The Art and Science of Transportation Research in the AI Era

SQL and Data Visualization

M.Sc. Hiba Karam

Learning Goals

#1 Learn more SQL Functions

#3 Understand Data visualisation

#2 Understand what is a SQL Join

#4 Differentiate between ‘bad’ and ‘good’ graph

Lecture Structure

#1 SQL Functions

#2 SQL Join

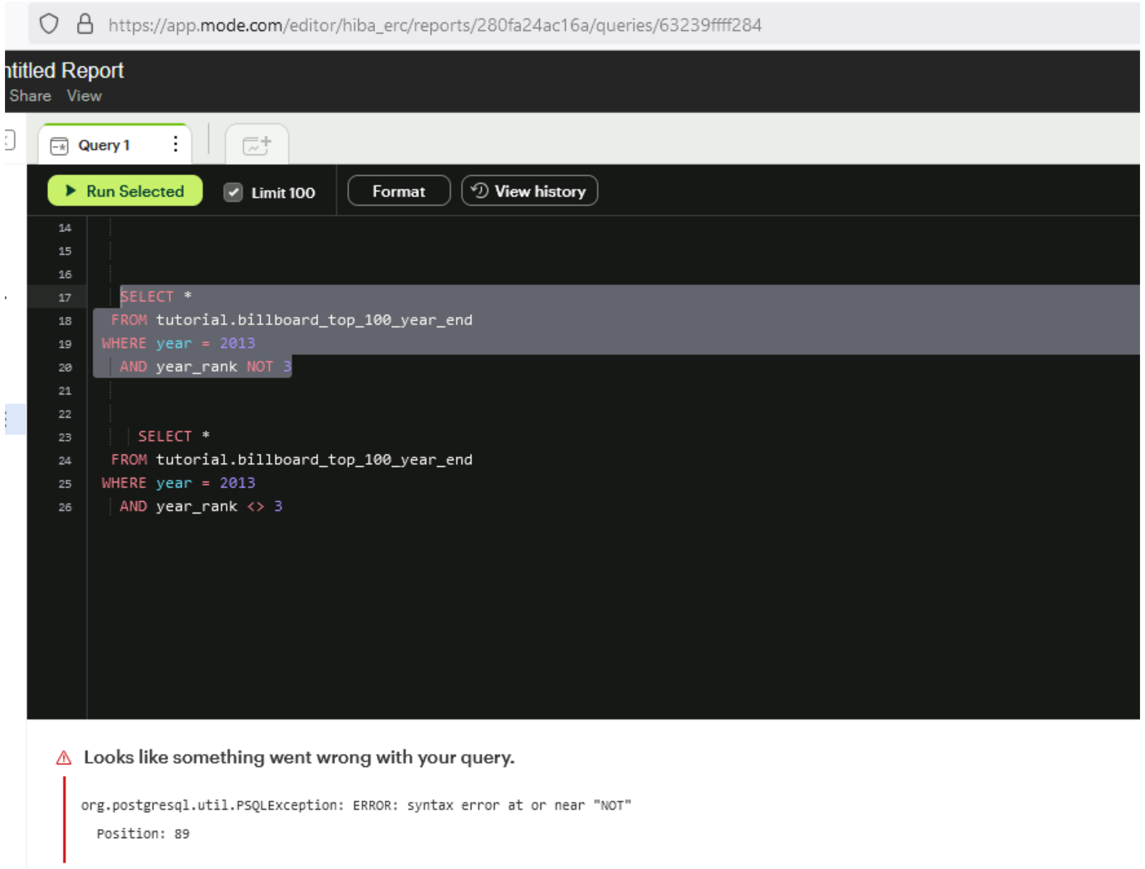
#3 Data visualisation

#4 Examples

#1.1 Recap SELECT and FROM

- Filtering using the WHERE clause happens **before** the results are returned.
- You can filter using a column that is not included in the SELECT statement, but it will not show if you do not explicitly select it.

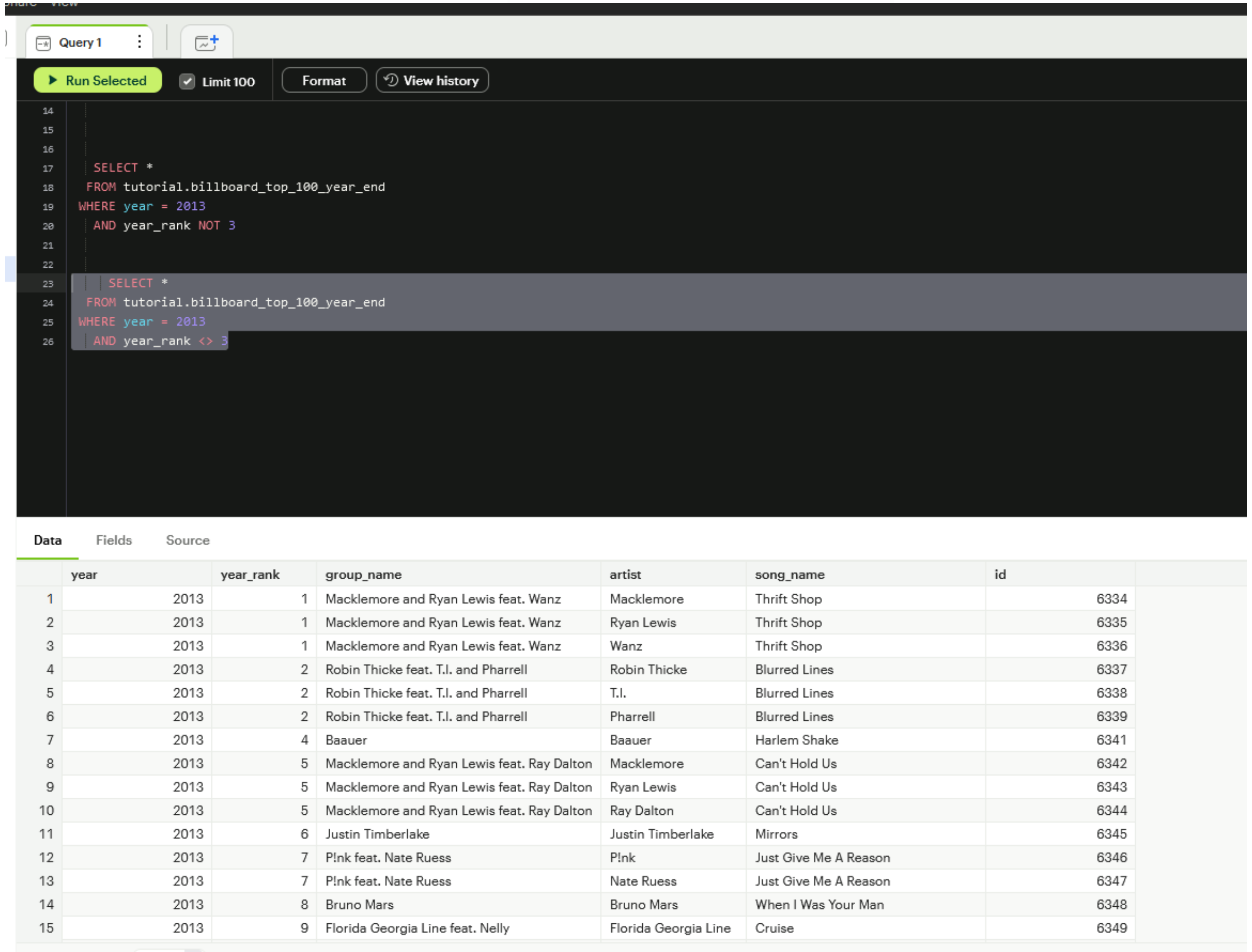
#1.1 Recap NOT and <>


https://app.mode.com/editor/hiba_erc/reports/280fa24ac16a/queries/63239ffff284
ntitled Report
Share View
Query 1
Run Selected
Limit 100
Format
View history

```
 SELECT *
FROM tutorial.billboard_top_100_year_end
WHERE year = 2013
 AND year_rank NOT 3

  SELECT *
FROM tutorial.billboard_top_100_year_end
WHERE year = 2013
 AND year_rank <> 3
```

Looks like something went wrong with your query.

```
org.postgresql.util.PSQLException: ERROR: syntax error at or near "NOT"
  Position: 89
```



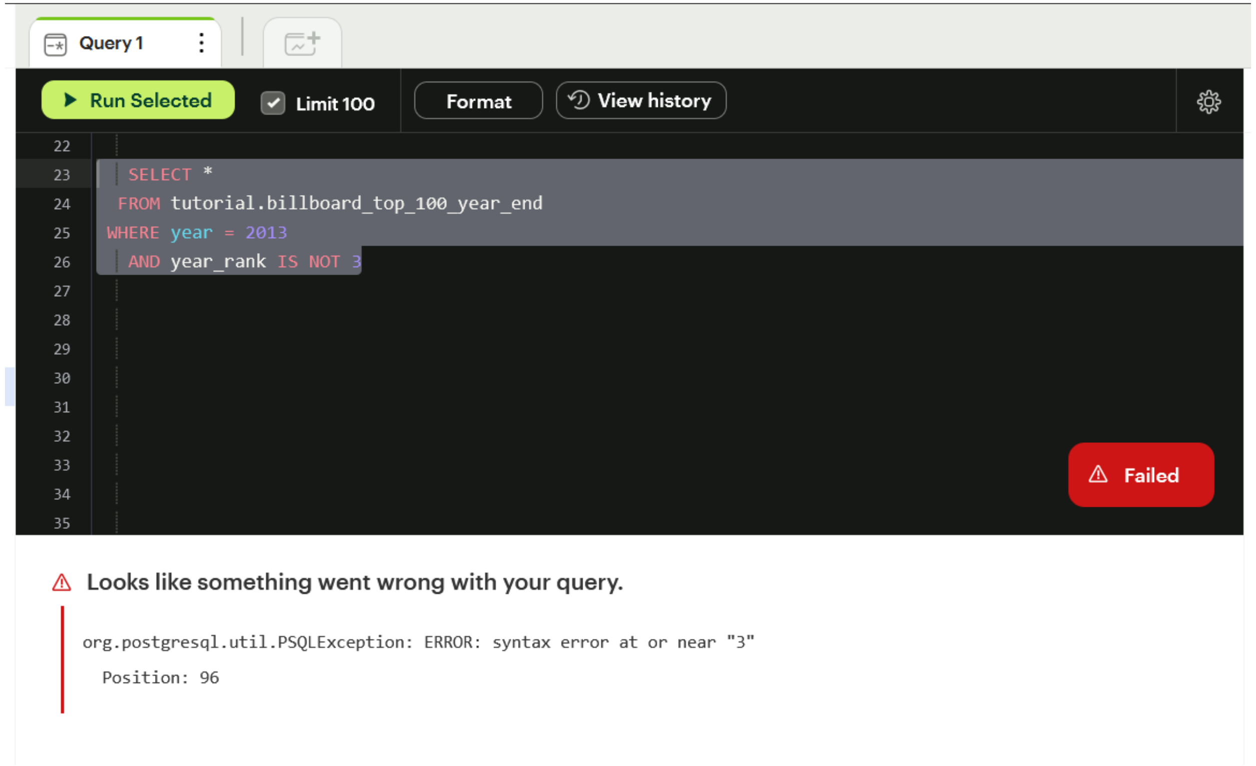
```
 SELECT *
FROM tutorial.billboard_top_100_year_end
WHERE year = 2013
 AND year_rank NOT 3

  SELECT *
FROM tutorial.billboard_top_100_year_end
WHERE year = 2013
 AND year_rank <> 3
```

	year	year_rank	group_name	artist	song_name	id
1	2013	1	Macklemore and Ryan Lewis feat. Wanz	Macklemore	Thrift Shop	6334
2	2013	1	Macklemore and Ryan Lewis feat. Wanz	Ryan Lewis	Thrift Shop	6335
3	2013	1	Macklemore and Ryan Lewis feat. Wanz	Wanz	Thrift Shop	6336
4	2013	2	Robin Thicke feat. T.I. and Pharrell	Robin Thicke	Blurred Lines	6337
5	2013	2	Robin Thicke feat. T.I. and Pharrell	T.I.	Blurred Lines	6338
6	2013	2	Robin Thicke feat. T.I. and Pharrell	Pharrell	Blurred Lines	6339
7	2013	4	Baauer	Baauer	Harlem Shake	6341
8	2013	5	Macklemore and Ryan Lewis feat. Ray Dalton	Macklemore	Can't Hold Us	6342
9	2013	5	Macklemore and Ryan Lewis feat. Ray Dalton	Ryan Lewis	Can't Hold Us	6343
10	2013	5	Macklemore and Ryan Lewis feat. Ray Dalton	Ray Dalton	Can't Hold Us	6344
11	2013	6	Justin Timberlake	Justin Timberlake	Mirrors	6345
12	2013	7	P!nk feat. Nate Ruess	P!nk	Just Give Me A Reason	6346
13	2013	7	P!nk feat. Nate Ruess	Nate Ruess	Just Give Me A Reason	6347
14	2013	8	Bruno Mars	Bruno Mars	When I Was Your Man	6348
15	2013	9	Florida Georgia Line feat. Nelly	Florida Georgia Line	Cruise	6349

Depends on the data that you work with.
What you intend on doing.

#1.1 Recap IS and =

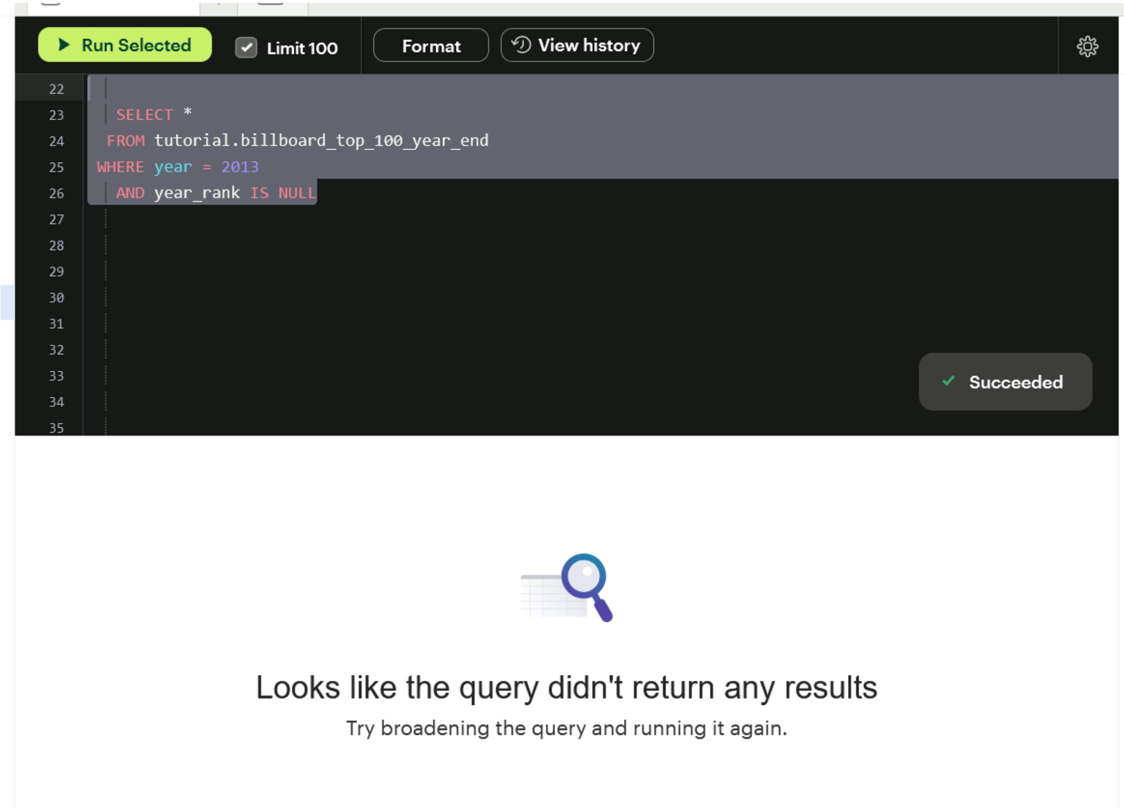
IS **cannot** be used by itself. It must be used with specific keywords like **NULL**, **TRUE**, or **FALSE – (**Boolean Data)



```
  SELECT *
 FROM tutorial.billboard_top_100_year_end
WHERE year = 2013
  AND year_rank IS NOT 3
```

Looks like something went wrong with your query.

```
org.postgresql.util.PSQLException: ERROR: syntax error at or near "3"
  Position: 96
```

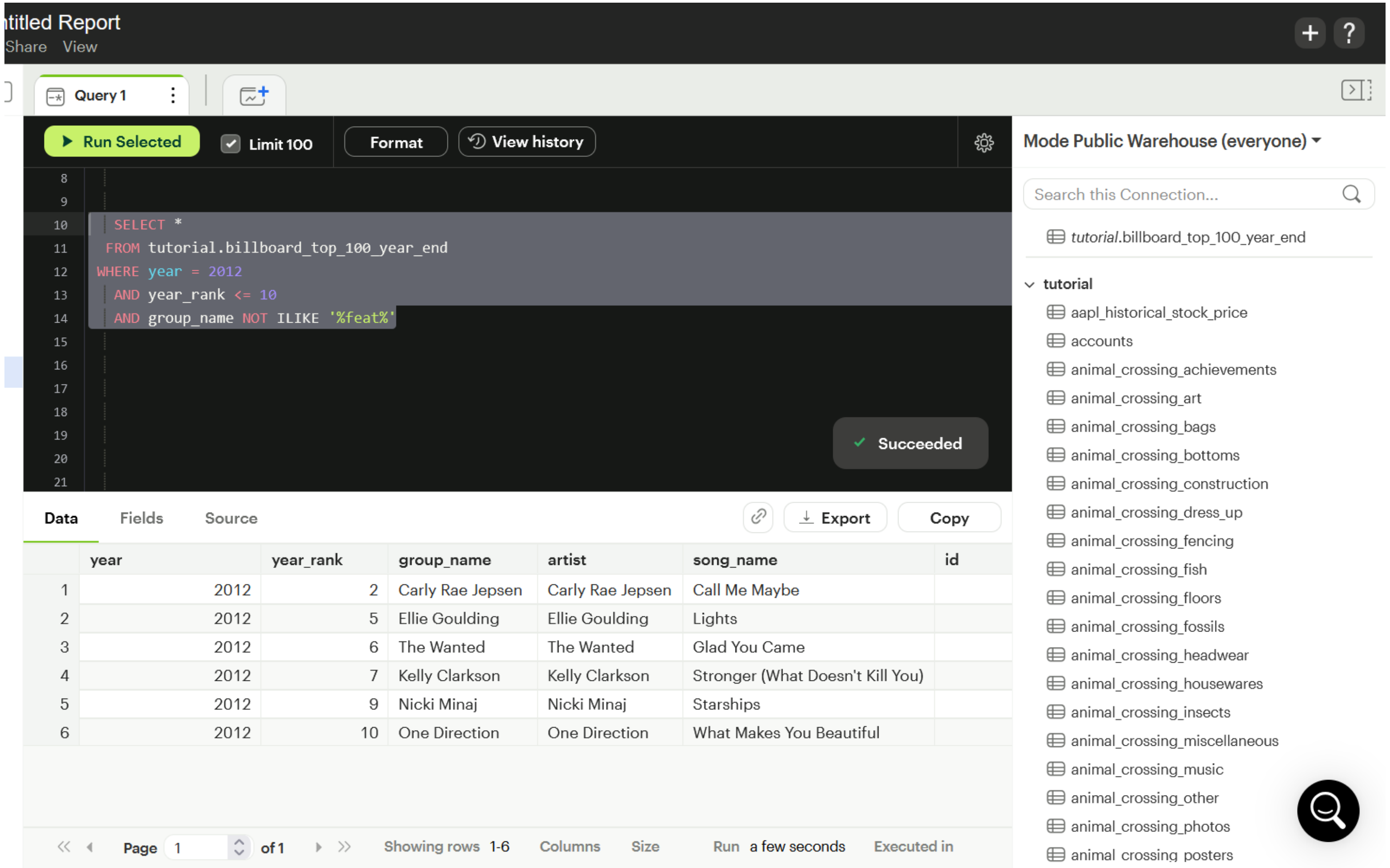


```
  SELECT *
 FROM tutorial.billboard_top_100_year_end
WHERE year = 2013
  AND year_rank IS NULL
```

Looks like the query didn't return any results

Try broadening the query and running it again.

#1.1 Recap NOT



```
  SELECT *
 FROM tutorial.billboard_top_100_year_end
WHERE year = 2012
  AND year_rank <= 10
  AND group_name NOT ILIKE '%feat%'
```

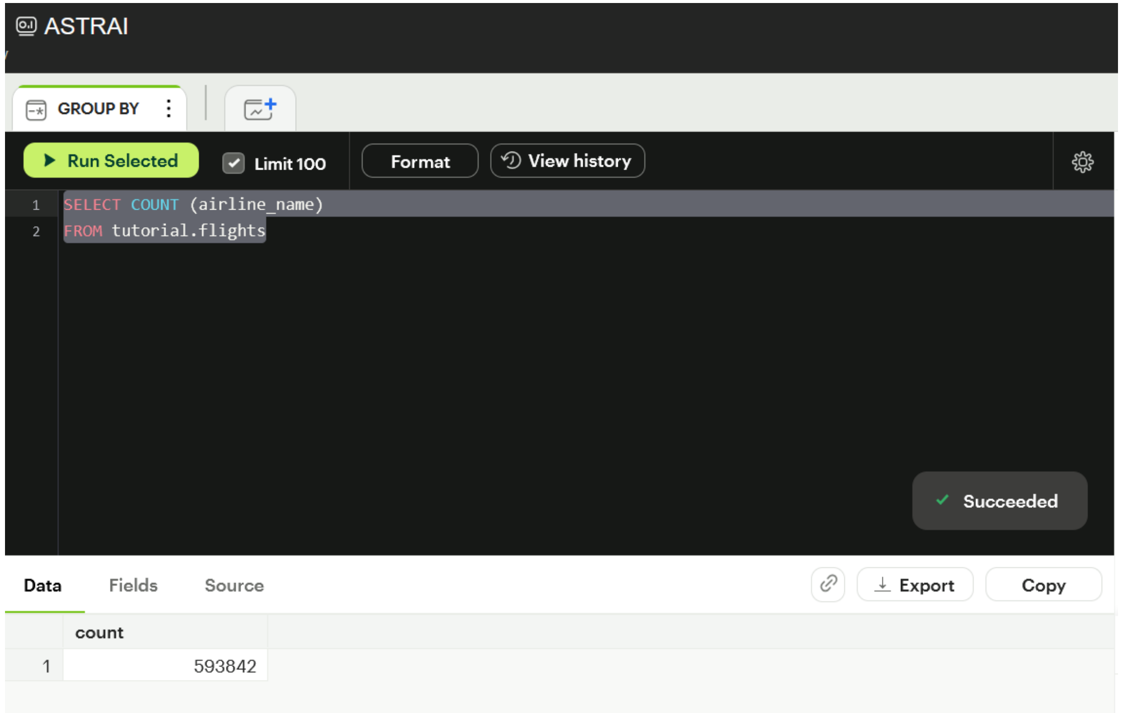
✓ Succeeded

	year	year_rank	group_name	artist	song_name	id
1	2012	2	Carly Rae Jepsen	Carly Rae Jepsen	Call Me Maybe	
2	2012	5	Ellie Goulding	Ellie Goulding	Lights	
3	2012	6	The Wanted	The Wanted	Glad You Came	
4	2012	7	Kelly Clarkson	Kelly Clarkson	Stronger (What Doesn't Kill You)	
5	2012	9	Nicki Minaj	Nicki Minaj	Starships	
6	2012	10	One Direction	One Direction	What Makes You Beautiful	

Page 1 of 1 Showing rows 1-6 Columns Size Run a few seconds Executed in

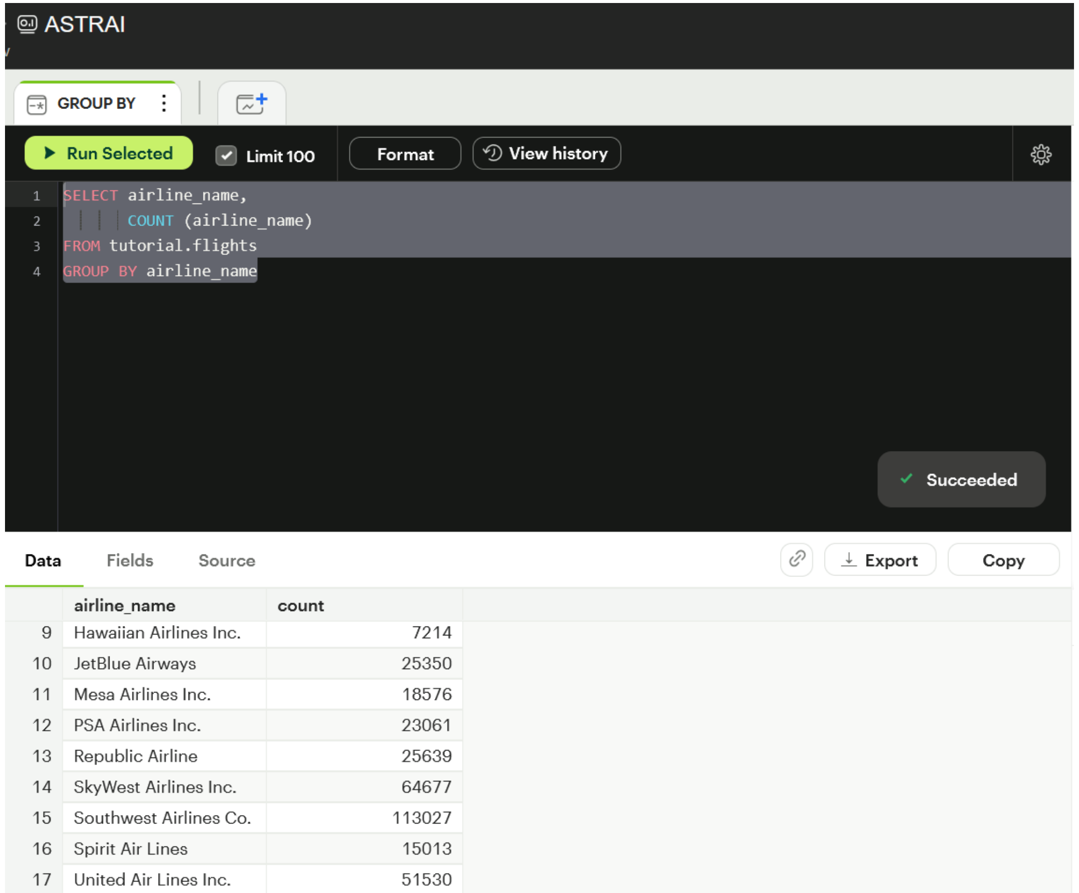
#1.2 GROUP BY

- Aggregation functions aggregate across the entire table.
- But what if you want to aggregate only part of a table?
- **GROUP BY** allows you to **separate data into groups**, which can be aggregated independently of one another.



```
SELECT COUNT (airline_name)
FROM tutorial.flights
```

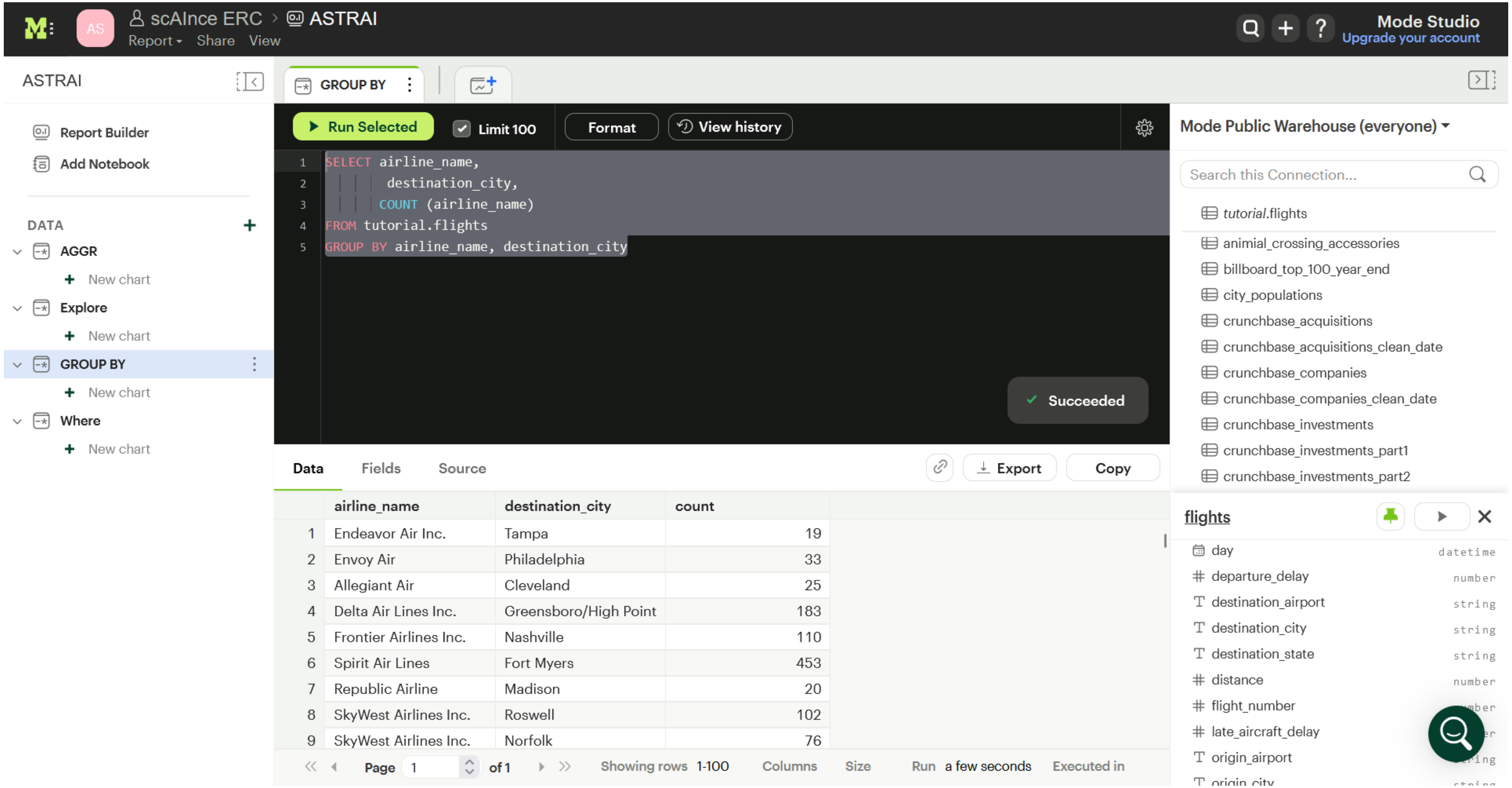
	count
1	593842



```
SELECT airline_name,
       COUNT (airline_name)
FROM tutorial.flights
GROUP BY airline_name
```

	airline_name	count
9	Hawaiian Airlines Inc.	7214
10	JetBlue Airways	25350
11	Mesa Airlines Inc.	18576
12	PSA Airlines Inc.	23061
13	Republic Airline	25639
14	SkyWest Airlines Inc.	64677
15	Southwest Airlines Co.	113027
16	Spirit Air Lines	15013
17	United Air Lines Inc.	51530

#1.2 GROUP BY



```sql
SELECT airline_name,
       destination_city,
       COUNT (airline_name)
FROM tutorial.flights
GROUP BY airline_name, destination_city
```

	airline_name	destination_city	count
1	Endeavor Air Inc.	Tampa	19
2	Envoy Air	Philadelphia	33
3	Allegiant Air	Cleveland	25
4	Delta Air Lines Inc.	Greensboro/High Point	183
5	Frontier Airlines Inc.	Nashville	110
6	Spirit Air Lines	Fort Myers	453
7	Republic Airline	Madison	20
8	SkyWest Airlines Inc.	Roswell	102
9	SkyWest Airlines Inc.	Norfolk	76

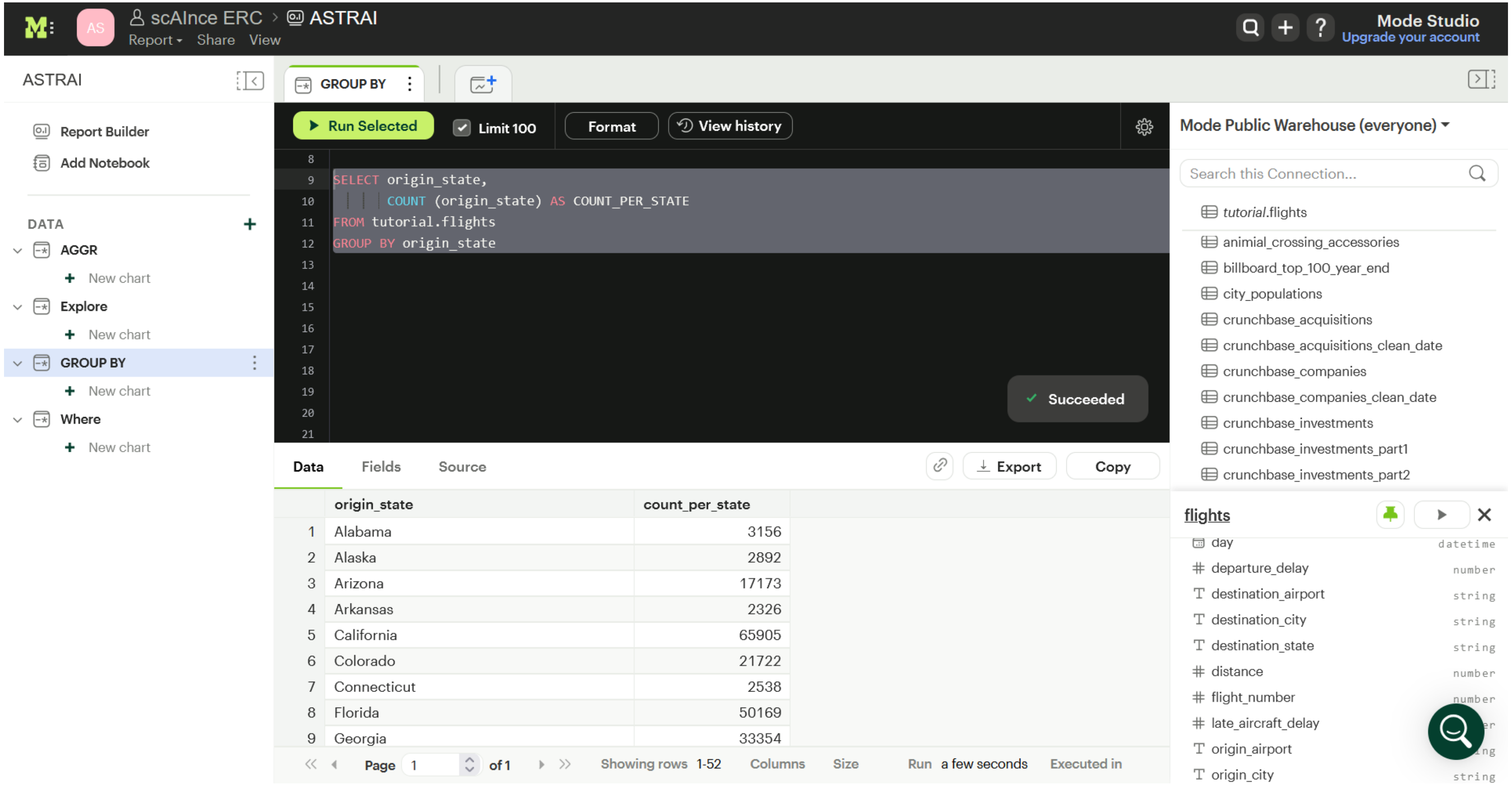
#1.2 GROUP BY

1. Count the total number of flights for each origin state?

#1.2 GROUP BY

TECHNISCHE UNIVERSITÄT DARMSTADT

IVV Institut für Verkehrsplanung und Verkehrstechnik TU Darmstadt



```
SELECT origin_state,
       COUNT (origin_state) AS COUNT_PER_STATE
FROM tutorial.flights
GROUP BY origin_state
```

	origin_state	count_per_state
1	Alabama	3156
2	Alaska	2892
3	Arizona	17173
4	Arkansas	2326
5	California	65905
6	Colorado	21722
7	Connecticut	2538
8	Florida	50169
9	Georgia	33354

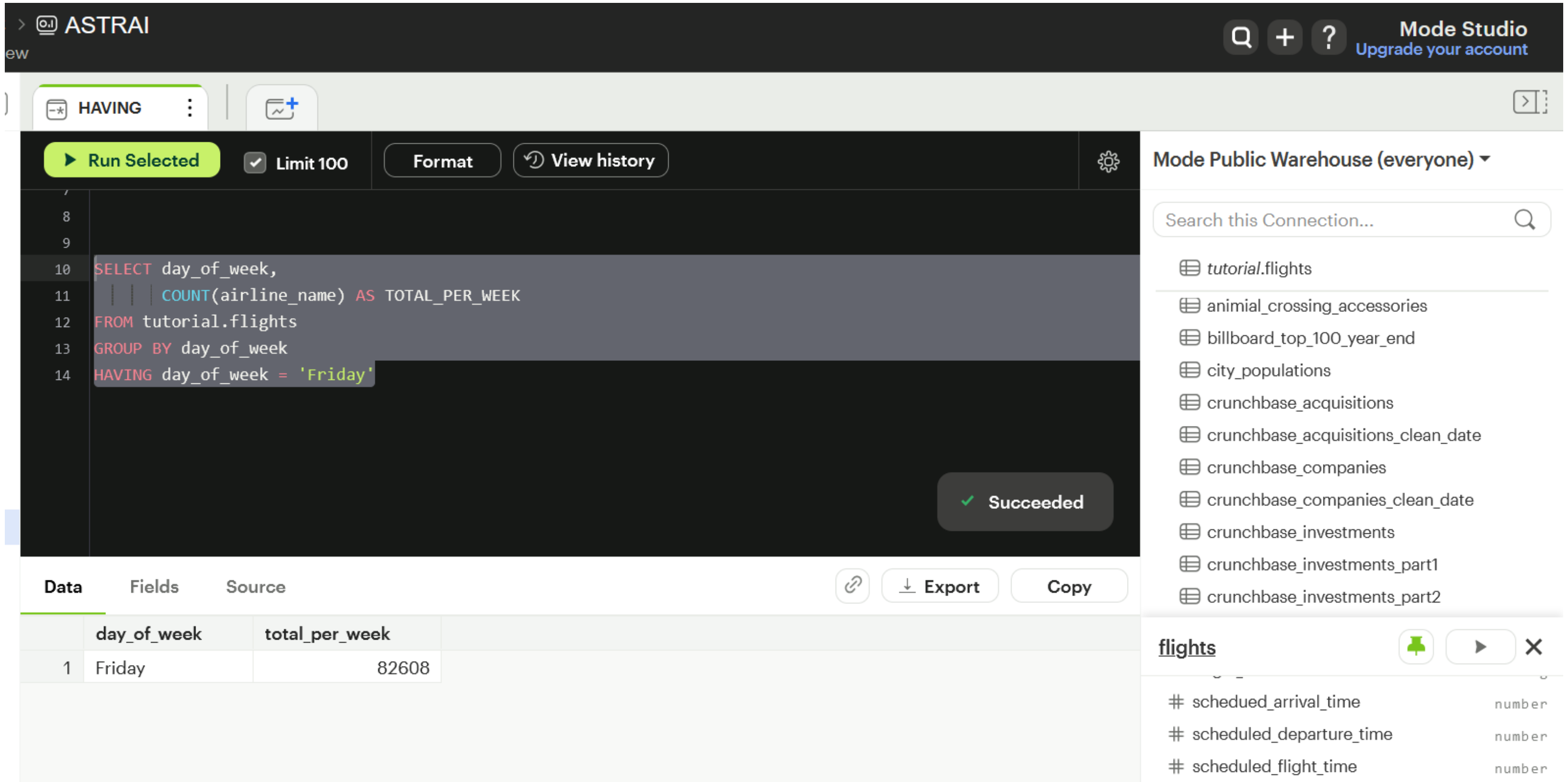
flights

day	datetime
departure_delay	number
destination_airport	string
destination_city	string
destination_state	string
distance	number
flight_number	number
late_aircraft_delay	[illegible]
origin_airport	[illegible]
origin_city	string

#1.3 HAVING

- The WHERE clause does not work with GROUP BY because it **doesn't allow you to filter on aggregate columns**—that's where the HAVING clause comes in.

- **HAVING: clause filters a query that has been aggregated.**

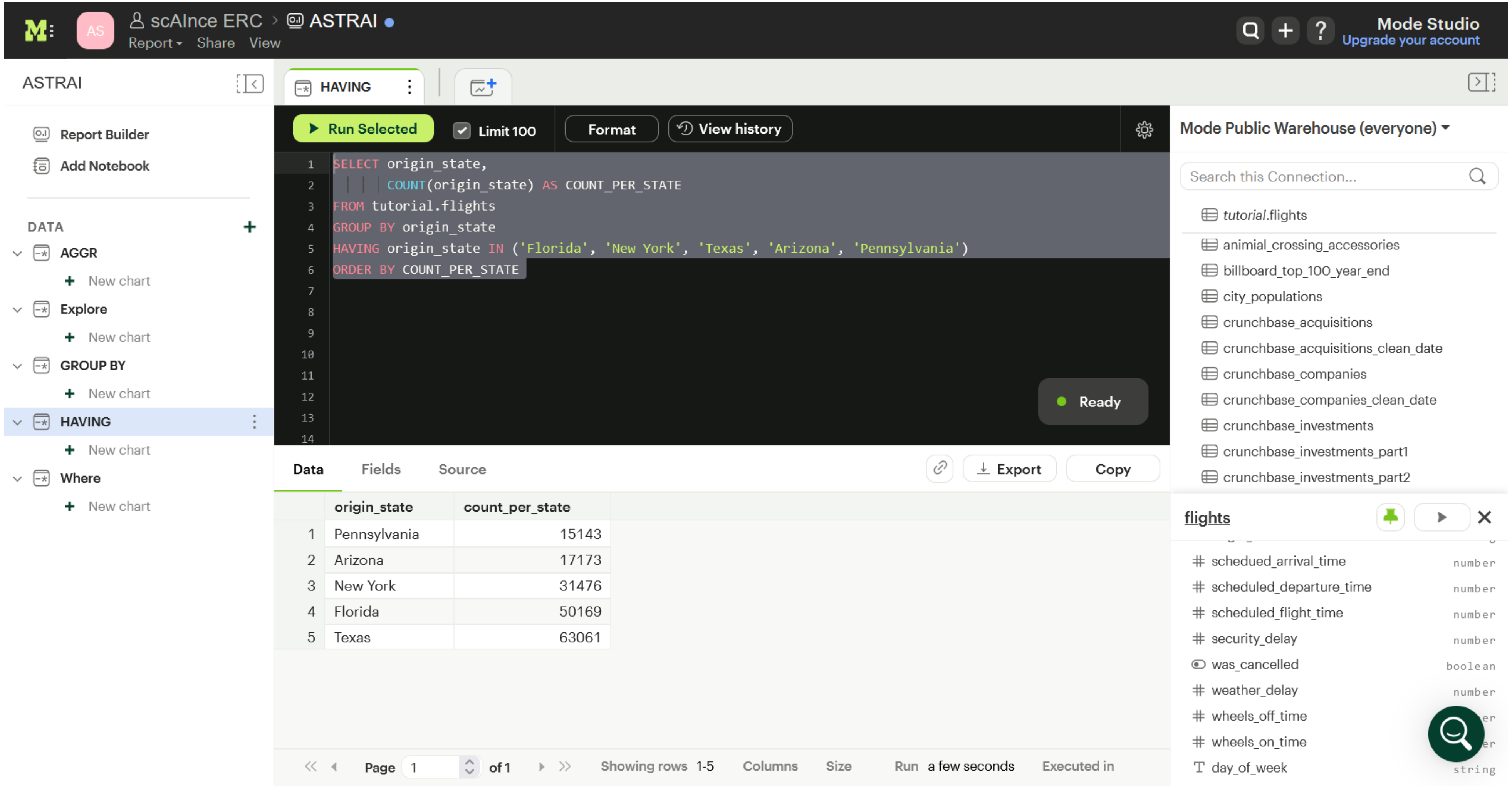


ASTRAI
ew
Mode Studio
Upgrade your account
HAVING
Run Selected
☑ Limit 100
Format
View history
Mode Public Warehouse (everyone)
Search this Connection...
tutorial.flights
animial_crossing_accessories
billboard_top_100_year_end
city_populations
crunchbase_acquisitions
crunchbase_acquisitions_clean_date
crunchbase_companies
crunchbase_companies_clean_date
crunchbase_investments
crunchbase_investments_part1
crunchbase_investments_part2
Succeeded
Data
Fields
Source
Export
Copy
flights
scheduled_arrival_time number
scheduled_departure_time number
scheduled_flight_time number

```sql
SELECT day_of_week,
       COUNT(airline_name) AS TOTAL_PER_WEEK
FROM tutorial.flights
GROUP BY day_of_week
HAVING day_of_week = 'Friday'
```

	day_of_week	total_per_week
1	Friday	82608

#1.3 HAVING



```sql
SELECT origin_state,
       COUNT(origin_state) AS COUNT_PER_STATE
FROM tutorial.flights
GROUP BY origin_state
HAVING origin_state IN ('Florida', 'New York', 'Texas', 'Arizona', 'Pennsylvania')
ORDER BY COUNT_PER_STATE
```

	origin_state	count_per_state
1	Pennsylvania	15143
2	Arizona	17173
3	New York	31476
4	Florida	50169
5	Texas	63061

#1.3 HAVING

As mentioned last week, the order in which you write the clauses is **important.**

Logical Execution:

1.FROM →
Start with the table tutorial.billboard_top_100_year_end

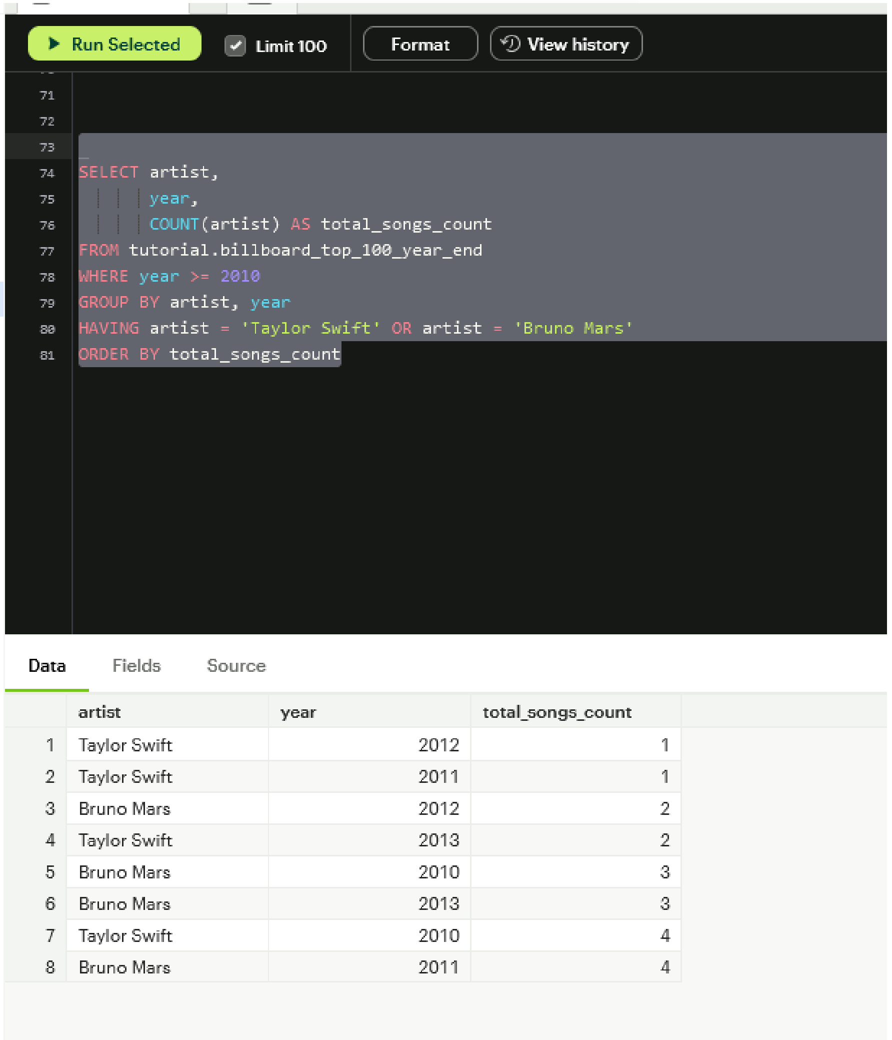
2.WHERE →
Filter rows where year >= 2010
This happens before grouping. Any rows that don't meet this condition are excluded immediately.

3. GROUP BY →
Group the filtered rows by artist and year
Each unique combination of artist and year becomes a group.

4. HAVING →
The HAVING clause filters groups after aggregation.
Here, only the groups where artist = 'Taylor Swift' or artist = 'Bruno Mars' are retained.

5.SELECT →
Choose columns artist, year and the count of artist as total_songs_count

5.ORDER BY →
Sort by the count number



```
SELECT artist,
       year,
       COUNT(artist) AS total_songs_count
FROM tutorial.billboard_top_100_year_end
WHERE year >= 2010
GROUP BY artist, year
HAVING artist = 'Taylor Swift' OR artist = 'Bruno Mars'
ORDER BY total_songs_count
```

	artist	year	total_songs_count
1	Taylor Swift	2012	1
2	Taylor Swift	2011	1
3	Bruno Mars	2012	2
4	Taylor Swift	2013	2
5	Bruno Mars	2010	3
6	Bruno Mars	2013	3
7	Taylor Swift	2010	4
8	Bruno Mars	2011	4

#1.3 HAVING

Logical Execution:

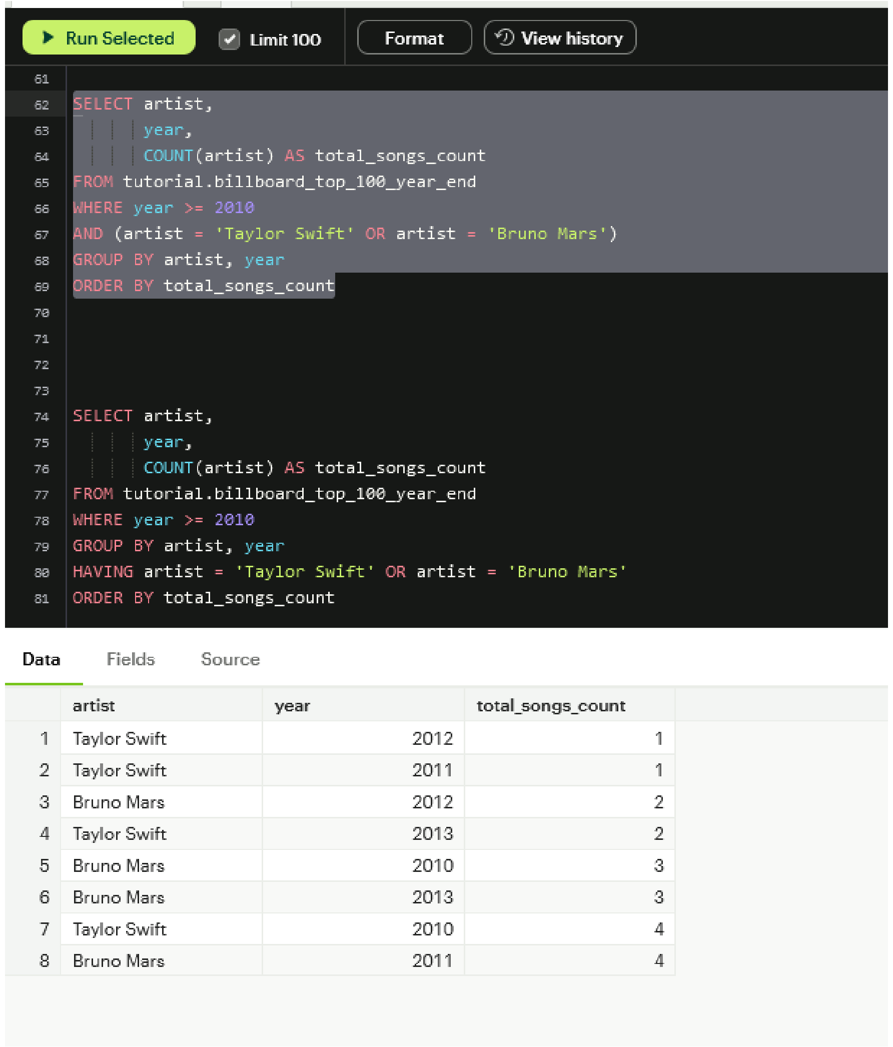
1. FROM →
Start with the table tutorial.billboard_top_100_year_end

2. WHERE →
Filter rows where
year >= 2010
artist name is either Taylor Swift OR Bruno Mars

3. GROUP BY →
Group the filtered rows by artist and year

4. SELECT →
Choose columns artist, year and the count of artist as total_songs_count

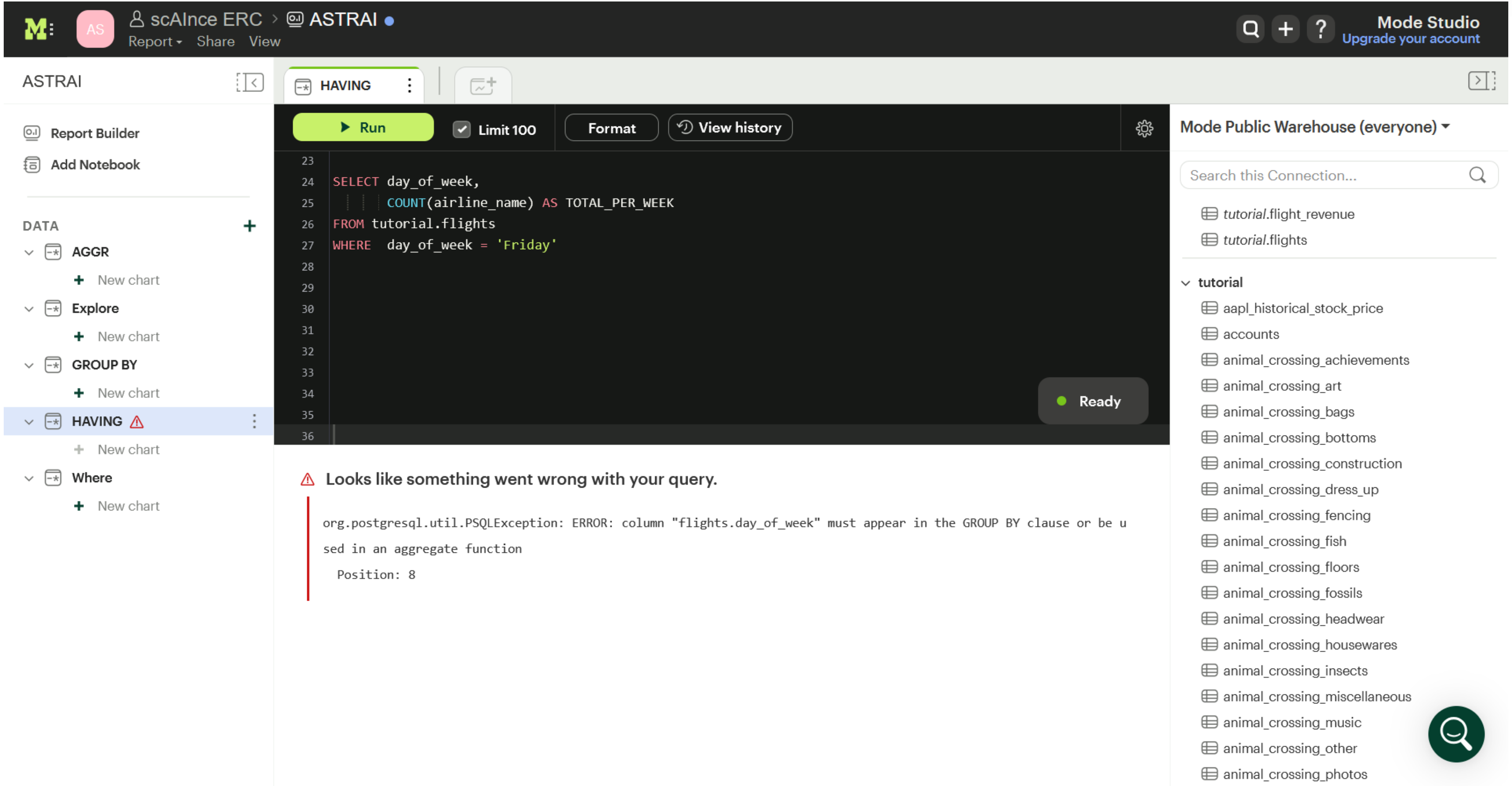
5.ORDER BY →
Sort by the count number



```
SELECT artist,
       year,
       COUNT(artist) AS total_songs_count
FROM tutorial.billboard_top_100_year_end
WHERE year >= 2010
AND (artist = 'Taylor Swift' OR artist = 'Bruno Mars')
GROUP BY artist, year
ORDER BY total_songs_count




SELECT artist,
       year,
       COUNT(artist) AS total_songs_count
FROM tutorial.billboard_top_100_year_end
WHERE year >= 2010
GROUP BY artist, year
HAVING artist = 'Taylor Swift' OR artist = 'Bruno Mars'
ORDER BY total_songs_count
```

	artist	year	total_songs_count
1	Taylor Swift	2012	1
2	Taylor Swift	2011	1
3	Bruno Mars	2012	2
4	Taylor Swift	2013	2
5	Bruno Mars	2010	3
6	Bruno Mars	2013	3
7	Taylor Swift	2010	4
8	Bruno Mars	2011	4

#1.3 HAVING



```sql
SELECT day_of_week,
        COUNT(airline_name) AS TOTAL_PER_WEEK
FROM tutorial.flights
WHERE  day_of_week = 'Friday'
```

Looks like something went wrong with your query.

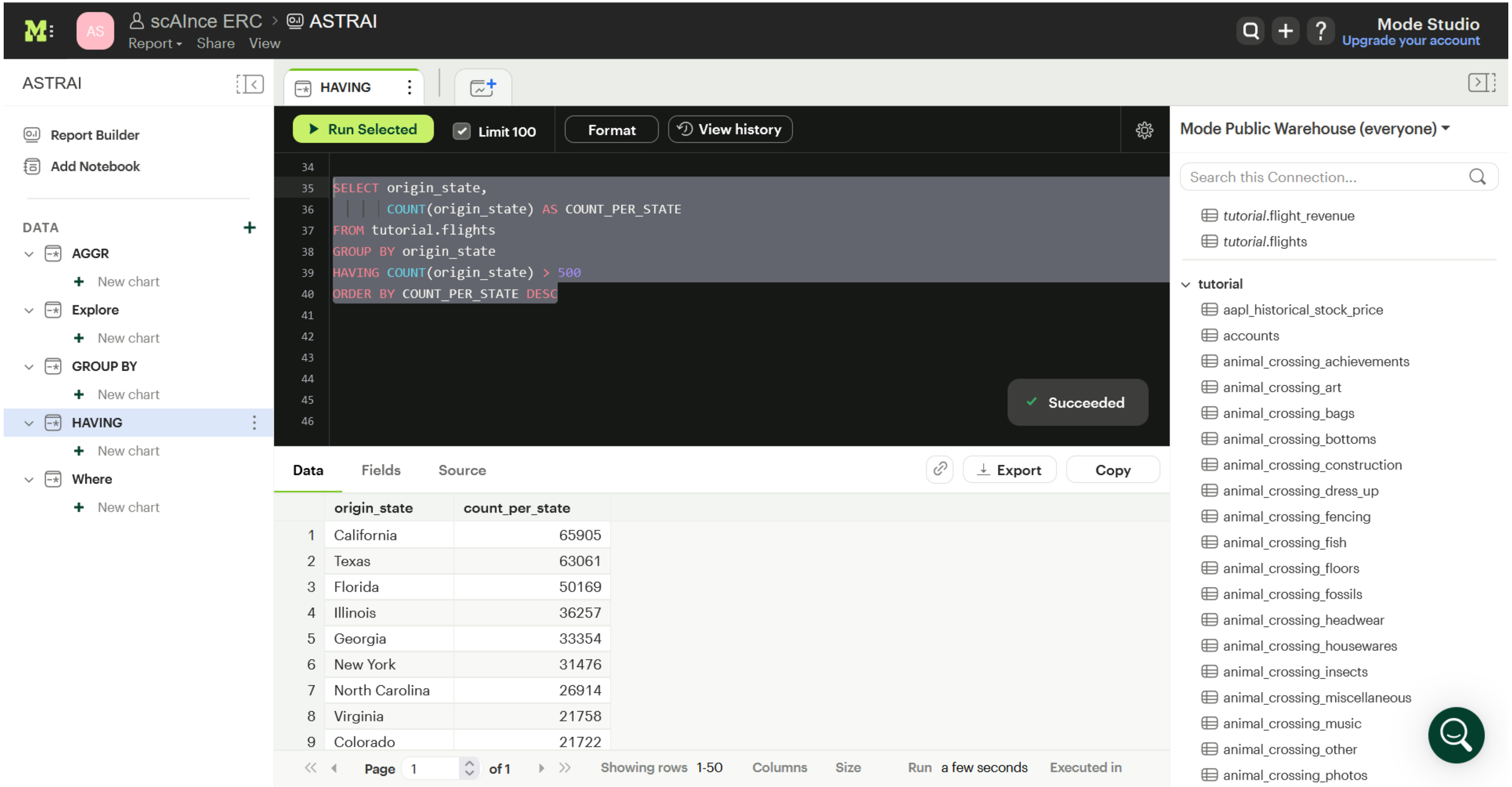
```
org.postgresql.util.PSQLException: ERROR: column "flights.day_of_week" must appear in the GROUP BY clause or be u
sed in an aggregate function
  Position: 8
```

Why do we have an error message?

#1.3 HAVING

2. Write a query to count and display the origin_state in the tutorial.flights table where the count of flights from each state exceeds 500 and sort the result in descending order of flight count.

#1.3 HAVING



```sql
SELECT origin_state,
       COUNT(origin_state) AS COUNT_PER_STATE
FROM tutorial.flights
GROUP BY origin_state
HAVING COUNT(origin_state) > 500
ORDER BY COUNT_PER_STATE DESC
```

	origin_state	count_per_state
1	California	65905
2	Texas	63061
3	Florida	50169
4	Illinois	36257
5	Georgia	33354
6	New York	31476
7	North Carolina	26914
8	Virginia	21758
9	Colorado	21722

#1.4 SQL JOIN

Up to this point, we've only been working with one table at a time. The real power of SQL, however, comes from **working with data from multiple tables at once**. If you remember from last week’s session, the term "relational database" refers to the fact that the tables within it **"relate" to one another**—they contain **common identifiers that allow information from multiple tables to be combined easily.**

#1.4 SQL JOIN

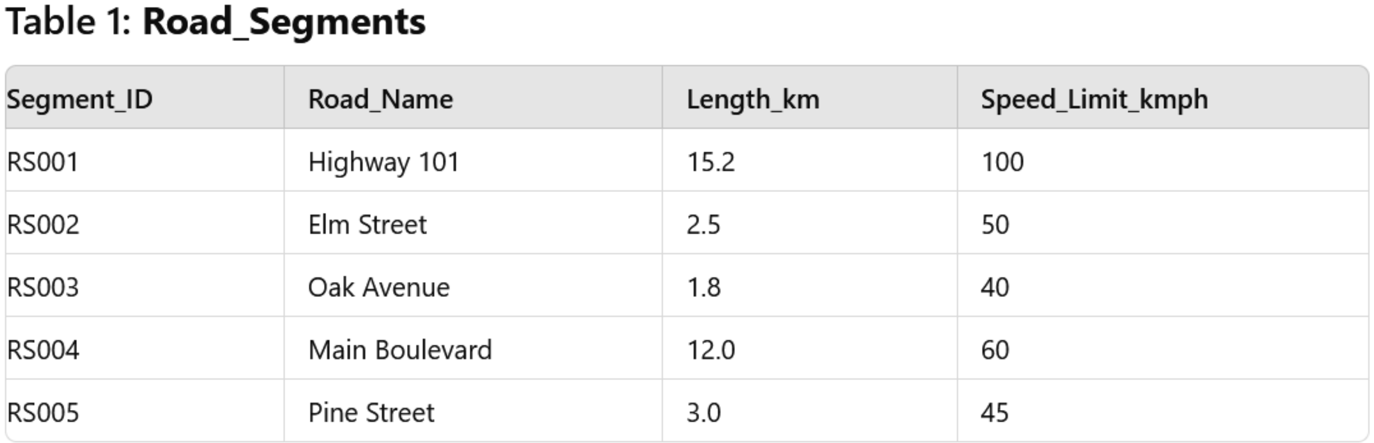


Table 1: **Road_Segments**

Segment_ID	Road_Name	Length_km	Speed_Limit_kmph
RS001	Highway 101	15.2	100
RS002	Elm Street	2.5	50
RS003	Oak Avenue	1.8	40
RS004	Main Boulevard	12.0	60
RS005	Pine Street	3.0	45

What is the Primary Key?

What is the forign Key?

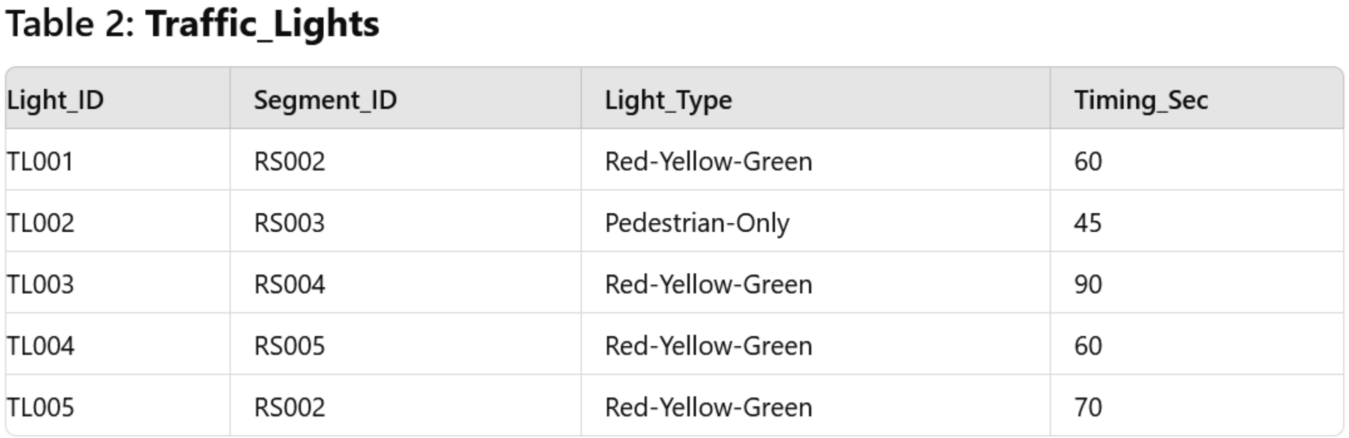
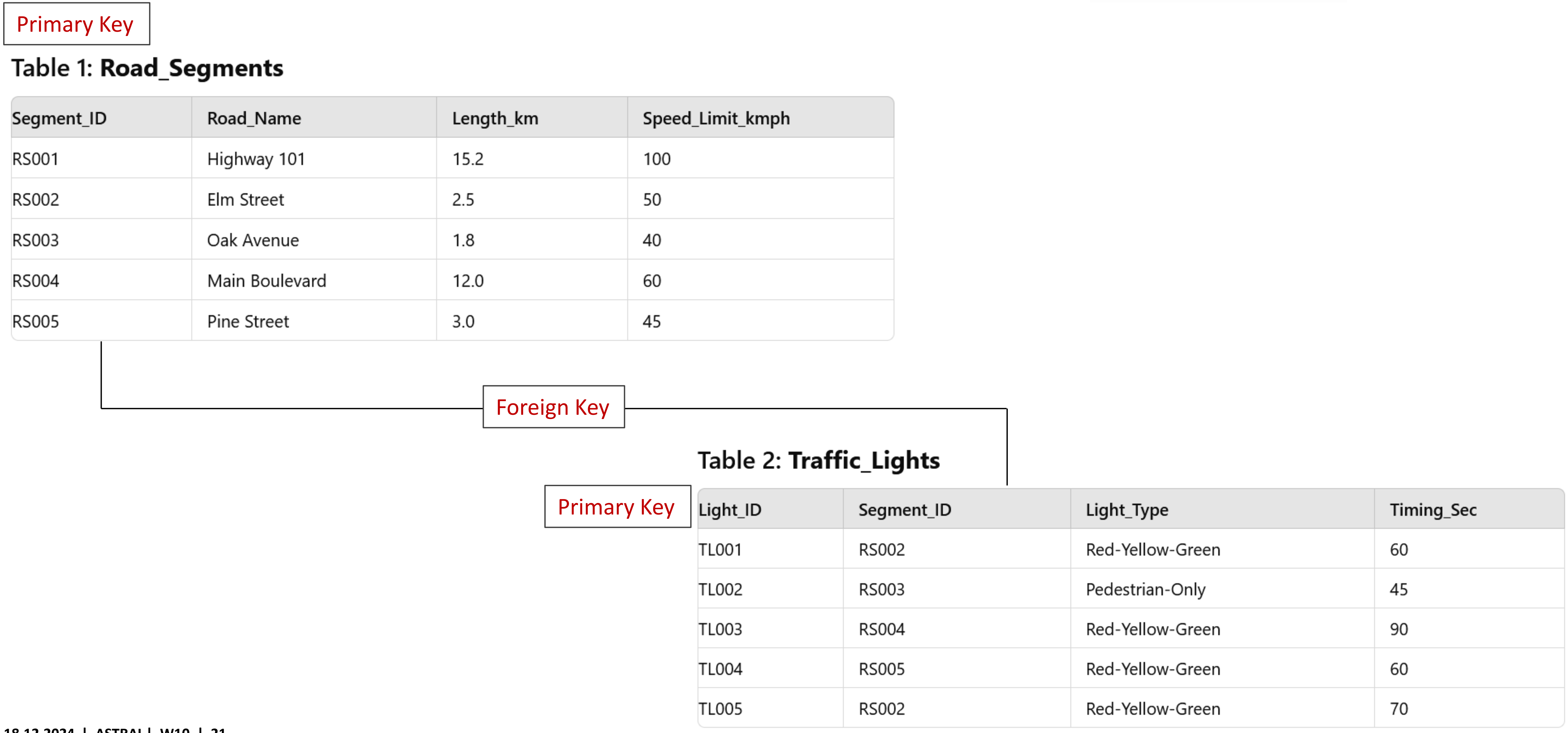


Table 2: **Traffic_Lights**

Light_ID	Segment_ID	Light_Type	Timing_Sec
TL001	RS002	Red-Yellow-Green	60
TL002	RS003	Pedestrian-Only	45
TL003	RS004	Red-Yellow-Green	90
TL004	RS005	Red-Yellow-Green	60
TL005	RS002	Red-Yellow-Green	70

What is the Primary Key?

#1.4 SQL JOIN



Primary Key

Table 1: **Road_Segments**

Segment_ID	Road_Name	Length_km	Speed_Limit_kmph
RS001	Highway 101	15.2	100
RS002	Elm Street	2.5	50
RS003	Oak Avenue	1.8	40
RS004	Main Boulevard	12.0	60
RS005	Pine Street	3.0	45

Foreign Key

Table 2: **Traffic_Lights**

Primary Key

Light_ID	Segment_ID	Light_Type	Timing_Sec
TL001	RS002	Red-Yellow-Green	60
TL002	RS003	Pedestrian-Only	45
TL003	RS004	Red-Yellow-Green	90
TL004	RS005	Red-Yellow-Green	60
TL005	RS002	Red-Yellow-Green	70

#1.4 SQL JOIN

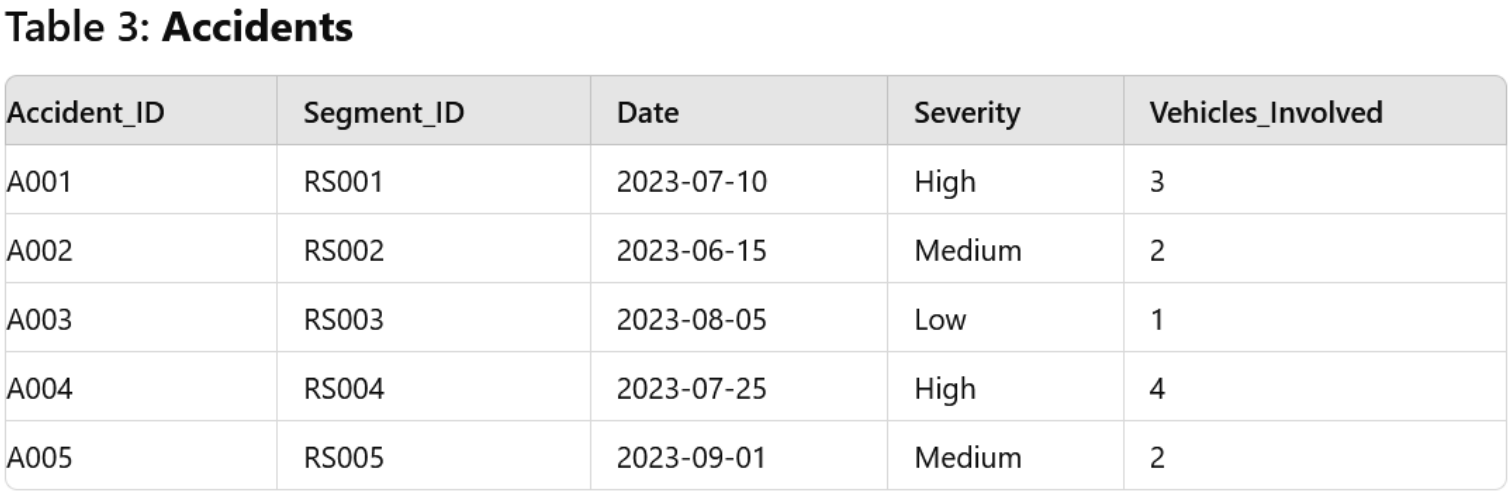


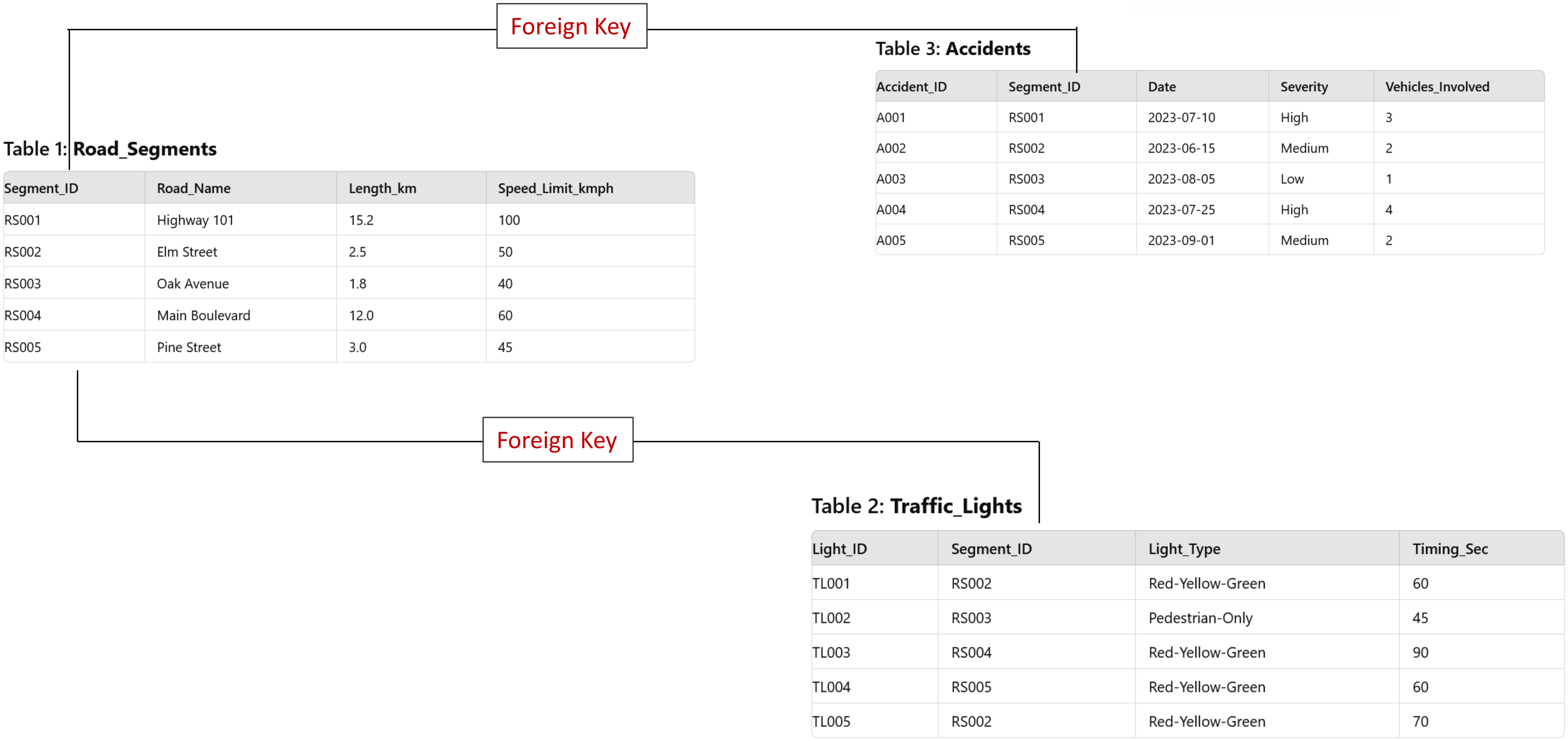
Table 3: **Accidents**

Accident_ID	Segment_ID	Date	Severity	Vehicles_Involved
A001	RS001	2023-07-10	High	3
A002	RS002	2023-06-15	Medium	2
A003	RS003	2023-08-05	Low	1
A004	RS004	2023-07-25	High	4
A005	RS005	2023-09-01	Medium	2

What is the Primary Key here?

To which table can I connect this table? How?

#1.4 SQL JOIN



Foreign Key

Table 3: **Accidents**

Accident_ID	Segment_ID	Date	Severity	Vehicles_Involved
A001	RS001	2023-07-10	High	3
A002	RS002	2023-06-15	Medium	2
A003	RS003	2023-08-05	Low	1
A004	RS004	2023-07-25	High	4
A005	RS005	2023-09-01	Medium	2

Table 1: **Road_Segments**

Segment_ID	Road_Name	Length_km	Speed_Limit_kmph
RS001	Highway 101	15.2	100
RS002	Elm Street	2.5	50
RS003	Oak Avenue	1.8	40
RS004	Main Boulevard	12.0	60
RS005	Pine Street	3.0	45

Foreign Key

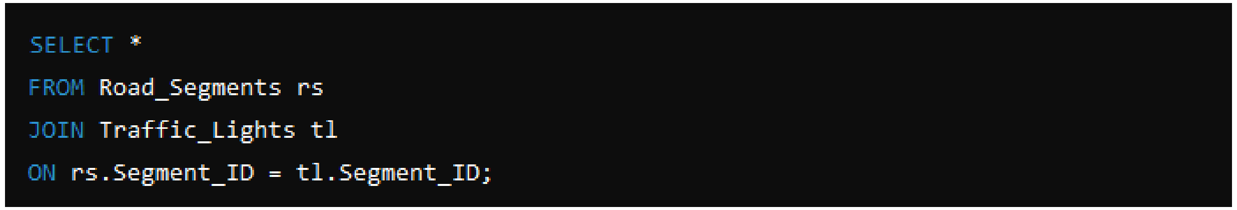
Table 2: **Traffic_Lights**

Light_ID	Segment_ID	Light_Type	Timing_Sec
TL001	RS002	Red-Yellow-Green	60
TL002	RS003	Pedestrian-Only	45
TL003	RS004	Red-Yellow-Green	90
TL004	RS005	Red-Yellow-Green	60
TL005	RS002	Red-Yellow-Green	70

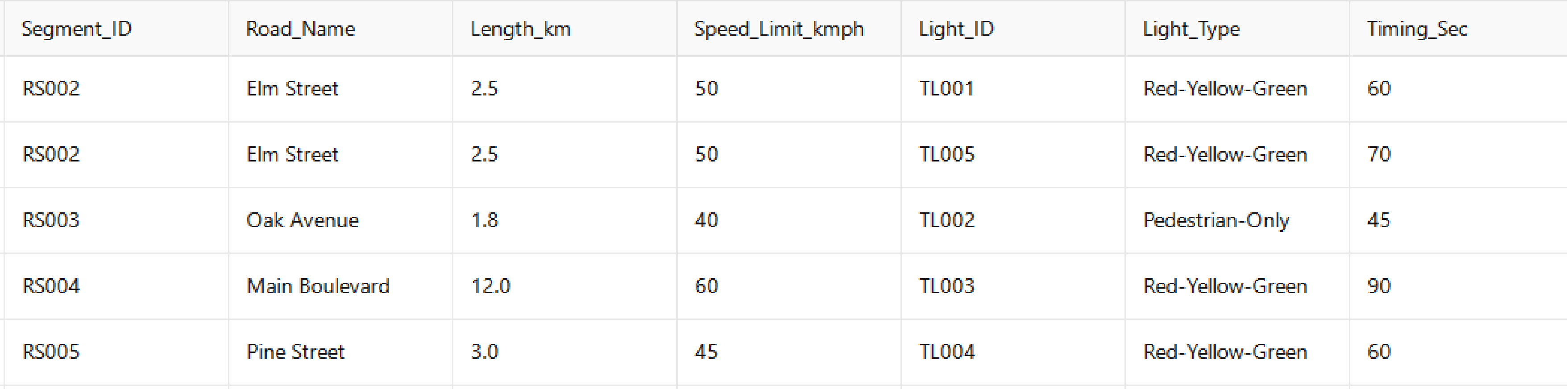
#1.4 SQL JOIN

When performing joins, it's easiest to give your table names aliases.

ON indicates how the two tables (the one after the FROM and the one after the JOIN) relate to each other.



```sql
SELECT *
FROM Road_Segments rs
JOIN Traffic_Lights tl
ON rs.Segment_ID = tl.Segment_ID;
```

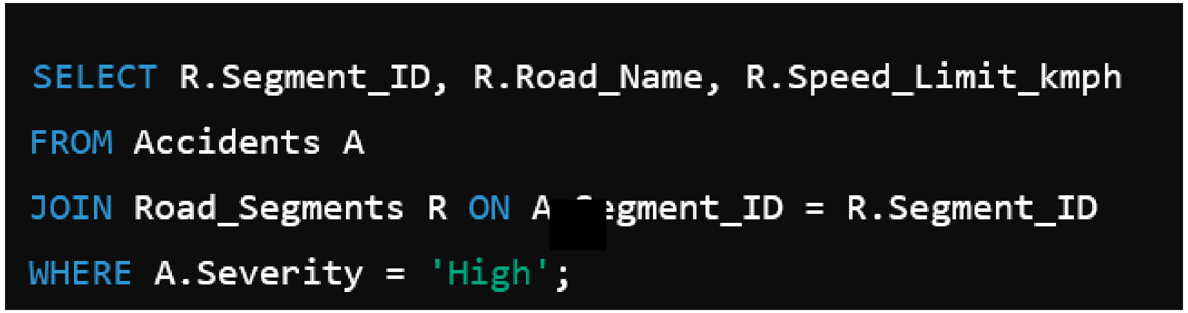


Segment_ID	Road_Name	Length_km	Speed_Limit_kmph	Light_ID	Light_Type	Timing_Sec
RS002	Elm Street	2.5	50	TL001	Red-Yellow-Green	60
RS002	Elm Street	2.5	50	TL005	Red-Yellow-Green	70
RS003	Oak Avenue	1.8	40	TL002	Pedestrian-Only	45
RS004	Main Boulevard	12.0	60	TL003	Red-Yellow-Green	90
RS005	Pine Street	3.0	45	TL004	Red-Yellow-Green	60

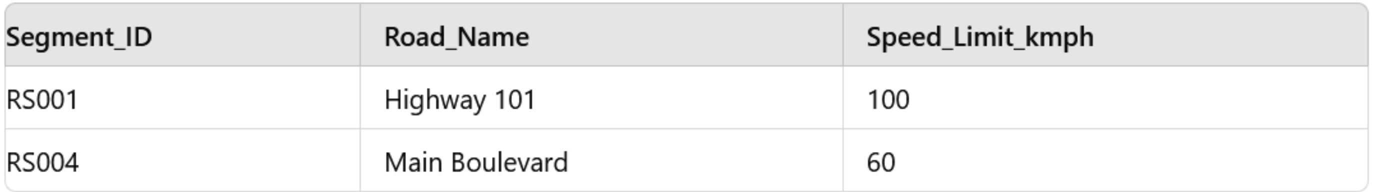
#1.4 SQL JOIN

I need the columns segment id, road name and speed limit WHERE severity column is = high

the filter happens after the tables are joined.



```
SELECT R.Segment_ID, R.Road_Name, R.Speed_Limit_kmph
FROM Accidents A
JOIN Road_Segments R ON A[redacted]gment_ID = R.Segment_ID
WHERE A.Severity = 'High';
```



Segment_ID	Road_Name	Speed_Limit_kmph
RS001	Highway 101	100
RS004	Main Boulevard	60

#1.4 SQL JOIN

You are given two tables on Mode:

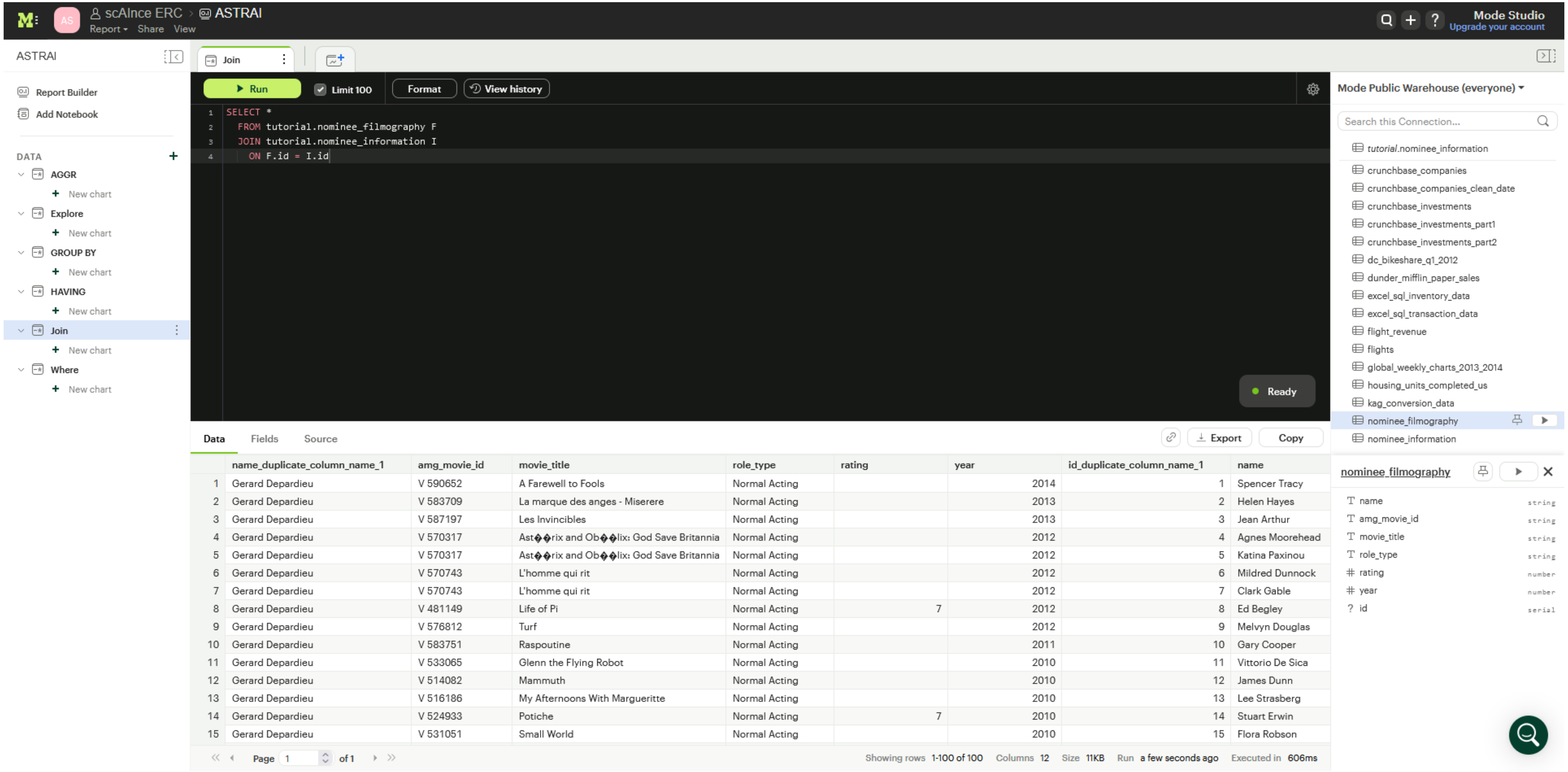
- tutorial.nominee_filmography
- tutorial.nominee_information

Your task is to:

1. Identify the foreign key that links the two tables.

2.Write an SQL query to join these tables based on the identified foreign key.

#1.4 SQL JOIN



```sql
SELECT *
  FROM tutorial.nominee_filmography F
  JOIN tutorial.nominee_information I
    ON F.id = I.id
```

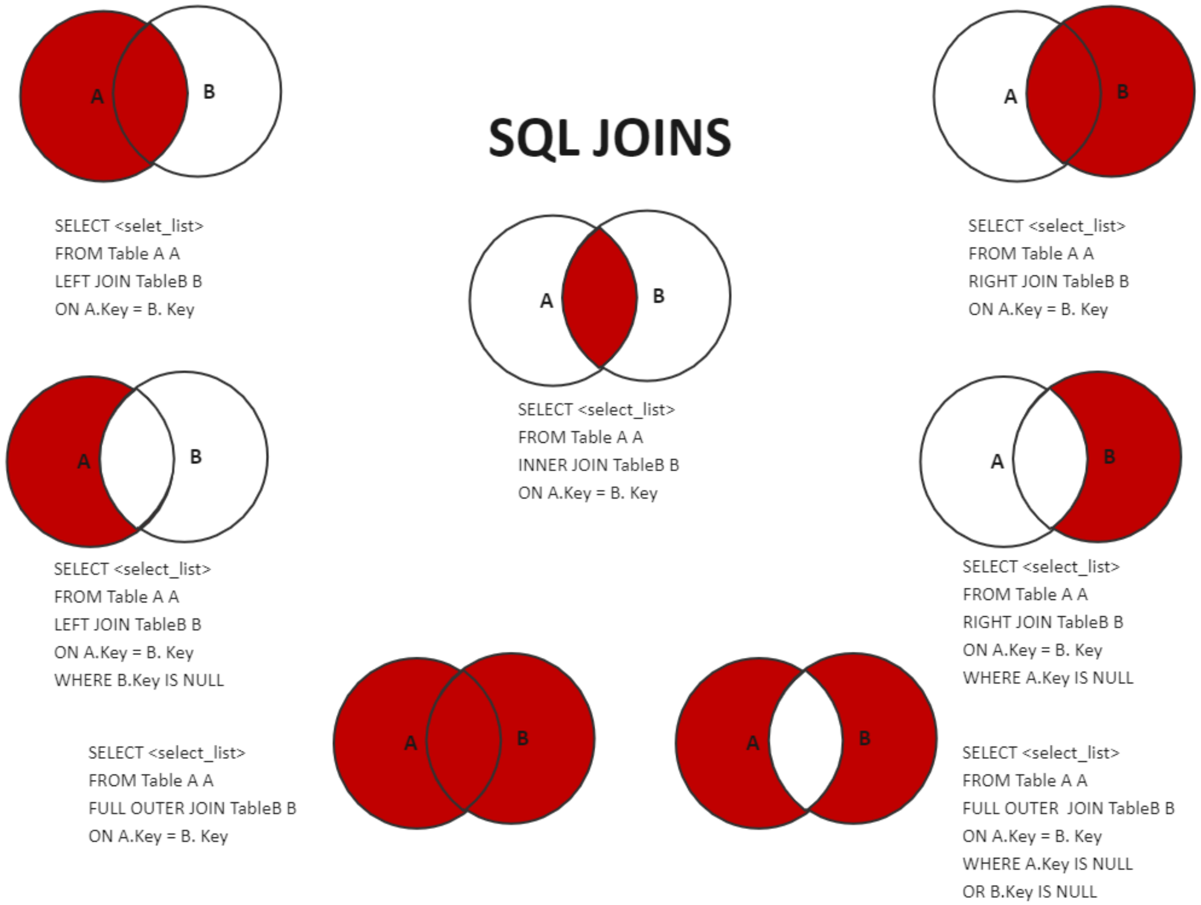
	name_duplicate_column_name_1	amg_movie_id	movie_title	role_type	rating	year	id_duplicate_column_name_1	name
1	Gerard Depardieu	V 590652	A Farewell to Fools	Normal Acting		2014	1	Spencer Tracy
2	Gerard Depardieu	V 583709	La marque des anges - Miserere	Normal Acting		2013	2	Helen Hayes
3	Gerard Depardieu	V 587197	Les Invincibles	Normal Acting		2013	3	Jean Arthur
4	Gerard Depardieu	V 570317	Ast��rix and Ob��lix: God Save Britannia	Normal Acting		2012	4	Agnes Moorehead
5	Gerard Depardieu	V 570317	Ast��rix and Ob��lix: God Save Britannia	Normal Acting		2012	5	Katina Paxinou
6	Gerard Depardieu	V 570743	L'homme qui rit	Normal Acting		2012	6	Mildred Dunnock
7	Gerard Depardieu	V 570743	L'homme qui rit	Normal Acting		2012	7	Clark Gable
8	Gerard Depardieu	V 481149	Life of Pi	Normal Acting	7	2012	8	Ed Begley
9	Gerard Depardieu	V 576812	Turf	Normal Acting		2012	9	Melvyn Douglas
10	Gerard Depardieu	V 583751	Raspoutine	Normal Acting		2011	10	Gary Cooper
11	Gerard Depardieu	V 533065	Glenn the Flying Robot	Normal Acting		2010	11	Vittorio De Sica
12	Gerard Depardieu	V 514082	Mammuth	Normal Acting		2010	12	James Dunn
13	Gerard Depardieu	V 516186	My Afternoons With Margueritte	Normal Acting		2010	13	Lee Strasberg
14	Gerard Depardieu	V 524933	Potiche	Normal Acting	7	2010	14	Stuart Erwin
15	Gerard Depardieu	V 531051	Small World	Normal Acting		2010	15	Flora Robson

Page 1 of 1   Showing rows 1-100 of 100   Columns 12   Size 11KB   Run a few seconds ago   Executed in 606ms

nominee_filmography

T name	string
T amg_movie_id	string
T movie_title	string
T role_type	string
# rating	number
# year	number
? id	serial

#1.4 SQL JOIN



SQL JOINS

```
SELECT <selet_list>
FROM Table A A
LEFT JOIN TableB B
ON A.Key = B. Key
```

```
SELECT <select_list>
FROM Table A A
INNER JOIN TableB B
ON A.Key = B. Key
```

```
SELECT <select_list>
FROM Table A A
RIGHT JOIN TableB B
ON A.Key = B. Key
```

```
SELECT <select_list>
FROM Table A A
LEFT JOIN TableB B
ON A.Key = B. Key
WHERE B.Key IS NULL
```

```
SELECT <select_list>
FROM Table A A
RIGHT JOIN TableB B
ON A.Key = B. Key
WHERE A.Key IS NULL
```

```
SELECT <select_list>
FROM Table A A
FULL OUTER JOIN TableB B
ON A.Key = B. Key
```

```
SELECT <select_list>
FROM Table A A
FULL OUTER  JOIN TableB B
ON A.Key = B. Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

ANY
QUESTIONS

#2 Data Visualization

- **Efficiently summarize large amounts of data through a graphical format.**

- There are many chart types available, each with their **own strengths and use cases.**

- One of the trickiest parts of the analysis process is **choosing the right way to represent your data** using one of these visualizations.

#2.1 Data Visualization

The process of visualizing your data:

1. Understand the context and have your questions and aim ready.

Build a clear understanding of **who** you are communicating to, **what** you need them to know or do, **how** you will communicate to them, and **what** data you have to back up your case.

#2.2 Data Visualization

2. Choose an appropriate visual display.

- When highlighting a number or two, simple text is best.
- Line charts are usually best for continuous data.
- Bar charts work great for categorical data and must have a zero baseline.
- Let the relationship you want to show guide the type of chart you choose

Visual Vocabulary:

https://public.tableau.com/app/profile/andy.kriebel/viz/VisualVocabulary/VisualVocabulary

#2.3 Data Visualization

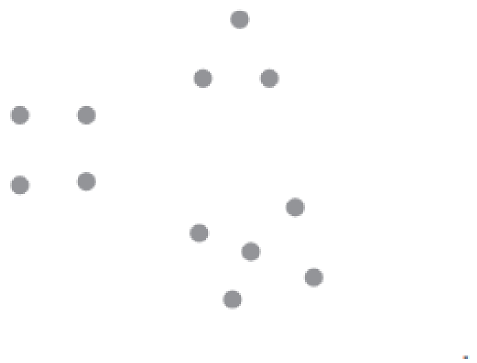
3. Identify elements that don't add informative value and remove them from your visuals.

- Remember: **clutter is your enemy!**
- Visual clutter creates excessive **cognitive load** that can **hinder** the transmission of our message.
- We do not see visuals with our eyes really, we see them with **our brains**.
- The **Gestalt Principles** provide a framework to **understand how people perceive and create order from visual stimuli.**
- Applying these principles can help you design visuals that align with how your audience naturally sees, allowing you to **identify** and **remove unnecessary visual elements** for clearer communication.

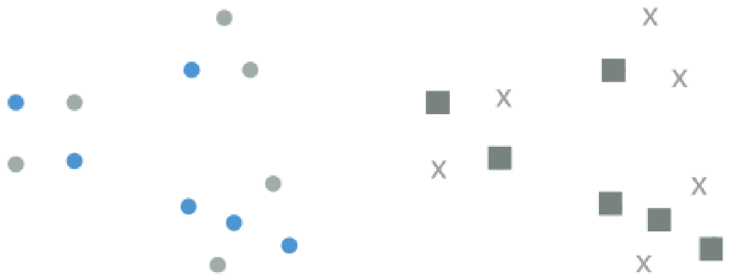
#2.3 Data Visualization

Proximity: We tend to think of objects that are physically close together as belonging to part of a group.

Similarity: Objects that are of similar color, shape, size, or orientation are perceived as related or belonging to part of a group.



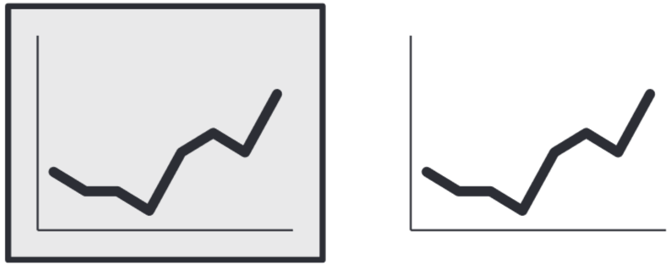
Gestalt principle of proximity



Gestalt principle of similarity

#2.3 Data Visualization

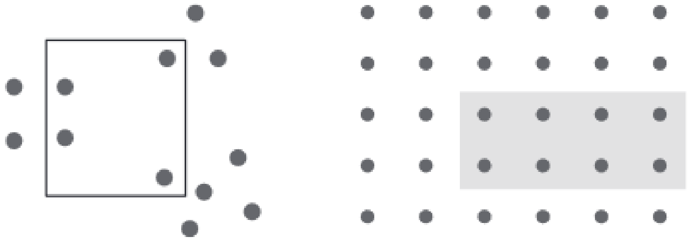
Closure: When parts of a whole are missing, our eyes naturally fill in the gap.



Gestalt principle of closure

your brain already sees the boundaries, so adding a box is redundant.

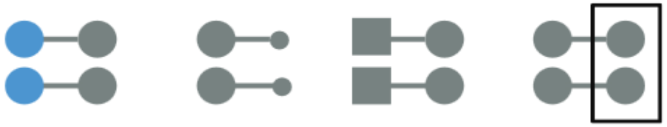
Enclosure: We think of objects that are physically enclosed together as belonging to part of a group. The aim is to highlight and group related content.



Gestalt principle of enclosure

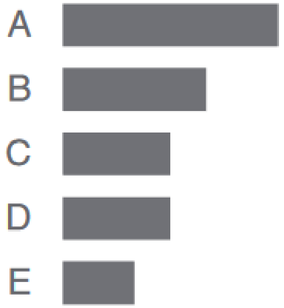
#2.3 Data Visualization

Connection: We tend to think of objects that are physically connected as part of a group. The connective property typically has a stronger associative value than similar color, size, or shape. Note when looking at the figure bellow, your eyes probably pair the shapes connected by lines (rather than similar color, size, or shape).



Gestalt principle of connection

Continuity is similar to closure: when looking at objects, our eyes seek the smoothest path and naturally create continuity in what we see even where it may not explicitly exist.



Gestalt principle of continuity and Simplicity

Simplicity: People tend to perceive and interpret visual elements in the simplest and most organized form possible.

Y-axis line removed stripping away unnecessary elements allows our data to stand out more.

#2.4 Data Visualization

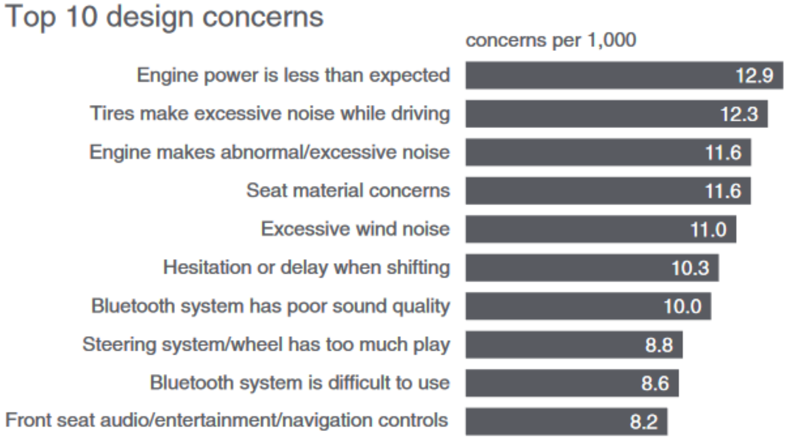
3. Focus attention where you want it.

- Employ the power of **preattentive attributes** like color, size, and position to signal what's important. Use these strategic attributes to **draw attention** to where you want your audience to look and guide your audience through your visual.

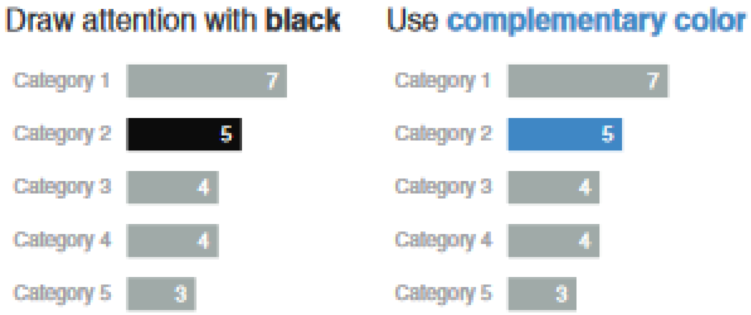
- Evaluate the effectiveness of preattentive attributes in your visual by applying the "where are your eyes drawn?"

- **Shapes Size matters.** Relative size denotes relative importance.

- Be **consistance** with colors.

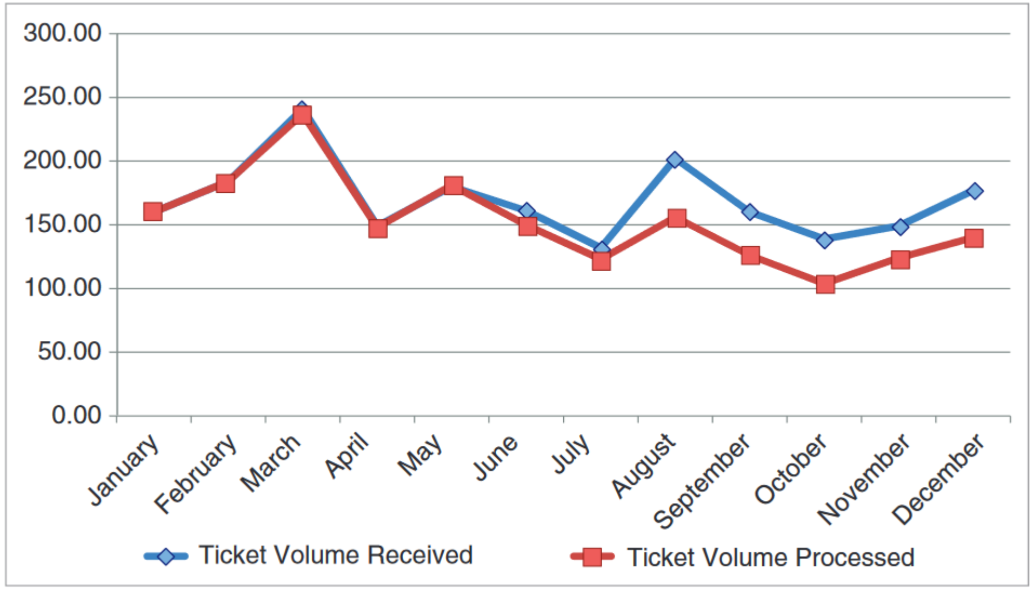


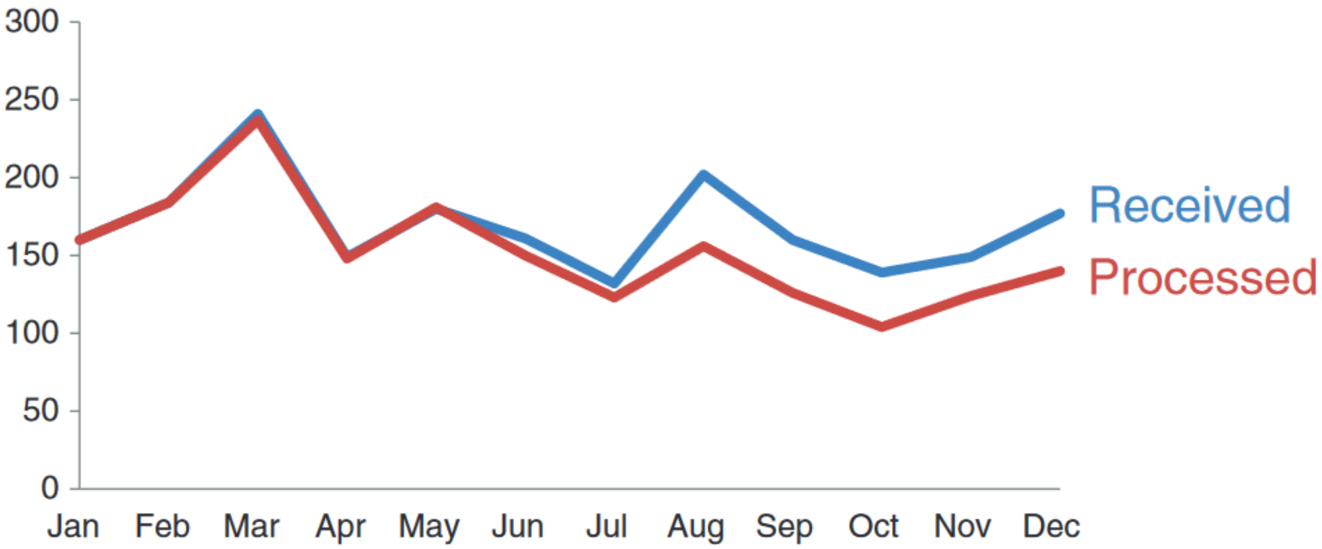
No preattentive attributes



Preattentive attributes

#4.1 Examples





1. Remove chart border. Closure Principle
2. Remove gridlines: This allows for greater contrast, and your data will stand out more. Simplicity Principle
3. Remove data markers. Continuity Principle
4. Clean up axis labels: will fit horizontally on the x-axis, eliminating the diagonal text and zeros. Simplicity Principle
5. Label data directly: Proximity and put the data labels right next to the data they describe. Proximity Principle
6. Leverage consistent color: Make the data labels the same color as the data they describe. Similarity Principle

#4.2 Examples

Research by MIT Prof. Sam Madden.

Telematics start-up

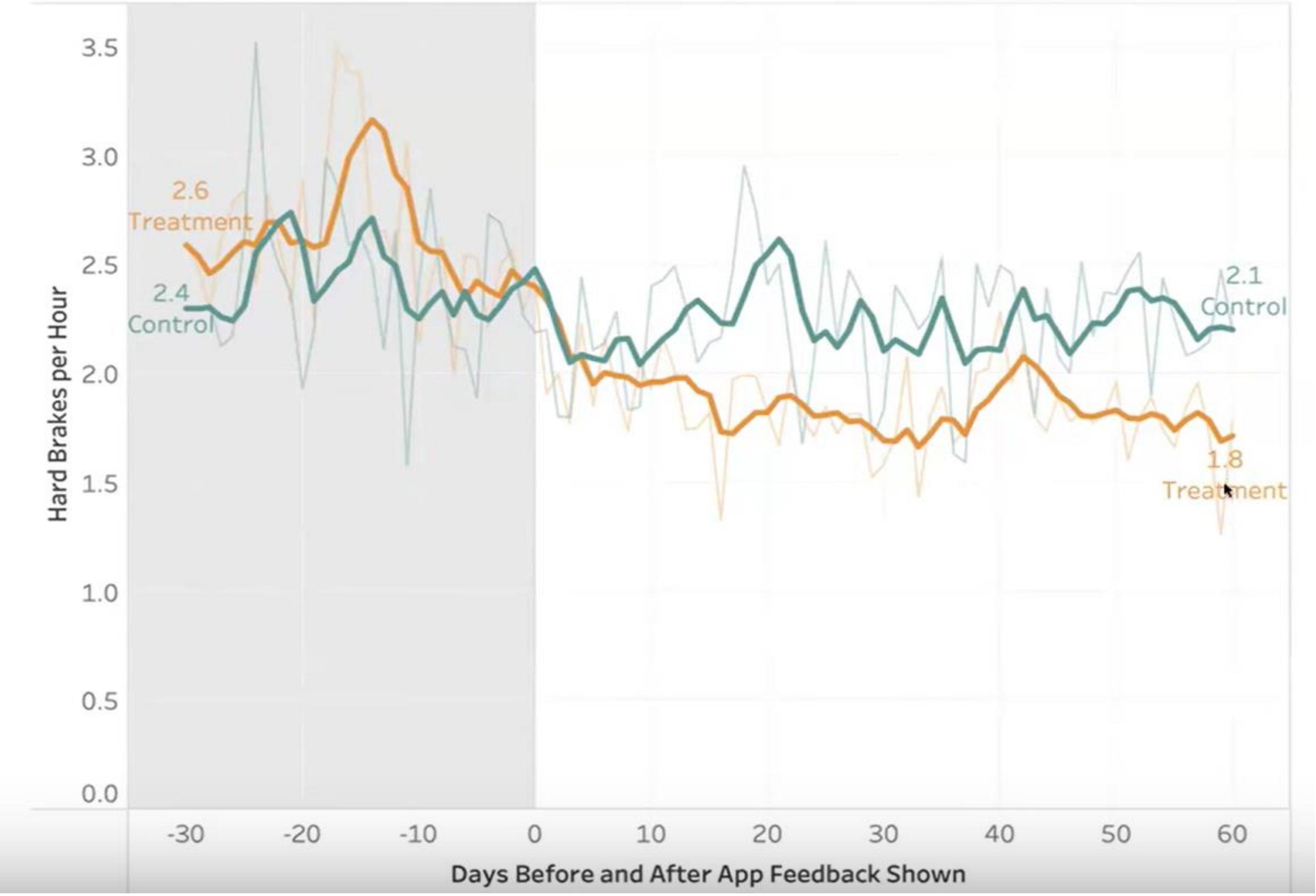
The mission is to enhance road and driver safety by connecting vehicles, gathering data, and using machine learning, AI, and sensor algorithms to provide insights and feedback that improve driver behavior.

Research: The Effect of Providing Driving Feedback – Braking

Are we going to make our roads safer by making drivers drive better?

The performance of two groups is compared based on their **harsh braking events** per hour over time.

The two groups: a control group with no intervention, and another group that received feedback.



Why is this a good visual?
Can you suggest something to make it better?

- **THANK YOU**
- **DANKE**