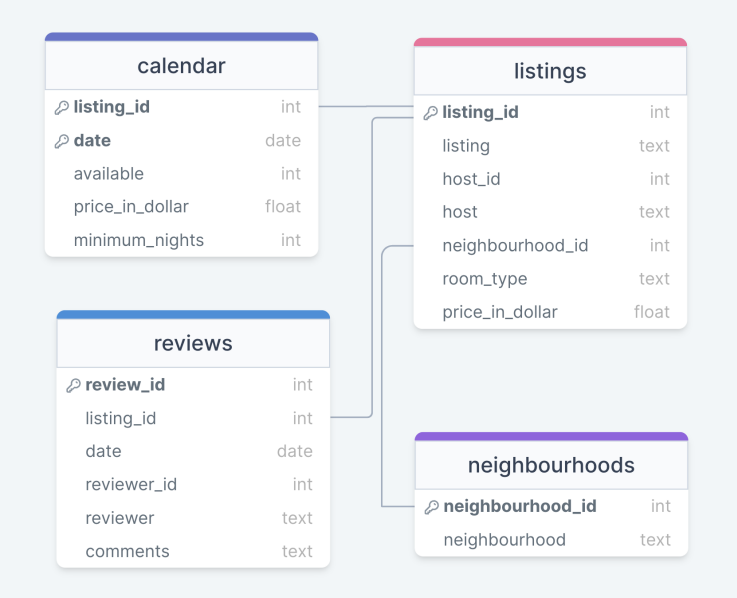
```
%%capture
!pip install git+https://github.com/sb2nov/sql-cc.git


import pandas as pd
from IPython.display import display, HTML
import sqlcc
from sqlcc import check

# Show all the rows (instead of only a few)
pd.set_option("display.max_rows", None)

# Set precision to max 2 decimals
pd.set_option('display.precision', 2)  # Use 'display.precision' instead of 'precision'

# Set CSS Style for Table
# Make it work with night & light mode
# - Alternating rows
# - th elements
# - td elements
css_style = '''
<style>
  html {
    --td-font-color: black;
    --font-color: black;
    --background-color: #e0e0e0;
  }
  html[theme=dark] {
    --td-font-color: white;
    --font-color: black;
    --background-color: #6688ff;
  }
  th {
    background: #fbd44c;
    color: var(--font-color);
    font-size: 16px;
    text-align: center;
    font-weight: bold;
  }
  tr:nth-child(even) {
    background-color: var(--background-color);
    color: var(--font-color);
  }
  td {
    font-size: 14px;
    color: var(--td-font-color);
  }
</style>
'''
```

```
def run(sql_query):
  df = sqlcc.run(sql_query)

  # Puts the scrollbar next to the DataFrame
  display(HTML(css_style +
               "<div style='max-height: 500px; overflow: auto; width: fit-content; border-style: solid;" +
               " border-width: 1px; border-color: #0139fe; font-family: GT Planar,Inter,Arial,sans-serif;'>" +
               df.style.render() +
               "</div>"))
```

Introduction

Airbnb Sydney database contains 4 tables: **listings**, **neighbourhoods**, **reviews**, and **calendar**.




```
### Question: retrieve data
query = """
SELECT * FROM listings
"""

run(query)
```

```
<ipython-input-29-86fbc71b6552>:55: FutureWarning: this method is deprecated in favour of `Styler.to_html()`
  df.style.render() +
```

	listing_id	listing	host_id	host	neighbourhood_id	room_type	price_in_dollar
0	574105250645758912	Peaceful 1 Bedroom Apartment in Bondi Beach	109067745	Andrew	35	Entire home/apt	150.000000
1	7874902	Bondi Vibes - Funky Designer Studio	41506490	Alex White	35	Entire home/apt	99.000000
2	4575789	Nice studio close to the beach!	22980172	María	35	Entire home/apt	125.000000
3	23077495	Just bring your beach towel	1305312	Gladys	35	Entire home/apt	119.000000
4	657377039990074112	Stylish lite 2b+2bth mod secure Beach apt with pkg	285488167	Rick	35	Entire home/apt	408.000000
5	53798702	Spacious and clean 2-bedroom apartment in Bondi	244604436	Tiina	35	Entire home/apt	96.000000
6	4344478	Bondi Beach Apartment 50m to beach	22553304	Stephen	35	Entire home/apt	101.000000
7	48699778	Calm & Coastal: Bronte Beach Studio with Parking	185783910	Annie	35	Entire home/apt	159.000000
8	12072720	Minutes to Bronte beach Ocean views	11745874	Angelika	35	Entire home/apt	210.000000
9	20255786	Terrace in heart of Bondi Junction	18901875	Astrid And Nick	35	Entire home/apt	850.000000
10	22973428	Bright stylish home in a great location	2537559	Sian	35	Entire home/apt	120.000000
11	22199388	Double room available in open and airy Bronte home	14228436	Audrey	35	Private room	43.000000
		Bondi Beachy - 2 bedroom in the heart of		Bondi Beach Holiday		Entire	

This vertically-scrollable table displays all of the data contained in the listings table.

Exploring data in other tables

```
### Question: Fetch data from the entire neighbourhoods table

query = """
```

```
query =
SELECT * FROM neighbourhoods
"""

run(query)
```

```
<ipython-input-29-86fbc71b6552>:55: FutureWarning: this method is deprecated in favour of `Styler.to_html()`
  df.style.render() +
```

	neighbourhood_id	neighbourhood
0	0	Ashfield
1	1	Auburn
2	2	Bankstown
3	3	Blacktown
4	4	Botany Bay
5	5	Burwood
6	6	Camden
7	7	Campbelltown
8	8	Canada Bay
9	9	Canterbury
10	10	City Of Kogarah
11	11	Fairfield
12	12	Holroyd
13	13	Hornsby
14	14	Hunters Hill
15	15	Hurstville
16	16	Ku-Ring-Gai
17	17	Lane Cove
18	18	Leichhardt
19	19	Liverpool

```
### Question: Fetch data from the entire reviews table
query = """
SELECT * FROM reviews
"""

run(query)
```

```
<ipython-input-29-86fbc71b6552>:55: FutureWarning: this method is deprecated in favour of `Styler.to_html()`
  df.style.render() +
```

	listing_id	review_id	date	reviewer_id	reviewer	comments
0	2033151	9717326	2014-01-06T00:00:00Z	10558887	James	Great host, great location and very comfortable room. Many thanks.
1	4344478	62879688	2016-02-18T00:00:00Z	40858536	Andrew	We had a fantastic time in Bondi and staying at Stephen's pad made it more worthwhile. Besides being located a few meters from the beach... gym, bus stops, eateries and supermarkets were all less than a 5min walk away. Stephen was a legend being just a text away if we needed any help. Top notch and friendly guy! We'll be back fosho!
2	7846383	114619982	2016-11-19T00:00:00Z	73029331	Florian	It was very nice to stay two weeks by Marcela. The flat war nice and clean and near to the beach. I allways eat my breakfast on the balconary. Short bus trip to city. Marcela helps me when i have questions about activities in Sydney. Thank you for the nice time.
3	16411678	123595331	2016-12-29T00:00:00Z	64486440	Kristin	The host canceled this reservation 3 days before arrival. This is an automated posting.
4	16411678	140591929	2017-03-30T00:00:00Z	14665890	Shao Ing	Alzbeta's studio is well look after, spacious and tastefully crafted. Clean yet cozy. Kitchen is well-equiped with nice setting. I enjoy the proximity to local amenities and various food choices. Alzbeta has been great with communication and a pleasure to meet in person. This is one great little studio that I would stay again. Many thanks!
5	7874902	197790998	2017-09-27T00:00:00Z	7140613	Anh	Cosy little place in a great location. Close to Bondi beach and shops. Good amenities and the bed was really comfortable!!
6	20628052	199418949	2017-10-01T00:00:00Z	477117	Cristina	Claire's place was a wonderful place to base ourselves over a long weekend. The apartment is private with a great courtyard, which was really handy to dry towels and beach gear since it's walking distance to the beach! The bed was comfortable and the kitchen was equipped with everything we needed. Claire was an incredibly hospitable host, helpful,

```
### Question: Fetch data from the entire calendar table

query = """
SELECT * FROM calendar
"""

run(query)
```

```
<ipython-input-29-86fbc71b6552>:55: FutureWarning: this method is deprecated in favour of `Styler.to_html()`
  df.style.render() +
```

	listing_id	date	available	price_in_dollar	minimum_nights
0	4344478	2022-09-09T00:00:00Z	0	101.000000	90
1	4344478	2022-09-10T00:00:00Z	0	101.000000	90
2	4344478	2022-09-11T00:00:00Z	0	101.000000	90
3	4344478	2022-09-12T00:00:00Z	0	101.000000	90
4	4344478	2022-09-13T00:00:00Z	0	101.000000	90
5	4344478	2022-09-14T00:00:00Z	0	101.000000	90
6	4344478	2022-09-15T00:00:00Z	0	101.000000	90
7	7581665	2022-09-09T00:00:00Z	1	300.000000	90
8	7581665	2022-09-10T00:00:00Z	0	300.000000	90
9	7581665	2022-09-11T00:00:00Z	0	300.000000	90
10	7581665	2022-09-12T00:00:00Z	0	300.000000	90
11	7581665	2022-09-13T00:00:00Z	1	300.000000	90
12	7581665	2022-09-14T00:00:00Z	1	300.000000	90
13	7581665	2022-09-15T00:00:00Z	1	300.000000	90

Printing list of host_id to host

Currently, the administrative system at our Airbnb Sydney office shows the host_id when the host calls our helpdesk. But this is very impersonal, and not ideal to answer the phone, “hello host_id 12345!” Due to company policy, helpdesk operators do not have access to the full database, and we also don’t have a system yet in place to offer only partial access. Still, we are determined to leave a good impression on everyone who calls us by knowing their name! One idea is to provide everyone access to a database print-out of the host_id with the host name.

```
### Question: Fetch host_id and host from the listings table
query = """
SELECT host_id, host
  FROM listings
"""

run(query)
```

```
<ipython-input-29-86fbc71b6552>:55: FutureWarning: this method is deprecated in favour of `Styler.to_html()`
  df.style.render() +
```

	host_id	host
0	109067745	Andrew
1	41506490	Alex White
2	22980172	María
3	1305312	Gladys
4	285488167	Rick
5	244604436	Tiina
6	22553304	Stephen
7	185783910	Annie
8	11745874	Angelika
9	18901875	Astrid And Nick
10	2537559	Sian
11	14228436	Audrey
12	113874	Bondi Beach Holiday Homes
13	43312353	Barry
14	37772146	Hannah
15	185126628	Evan
16	43502287	Jackie
17	1043937	Alzbeta
18	120290790	Shu
19	254427140	The Jensen Potts Point

How would a helpdesk operator find a host quickly when the list is not ordered? **Let's fix the ordering by adapting the query to make it order descendingly on host_id.**

```
### Question:
# 1/ Fetch host_id and host from the listings table;
# 2/ Make sure the output is sorted by host_id in descending order

query = """
SELECT host_id, host
  FROM listings
    ORDER BY host_id DESC
"""

run(query)
```

```
<ipython-input-29-86fbc71b6552>:55: FutureWarning: this method is deprecated in favour of `Styler.to_html()`
  df.style.render() +
```

	host_id	host
0	465696336	Bin
1	457666854	Zeam
2	440651046	Victoria
3	301753450	AirPillows
4	285488167	Rick
5	269689583	Jill
6	254427140	The Jensen Potts Point
7	244604436	Tiina
8	234423891	Duffotel
9	215158027	Peter & Stephanie
10	213725582	Shaoyong
11	185783910	Annie
12	185126628	Evan
13	169795871	Yihan
14	163419570	Zach
15	163113803	Ken
16	136836511	Fiona

Additional filters on the printed output

Unfortunately, asking employees to look at the printout of this list isn't going to be a scalable solution especially as we continually add new neighbourhoods. As a temporary solution, let's assign one neighbourhood to each of our employees so they have a smaller printout of listings to keep track of.

How would we adapt the query to be sorted by *host_id*, to display the host_id and the host, and to be restricted to the neighbourhood_id of a particular neighbourhood, let's say number 35?

```
### Question:
# 1/ Fetch only host_id, host from the listings table;
# 2/ Make sure you filtered the data to just neighbourhood_id=35;
# 3/ Make sure the output is sorted by host_id in descending order
query = """
SELECT host_id, host
  FROM listings
    WHERE neighbourhood_id=35
      ORDER BY host_id DESC
"""

run(query)
```

```
<ipython-input-29-86fbc71b6552>:55: FutureWarning: this method is deprecated in favour of `Styler.to_html()`
  df.style.render() +
```

	host_id	host
0	285488167	Rick
1	244604436	Tiina
2	185783910	Annie
3	109067745	Andrew
4	41506490	Alex White
5	22980172	María
6	22553304	Stephen
7	18901875	Astrid And Nick
8	14228436	Audrey
9	11745874	Angelika
10	2537559	Sian
11	1305312	Gladys
12	113874	Bondi Beach Holiday Homes

The number of people that we have currently employed has increased by a staggering 120%. So, in the future, there is potential to split up our list even further and divide it among many more employees who can be on-call. Our best option for now is splitting the list on the number of the neighbourhood AND room_type. **Find all the listings where we set our neighbourhood_id to 27 and "Private room".**

```
### Question:
# 1/ Fetch host_id, host from the listings table;
# 2/ Make sure you filtered the data to just neighbourhood_id=27 and room_type='Private room';
# 3/ Make sure the output is sorted by host_id in descending order
query = """
SELECT host_id, host
  FROM listings
    WHERE neighbourhood_id=27 AND room_type='Private room'
      ORDER BY host_id DESC
"""
```

```
run(query)
```

```
<ipython-input-29-86fbc71b6552>:55: FutureWarning: this method is deprecated in favour of `Styler.to_html()`
  df.style.render() +
```

	host_id	host
0	61907880	Malin
1	40070341	Marcela
2	16006077	Nathalia

You're doing a shift of answering phones, and a host calls with an important question. Her name is Maggie, and she is known to us by the *listing_id* of 7581665.

She is curious whether the changes she made to the calendar on the website are reflected in our system. Maggie generally charges 300 dollars per night. She has noticed however, that can charge a little bit more on Friday and Saturday nights. She updated the price to 350 dollars for Friday, September 9th 2022 and Saturday, September 10th in 2022. She wants to double check that change went through and if that corresponds with our database.

Please help confirm this for Maggie by writing a query for the calendar table, to provide us with all of the calendar listings between the date of 2022-09-09 and 2022-09-11.

```
### Question:
# 1/ Fetch all columns from the calendar table;
# 2/ Make sure you filtered the data to dates between 2022-09-09 and 2022-09-11
query = """
SELECT *
  FROM calendar
    WHERE date BETWEEN '2022-09-09' AND '2022-09-11'
"""
```

```
run(query)
```

```
<ipython-input-29-86fbc71b6552>:55: FutureWarning: this method is deprecated in favour of `Styler.to_html()`
  df.style.render() +
```

	listing_id	date	available	price_in_dollar	minimum_nights
0	4344478	2022-09-09T00:00:00Z	0	101.000000	90
1	4344478	2022-09-10T00:00:00Z	0	101.000000	90
2	7581665	2022-09-09T00:00:00Z	1	300.000000	90
3	7581665	2022-09-10T00:00:00Z	0	300.000000	90
4	7846383	2022-09-09T00:00:00Z	0	110.000000	2
5	7846383	2022-09-10T00:00:00Z	0	110.000000	2
6	8476652	2022-09-09T00:00:00Z	0	60.000000	90
7	8476652	2022-09-10T00:00:00Z	0	60.000000	90
8	8925052	2022-09-09T00:00:00Z	0	120.000000	90
9	8925052	2022-09-10T00:00:00Z	0	120.000000	90
10	9012529	2022-09-09T00:00:00Z	0	947.000000	90
11	9012529	2022-09-10T00:00:00Z	0	947.000000	90
12	9354832	2022-09-09T00:00:00Z	0	222.000000	90
13	9354832	2022-09-10T00:00:00Z	0	222.000000	90
14	9376988	2022-09-09T00:00:00Z	0	2000.000000	5
15	9376988	2022-09-10T00:00:00Z	0	2000.000000	5
16	10103846	2022-09-09T00:00:00Z	0	400.000000	7
17	10103846	2022-09-10T00:00:00Z	0	400.000000	7
18	10111788	2022-09-09T00:00:00Z	0	99999.000000	90
19	10111788	2022-09-10T00:00:00Z	0	99999.000000	90

Add another condition to the query of the previous step. Let's filter on listing_id 7581665.

```
### Question:
# 1/ Fetch all columns from the calendar table;
# 2/ Make sure you filtered the data to dates between 2022-09-09 and 2022-09-11;
# 3/ Make sure you filtered the data to listing_id=7581665
query = """
SELECT *
  FROM calendar
       WHERE date BETWEEN '2022-09-09' AND '2022-09-11' AND
             listing_id=7581665
"""
```

```
run(query)
```

```
<ipython-input-29-86fbc71b6552>:55: FutureWarning: this method is deprecated in favour of `Styler.to_html()`
  df.style.render() +
```

	listing_id	date	available	price_in_dollar	minimum_nights
0	7581665	2022-09-09T00:00:00Z	1	300.000000	90
1	7581665	2022-09-10T00:00:00Z	0	300.000000	90

You just got off a call with a dissatisfied host who has a *listing_id* of *657377039990074158*. His name is Rick, and he was angry that no customers seemed to book his listing for the coming week (or past weeks for that matter). He wants us to promote his listing on our website more prominently, otherwise he might stop being our customer. We believe there might be some other issues at hand. Could you investigate and share us some of your findings as to why nobody is booking Rick's listing? We suggest you look at the **calendar**, **reviews**, and **listings**. Please report any findings you deem relevant.

```
### Question:
# 1/ Fetch all columns from the calendar table;
# 2/ Make sure you filtered the data to listing_id=657377039990074158

query = """
SELECT *
  FROM calendar
      WHERE listing_id=657377039990074158
"""

run(query)
```

```
<ipython-input-29-86fbc71b6552>:55: FutureWarning: this method is deprecated in favour of `Styler.to_html()`
  df.style.render() +
```

	listing_id	date	available	price_in_dollar	minimum_nights
0	657377039990074112	2022-09-10T00:00:00Z	0	430.000000	2
1	657377039990074112	2022-09-11T00:00:00Z	0	399.000000	2
2	657377039990074112	2022-09-12T00:00:00Z	0	399.000000	2
3	657377039990074112	2022-09-13T00:00:00Z	0	399.000000	2
4	657377039990074112	2022-09-14T00:00:00Z	0	399.000000	2
5	657377039990074112	2022-09-15T00:00:00Z	0	399.000000	2

```
### Question:
# 1/ Fetch all columns from the reviews table;
# 2/ Make sure you filtered the data to listing_id=657377039990074158
query = """
SELECT *
  FROM reviews
    WHERE listing_id=657377039990074158
"""

run(query)
```

```
<ipython-input-29-86fbc71b6552>:55: FutureWarning: this method is deprecated in favour of `Styler.to_html()`
  df.style.render() +
```

listing_id	review_id	date	reviewer_id	reviewer	comments

```
### Question:
# 1/ Fetch all columns from the listings table;
# 2/ Make sure you filtered the data to listing_id=657377039990074158
query = """
SELECT *
  FROM listings
    WHERE listing_id=657377039990074158
"""

run(query)
```

```
<ipython-input-29-86fbc71b6552>:55: FutureWarning: this method is deprecated in favour of `Styler.to_html()`
  df.style.render() +
```

	listing_id	listing	host_id	host	neighbourhood_id	room_type	price_in_dollar
0	657377039990074112	Stylish lite 2b+2bth mod secure Beach apt with pkg	285488167	Rick	35	Entire home/apt	408.000000

The contributing factors we found are: the price of his apartment is a bit steep compared to others, it has no reviews, and it is not available for booking on any of the given days this week.

A host calls in and is wondering how to price her listing. She wants to price her listing at least $400 since it's a really nice space. She's concerned she'll be too expensive for Airbnb Sydney, and considering a competitor VRBO. Oh no! It's your job as GM to get her on Airbnb. First, let's write a query to show how many listings are greater than or equal to $400. You are pretty sure there are some listings at her price range.

```
### Question:
# 1/ Fetch all columns from the listings table;
# 2/ Make sure you filter listings, having a price greater than or equal to 400 dollars;
# 3/ Make sure you order the list by price_in_dollar ascendingly
query = """
SELECT *
  FROM listings
    WHERE price_in_dollar >= 400
      ORDER BY price_in_dollar
"""

run(query)
```

```
<ipython-input-29-86fbc71b6552>:55: FutureWarning: this method is deprecated in favour of `Styler.to_html()`
  df.style.render() +
```

	listing_id	listing	host_id	host	neighbourhood_id	room_type	price_in_dollar
0	10103846	Clifton Road Clovelly	51850460	Craig	27	Entire home/apt	400.000000
1	657377039990074112	Stylish lite 2b+2bth mod secure Beach apt with pkg	285488167	Rick	35	Entire home/apt	408.000000
2	21443493	Perfect beachside getaway footsteps to the beach.	88281040	Mel	20	Entire home/apt	427.000000
3	644585184959945856	Silver Beach Penthouse	440651046	Victoria	31	Entire home/apt	465.000000
4	34027782	Luxurious Cozy Bedroom, Explore @Olympic Park	163419570	Zach	1	Private room	500.000000
5	10729688	Manly Beach House	13657320	Rebecca	20	Entire home/apt	557.000000
6	20255786	Terrace in heart of Bondi Junction	18901875	Astrid And Nick	35	Entire home/apt	850.000000
7	9012529	Balgowlah Beach House	22227415	Marnie	20	Entire home/apt	947.000000
8	46159786	Warriewood Beach House	136836511	Fiona	26	Entire home/apt	1000.000000
9	9376988	Family Beach House🌴	11196275	Emma	26	Entire home/apt	2000.000000
10	10111788	******Paddington Sydney*****	51892300	Tom	37	Private room	28613.000000

Now, we want to send her a list of listings to check out greater than or equal to $400, but just Entire home/apt in neighborhood's 20 and 26 (near where she lives).

```
### Question:
# 1/ Fetch all columns from the listings table;
# 2/ Make sure you filter listings, having a price greater than or equal to than 400 dollars;
# 3/ Make sure you filter listings also on having neighbourhood id 20 or 26;
# 4/ Make sure you order the list by price_in_dollar ascendingly

query = """
SELECT *
  FROM listings
    WHERE price_in_dollar >= 400 AND neighbourhood_id IN (20,26)
      ORDER BY price_in_dollar
"""

run(query)
```

```
<ipython-input-29-86fbc71b6552>:55: FutureWarning: this method is deprecated in favour of `Styler.to_html()`
  df.style.render() +
```

	listing_id	listing	host_id	host	neighbourhood_id	room_type	price_in_dollar
0	21443493	Perfect beachside getaway footsteps to the beach.	88281040	Mel	20	Entire home/apt	427.000000
1	10729688	Manly Beach House	13657320	Rebecca	20	Entire home/apt	557.000000
2	9012529	Balgowlah Beach House	22227415	Marnie	20	Entire home/apt	947.000000
3	46159786	Warriewood Beach House	136836511	Fiona	26	Entire home/apt	1000.000000
4	9376988	Family Beach House🌴	11196275	Emma	26	Entire home/apt	2000.000000

Great, there should be a couple in those neighbourhoods! She wants to read the reviews of all these listings if possible. Retrieve the listings for all of these places?

```
### Question:
# 1/ Fetch all columns from the reviews table;
# 2/ Make sure you filter reviews based on listing ids: 21443493, 10729688, 9012529, 46159786, and 9376988
# 3/ Make sure you order the list by listing_id ascendingly
query = """
SELECT *
  FROM reviews
    WHERE listing_id IN (21443493, 10729688, 9012529, 46159786, 9376988)
      ORDER BY listing_id
"""

run(query)
```

```
<ipython-input-29-86fbc71b6552>:55: FutureWarning: this method is deprecated in favour of `Styler.to_html()`
  df.style.render() +
```

	listing_id	review_id	date	reviewer_id	reviewer	comments
0	9376988	589400048	2020-01-05T00:00:00Z	227180362	Natalie	Emma's place was a great sized family home. The outdoor area with the pool was great! The location was a walk to North Avalon beach and a 15 min walk to the main Avalon village. We all had a great stay! Thank you Emma!

Analysis reveals there to be a big difference in prices per listing (up to 500x). This is to be expected since some areas are closer to points of interests and offer more luxury.

More expensive areas can net us higher sales since our cut/fee is a percentage of the price of the listing. However, cheaper areas potentially are more likely to be booked.

So for now we are interested in those areas to analyze their avaiblility and their potential differences.

Let's first start with the cheapest neighbourhoods. The top three cheapest are neighbourhood ids number 21, 28 and 4.

```
### Question:
# 1/ Fetch all columns from the listings table
# 2/ Make sure you filter the data using WHERE-IN

query = """
SELECT *
  FROM listings
    WHERE neighbourhood_id IN (21,28,4)
"""

run(query)
```

```
<ipython-input-29-86fbc71b6552>:55: FutureWarning: this method is deprecated in favour of `Styler.to_html()`
  df.style.render() +
```

	listing_id	listing	host_id	host	neighbourhood_id	room_type	price_in_dollar
0	8476652	Cozy double room @ King St, Newtown	38507923	Austin	21	Private room	60.000000
1	22296011	Large private room on Camperdown park & Newtown	10873080	Joshua	21	Private room	40.000000
2	28268415	Amazing private room in Wolli Creek	42427723	Maíra	28	Private room	60.000000
3	40426054	Christmas rental short term	78397109	Brenda	4	Private room	80.000000

The prices of these four listings range between 40 to 80 dollar. What about the most expensive neighbourhoods with id 1, 26 and 37? These listings are expected to be more and a lot more expensive listings so please order the list by price_in_dollar.

```
### Question:
# 1/ Fetch all columns from the listings table
# 2/ Make sure you filter the data using WHERE-IN
# 3/ Make sure you order the list by price_in_dollar ascendingly
query = """
SELECT *
  FROM listings
    WHERE neighbourhood_id IN (1,26,37)
      ORDER BY price_in_dollar
"""

run(query)
```

```
<ipython-input-29-86fbc71b6552>:55: FutureWarning: this method is deprecated in favour of `Styler.to_html()`
  df.style.render() +
```

	listing_id	listing	host_id	host	neighbourhood_id	room_type	price_in_dollar
0	658279799851154944	Lovely bedroom by at the sea.	3526500	Maurice	37	Private room	75.000000
1	10549608	Amazing Double Bay Apartment with High Ceilings	23953750	Cyrus	37	Private room	200.000000
2	41983356	Duffotel Paddington EXEC 2-Bedroom Apartment	234423891	Duffotel	37	Entire home/apt	211.000000
3	44284905	BEACHCOMBER NORTH AVALON	269689583	Jill	26	Entire home/apt	220.000000
4	2033151	Beautifully Renovated Paddington Apartment	5727462	Julia	37	Entire home/apt	271.000000
5	34027782	Luxurious Cozy Bedroom, Explore @Olympic Park	163419570	Zach	1	Private room	500.000000
6	46159786	Warriewood Beach House	136836511	Fiona	26	Entire home/apt	1000.000000
7	9376988	Family Beach House🌴	11196275	Emma	26	Entire home/apt	2000.000000
8	10111788	******Paddington Sydney*****	51892300	Tom	37	Private room	28613.000000

Here it's revealed that we have 9 listings, with a varied amount of prices.

Now, let's start comparing the calendar from the cheapest against the most expensive, which is available the most. The cheapest are listing_id: 22296011, 8476652, 28268415, and 40426054

```
### Question:
# 1/ Fetch all columns from the calendar table
# 2/ Make sure you filter the data using WHERE-IN

query = """
SELECT *
  FROM calendar
    WHERE listing_id IN (22296011, 8476652, 28268415, 40426054)
"""

run(query)
```

```
<ipython-input-29-86fbc71b6552>:55: FutureWarning: this method is deprecated in favour of `Styler.to_html()`
  df.style.render() +
```

	listing_id	date	available	price_in_dollar	minimum_nights
0	8476652	2022-09-09T00:00:00Z	0	60.000000	90
1	8476652	2022-09-10T00:00:00Z	0	60.000000	90
2	8476652	2022-09-11T00:00:00Z	0	60.000000	90
3	8476652	2022-09-12T00:00:00Z	0	60.000000	90
4	8476652	2022-09-13T00:00:00Z	0	60.000000	90
5	8476652	2022-09-14T00:00:00Z	0	60.000000	90
6	8476652	2022-09-15T00:00:00Z	0	60.000000	90
7	22296011	2022-09-09T00:00:00Z	0	40.000000	90
8	22296011	2022-09-10T00:00:00Z	0	40.000000	90
9	22296011	2022-09-11T00:00:00Z	0	40.000000	90
10	22296011	2022-09-12T00:00:00Z	0	40.000000	90
11	22296011	2022-09-13T00:00:00Z	0	40.000000	90
12	22296011	2022-09-14T00:00:00Z	0	40.000000	90
13	22296011	2022-09-15T00:00:00Z	0	40.000000	90
14	28268415	2022-09-10T00:00:00Z	0	60.000000	90
15	28268415	2022-09-11T00:00:00Z	0	60.000000	90
16	28268415	2022-09-12T00:00:00Z	0	60.000000	90
17	28268415	2022-09-13T00:00:00Z	0	60.000000	90

Now also let's do that for the listings from the most expensive neighbourhoods. These have the listing_id: 658279799851154919, 10549608, 41983356, 44284905, 2033151, 34027782, 46159786, 9376988, and 10111788

```
### Question:
# 1/ Fetch all columns from the calendar table
# 2/ Make sure you filter the data using WHERE-IN

query = """
SELECT *
  FROM calendar
    WHERE listing_id IN (658279799851154919, 10549608, 41983356,
                44284905, 2033151, 34027782, 46159786, 9376988, 10111788)
"""

run(query)
```

```
<ipython-input-29-86fbc71b6552>:55: FutureWarning: this method is deprecated in favour of `Styler.to_html()`
  df.style.render() +
```

	listing_id	date	available	price_in_dollar	minimum_nights
0	2033151	2022-09-09T00:00:00Z	0	320.000000	2
1	2033151	2022-09-10T00:00:00Z	0	327.000000	2
2	2033151	2022-09-11T00:00:00Z	0	222.000000	2
3	2033151	2022-09-12T00:00:00Z	0	201.000000	2
4	2033151	2022-09-13T00:00:00Z	0	219.000000	2
5	2033151	2022-09-14T00:00:00Z	0	255.000000	2
6	2033151	2022-09-15T00:00:00Z	0	292.000000	2
7	9376988	2022-09-09T00:00:00Z	0	2000.000000	5
8	9376988	2022-09-10T00:00:00Z	0	2000.000000	5
9	9376988	2022-09-11T00:00:00Z	0	2000.000000	5
10	9376988	2022-09-12T00:00:00Z	0	2000.000000	5
11	9376988	2022-09-13T00:00:00Z	0	2000.000000	5
12	9376988	2022-09-14T00:00:00Z	0	2000.000000	5
13	9376988	2022-09-15T00:00:00Z	0	2000.000000	5
14	10111788	2022-09-09T00:00:00Z	0	99999.000000	90
15	10111788	2022-09-10T00:00:00Z	0	99999.000000	90
16	10111788	2022-09-11T00:00:00Z	0	59.000000	90
17	10111788	2022-09-12T00:00:00Z	0	59.000000	90
18	10111788	2022-09-13T00:00:00Z	0	59.000000	90
19	10111788	2022-09-14T00:00:00Z	0	59.000000	90

We find that for the least expensive neighbourhouds we have 26 days registered, of which none are available for renting, which means for the coming week we have 0% availability in the cheap neighbourhoods. However, the most expensive neighbourhoods have 62 days registered of which the coming week 4 days are available for renting, which means around 6% availability for next week.

On a larger scale these differences in availability of listings and thus revenue stream generated per listing are important. These data points might guide your focus as a General manager at Sydney Airbnb or pretty much any company which tries to make data-driven decisions.