu cuenta.  Explica:  cómo planteaste las hipótesis nula y alternativa  qué criterio usaste para probar las hipótesis y por qué  ¿Cómo será evaluado mi proyecto?  Estos son los criterios de evaluación del proyecto. Léelos atentamente antes de empezar a trabajar.  Esto es lo que buscará el revisor del proyecto al evaluar tu proyecto:  cómo recuperas los datos del sitio web  cómo creas slices de datos  cómo agrupas los datos  si utilizas los métodos correctos para unir tablas  cómo formulas las hipótesis  qué criterios utilizas para probar las hipótesis y por qué  a qué conclusiones llegas  si dejas comentarios en cada paso  Las hojas informativas y resúmenes de las lecciones anteriores tienen todo lo que necesitas para completar el proyecto. |
|  |

¡Buena suerte!

Estás trabajando como analista para Zuber, una nueva empresa de viajes compartidos que se está lanzando en Chicago. Tu tarea es encontrar patrones en la información disponible. Quieres comprender las preferencias de los pasajeros y el impacto de los factores externos en los viajes.

Estudiarás una base de datos, analizarás los datos de los competidores y probarás una hipótesis sobre el impacto del clima en la frecuencia de los viajes.

En esta tarea completarás el primer paso del proyecto.

Usa el material del capítulo "Recuperación de datos de recursos en línea".

import pandas as pd

import requests

from bs4 import BeautifulSoup

# URL del sitio web que contiene los datos

url = "https://practicum-content.s3.us-west-1.amazonaws.com/data-analyst-eng/moved\_chicago\_weather\_2017.html"

# Realizamos una solicitud GET para obtener el contenido HTML

response = requests.get(url)

# Comprobamos si la solicitud fue exitosa (código de estado 200)

if response.status\_code == 200:

# Parseamos el contenido HTML usando BeautifulSoup

soup = BeautifulSoup(response.content, "html.parser")

# Encontramos todos los elementos <tr> dentro del elemento con id="weather\_records"

table\_rows = soup.find("table", attrs={"id": "weather\_records"}).find\_all("tr")

# Extraemos los encabezados de la tabla (primera fila)

headers = [header.text.strip() for header in table\_rows[0].find\_all("th")]

# Extraemos los datos de la tabla (las filas restantes)

data = []

for row in table\_rows[1:]:

data.append([cell.text.strip() for cell in row.find\_all("td")])

# Creamos el DataFrame

weather\_records = pd.DataFrame(data, columns=headers)

# Imprimimos el DataFrame completo

print(weather\_records)

else:

print(f"No se pudo acceder al contenido del sitio web. Código de estado: {response.status\_code}")

Ejercicio 1

Imprime el campo *company\_name*. Encuentra la cantidad de viajes en taxi para cada compañía de taxis para el 15 y 16 de noviembre de 2017, asigna al campo resultante el nombre *trips\_amount* e imprímelo también. Ordena los resultados por el campo *trips\_amount* en orden descendente.

en sql Imprime el campo company\_name. Encuentra la cantidad de viajes en taxi para cada compañía de taxis para el 15 y 16 de noviembre de 2017, asigna al campo resultante el nombre trips\_amount e imprímelo también. Ordena los resultados por el campo trips\_amount en orden descendente.

ChatGPT

Para lograr esto, necesitamos realizar una consulta SQL que una las tablas trips y cabs, filtre los viajes realizados el 15 y 16 de noviembre de 2017, cuente la cantidad de viajes por compañía de taxis y ordene los resultados en orden descendente. Aquí está la consulta SQL para hacerlo:

sql

Copiar código

SELECT

c.company\_name,

COUNT(t.trip\_id) AS trips\_amount

FROM

trips t

JOIN

cabs c ON t.cab\_id = c.cab\_id

WHERE

t.start\_ts >= '2017-11-15 00:00:00' AND t.start\_ts < '2017-11-17 00:00:00'

GROUP BY

c.company\_name

ORDER BY

trips\_amount DESC;

**Explicación de la consulta:**

1. **Seleccionar columnas**:
   * c.company\_name: El nombre de la empresa de taxis.
   * COUNT(t.trip\_id) AS trips\_amount: La cantidad de viajes (contamos los trip\_id).
2. **Join de las tablas**:
   * JOIN cabs c ON t.cab\_id = c.cab\_id: Unimos las tablas trips y cabs mediante el campo cab\_id.
3. **Filtro por fecha**:
   * t.start\_ts >= '2017-11-15 00:00:00' AND t.start\_ts < '2017-11-17 00:00:00': Filtramos los viajes que ocurrieron el 15 y 16 de noviembre de 2017. Utilizamos un rango de fechas que incluye desde el inicio del 15 de noviembre hasta el final del 16 de noviembre.
4. **Agrupar y contar**:
   * GROUP BY c.company\_name: Agrupamos los resultados por el nombre de la compañía.
   * COUNT(t.trip\_id): Contamos la cantidad de viajes por cada grupo.
5. **Ordenar los resultados**:
   * ORDER BY trips\_amount DESC: Ordenamos los resultados por trips\_amount en orden descendente.

**Resultado esperado:**

La consulta imprimirá el nombre de la empresa de taxis (company\_name) y la cantidad de viajes (trips\_amount) realizados por cada empresa el 15 y 16 de noviembre de 2017, ordenados de mayor a menor cantidad de viajes.

| **ompany\_name** | **trips\_amount** |
| --- | --- |
| Flash Cab | 19558 |
| Taxi Affiliation Services | 11422 |
| Medallion Leasin | 10367 |
| Yellow Cab | 9888 |
| Taxi Affiliation Service Yellow | 9299 |
| Chicago Carriage Cab Corp | 9181 |
| City Service | 8448 |
| Sun Taxi | 7701 |
| Star North Management LLC | 7455 |
| Blue Ribbon Taxi Association Inc. | 5953 |
| Choice Taxi Association | 5015 |
| Globe Taxi | 4383 |
| Dispatch Taxi Affiliation | 3355 |
| Nova Taxi Affiliation Llc | 3175 |
| Patriot Taxi Dba Peace Taxi Associat | 2235 |
| Checker Taxi Affiliation | 2216 |
| Blue Diamond | 2070 |
| Chicago Medallion Management | 1955 |
| 24 Seven Taxi | 1775 |
| Chicago Medallion Leasing INC | 1607 |
| Checker Taxi | 1486 |
| American United | 1404 |
| Chicago Independents | 1296 |
| KOAM Taxi Association | 1259 |
| Chicago Taxicab | 1014 |
| Top Cab Affiliation | 978 |
| Gold Coast Taxi | 428 |
| Service Taxi Association | 402 |
| 5 Star Taxi | 310 |
| 303 Taxi | 250 |
| Setare Inc | 230 |
| American United Taxi Affiliation | 210 |
| Leonard Cab Co | 147 |
| Metro Jet Taxi A | 146 |
| Norshore Cab | 127 |
| 6742 - 83735 Tasha ride inc | 39 |
| 3591 - 63480 Chuks Cab | 37 |
| 1469 - 64126 Omar Jada | 36 |
| 0118 - 42111 Godfrey S.Awir | 33 |
| 6743 - 78771 Luhak Corp | 33 |
| 6574 - Babylon Express Inc. | 31 |
| Chicago Star Taxicab | 29 |
| 1085 - 72312 N and W Cab Co | 29 |
| 2809 - 95474 C & D Cab Co Inc. | 29 |
| 2092 - 61288 Sbeih company | 27 |
| 3011 - 66308 JBL Cab Inc. | 25 |
| 3620 - 52292 David K. Cab Corp. | 21 |
| 4615 - 83503 Tyrone Henderson | 21 |
| 3623 - 72222 Arrington Enterprises | 20 |
| 5074 - 54002 Ahzmi Inc | 16 |
| 2823 - 73307 Lee Express Inc | 15 |
| 4623 - 27290 Jay Kim | 15 |
| 3721 - Santamaria Express, Alvaro Santamaria | 14 |
| 5006 - 39261 Salifu Bawa | 14 |
| 2192 - 73487 Zeymane Corp | 14 |
| 6057 - 24657 Richard Addo | 13 |
| 5997 - 65283 AW Services Inc. | 12 |
| Metro Group | 11 |
| 5062 - 34841 Sam Mestas | 8 |
| 4053 - 40193 Adwar H. Nikola | 7 |
| 2733 - 74600 Benny Jona | 7 |
| 5874 - 73628 Sergey Cab Corp. | 5 |
| 2241 - 44667 - Felman Corp, Manuel Alonso | 3 |
| 3556 - 36214 RC Andrews Cab | 2 |

Ejercicio 2

Encuentra la cantidad de viajes para cada empresa de taxis cuyo nombre contenga las palabras "Yellow" o "Blue" del 1 al 7 de noviembre de 2017. Nombra la variable resultante *trips\_amount.* Agrupa los resultados por el campo *company\_name*.

Para encontrar la cantidad de viajes para cada empresa de taxis cuyo nombre contenga las palabras "Yellow" o "Blue" del 1 al 7 de noviembre de 2017, agrupando los resultados por el campo company\_name, puedes usar la siguiente consulta SQL:

sql

Copiar código

SELECT

company.company\_name, -- El nombre de la empresa de taxis.

COUNT(trips.trip\_id) AS trips\_amount -- La cantidad de viajes (contamos los trip\_id).

FROM

trips

JOIN

cabs company ON trips.cab\_id = company.cab\_id -- Unimos las tablas trips y cabs mediante el campo cab\_id.

WHERE

(company.company\_name LIKE '%Yellow%' OR company.company\_name LIKE '%Blue%') -- Filtramos las empresas cuyos nombres contienen "Yellow" o "Blue".

AND trips.start\_ts >= '2017-11-01 00:00:00' AND trips.start\_ts < '2017-11-08 00:00:00' -- Filtramos los viajes que ocurrieron del 1 al 7 de noviembre de 2017.

GROUP BY

company.company\_name -- Agrupamos los resultados por el nombre de la compañía.

ORDER BY

trips\_amount DESC; -- Ordenamos los resultados por trips\_amount en orden descendente.

**Descripción paso a paso:**

1. **Seleccionar columnas**: Seleccionamos company\_name de la tabla cabs y contamos los trip\_id de la tabla trips, asignándole el alias trips\_amount.
2. **Unir tablas**: Realizamos un JOIN entre la tabla trips y la tabla cabs, utilizando cab\_id como la clave de unión.
3. **Filtrar por nombre de compañía y fechas**: Filtramos las compañías cuyos nombres contienen "Yellow" o "Blue" usando la cláusula LIKE y filtramos los viajes que ocurrieron entre las 00:00 del 1 de noviembre de 2017 y las 23:59 del 7 de noviembre de 2017.
4. **Agrupar resultados**: Agrupamos los resultados por company\_name para obtener la cantidad de viajes por cada compañía de taxis.
5. **Ordenar resultados**: Ordenamos los resultados en orden descendente basado en trips\_amount.

Esta consulta SQL proporcionará la cantidad de viajes realizados por cada compañía de taxis cuyo nombre contiene las palabras "Yellow" o "Blue" en el período especificado, ordenados de mayor a menor cantidad de viajes.

| **company\_name** | **trips\_amount** |
| --- | --- |
| Yellow Cab | 33668 |
| Taxi Affiliation Service Yellow | 29213 |
| Blue Ribbon Taxi Association Inc. | 17675 |
| Blue Diamond | 6764 |

Ejercicio 3

Del 1 al 7 de noviembre de 2017, las empresas de taxis más populares fueron Flash Cab y Taxi Affiliation Services. Encuentra el número de viajes de estas dos empresas y asigna a la variable resultante el nombre *trips\_amount.* Junta los viajes de todas las demás empresas en el grupo "Other". Agrupa los datos por nombres de empresas de taxis. Asigna el nombre *company* al campo con nombres de empresas de taxis. Ordena el resultado en orden descendente por *trips\_amount*.

Para encontrar el número de viajes de las empresas de taxis "Flash Cab" y "Taxi Affiliation Services" del 1 al 7 de noviembre de 2017, y agrupar los viajes de todas las demás empresas en el grupo "Other", puedes usar la siguiente consulta SQL:

sql

Copiar código

SELECT

CASE

WHEN company.company\_name = 'Flash Cab' THEN 'Flash Cab'

WHEN company.company\_name = 'Taxi Affiliation Services' THEN 'Taxi Affiliation Services'

ELSE 'Other'

END AS company, -- Asigna "Flash Cab", "Taxi Affiliation Services" o "Other" según corresponda.

COUNT(trips.trip\_id) AS trips\_amount -- La cantidad de viajes (contamos los trip\_id).

FROM

trips

JOIN

cabs company ON trips.cab\_id = company.cab\_id -- Unimos las tablas trips y cabs mediante el campo cab\_id.

WHERE

trips.start\_ts >= '2017-11-01 00:00:00' AND trips.start\_ts < '2017-11-08 00:00:00' -- Filtramos los viajes que ocurrieron del 1 al 7 de noviembre de 2017.

GROUP BY

company -- Agrupamos los resultados por el campo calculado company.

ORDER BY

trips\_amount DESC; -- Ordenamos los resultados por trips\_amount en orden descendente.

**Descripción paso a paso:**

1. **Seleccionar columnas**:
   * Utilizamos una cláusula CASE para asignar el nombre "Flash Cab" o "Taxi Affiliation Services" si el company\_name coincide con estos nombres específicos. Para todas las demás compañías, asignamos el nombre "Other".
   * Contamos los trip\_id de la tabla trips, asignándole el alias trips\_amount.
2. **Unir tablas**: Realizamos un JOIN entre la tabla trips y la tabla cabs, utilizando cab\_id como la clave de unión.
3. **Filtrar por fechas**: Filtramos los viajes que ocurrieron entre las 00:00 del 1 de noviembre de 2017 y las 23:59 del 7 de noviembre de 2017.
4. **Agrupar resultados**: Agrupamos los resultados por el campo calculado company (que puede ser "Flash Cab", "Taxi Affiliation Services" o "Other").
5. **Ordenar resultados**: Ordenamos los resultados en orden descendente basado en trips\_amount.

Esta consulta SQL proporcionará el número de viajes realizados por "Flash Cab", "Taxi Affiliation Services" y un grupo combinado de todas las demás compañías ("Other") en el período especificado, ordenados de mayor a menor cantidad de viajes.

| **company** | **trips\_amount** |
| --- | --- |
| Other | 335771 |
| Flash Cab | 64084 |
| Taxi Affiliation Services | 37583 |

Ejercicio 4

Recupera los identificadores de los barrios de O'Hare y Loop de la tabla *neighborhoods*.

SELECT

neighborhood\_id, -- Recupera el identificador del barrio.

name -- Recupera el nombre del barrio.

FROM

neighborhoods

WHERE

name IN ('O''Hare', 'Loop'); -- Filtra por los nombres de los barrios O'Hare y Loop.

Para recuperar los identificadores de los barrios de O'Hare y Loop de la tabla neighborhoods, puedes usar una consulta SQL simple que filtre por el nombre del barrio. A continuación se muestra la consulta:

**Descripción paso a paso:**

1. **Seleccionar columnas**: Seleccionamos neighborhood\_id y name de la tabla neighborhoods.
2. **Filtrar por nombres de barrios**: Utilizamos la cláusula WHERE con la condición IN para filtrar los registros donde el name es 'O'Hare' o 'Loop'. Nótese el uso de la barra invertida (\) para escapar la comilla simple en 'O'Hare'.

Esta consulta SQL recuperará los identificadores de los barrios de O'Hare y Loop. Aquí tienes el código con los comentarios añadidos:

* name IN ('O''Hare', 'Loop'): En esta parte de la consulta, utilizamos '' para representar la comilla simple dentro de la cadena literal. Esto asegura que la consulta SQL entienda que estamos buscando los nombres exactos 'O'Hare' y 'Loop'.

Con esta corrección, la consulta debería ejecutarse sin problemas y recuperar los identificadores de los barrios de O'Hare y Loop de la tabla neighborhoods.

Ejercicio 5

Para cada hora recupera los registros de condiciones meteorológicas de la tabla *weather\_records*. Usando el operador CASE, divide todas las horas en dos grupos: Bad si el campo *description* contiene las palabras rain o storm, y Good para los demás. Nombra el campo resultante *weather\_conditions*. La tabla final debe incluir dos campos: fecha y hora (*ts*) y *weather\_conditions*.

Para lograr esto, necesitamos recuperar registros de condiciones meteorológicas por cada hora de la tabla weather\_records, y luego usar un operador CASE para clasificar las condiciones meteorológicas como "Bad" o "Good" según el campo description. Aquí está la consulta SQL que cumple con estos requisitos:

sql

Copiar código

SELECT

ts AS fecha\_y\_hora, -- Renombramos el campo ts como fecha\_y\_hora.

CASE

WHEN LOWER(description) LIKE '%rain%' OR LOWER(description) LIKE '%storm%' THEN 'Bad'

ELSE 'Good'

END AS weather\_conditions -- Usamos CASE para clasificar las condiciones meteorológicas.

FROM

weather\_records

ORDER BY

ts; -- Ordenamos por ts, asumiendo que representa la fecha y hora de cada registro.

**Explicación paso a paso:**

1. **Seleccionar columnas**: Seleccionamos ts de la tabla weather\_records como fecha\_y\_hora para representar la fecha y hora del registro meteorológico.
2. **Usar CASE para clasificar condiciones meteorológicas**: Utilizamos un operador CASE para evaluar el campo description. Si el texto en description contiene las palabras "rain" o "storm" (considerando mayúsculas y minúsculas con LOWER()), se clasifica como 'Bad'. En caso contrario, se clasifica como 'Good'.
3. **Alias del campo resultante**: El resultado del CASE se asigna al campo weather\_conditions.
4. **Ordenar resultados**: Ordenamos los resultados por ts para asegurar que los registros se presenten en orden cronológico.

Esta consulta SQL generará una tabla que incluye la fecha y hora del registro meteorológico (fecha\_y\_hora) y las condiciones meteorológicas clasificadas como "Bad" o "Good" (weather\_conditions). Asegúrate de ajustar el ordenamiento (ORDER BY ts) según la estructura de tu tabla weather\_records y el formato de fecha y hora utilizado.

| **fecha\_y\_hora** | **weather\_conditions** |
| --- | --- |
| 2017-11-01 00:00:00 | Good |
| 2017-11-01 01:00:00 | Good |
| 2017-11-01 02:00:00 | Good |
| 2017-11-01 03:00:00 | Good |
| 2017-11-01 04:00:00 | Good |
| 2017-11-01 05:00:00 | Good |
| 2017-11-01 06:00:00 | Good |
| 2017-11-01 07:00:00 | Good |
| 2017-11-01 08:00:00 | Good |
| 2017-11-01 09:00:00 | Good |
| 2017-11-01 10:00:00 | Good |
| 2017-11-01 11:00:00 | Good |
| 2017-11-01 12:00:00 | Good |
| 2017-11-01 13:00:00 | Good |
| 2017-11-01 14:00:00 | Good |
| 2017-11-01 15:00:00 | Good |
| 2017-11-01 16:00:00 | Good |
| 2017-11-01 17:00:00 | Good |
| 2017-11-01 18:00:00 | Good |
| 2017-11-01 19:00:00 | Good |
| 2017-11-01 20:00:00 | Good |
| 2017-11-01 21:00:00 | Good |
| 2017-11-01 22:00:00 | Good |
| 2017-11-01 23:00:00 | Good |
| 2017-11-02 00:00:00 | Good |
| 2017-11-02 01:00:00 | Good |
| 2017-11-02 02:00:00 | Good |
| 2017-11-02 03:00:00 | Bad |
| 2017-11-02 04:00:00 | Bad |
| 2017-11-02 05:00:00 | Bad |
| 2017-11-02 06:00:00 | Bad |
| 2017-11-02 07:00:00 | Bad |
| 2017-11-02 08:00:00 | Good |
| 2017-11-02 09:00:00 | Good |
| 2017-11-02 10:00:00 | Good |
| 2017-11-02 11:00:00 | Good |
| 2017-11-02 12:00:00 | Bad |
| 2017-11-02 13:00:00 | Good |
| 2017-11-02 14:00:00 | Good |
| 2017-11-02 15:00:00 | Good |
| 2017-11-02 16:00:00 | Good |
| 2017-11-02 17:00:00 | Good |
| 2017-11-02 18:00:00 | Good |
| 2017-11-02 19:00:00 | Good |
| 2017-11-02 20:00:00 | Bad |
| 2017-11-02 21:00:00 | Bad |
| 2017-11-02 22:00:00 | Good |
| 2017-11-02 23:00:00 | Good |
| 2017-11-03 00:00:00 | Bad |
| 2017-11-03 01:00:00 | Good |
| 2017-11-03 02:00:00 | Good |
| 2017-11-03 03:00:00 | Good |
| 2017-11-03 04:00:00 | Good |
| 2017-11-03 05:00:00 | Good |
| 2017-11-03 06:00:00 | Good |
| 2017-11-03 07:00:00 | Good |
| 2017-11-03 08:00:00 | Good |
| 2017-11-03 09:00:00 | Good |
| 2017-11-03 10:00:00 | Good |
| 2017-11-03 11:00:00 | Good |
| 2017-11-03 12:00:00 | Good |
| 2017-11-03 13:00:00 | Good |
| 2017-11-03 14:00:00 | Good |
| 2017-11-03 15:00:00 | Good |
| 2017-11-03 16:00:00 | Good |
| 2017-11-03 17:00:00 | Good |
| 2017-11-03 18:00:00 | Good |
| 2017-11-03 19:00:00 | Good |
| 2017-11-03 20:00:00 | Good |
| 2017-11-03 21:00:00 | Good |
| 2017-11-03 22:00:00 | Good |
| 2017-11-03 23:00:00 | Good |
| 2017-11-04 00:00:00 | Good |
| 2017-11-04 01:00:00 | Good |
| 2017-11-04 02:00:00 | Good |
| 2017-11-04 03:00:00 | Good |
| 2017-11-04 04:00:00 | Good |
| 2017-11-04 05:00:00 | Good |
| 2017-11-04 06:00:00 | Good |
| 2017-11-04 07:00:00 | Good |
| 2017-11-04 08:00:00 | Good |
| 2017-11-04 09:00:00 | Good |
| 2017-11-04 10:00:00 | Good |
| 2017-11-04 11:00:00 | Good |
| 2017-11-04 12:00:00 | Good |
| 2017-11-04 13:00:00 | Good |
| 2017-11-04 14:00:00 | Good |
| 2017-11-04 15:00:00 | Good |
| 2017-11-04 16:00:00 | Bad |
| 2017-11-04 17:00:00 | Bad |
| 2017-11-04 18:00:00 | Bad |
| 2017-11-04 19:00:00 | Good |
| 2017-11-04 20:00:00 | Good |
| 2017-11-04 21:00:00 | Good |
| 2017-11-04 22:00:00 | Good |
| 2017-11-04 23:00:00 | Good |
| 2017-11-05 00:00:00 | Good |
| 2017-11-05 01:00:00 | Bad |
| 2017-11-05 02:00:00 | Good |
| 2017-11-05 03:00:00 | Good |
| 2017-11-05 04:00:00 | Bad |
| 2017-11-05 05:00:00 | Bad |
| 2017-11-05 06:00:00 | Good |
| 2017-11-05 07:00:00 | Good |
| 2017-11-05 08:00:00 | Good |
| 2017-11-05 09:00:00 | Good |
| 2017-11-05 10:00:00 | Good |
| 2017-11-05 11:00:00 | Good |
| 2017-11-05 12:00:00 | Good |
| 2017-11-05 13:00:00 | Good |
| 2017-11-05 14:00:00 | Bad |
| 2017-11-05 15:00:00 | Good |
| 2017-11-05 16:00:00 | Bad |
| 2017-11-05 17:00:00 | Good |
| 2017-11-05 18:00:00 | Bad |
| 2017-11-05 19:00:00 | Bad |
| 2017-11-05 20:00:00 | Bad |
| 2017-11-05 21:00:00 | Good |
| 2017-11-05 22:00:00 | Good |
| 2017-11-05 23:00:00 | Good |
| 2017-11-06 00:00:00 | Good |
| 2017-11-06 01:00:00 | Good |
| 2017-11-06 02:00:00 | Good |
| 2017-11-06 03:00:00 | Good |
| 2017-11-06 04:00:00 | Good |
| 2017-11-06 05:00:00 | Good |
| 2017-11-06 06:00:00 | Good |
| 2017-11-06 07:00:00 | Good |
| 2017-11-06 08:00:00 | Good |
| 2017-11-06 09:00:00 | Good |
| 2017-11-06 10:00:00 | Good |
| 2017-11-06 11:00:00 | Good |
| 2017-11-06 12:00:00 | Good |
| 2017-11-06 13:00:00 | Good |
| 2017-11-06 14:00:00 | Good |
| 2017-11-06 15:00:00 | Good |
| 2017-11-06 16:00:00 | Good |
| 2017-11-06 17:00:00 | Good |
| 2017-11-06 18:00:00 | Good |
| 2017-11-06 19:00:00 | Good |
| 2017-11-06 20:00:00 | Good |
| 2017-11-06 21:00:00 | Good |
| 2017-11-06 22:00:00 | Good |
| 2017-11-06 23:00:00 | Good |
| 2017-11-07 00:00:00 | Good |
| 2017-11-07 01:00:00 | Good |
| 2017-11-07 02:00:00 | Good |
| 2017-11-07 03:00:00 | Good |
| 2017-11-07 04:00:00 | Good |
| 2017-11-07 05:00:00 | Good |
| 2017-11-07 06:00:00 | Good |
| 2017-11-07 07:00:00 | Good |
| 2017-11-07 08:00:00 | Good |
| 2017-11-07 09:00:00 | Good |
| 2017-11-07 10:00:00 | Good |
| 2017-11-07 11:00:00 | Good |
| 2017-11-07 12:00:00 | Good |
| 2017-11-07 13:00:00 | Good |
| 2017-11-07 14:00:00 | Good |
| 2017-11-07 15:00:00 | Good |
| 2017-11-07 16:00:00 | Good |
| 2017-11-07 17:00:00 | Good |
| 2017-11-07 18:00:00 | Good |
| 2017-11-07 19:00:00 | Good |
| 2017-11-07 20:00:00 | Good |
| 2017-11-07 21:00:00 | Good |
| 2017-11-07 22:00:00 | Good |
| 2017-11-07 23:00:00 | Bad |
| 2017-11-08 00:00:00 | Bad |
| 2017-11-08 01:00:00 | Good |
| 2017-11-08 02:00:00 | Good |
| 2017-11-08 03:00:00 | Good |
| 2017-11-08 04:00:00 | Good |
| 2017-11-08 05:00:00 | Good |
| 2017-11-08 06:00:00 | Good |
| 2017-11-08 07:00:00 | Good |
| 2017-11-08 08:00:00 | Good |
| 2017-11-08 09:00:00 | Good |
| 2017-11-08 10:00:00 | Good |
| 2017-11-08 11:00:00 | Good |
| 2017-11-08 12:00:00 | Good |
| 2017-11-08 13:00:00 | Good |
| 2017-11-08 14:00:00 | Good |
| 2017-11-08 15:00:00 | Good |
| 2017-11-08 16:00:00 | Good |
| 2017-11-08 17:00:00 | Good |
| 2017-11-08 18:00:00 | Good |
| 2017-11-08 19:00:00 | Good |
| 2017-11-08 20:00:00 | Good |
| 2017-11-08 21:00:00 | Good |
| 2017-11-08 22:00:00 | Good |
| 2017-11-08 23:00:00 | Good |
| 2017-11-09 00:00:00 | Good |
| 2017-11-09 01:00:00 | Good |
| 2017-11-09 02:00:00 | Good |
| 2017-11-09 03:00:00 | Good |
| 2017-11-09 04:00:00 | Good |
| 2017-11-09 05:00:00 | Good |
| 2017-11-09 06:00:00 | Good |
| 2017-11-09 07:00:00 | Good |

Ejercicio 6

Recupera de la tabla de *trips* todos los viajes que comenzaron en el Loop (*pickup\_location\_id:* 50) el sábado y terminaron en O'Hare (*dropoff\_location\_id*: 63). Obtén las condiciones climáticas para cada viaje. Utiliza el método que aplicaste en la tarea anterior. Recupera también la duración de cada viaje. Ignora los viajes para los que no hay datos disponibles sobre las condiciones climáticas.

Las columnas de la tabla deben estar en el siguiente orden:

* *start\_ts*
* *weather\_conditions*
* *duration\_seconds*

Ordena por *trip\_id.*

Entendido, vamos a retomar la consulta anterior pero sin abreviar las variables y asegurándonos de incluir todos los detalles solicitados:

sql

Copiar código

SELECT

trips.start\_ts AS start\_ts,

CASE

WHEN LOWER(weather.description) LIKE '%rain%' OR LOWER(weather.description) LIKE '%storm%' THEN 'Bad'

ELSE 'Good'

END AS weather\_conditions,

trips.duration\_seconds AS duration\_seconds

FROM

trips

JOIN

weather\_records weather ON DATE\_TRUNC('hour', trips.start\_ts) = DATE\_TRUNC('hour', weather.ts)

WHERE

trips.pickup\_location\_id = 50 -- Viajes que comenzaron en el Loop (pickup\_location\_id: 50).

AND trips.dropoff\_location\_id = 63 -- Viajes que terminaron en O'Hare (dropoff\_location\_id: 63).

AND EXTRACT('dow' FROM trips.start\_ts) = 6 -- Filtrar solo los viajes que comenzaron un sábado (dow=6).

AND weather.description IS NOT NULL -- Ignorar los viajes sin datos disponibles sobre las condiciones climáticas.

ORDER BY

trips.trip\_id; -- Ordenar por trip\_id.

**Explicación detallada:**

1. **Seleccionar columnas**: Seleccionamos trips.start\_ts para la fecha y hora de inicio del viaje, trips.duration\_seconds para la duración del viaje y usamos CASE para clasificar las condiciones climáticas como "Bad" o "Good".
2. **Unir tablas y obtener condiciones climáticas**: Unimos trips con weather\_records usando DATE\_TRUNC para asegurar que coincidamos con las condiciones climáticas por cada hora del viaje (trips.start\_ts y weather.ts). Usamos un CASE dentro de SELECT para clasificar las condiciones meteorológicas.
3. **Filtrar por condiciones**: Filtramos los viajes que comenzaron en el Loop (pickup\_location\_id: 50) y terminaron en O'Hare (dropoff\_location\_id: 63). Además, filtramos para asegurar que los viajes comenzaron un sábado (EXTRACT('dow' FROM trips.start\_ts) = 6).
4. **Ignorar viajes sin datos climáticos**: Usamos weather.description IS NOT NULL para asegurar que solo seleccionamos viajes para los cuales tenemos datos disponibles sobre las condiciones climáticas.
5. **Ordenar resultados**: Ordenamos los resultados por trips.trip\_id para mantener un orden coherente.

Esta consulta SQL te proporcionará los detalles de todos los viajes que comenzaron en el Loop y terminaron en O'Hare un sábado, incluyendo la fecha y hora de inicio (start\_ts), las condiciones climáticas clasificadas (weather\_conditions) y la duración del viaje en segundos (duration\_seconds). Asegúrate de ajustar los nombres exactos de las tablas y los campos según la estruct

| **start\_ts** | **weather\_conditions** | **duration\_seconds** |
| --- | --- | --- |
| 2017-11-25 12:00:00 | Good | 1380 |
| 2017-11-25 16:00:00 | Good | 2410 |
| 2017-11-25 14:00:00 | Good | 1920 |
| 2017-11-25 12:00:00 | Good | 1543 |
| 2017-11-04 10:00:00 | Good | 2512 |
| 2017-11-11 07:00:00 | Good | 1440 |
| 2017-11-11 04:00:00 | Good | 1320 |
| 2017-11-04 16:00:00 | Bad | 2969 |
| 2017-11-18 11:00:00 | Good | 2280 |
| 2017-11-04 16:00:00 | Bad | 3120 |
| 2017-11-11 15:00:00 | Good | 4800 |
| 2017-11-04 05:00:00 | Good | 1260 |
| 2017-11-11 06:00:00 | Good | 1346 |
| 2017-11-04 04:00:00 | Good | 1333 |
| 2017-11-04 11:00:00 | Good | 2574 |
| 2017-11-11 12:00:00 | Good | 2441 |
| 2017-11-04 14:00:00 | Good | 3300 |
| 2017-11-11 14:00:00 | Good | 2460 |
| 2017-11-11 12:00:00 | Good | 2040 |
| 2017-11-18 06:00:00 | Good | 1500 |
| 2017-11-04 11:00:00 | Good | 2040 |
| 2017-11-11 08:00:00 | Good | 1470 |
| 2017-11-04 08:00:00 | Good | 1546 |
| 2017-11-11 16:00:00 | Good | 2100 |
| 2017-11-25 13:00:00 | Good | 60 |
| 2017-11-04 12:00:00 | Good | 2640 |
| 2017-11-25 10:00:00 | Good | 1502 |
| 2017-11-11 12:00:00 | Good | 1915 |
| 2017-11-04 12:00:00 | Good | 2769 |
| 2017-11-11 13:00:00 | Good | 2250 |
| 2017-11-11 04:00:00 | Good | 1260 |
| 2017-11-18 14:00:00 | Good | 2826 |
| 2017-11-04 14:00:00 | Good | 3360 |
| 2017-11-04 14:00:00 | Good | 3180 |
| 2017-11-25 20:00:00 | Good | 2100 |
| 2017-11-04 10:00:00 | Good | 1800 |
| 2017-11-11 12:00:00 | Good | 2289 |
| 2017-11-04 08:00:00 | Good | 1494 |
| 2017-11-11 11:00:00 | Good | 1560 |
| 2017-11-18 12:00:00 | Bad | 1980 |
| 2017-11-11 13:00:00 | Good | 2115 |
| 2017-11-11 10:00:00 | Good | 1506 |
| 2017-11-04 12:00:00 | Good | 2580 |
| 2017-11-04 17:00:00 | Bad | 2460 |
| 2017-11-11 09:00:00 | Good | 1620 |
| 2017-11-04 06:00:00 | Good | 1163 |
| 2017-11-04 05:00:00 | Good | 1533 |
| 2017-11-11 04:00:00 | Good | 1477 |
| 2017-11-11 19:00:00 | Good | 1984 |
| 2017-11-04 13:00:00 | Good | 2940 |
| 2017-11-04 07:00:00 | Good | 1320 |
| 2017-11-04 06:00:00 | Good | 1440 |
| 2017-11-11 06:00:00 | Good | 1260 |
| 2017-11-11 08:00:00 | Good | 1560 |
| 2017-11-04 09:00:00 | Good | 1683 |
| 2017-11-11 05:00:00 | Good | 1343 |
| 2017-11-18 06:00:00 | Good | 1742 |
| 2017-11-04 09:00:00 | Good | 1560 |
| 2017-11-11 08:00:00 | Good | 1358 |
| 2017-11-11 12:00:00 | Good | 1980 |
| 2017-11-04 16:00:00 | Bad | 2760 |
| 2017-11-18 12:00:00 | Bad | 2460 |
| 2017-11-18 10:00:00 | Bad | 1440 |
| 2017-11-25 14:00:00 | Good | 1620 |
| 2017-11-11 08:00:00 | Good | 1415 |
| 2017-11-25 05:00:00 | Good | 1325 |
| 2017-11-18 06:00:00 | Good | 2100 |
| 2017-11-11 08:00:00 | Good | 1320 |
| 2017-11-11 10:00:00 | Good | 1260 |
| 2017-11-11 06:00:00 | Good | 1260 |
| 2017-11-11 08:00:00 | Good | 1200 |
| 2017-11-25 08:00:00 | Good | 1320 |
| 2017-11-11 18:00:00 | Good | 2280 |
| 2017-11-25 10:00:00 | Good | 1320 |
| 2017-11-04 06:00:00 | Good | 1140 |
| 2017-11-11 18:00:00 | Good | 2520 |
| 2017-11-18 16:00:00 | Bad | 3000 |
| 2017-11-11 19:00:00 | Good | 1920 |
| 2017-11-04 18:00:00 | Bad | 2363 |
| 2017-11-04 14:00:00 | Good | 3084 |
| 2017-11-11 08:00:00 | Good | 1380 |
| 2017-11-11 08:00:00 | Good | 1380 |
| 2017-11-04 09:00:00 | Good | 1380 |
| 2017-11-04 09:00:00 | Good | 1380 |
| 2017-11-11 12:00:00 | Good | 2213 |
| 2017-11-11 08:00:00 | Good | 1140 |
| 2017-11-11 10:00:00 | Good | 1435 |
| 2017-11-04 10:00:00 | Good | 2460 |
| 2017-11-11 07:00:00 | Good | 1200 |
| 2017-11-04 15:00:00 | Good | 3201 |
| 2017-11-11 11:00:00 | Good | 2074 |
| 2017-11-18 11:00:00 | Good | 2843 |
| 2017-11-11 17:00:00 | Good | 2426 |
| 2017-11-04 09:00:00 | Good | 1740 |
| 2017-11-25 07:00:00 | Good | 2340 |
| 2017-11-18 05:00:00 | Good | 2075 |
| 2017-11-18 07:00:00 | Bad | 1511 |
| 2017-11-11 18:00:00 | Good | 2220 |
| 2017-11-04 10:00:00 | Good | 2551 |
| 2017-11-11 16:00:00 | Good | 2062 |
| 2017-11-04 12:00:00 | Good | 2999 |
| 2017-11-04 08:00:00 | Good | 1677 |
| 2017-11-04 06:00:00 | Good | 1177 |
| 2017-11-11 06:00:00 | Good | 1475 |
| 2017-11-25 08:00:00 | Good | 1277 |
| 2017-11-11 04:00:00 | Good | 1213 |
| 2017-11-18 13:00:00 | Bad | 4044 |
| 2017-11-04 21:00:00 | Good | 1680 |
| 2017-11-04 18:00:00 | Bad | 1980 |
| 2017-11-25 18:00:00 | Good | 2760 |
| 2017-11-11 09:00:00 | Good | 1380 |
| 2017-11-11 07:00:00 | Good | 1380 |
| 2017-11-18 08:00:00 | Bad | 1320 |
| 2017-11-11 16:00:00 | Good | 2591 |
| 2017-11-11 08:00:00 | Good | 1260 |
| 2017-11-11 07:00:00 | Good | 1380 |
| 2017-11-11 10:00:00 | Good | 1440 |
| 2017-11-04 14:00:00 | Good | 3240 |
| 2017-11-04 16:00:00 | Bad | 2782 |
| 2017-11-04 14:00:00 | Good | 3120 |
| 2017-11-04 19:00:00 | Good | 1869 |
| 2017-11-11 06:00:00 | Good | 1218 |
| 2017-11-11 11:00:00 | Good | 1620 |
| 2017-11-11 10:00:00 | Good | 1380 |
| 2017-11-11 08:00:00 | Good | 1380 |
| 2017-11-11 09:00:00 | Good | 1380 |
| 2017-11-11 07:00:00 | Good | 1320 |
| 2017-11-04 14:00:00 | Good | 3300 |
| 2017-11-04 13:00:00 | Good | 3060 |
| 2017-11-11 13:00:00 | Good | 2100 |
| 2017-11-18 14:00:00 | Good | 3540 |
| 2017-11-11 21:00:00 | Good | 1920 |
| 2017-11-11 17:00:00 | Good | 2160 |
| 2017-11-11 13:00:00 | Good | 2123 |
| 2017-11-11 07:00:00 | Good | 1384 |
| 2017-11-11 10:00:00 | Good | 1260 |
| 2017-11-18 15:00:00 | Good | 3480 |
| 2017-11-11 12:00:00 | Good | 2071 |
| 2017-11-18 13:00:00 | Bad | 3300 |
| 2017-11-25 13:00:00 | Good | 1560 |
| 2017-11-04 12:00:00 | Good | 2760 |
| 2017-11-18 12:00:00 | Bad | 3024 |
| 2017-11-11 11:00:00 | Good | 1380 |
| 2017-11-25 06:00:00 | Good | 1200 |
| 2017-11-11 06:00:00 | Good | 1667 |
| 2017-11-18 18:00:00 | Good | 2056 |
| 2017-11-11 10:00:00 | Good | 1473 |
| 2017-11-11 17:00:00 | Good | 2460 |
| 2017-11-11 10:00:00 | Good | 1740 |
| 2017-11-11 14:00:00 | Good | 2340 |
| 2017-11-04 16:00:00 | Bad | 3180 |
| 2017-11-04 11:00:00 | Good | 2220 |
| 2017-11-11 12:00:00 | Good | 2240 |
| 2017-11-04 14:00:00 | Good | 2778 |
| 2017-11-18 06:00:00 | Good | 1420 |
| 2017-11-04 14:00:00 | Good | 3480 |
| 2017-11-25 20:00:00 | Good | 1980 |
| 2017-11-18 10:00:00 | Bad | 2055 |
| 2017-11-11 15:00:00 | Good | 2380 |
| 2017-11-04 08:00:00 | Good | 1539 |
| 2017-11-25 10:00:00 | Good | 1591 |
| 2017-11-18 14:00:00 | Good | 2588 |
| 2017-11-11 07:00:00 | Good | 0 |
| 2017-11-04 22:00:00 | Good | 1380 |
| 2017-11-04 12:00:00 | Good | 2220 |
| 2017-11-04 08:00:00 | Good | 1380 |
| 2017-11-04 14:00:00 | Good | 2820 |
| 2017-11-11 09:00:00 | Good | 0 |
| 2017-11-18 09:00:00 | Bad | 1260 |
| 2017-11-18 13:00:00 | Bad | 2940 |
| 2017-11-18 16:00:00 | Bad | 2340 |
| 2017-11-18 12:00:00 | Bad | 2220 |
| 2017-11-25 11:00:00 | Good | 1140 |
| 2017-11-11 10:00:00 | Good | 1239 |
| 2017-11-04 16:00:00 | Bad | 3130 |
| 2017-11-18 15:00:00 | Good | 2877 |
| 2017-11-11 03:00:00 | Good | 1162 |
| 2017-11-04 14:00:00 | Good | 3060 |
| 2017-11-11 13:00:00 | Good | 1680 |
| 2017-11-04 10:00:00 | Good | 2112 |
| 2017-11-04 11:00:00 | Good | 2328 |
| 2017-11-11 05:00:00 | Good | 1504 |
| 2017-11-04 06:00:00 | Good | 1439 |
| 2017-11-18 16:00:00 | Bad | 2811 |
| 2017-11-18 11:00:00 | Good | 2094 |
| 2017-11-11 06:00:00 | Good | 1430 |
| 2017-11-18 12:00:00 | Bad | 3026 |
| 2017-11-04 14:00:00 | Good | 3120 |
| 2017-11-25 12:00:00 | Good | 1380 |
| 2017-11-18 14:00:00 | Good | 2994 |
| 2017-11-11 11:00:00 | Good | 1620 |
| 2017-11-04 12:00:00 | Good | 2640 |
| 2017-11-18 00:00:00 | Bad | 480 |
| 2017-11-18 19:00:00 | Good | 0 |
| 2017-11-11 10:00:00 | Good | 1414 |
| 2017-11-11 12:00:00 | Good | 1981 |
| 2017-11-11 08:00:00 | Good | 1200 |
| 2017-11-04 11:00:00 | Good | 2160 |
| 2017-11-11 15:00:00 | Good | 2400 |
| 2017-11-11 20:00:00 | Good | 1500 |

ura de tu base de datos.

Paso 4. Análisis exploratorio de datos (Python)

Además de los datos que recuperaste en las tareas anteriores te han dado un segundo archivo. Ahora tienes estos dos CSV:

[/datasets/project\_sql\_result\_01.csv](https://practicum-content.s3.us-west-1.amazonaws.com/learning-materials/data-analyst-eng/moved_project_sql_result_01.csv). contiene los siguientes datos:

*company\_name*: nombre de la empresa de taxis

*trips\_amount*: el número de viajes de cada compañía de taxis el 15 y 16 de noviembre de 2017.

[/datasets/project\_sql\_result\_04.csv](https://practicum-content.s3.us-west-1.amazonaws.com/learning-materials/data-analyst-eng/moved_project_sql_result_04.csv). contiene los siguientes datos:

*dropoff\_location\_name*: barrios de Chicago donde finalizaron los viajes

*average\_trips*: el promedio de viajes que terminaron en cada barrio en noviembre de 2017.

Para estos dos datasets ahora necesitas

* importar los archivos
* estudiar los datos que contienen
* asegurarte de que los tipos de datos sean correctos
* identificar los 10 principales barrios en términos de finalización del recorrido
* hacer gráficos: empresas de taxis y número de viajes, los 10 barrios principales por número de finalizaciones
* sacar conclusiones basadas en cada gráfico y explicar los resultados

Paso 5. Prueba de hipótesis (Python)

[/datasets/project\_sql\_result\_07.csv](https://practicum-content.s3.us-west-1.amazonaws.com/learning-materials/data-analyst-eng/moved_project_sql_result_07.csv) — el resultado de la última consulta. Contiene datos sobre viajes desde el Loop hasta el Aeropuerto Internacional O'Hare. Recuerda, estos son los valores de campo de la tabla:

* *start\_ts*: fecha y hora de la recogida
* *weather\_conditions*: condiciones climáticas en el momento en el que comenzó el viaje
* *duration\_seconds*: duración del viaje en segundos

Prueba la hipótesis:

"La duración promedio de los viajes desde el Loop hasta el Aeropuerto Internacional O'Hare cambia los sábados lluviosos".

Decide por tu cuenta dónde establecer el nivel de significación (alfa).

Explica:

* cómo planteaste las hipótesis nula y alternativa
* qué criterio usaste para probar las hipótesis y por qué

**¿Cómo será evaluado mi proyecto?**

Estos son los criterios de evaluación del proyecto. Léelos atentamente antes de empezar a trabajar.

Esto es lo que buscará el revisor del proyecto al evaluar tu proyecto:

* cómo recuperas los datos del sitio web
* cómo creas slices de datos
* cómo agrupas los datos
* si usas correctamente los diversos métodos para combinar datos
* cómo formulas las hipótesis
* qué criterios utilizas para probar las hipótesis y por qué
* a qué conclusiones llegas
* si dejas comentarios en cada paso

Las fichas de ayuda y resúmenes de las lecciones anteriores tienen todo lo que necesitas para completar el proyecto.