SQL webinar - Basics and tricks

Marc Herrera

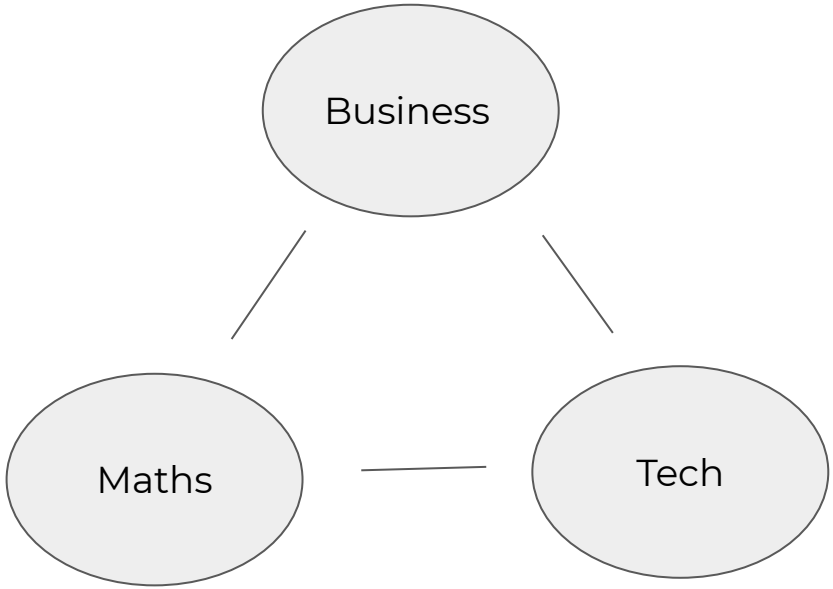
Senior Looker Specialist

Table of contents

Introduction

I want to get to know you better

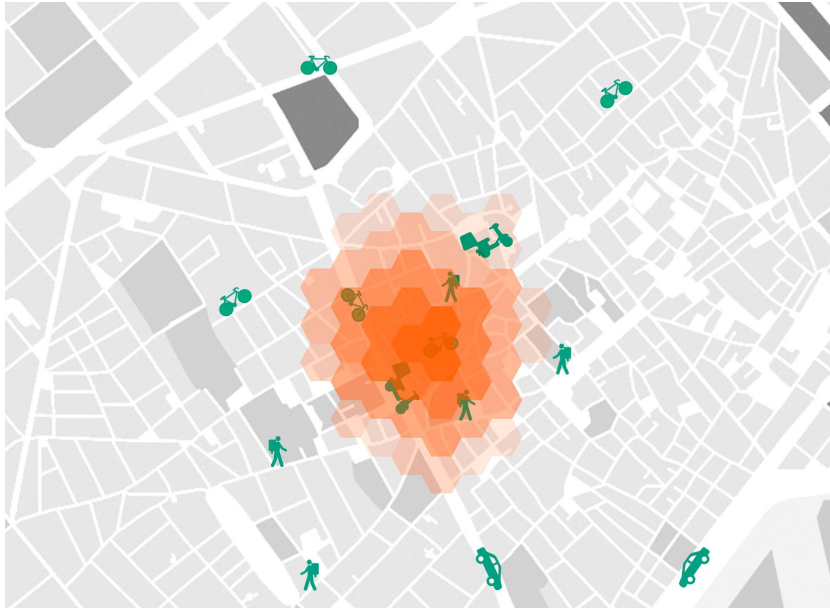
- How do you see yourself in the data triangle?
- Biggest SQL pain.



A bit about myself

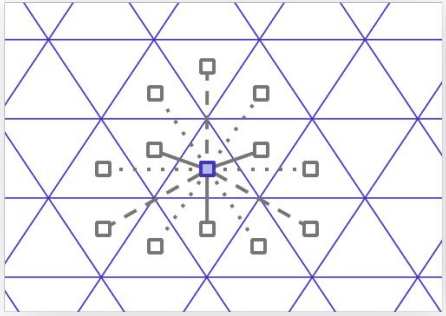
- Studied Economics on 2010.
- Like Sailing and going to Cadaqués.
- Bikeaholic; I bike even to go clubbing.
- I once subscribed and terminated a Gym subscription without going even once.
- Work in Glovo since 2018.

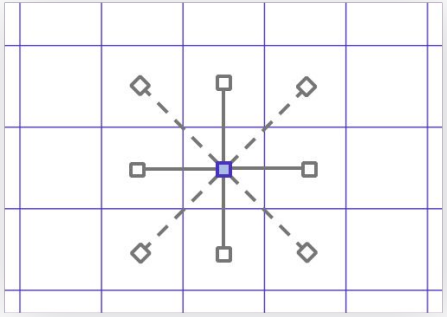
I started working in content geoanalysis

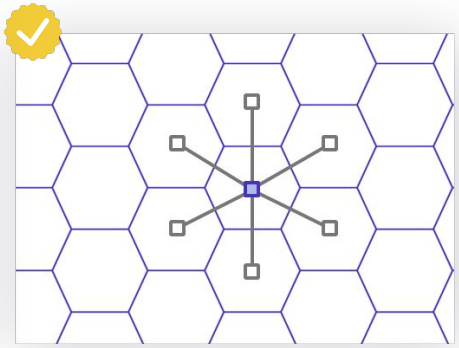


Geo-spatial analysis

Hex are separated equally.





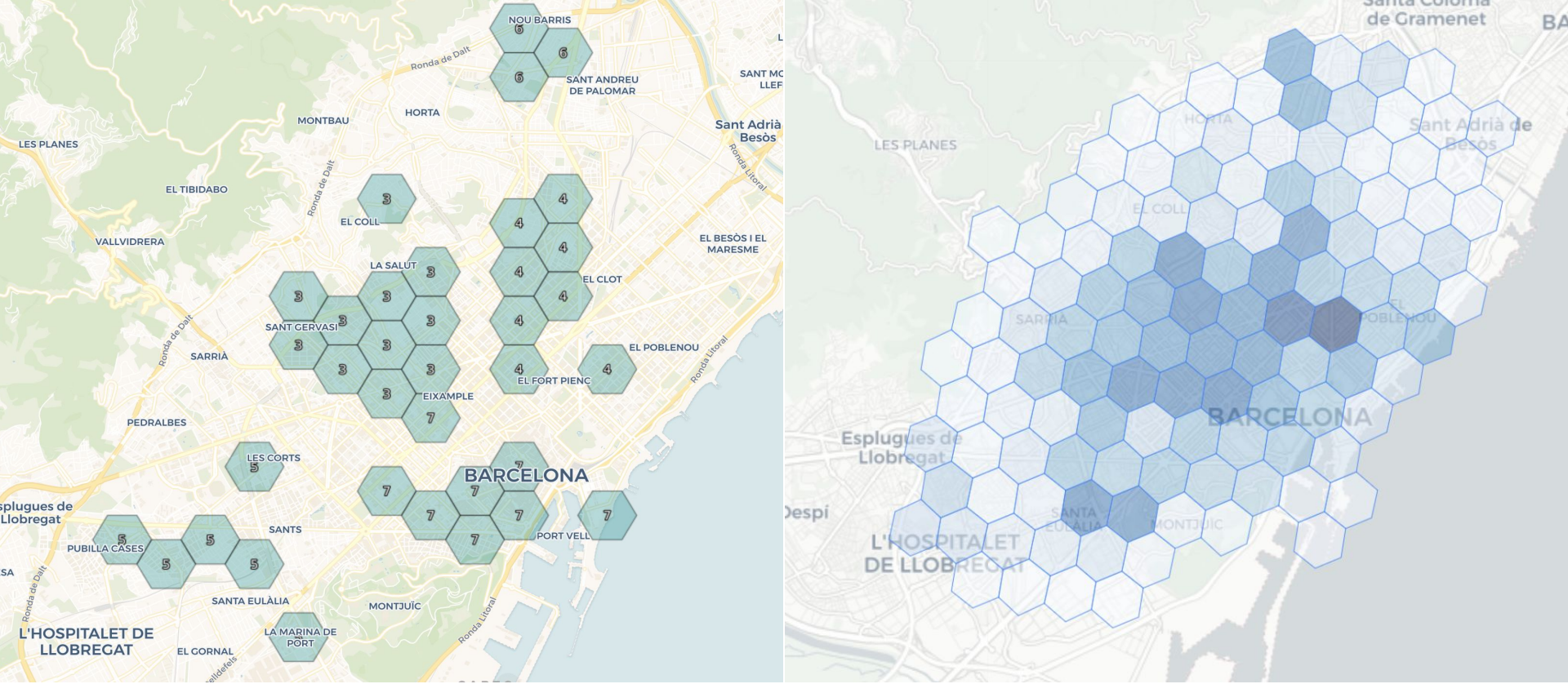


Walkers

Bicycle

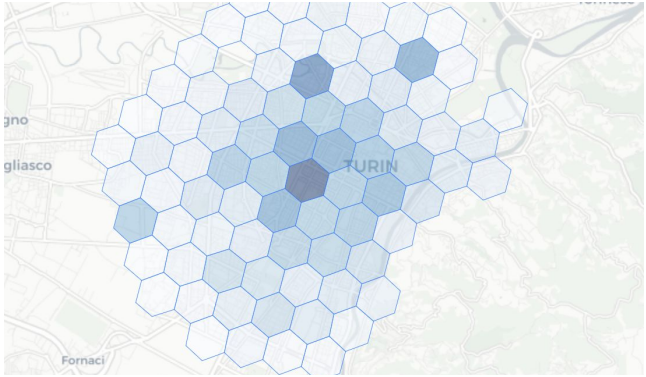
Motorbike

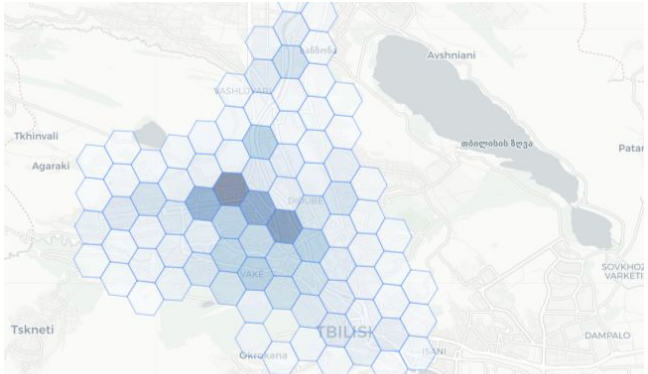
Car

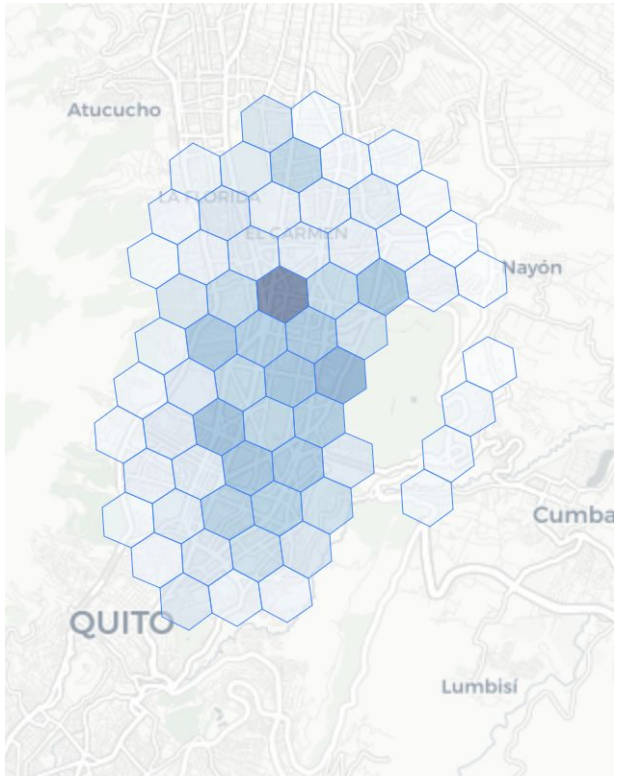


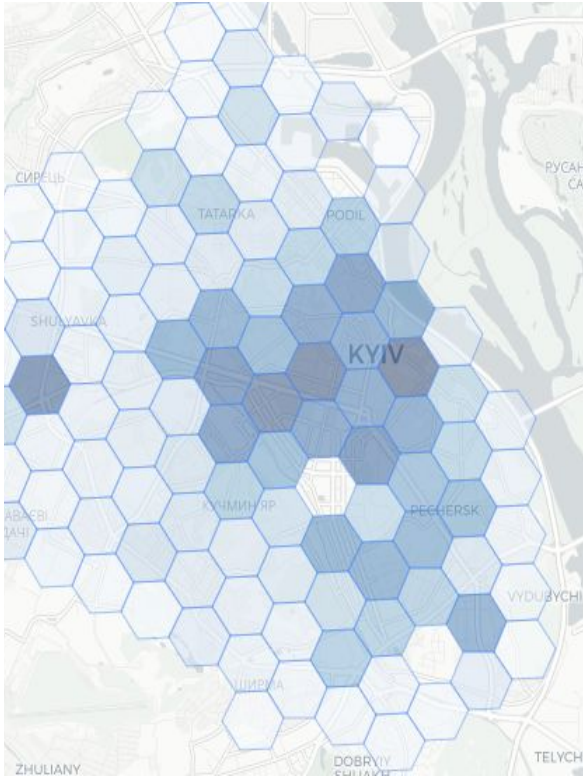
And reframed the calculation to find the optimal dark kitchen location

So the analysis could be scaled in a worldwide scale



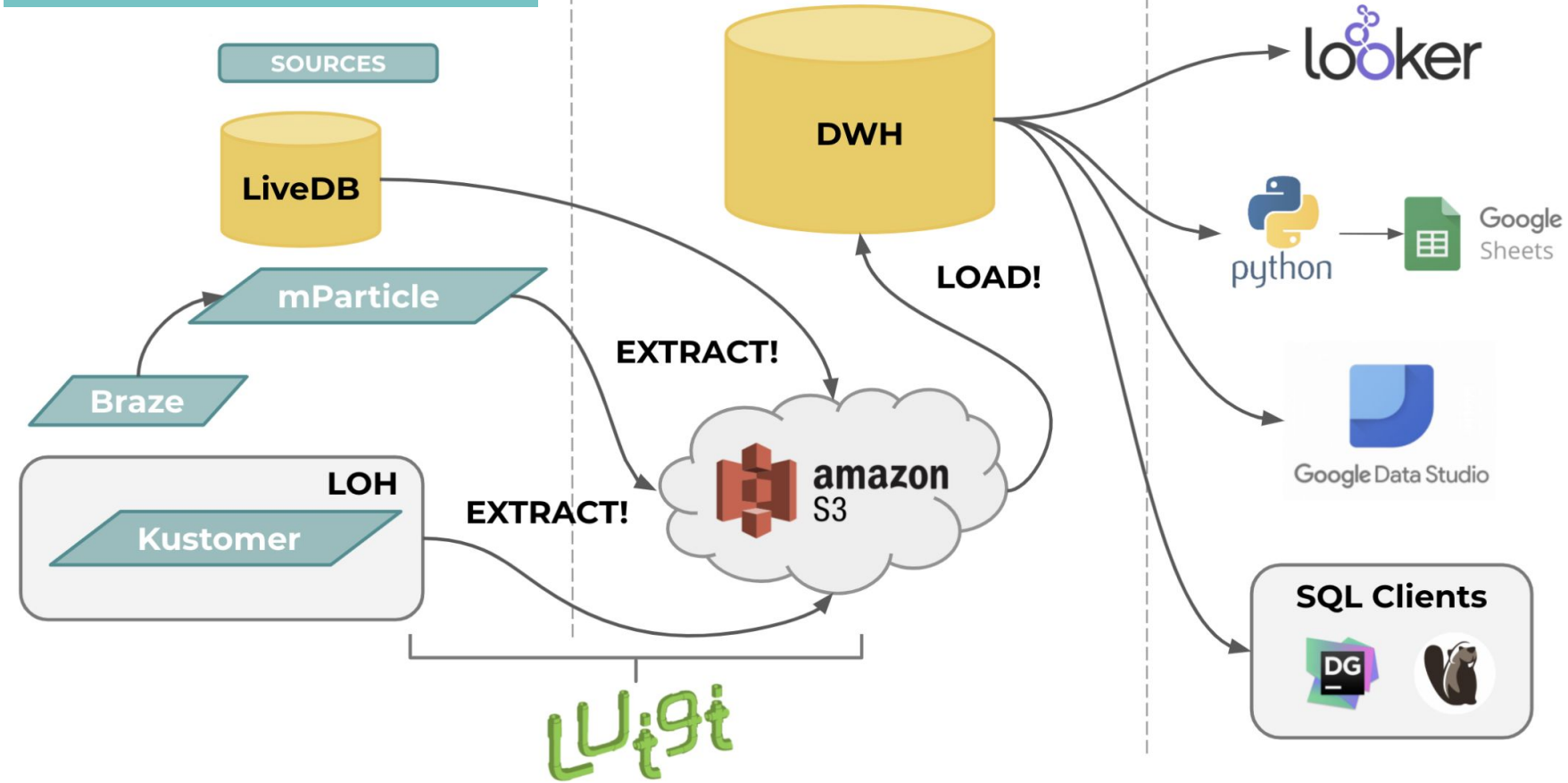






Current database usage in companies from analytics perspective

Data structure example



Threats of building a data driven company

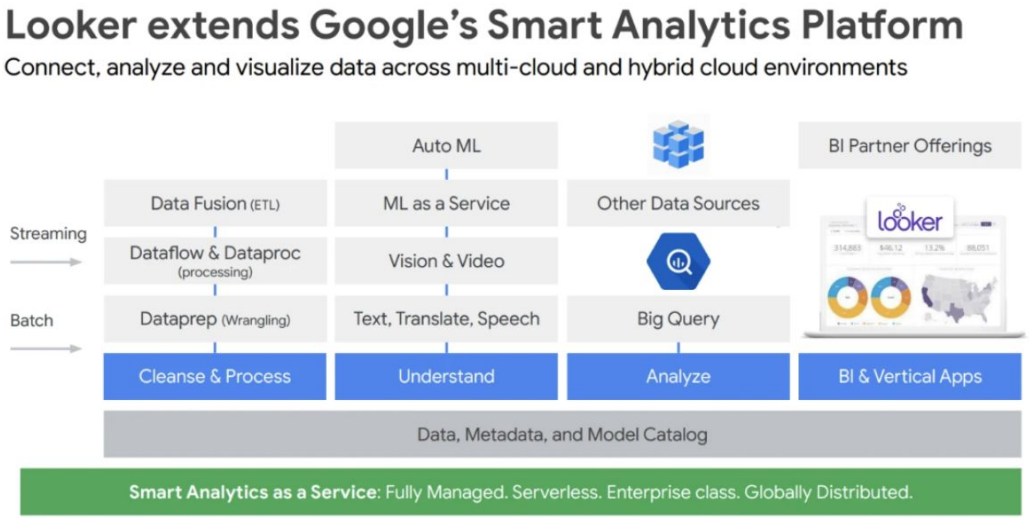
- **SQL is needed for 90% of the members of a data driven company** but generally its knowledge is very scarce.
- Even among people who use SQL, it's very difficult to make sure that each of the KPI's or queries are done with the same criteria. We may find several people looking for the same thing and getting into different results. **Data discrepancy.**

Looker, a SQL code builder

- Business intelligence software that **helps making data available to everyone** disregarding on their SQL level.
- Looker is run by its **own coding language named “LookMl”**. This code aims to scale and interpret the nature of the tables relationships. This code is modified and managed by the BI developers so we make sure there is one sole way to define each of the KPIs.

Google acquired Looker in 2020

- Aims to be the visualization tool of the future.
- It also integrates with other non-Google products such as redshift databases and Mysql servers.



Looker

Main competence: Tableau

- Tableau is more commonly used. However, Tableau does not have a way to escalate the SQL code so you keep needing SQL knowledge for all the tool users.
- Looker is cheaper and easier to use for non technical users.

SQL

What is SQL?

- Structured Query Language.
- Created on the 70's by the IBM researchers Raymond Boyce and Donald Chamberlin.
- For the exercises I used the sample database provided by Dbeaver. Link to download. → SQLite

Content on structure level; order matters

SELECT	Columns you want to retrieve
FROM	Base table from which you extract the information
JOIN	Extra tables you want to join with base table.
WHERE	Filtering to be applied at row level before aggregates occur.
GROUP BY	Clause to specify the dimensions that will pivot the inserted calculation.
HAVING	Filtering applied once aggregation is done (like a conditional Limit).
ORDER BY	Sorting (ASC/DESC).
LIMIT	Count of displayed records.

Aggregates

SELECT

FROM

JOIN

WHERE

GROUP BY

HAVING

ORDER BY

LIMIT

Most common aggregates; Max, min, avg, count, sum.

```
SELECT
        BillingCountry ,
        BillingCity,
        COUNT(*),
        COUNT(BillingState) ,
        MAX (Total),
        MIN (Total),
        AVG     (Total)
FROM
        Invoice
GROUP BY
        1,
        2
order by
        3 desc
```

BillingCountry	BillingCity	COUNT(*)	COUNT(BillingSt	MAX (Total)	MIN (Total)	AVG(Total)
Brazil	São Paulo	14	14	13,86	0,99	5,3742857143
Czech Republic	Prague	14	0	25,86	0,99	6,4457142857

Difference between having / where / no filter


```
SELECT
        BillingCountry ,
        BillingCity,
        COUNT(*),
        COUNT(BillingState) ,
        MAX (Total),
        MIN (Total),
        AVG      (Total)
FROM
        Invoice
--WHERE Total > 2
GROUP BY
        1,
        2
--HAVING AVG (Total)>5
order by
        3 desc
```

No filter

BillingCountry	BillingCity	COUNT(*)	COUNT(BillingSt	MAX (Total)	MIN (Total)	AVG(Total)
Brazil	São Paulo	14	14	13,86	0,99	5,3742857143
Czech Republic	Prague	14	0	25,86	0,99	6,4457142857

Having

BillingCountry	BillingCity	COUNT(*)	COUNT(BillingSt	MAX (Total)	MIN (Total)	AVG(Total)
Brazil	São Paulo	14	14	13,86	0,99	5,3742857143
Czech Republic	Prague	14	0	25,86	0,99	6,4457142857

Where

	BillingCountry	BillingCity	COUNT(*)	COUNT(BillingSt	MAX (Total)	MIN (Total)	AVG(Total)
3	Brazil	São Paulo	8	8	13,86	3,96	8,1675
4	Czech Republic	Prague	8	0	25,86	3,96	10,0425

Where + Having

	BillingCountry	BillingCity	COUNT(*)	COUNT(BillingSt	MAX (Total)	MIN (Total)	AVG(Total)
3	Brazil	São Paulo	8	8	13,86	3,96	8,1675
4	Czech Republic	Prague	8	0	25,86	3,96	10,0425

Date management

SELECT

FROM

JOIN

WHERE

GROUP BY

HAVING

ORDER BY

LIMIT

Datetime() or *getdate()* usage

```
SELECT
        DATETIME() as UTC,
        DATETIME(DATETIME(), '+2 hours') as CET,
        STRFTIME('%H', DATETIME(DATETIME(), '+2 hours')) as CEThour,
        DATE(DATETIME(), 'start of day') as Day_trunc,
        DATE(DATETIME('now', '-1 days'), 'start of day') as Prev_Day_trunc,
        DATE(DATETIME(), 'start of month') as Month_trunc,
        DATE(DATETIME('now', '-1 months'), 'start of month') as Prev_Month_trunc,
        DATE(DATETIME(), 'weekday 1') as Week_trunc,
        DATE(DATETIME('now', '-7 days'), 'weekday 1') as Prev_Week_trunc
```

- Very useful to test time frame management before filtering.
- These statements prove there is no need to extract information from a table to execute a SQL query.
- Date trunc formats are valid to group by aggregates.

UTC	CET	CEThou	Day_trunc	Prev_Day_trunc	Month_trunc	Prev_Month_trunc	Week_trunc	Prev_Week_trunc
2020-06-15 15:59:26	2020-06-15 17:59:26	17	2020-06-15	2020-06-14	2020-06-01	2020-05-01	2020-06-15	2020-06-08

Date management syntaxis varies a lot on different relational DB management systems

- datepart() = EXTRACT (Big query)
- datetrunc()= remains
- For current time extract PostgreSQL (redshift) uses getdate(), SQLite uses Datetime() and Bigquery current_date(). Link for the documentation on BigQuery time management.

```
]:
QUERY4 = """
SELECT current_date(), current_time(), current_datetime()
"""
query_job4 = client.query(QUERY4)
df4 = query_job4.to_dataframe()
df4.head()
```

]:

	f0_	f1_	f2_
0	2020-06-08	22:13:05.911962	2020-06-08 22:13:05.911962

Do you know what is the difference between the *rank()* and the *row_number()* function?

Window functions

SELECT

FROM

JOIN

WHERE

GROUP BY

HAVING

ORDER BY

LIMIT

Does the calculations without grouping the dimensions.

```
SELECT
        BillingCountry ,
        BillingCity,
        Total,
        ROW_NUMBER () OVER ( partition by BillingCity ORDER BY Total desc),
        RANK () OVER ( partition by BillingCity ORDER BY Total desc),
        SUM (Total) OVER ( partition by BillingCity)
FROM
        Invoice
order by
        1 asc
```

	BillingCountry	BillingCity	Total	ROW...	RANK()	sum (Total)
371	USA	Reno	13,86	1	1	37,62
372	USA	Reno	8,91	2	2	37,62
373	USA	Reno	5,94	3	3	37,62
374	USA	Reno	3,96	4	4	37,62
375	USA	Reno	1,98	5	5	37,62
376	USA	Reno	1,98	6	5	37,62
377	USA	Reno	0,99	7	7	37,62
378	USA	Salt Lake City	13,86	1	1	43,62
379	USA	Salt Lake City	11,94	2	2	43,62

Row number most useful use cases

- For filtering only the Nth biggest product for each city.

```
SELECT *
FROM
        (SELECT
        BillingCountry , BillingCity, Total,
        ROW_NUMBER () OVER ( partition by BillingCity ORDER BY Total desc) as rownumber
        FROM Invoice)
WHERE rownumber <4
```

BillingCountry	BillingCity	Total	rownumber
Netherlands	Amsterdam	13,86	1
Netherlands	Amsterdam	8,94	2
Netherlands	Amsterdam	8,91	3
India	Bangalore	13,86	1
India	Bangalore	8,91	2
India	Bangalore	5,94	3
Germany	Berlin	13,86	1

Row number in funnel building

- For building up funnel studies. Imagine a table in which you got each event a user does while navigating. Step one; declare the order of the events based on the time it happened and separing at session level. Step two; append it with prior one.

Row function:

```
row_number() over (partition by mp.session_id order by mp.creation_time asc) as ordersequence
```

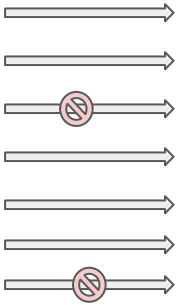
Join used:

```
from eventstable
        left join eventstable destin
                  on destin.ordersequence - 1 = eventstable.ordersequence and destin.session_id = eventstable.session_id
```

Row number in funnel building

- Note that when joining then, all events churning will get a null.
- This permits seeing how many sessions are you churning in every step.
- Although the purchase has no pair, it's okay as in the purchase the conversion is met.

Session id	Event	Row number
Session 1	Opens app	1
Session 1	Opens store wall	2
Session 1	Goes to checkout	3
Session 2	Opens app	1
Session 2	Opens store wall	2
Session 2	Goes to checkout	3
Session 2	Purchases	4



Session id	Event	Row number
Session 1	Opens app	1
Session 1	Opens store wall	2
Session 1	Goes to checkout	3
Session 2	Opens app	1
Session 2	Opens store wall	2
Session 2	Goes to checkout	3
Session 2	Purchases	4

Do you know what is the difference between a *left join* and a *left outer join*?

Should I use a inner join or a left join?

SELECT

FROM

JOIN

WHERE

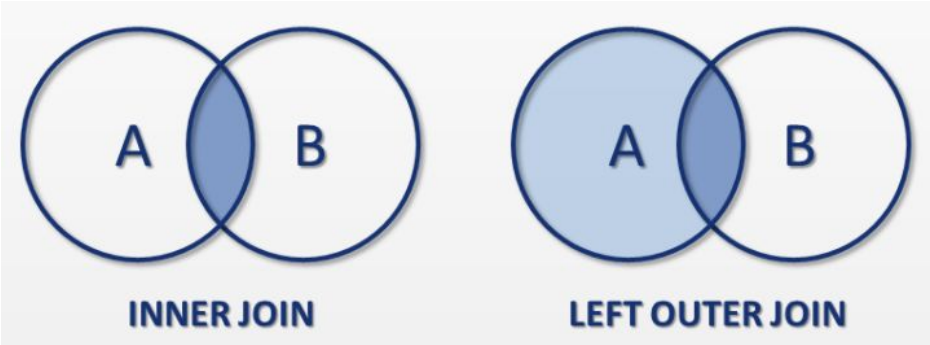
GROUP BY

HAVING

ORDER BY

LIMIT

Generally only the left join is used.



- When using the inner join you will only have records for the rows where the join condition is met.

```
SELECT count(*)
FROM Artist
  JOIN album on album.ArtistId = Artist.ArtistId
```

=

```
SELECT count(*)
FROM Artist
   LEFT JOIN album on album.ArtistId = Artist.ArtistId
WHERE album.ArtistId NOTNULL
```

Do you know what is the default *join*

applied when typing only “*join*”?

Should I use a inner join or a left join?

- Recommendation; use always a left join but **avoid counting rows.**
- Generally it's better to add the filter than using the inner join to make the code more intuitive for others.

```
SELECT Artist.Name, count(*) , count(album.AlbumId )
FROM Artist
  LEFT JOIN album on album.ArtistId = Artist.ArtistId
GROUP BY 1
ORDER BY 3 DESC
```

	Name	count(*)	count(album.AlbumId)
1	Iron Maiden	21	21
2	Led Zeppelin	14	14
204	Aaron Copland & Lo	1	1
205	Youssou N'Dour	1	0
206	Xis	1	0

Beware with fanouts

- Fanouts happen when we join a table to a biggest one.
- Beware when relationship is “one to many”.
- I recommend always building queries from the biggest table.

```
SELECT Artist.Name, count(album.AlbumId) , count(Track.Trackid )
FROM Artist
  LEFT JOIN album on album.ArtistId = Artist.ArtistId
  LEFT JOIN Track on Track.Composer =artist.Name
GROUP BY 1
ORDER BY 3 DESC
```

	Name	count(album.AlbumId)	count(Track.Trackid)
1	U2	440	440
2	Metallica	80	80
3	Miles Davis	69	69

```
SELECT Artist.Name, count(distinct album.AlbumId) , count(distinct Track.Trackid )
FROM Artist
  LEFT JOIN album on album.ArtistId = Artist.ArtistId
  LEFT JOIN Track on Track.Composer =artist.Name
GROUP BY 1
ORDER BY 2 DESC
```

	Name	count(distinct album.AlbumId)	count(distinct Track.Trackid)
1	Iron Maiden	21	0
2	Led Zeppelin	14	0
3	Deep Purple	11	0
4	U2	10	44
5	Metallica	10	8
6	Ozzy Osbourne	6	0
7	Pearl Jam	5	2

Filtering clauses

SELECT

FROM

JOIN

WHERE

GROUP BY

HAVING

ORDER BY

LIMIT

Most frequent filtering clauses;

- Nulls: ISNULL, NOTNULL.
- Dates: refer to the date slides.
- Values: >, <, <=, >=.
- Strings: two main ways;
    - IN: when filtering for one / more string.
    - Like: good for conditions (starts/finishes with *x*)

```
SELECT
        email,
        FirstName,
        Company
from
        Customer
Where
        -- Company notnull
         Company isnull
        -- Email LIKE '%mail.com'
        -- Email IN ('hleacock@gmail.com', 'kachase@hotmail.com')
```

Email	FirstName
ftremblay@gmail.com	François
hholy@gmail.com	Helena
kachase@hotmail.com	Kathy

Trick for splitting dimensions

Which is the most common email domain in the customer database?

```
SELECT
  substr(Email , instr(Email, '@') + 1) as domain,
-- replace(substr(Email, instr(Email, '@') + 1), ltrim(substr(Email, instr(Email, '@') + 1), replace(substr(Email, instr(Email, '@') + 1), '.', '')), '') as company,
  count(*)
from
  Customer
group by 1
order by 2 desc
```

gmail.com	8
hotmail.com	4
shaw.ca	3
yahoo.fr	2
yahoo.de	2
yahoo.com	2
uol.com.br	2
surfeu.de	2
aol.com	2
yahoo.uk	1
yahoo.se	1

yahoo	18
gmail	8
apple	7
hotmail	4
shaw	3
uol	2
surfeu	2
aol	2
yachoo	1
wp	1

Thanks =)