

ELECTIVE III : Data Analytics - Use of Select Tools and Techniques

CSE 457

CA4 Project

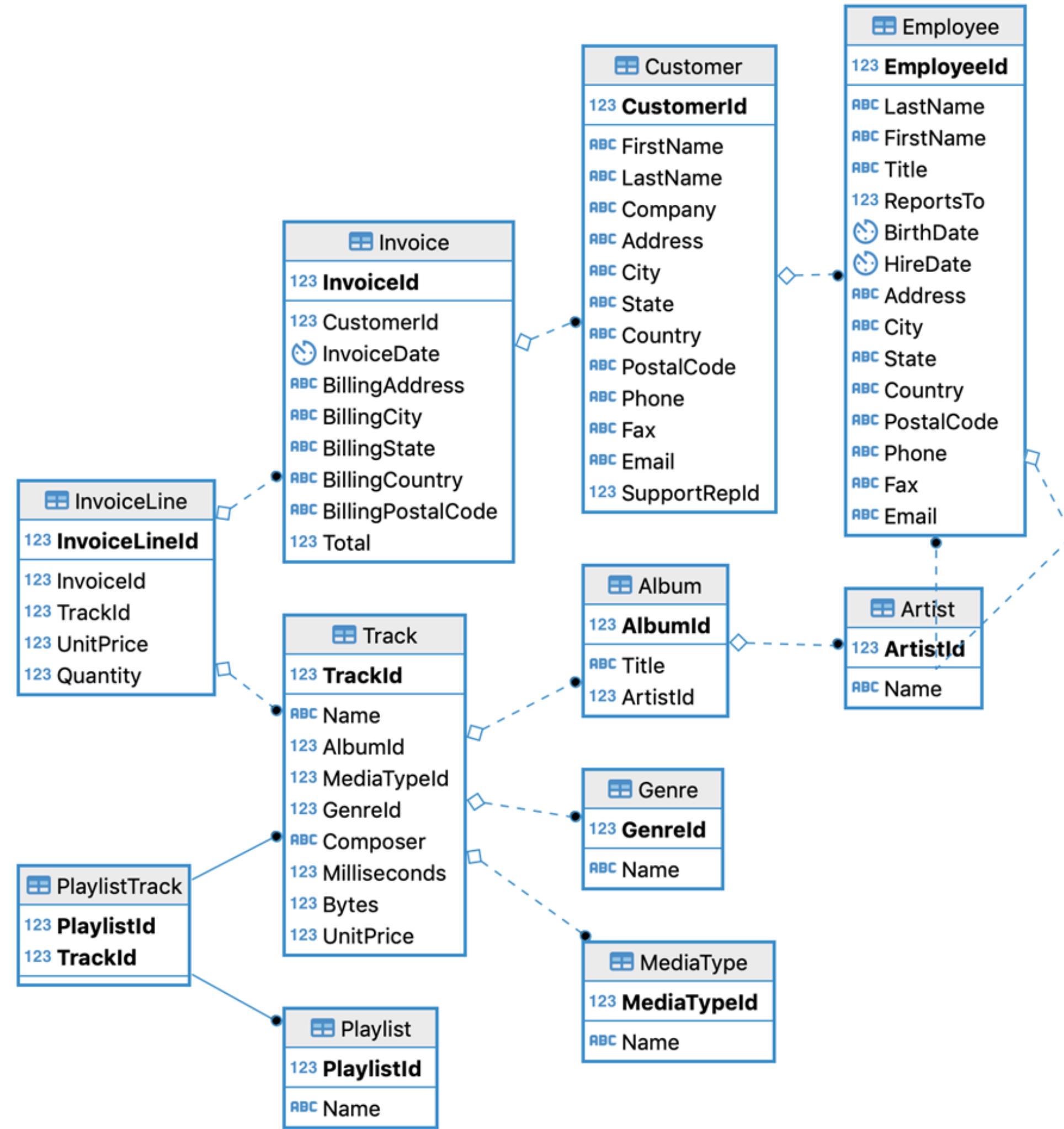
Done by

Gokul P - E0119056
Saai Swetha M K - E0119016

- **Chinook Db**
data model represents a digital music media store, including tables for artists, albums, media tracks, invoices and customers
- **Car India Dataset**
information on each car's model type, makers, no.of cylinders, height, width etc.

Datasets

1. Chinook Database
2. Cars India dataset



Postgres

https://raw.githubusercontent.com/xivSolutions/ChinookDb_Pg_Modified/master/chinook_pg_serial_pk_proper_naming.sql

The screenshot shows the pgAdmin 4 interface with the following details:

- Servers:** PostgreSQL > chinook
- Databases:** chinook, dvdrental, postgres
- Login/Group Roles:** 12
- Tablespaces:** 2
- PostgreSQL 14**

Query History:

```
1 ****
2 Chinook Database - Version 1.4
3 Script: Chinook_PostgreSQL.sql
4 Description: Creates and populates the Chinook database.
5 DB Server: PostgreSQL
6 Author: Luis Rocha
7 License: http://www.codeplex.com/ChinookDatabase/license
8
9
10 Modified By: John Atten
11 Modification Date: 2/1/2014
```

Data output:

ALTER TABLE

Query returned successfully in 6 secs 901 msec.

	track_id [PK] integer	name character varying (200)	album_id integer	media_type_id integer	genre_id integer	composer character varying (220)	milliseconds integer	bytes integer	unit_price numeric (10
1	1	For Those About To Ro...	1	1	1	Angus Young, Malcolm...	343719	11170334	
2	2	Balls to the Wall	2	2	1	[null]	342562	5510424	
3	3	Fast As a Shark	3	2	1	E. Baltes, S. Kaufman, ...	230619	3990994	
4	4	Restless and Wild	3	2	1	E. Baltes, R.A. Smith-Di...	252051	4331779	
5	5	Princess of the Dawn	3	2	1	Deaffy & R.A. Smith-Di...	375418	6290521	
6	6	Put The Finger On You	1	1	1	Angus Young, Malcolm...	205662	6713451	
7	7	Let's Get It Up	1	1	1	Angus Young, Malcolm...	233926	7636561	
8	8	Inject The Venom	1	1	1	Angus Young, Malcolm...	210834	6852860	
9	9	Snowballed	1	1	1	Angus Young, Malcolm...	203102	6599424	
10	10	Evil Walks	1	1	1	Angus Young, Malcolm...	263497	8611245	

Postgres

**No.of songs whose duration is less than or equal to 2 mins or
120000ms**

Query Query History

5
6 select count(*) from track where milliseconds<=120000;

Data output Messages Notifications

☰ + ↻ ↴ ↵ ↷ ↸ ↹

	count	bigint	🔒
1	94		

Postgres

Average price of the music tracks composed by Steven Tyler

Query Query History

```
7 /*average unit price of the tracks composed by steven tyler*/
8 select avg(unit_price) from track where composer like '%Steven Tyler%';
```

Data output Messages Notifications

avg
numeric 

	avg
1	0.990000000

Postgres

Country wise total sales in the year 2011

```
14 select billing_country,sum(total) as TotalSales from invoice where EXTRACT(year FROM invoice_date)=2011  
15 group by billing_country;  
16
```

Data output Messages Notifications

The screenshot shows a database client interface with a query results window. The query selects total sales by country for the year 2011. The results are displayed in a table with two columns: 'billing_country' and 'totalsales'. The table has 12 rows, each representing a country and its total sales for 2011. The client interface includes a toolbar with various icons and a menu bar at the top.

	billing_country	totalsales
1	Argentina	0.99
2	Australia	1.98
3	Belgium	24.75
4	Brazil	19.80
5	Canada	55.44
6	Chile	5.94
7	Czech Republic	12.87
8	Finland	15.88
9	France	42.61
10	Germany	48.57
11	India	24.75
12	Ireland	32.75

Postgres

Country with highest sales in the year 2009

```
19 select billing_country,sum(total) as TotalSales from invoice  
20 where EXTRACT(year FROM invoice_date)=2009  
21 group by billing_country  
22 order by TotalSales desc limit 1;  
23
```

Data output Messages Notifications



	billing_country	totalsales
	character varying (40)	numeric
1	USA	103.95

Postgres

Country with most invoices

```
27 SELECT billing_country, COUNT(invoice_id) FROM invoice  
28 GROUP BY 1  
29 ORDER BY 2 DESC;  
30
```

Data output Messages Notifications



	billing_country character varying (40)	count bigint
1	USA	91
2	Canada	56
3	France	35
4	Brazil	35
5	Germany	28
6	United Kingdom	21
7	Portugal	14
8	Czech Republic	14
9	India	13
10	Norway	7
11	Sweden	7

Postgres

Country with most invoices

```
27 SELECT billing_country, COUNT(invoice_id) FROM invoice  
28 GROUP BY 1  
29 ORDER BY 2 DESC;  
30
```

Data output Messages Notifications



	billing_country character varying (40)	count bigint
1	USA	91
2	Canada	56
3	France	35
4	Brazil	35
5	Germany	28
6	United Kingdom	21
7	Portugal	14
8	Czech Republic	14
9	India	13
10	Norway	7
11	Sweden	7

Postgres

Highest no.of music tracks purchased by a single customer

```
21  SELECT count(*)
22  FROM CUSTOMER C
23  JOIN INVOICE I ON C.customer_id = I.customer_id
24  GROUP BY I.customer_id
25  ORDER BY 1 DESC Limit 1;
26
```

Data output Messages Notifications



	count	
	bigint	🔒
1	7	

Postgres

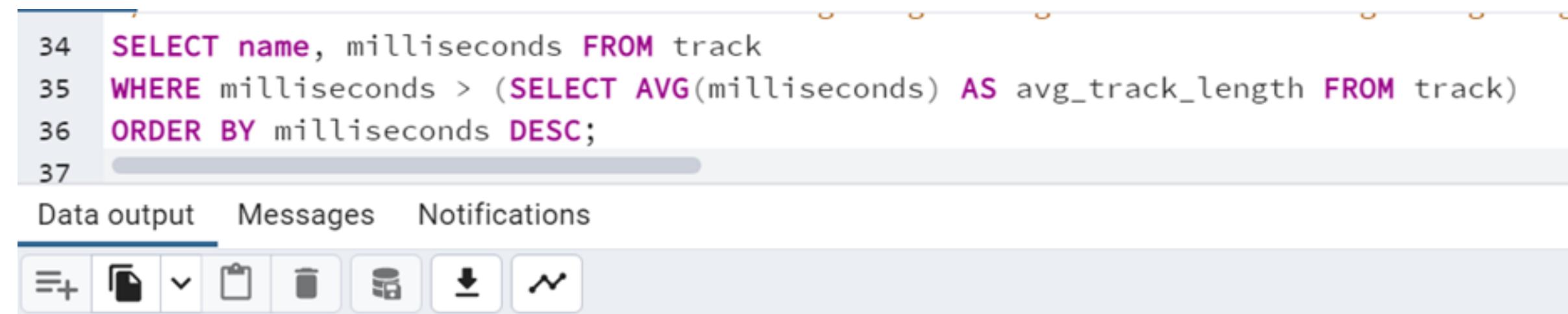
No.of songs per Genre

Postgres

List of all music tracks that has a song length greater than avg song length

```
34  SELECT name, milliseconds FROM track
35  WHERE milliseconds > (SELECT AVG(milliseconds) AS avg_track_length FROM track)
36  ORDER BY milliseconds DESC;
37
```

Data output Messages Notifications



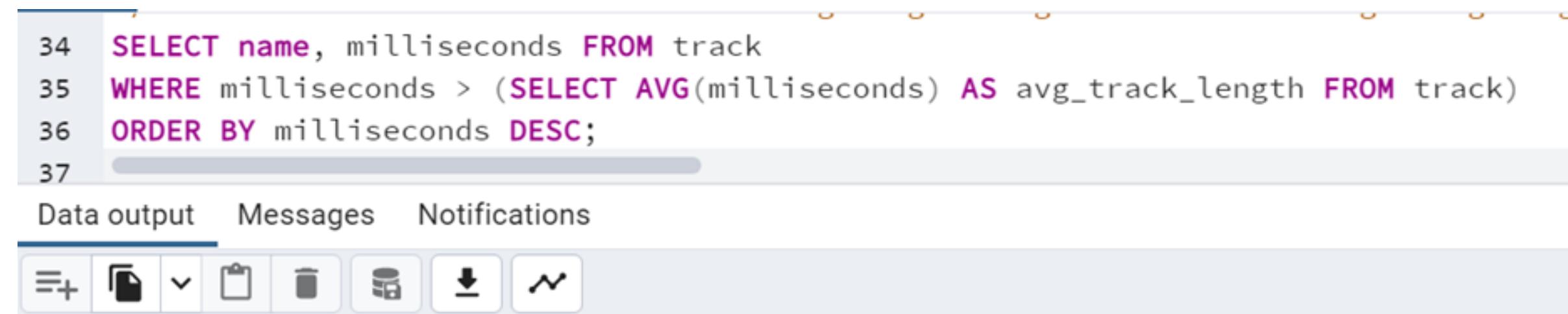
	name character varying (200)	milliseconds integer
1	Occupation / Precipice	5286953
2	Through a Looking Gla...	5088838
3	Greetings from Earth, ...	2960293
4	The Man With Nine Liv...	2956998
5	Battlestar Galactica, Pt...	2956081
6	Battlestar Galactica, Pt...	2952702
7	Murder On the Rising S...	2935894
8	Battlestar Galactica, Pt...	2927802
9	Take the Celestra	2927677
10	Fire In Space	2926593
11	The Long Patrol	2925008

Postgres

List of all music tracks that has a song length greater than avg song length

```
34  SELECT name, milliseconds FROM track
35  WHERE milliseconds > (SELECT AVG(milliseconds) AS avg_track_length FROM track)
36  ORDER BY milliseconds DESC;
37
```

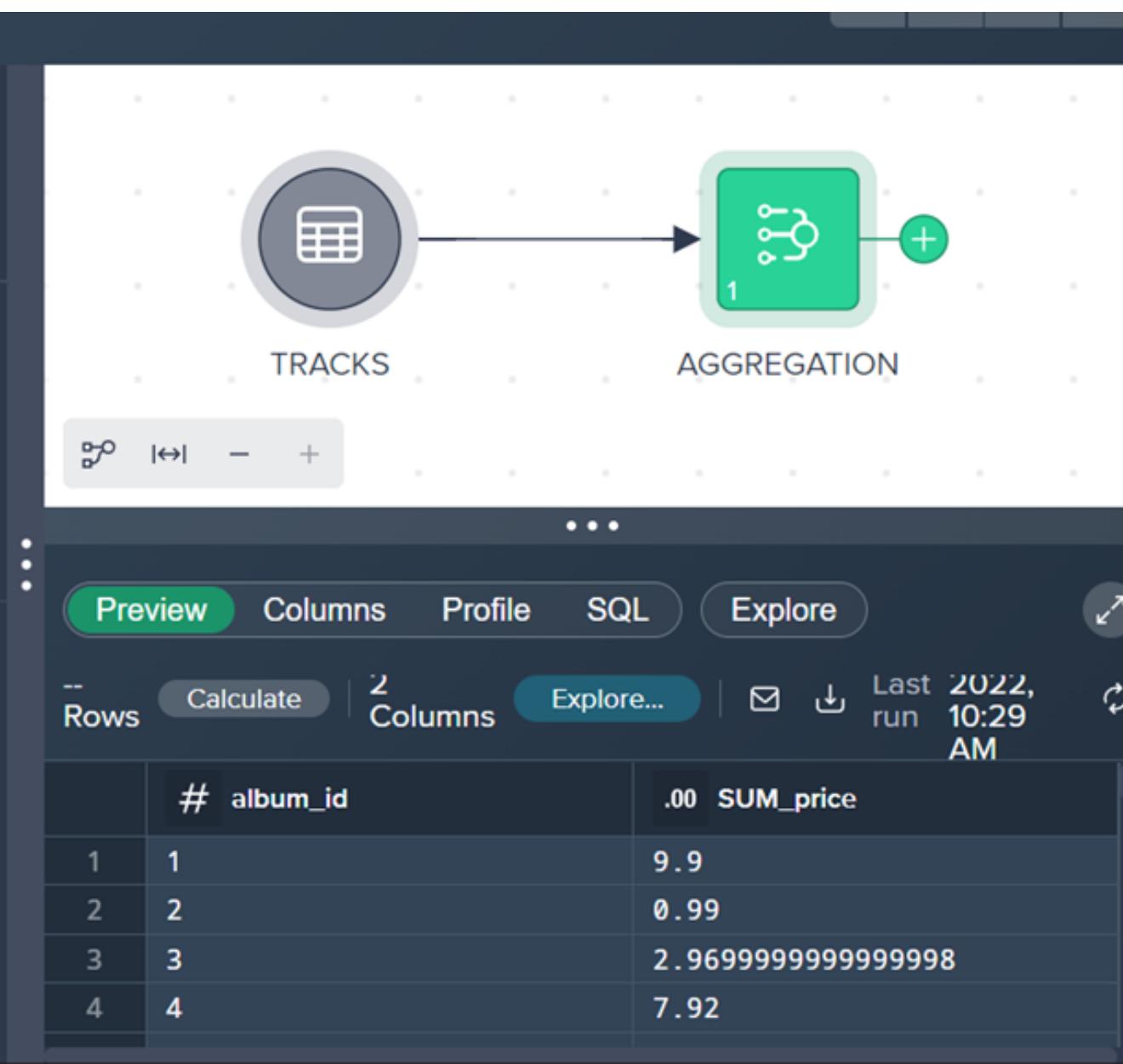
Data output Messages Notifications



	name character varying (200)	milliseconds integer
1	Occupation / Precipice	5286953
2	Through a Looking Gla...	5088838
3	Greetings from Earth, ...	2960293
4	The Man With Nine Liv...	2956998
5	Battlestar Galactica, Pt...	2956081
6	Battlestar Galactica, Pt...	2952702
7	Murder On the Rising S...	2935894
8	Battlestar Galactica, Pt...	2927802
9	Take the Celestra	2927677
10	Fire In Space	2926593
11	The Long Patrol	2925008

Datameer

Album wise total price of music tracks



Aggregate

Apply

#	album_id	.00 SUM_price
1		9.9
2		0.99
3		2.969999999999998
4		7.92
5		14.85
6		12.87
7		11.87999999999999
8		13.86
9		7.92
11		11.87999999999999
12		11.87999999999999
13		7.92
15		4.95
18		16.83
19		10.89
20		10.89
21		17.82
23		33.66
24		22.22

Group Bys +

Field: album_id

Measures +

Include row count (i.e. COUNT)

Field: price

Aggregate Function: Sum

Datameer

No.of songs composed by each composer album wise

The screenshot displays the Datameer platform interface, illustrating a data pipeline and its configuration.

Data Pipeline: A flow diagram at the top shows a "TRACKS" source connected to an "AGGREGATION 3" step, which contains a single aggregation node.

Preview Table: Below the pipeline, a preview table shows the raw data from the "TRACKS" source. The columns are "composer" and "# album_id". The data includes entries for various artists like Angus Young, F. Baltes, and AC/DC across different album IDs.

Aggregation Configuration: A modal window titled "Aggregate" details the grouping and measurement for the aggregation step. It shows "Group Bys" set to "composer" and "album_id", and a "Measures" section where "album_id" is aggregated using the "Count" function.

Row	composer	# album_id	# COUNT
1	Angus Young, Malcolm Young, Brian Johnson	1	10
2	F. Baltes, S. Kaufman, U. Dirksneider & W. Hoffman	3	1
3	F. Baltes, R.A. Smith-Diesel, S. Kaufman	3	1
4	Deaffy & R.A. Smith-Diesel	3	1
5	AC/DC	4	8
6	Steven Tyler, Joe Perry, Jack Blades, Tommy Shaw	5	1
7	Steven Tyler, Joe Perry	5	1
8	Steven Tyler, Joe Perry, Jim Vallance, Holly Knight	5	1
9	Steven Tyler, Joe Perry, Desmond Child	5	3
10	Steven Tyler, Tom Hamilton	5	1
11	Steven Tyler, Joe Perry, Taylor Rhodes	5	2
12	Steven Tyler, Richie Supa	5	1
13	Steven Tyler, Jim Vallance	5	2
14	Steven Tyler, Joe Perry, Jim Vallance	5	1
15	Steven Tyler, Desmond Child	5	1
16	Alanis Morissette & Glenn Ballard	6	13
17	Jerry Cantrell, Layne Staley	7	4
18	Jerry Cantrell, Michael Starr, Layne Staley	7	1
19	Annotations	0	0

Datameer

No.of songs genre and price wise - using pivot table

The screenshot shows the Datameer Data Pipeline interface. A data flow is visible on the left, starting with a 'TRACKS' source and leading to a 'PIVOT' step. The 'PIVOT' step has three output columns: '.00 price', '# 1_ANY_genre_id', and '# 2_ANY_genre_id'. The preview pane below shows two rows of data:

	.00 price	# 1_ANY_genre_id	# 2_ANY_genre_id
1	0.99	1	1
2	1.99		

The right side of the interface shows the 'Transform' tab of the 'PIVOT' step configuration. It includes a 'Source' section set to 'TRACKS', a 'Recipe' section for chaining transformations, and detailed configuration panels for 'Rows', 'Columns', and 'Measures'. The 'Rows' panel shows the field 'price' selected. The 'Columns' panel shows 'media_type_id' selected with values '1, 2, 3, 4, 5 (5 values)'. The 'Measures' panel shows the field 'genre_id'.

Transform Step Configuration (PIVOT)

Source: TRACKS

Recipe: Chain data transformations together in a single view by adding steps to your recipe below.

Rows:

Field	Value
price	0.99
price	1.99

Columns:

Field	Values
media_type_id	1, 2, 3, 4, 5 (5 values)

Measures:

Field	Value
genre_id	

Datameer

Join tables track and album

The screenshot illustrates the Datameer platform interface for joining two datasets: TRACKS and ALBUMS.

Left Panel: Shows a visual workflow diagram where the TRACKS dataset is connected to the ALBUMS dataset via a JOIN node. The preview table below shows the joined data with columns: #, track_id, ABC name, #, album_id, and #, media_type.

#	track_id	ABC name	#	album_id	#	media_type
1	1	For Those About To Rock (We Salute You)	1			1
2	2	Balls to the Wall	2			2
3	3	Fast As a Shark	3			2
4	4	Restless and Wild	3			2
5	5	Princess of the Dawn	3			2

Middle Panel: The Transform tab of the configuration panel is active, showing the Source as TRACKS. The Recipe section allows chaining data transformations.

Right Panel: The Join Configuration panel is open, showing the SOURCES step with TRACKS and ALBUMS selected. It also shows the Choose Join Mode options: Inner join (selected), Outer left, Outer right, and Full outer join.

Bottom Panel: The COLUMNS step is visible, showing the selection of columns from both datasets.

Datameer

No.of artists who sang in each album - using both track and album tables

The screenshot illustrates a data pipeline setup in the Datameer platform. The process starts with two input tables: 'TRACKS' and 'ALBUMS'. These are joined together, and the resulting data is then aggregated. The final output is a preview table showing the count of artists per album.

Workflow Components:

- Source:** TRACKS, ALBUMS
- Transform:** JOIN, AGGREGATION
- Preview:** ABC title, # COUNT

Preview Data:

	ABC title	# COUNT
1	For Those About To Rock We Salute You	10
2	Balls to the Wall	1
3	Restless and Wild	3

Aggregate Step:

Chain data transformations together in a single view by adding steps to your recipe below.

+ Add to Recipe

Aggregate Data:

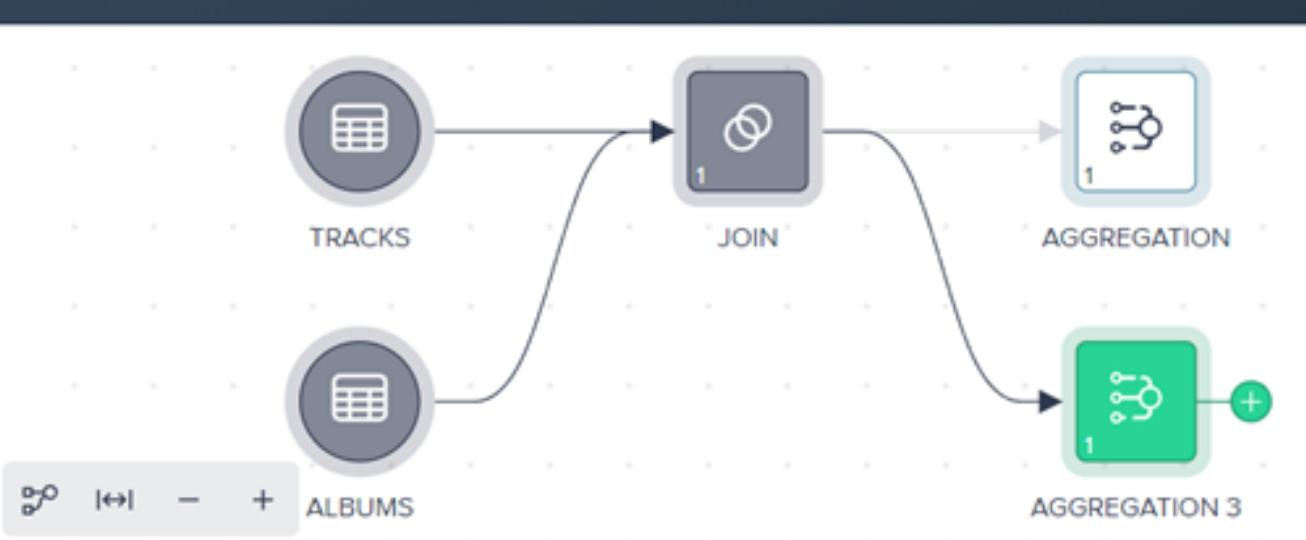
	ABC title	# COUNT
1	For Those About To Rock We Salute You	10
2	Balls to the Wall	1
3	Restless and Wild	3
4	Let There Be Rock	8
5	Big Ones	15
6	Jagged Little Pill	13
7	Warner 25 Anos	14
8	Plays Metallica By Four Cellos	8
9	Audioslave	14
10	BackBeat Soundtrack	12
11	The Best Of Billy Cobham	8
12	Alcohol Fueled Brutality Live! [Disc 1]	13

Group Bys: title

Measures: artist_id, Count

Datameer

No.of albums composed by each composer along with total price of all the songs and total no.of genres composed



The screenshot shows the Datameer Data Pipeline interface. At the top left, a flow diagram illustrates the data transformation process: 'TRACKS' and 'ALBUMS' tables are joined, then aggregated, and finally aggregated again ('AGGREGATION 3'). On the right, a detailed view of the 'Aggregate' step is shown, featuring tabs for 'Transform' (selected), 'Source', 'Exchange', and 'Info'. The 'Transform' tab displays the query: 'SELECT # COUNT AS COUNT FROM (SELECT album_id, COUNT(*) AS COUNT FROM tracks GROUP BY album_id) AS T1 GROUP BY album_id'. Below this is a preview of the data, showing columns: 'ABC composer', '# COUNT', '.00 SUM_price', and '# COUNT_2'. The preview table contains 26 rows of data, such as 'Angus Young, Malcolm Young, Brian Johnson' with a count of 10 and a total price of 9.9. To the right of the preview are sections for 'Group By's, 'Measures', and 'Fields', which define the aggregation logic.

	ABC composer	# COUNT	.00 SUM_price	# COUNT_2
1	Angus Young, Malcolm Young, Brian Johnson	10	9.9	10
2		978	1181.22	978
3	F. Baltes, S. Kaufman, U. Dirksneider &	1	0.99	1
4	Deaffy & R.A. Smith-Diesel	1	0.99	1
5	AC/DC	8	7.92	8
6	Steven Tyler, Joe Perry, Jack Blades, Tommy Shaw	1	0.99	1
7	Steven Tyler, Joe Perry	1	0.99	1
8	Steven Tyler, Joe Perry, Desmond Child	3	2.9699999999999998	3
9	Steven Tyler, Richie Supa	1	0.99	1
10	Steven Tyler, Jim Vallance	2	1.98	2
11	Steven Tyler, Joe Perry, Jim Vallance	1	0.99	1
12	Steven Tyler, Desmond Child	1	0.99	1
13	Alanis Morissette & Glenn Ballard	13	12.87	13
14	Jerry Cantrell	6	5.939999999999995	6
15	Jerry Cantrell, Layne Staley	4	3.96	4
16	Apocalyptica	8	7.92	8
17	Audioslave/Chris Cornell	14	13.86	14
18	Cornell, Commerford, Morello, Wilk	12	11.879999999999999	12
19	Larry Williams	2	1.98	2
20	Brian Holland/Freddie Gorman/Georgia Dobbins/Rob	1	0.99	1
21	Bo Diddley	1	0.99	1
22	Ned Fairchild	1	0.99	1
23	George Duke	1	0.99	1
24	Tony Iommi, Bill Ward, Geezer Butler, Ozzy Osbourne	11	10.89	11
25	Eddie Casillas/Roy Z	1	0.99	1
26	Tradicional	1	0.99	1

Datameer

Join 3 tables - track, album and artist

The screenshot illustrates the Datameer Data Integration interface, showing a workflow for joining three tables: ARTIST, TRACKS, and ALBUMS.

Workflow Diagram:

```
graph LR; ARTIST((ARTIST)) --> JOIN1((JOIN)); TRACKS((TRACKS)) --> JOIN1; JOIN1 --> JOIN2((JOIN 2)); JOIN1 --> AGGREGATION1[AGGREGATION]; ALBUMS((ALBUMS)) --> JOIN1; JOIN2 --> AGGREGATION2[AGGREGATION 3];
```

Pipeline Components:

- ARTIST**: Represented by a circular icon with a grid symbol.
- TRACKS**: Represented by a circular icon with a grid symbol.
- ALBUMS**: Represented by a circular icon with a grid symbol.
- JOIN 1**: A square icon with a circular join symbol.
- JOIN 2**: A green square icon with a circular join symbol.
- AGGREGATION**: A blue square icon with a circular join symbol.
- AGGREGATION 3**: A blue square icon with a circular join symbol.

Preview View:

Shows a preview of the joined data with 21 columns. The first few rows of the **ABC name** column are:

#	track_id	ABC name
1	1	For Those About To Rock (We Salute You)
2	2	Balls to the Wall
3	3	Fast As a Shark

Join Configuration:

The **Join Configuration** dialog is open, showing the configuration for the **JOIN 1** step.

SOURCES: Choose Sources: **JOIN** (radio button selected) **ARTIST** (dropdown menu).

Choose Join Mode: **Inner Join** (selected), **outer left**, **outer right**, **full outer join**.

COLUMNS: Choose Columns: **Select C...** (dropdown menu).

Preview: Shows the joined data with columns: #, track_id, ABC name, #, album_id, #, media_type.

#	track_id	ABC name	#	album_id	#	media_type
1	1	For Those About To Rock (We Salute You)	1		1	
2	2	Balls to the Wall	2		2	
3	3	Fast As a Shark	3		2	
4	4	Restless and Wild	3		2	
5	5	Princess of the Dawn	3		2	
6	6	Put The Finger On You	1		1	
7	7	Let's Get It Up	1		1	
8	8	Inject The Venom	1		1	
9	9	Snowballed	1		1	
10	10	Evil Walks	1		1	
11	11	C.O.D.	1		1	
12	12	Breaking The Rules	1		1	
13	13	Night Of The Long Knives	1		1	
14	14	Spellbound	1		1	
15	15	Go Down	4		1	
16	16	Dog Eat Dog	4		1	

Splunk

List of all petrol car models with high safety measures i.e NCAP Rating greater than or equal to 4

New Search

```
ncap_rating >=4 fuel="Petrol" |stats values(model)
```

✓ 122 events (before 10/14/22 6:50:10.000 PM) No Event Sampling ▾

Events Patterns Statistics (1) Visualization

20 Per Page ▾ ✎ Format Preview ▾

values(model) ▾

- Alcazar
- Amaze
- Aura
- C3
- City 4th Gen
- City 5th Gen
- Fortuner
- GT-R
- Glanza
- Jazz
- Kicks
- Kiger
- Magnite
- Scorpio-N

Splunk

List of all car makers who produce cars with high safety measure with an NCAP Rating of 5

New Search

```
ncap_rating=5 | stats values(maker)
```

✓ 72 events (before 10/14/22 6:47:23.000 PM) No Event Sampling ▾

Events Patterns Statistics (1) Visualization

20 Per Page ▾ ✎ Format Preview ▾

values(maker) ▾

- Honda
- Hyundai
- Kia
- Mahindra
- Nissan
- Toyota
- Volkswagen

Splunk

No.of different Sedan type car models produced by each maker with more than 4 seats

New Search Save As Create Table View Close

```
type="Sedan" | seats>=4 | stats count(model) by maker
```

All time Smart Mode

✓ 26 events (before 10/14/22 7:01:15.000 PM) No Event Sampling Job

Events Patterns Statistics (3) Visualization

20 Per Page Format Preview

maker	count(model)
Honda	10
Hyundai	10
Volkswagen	6

Splunk

Find list of types of Petrol based SUV cars that have fuel tank capacity >= 50 and with 5 seats

New Search

```
fuel="Petrol" type="SUV" fuel_tank_capacity>=50 seats=5 |stats values(maker)
```

✓ 30 events (before 10/14/22 7:43:30.000 PM) No Event Sampling ▾

Events Patterns Statistics (1) Visualization

20 Per Page ▾ ✓ Format Preview ▾

values(maker) ▾

Hyundai
Kia
Mahindra
Nissan
Volkswagen

Splunk

Type wise no.of cars produced by each maker

New Search Save As Create Table View Close

```
source="cars_india_dataset.csv" | stats count(model) by maker,type
```

All time Smart Mode

✓ 468 events (before 10/14/22 7:11:22.000 PM) No Event Sampling Job II ■ + ◐ ↓

Events Patterns Statistics (31) Visualization

20 Per Page Format Preview < Prev 1 2 Next >

maker	type	count(model)
Citroen	Hatchback	4
Citroen	SUV	2
Honda	Compact SUV	4
Honda	Compact Sedan	8
Honda	Hatchback	4
Honda	Sedan	10
Hyundai	Compact SUV	10
Hyundai	Compact Sedan	8
Hyundai	Hatchback	22
Hyundai	SUV	34
Hyundai	Sedan	10
Kia	Compact SUV	10

Splunk

List of all Diesel cars that have a fuel efficiency greater than the average fuel efficiency

New Search Save As ▾ Create Table View Close

```
fuel="Diesel" | eventstats mean(fuel_efficiency) as avg | where fuel_efficiency>=avg | table model
```

All time ▾

✓ 48 events (before 10/14/22 7:36:32.000 PM) No Event Sampling ▾ Job ▾ Smart Mode ▾

Events Patterns Statistics (48) Visualization

20 Per Page ▾ Preview ▾ < Prev 1 2 3 Next >

model ▾

model
Amaze
Sonet
C5 Aircross
WR-V
Amaze
XUV 300
City 5th Gen
i20
Sonet
Venue
Creta
Alivazor

Splunk

Model of the car with highest displacement

New Search Save As ▾ Create Table View Close

```
source="cars_india_dataset.csv" | eventstats max(displacement) as mx | where displacement=mx | table model,displacement
```

All time ▾ Q

✓ 2 events (before 10/14/22 7:58:26.000 PM) No Event Sampling ▾ Job ▾ II ■ ▶ ⏪ ⏴ ⏵ ⏴ Smart Mode ▾

Events Patterns Statistics (2) Visualization

20 Per Page ▾ Format Preview ▾

model	displacement
GT-R	3799
GT-R	3799

Splunk

Distinct car models that has 7 seats and uses diesel consumption

New Search

```
fuel="Diesel" seats>=7 |stats values(model)
```

✓ 48 events (before 10/14/22 8:03:08.000 PM) No Event Sampling ▾

Events Patterns Statistics (1) Visualization

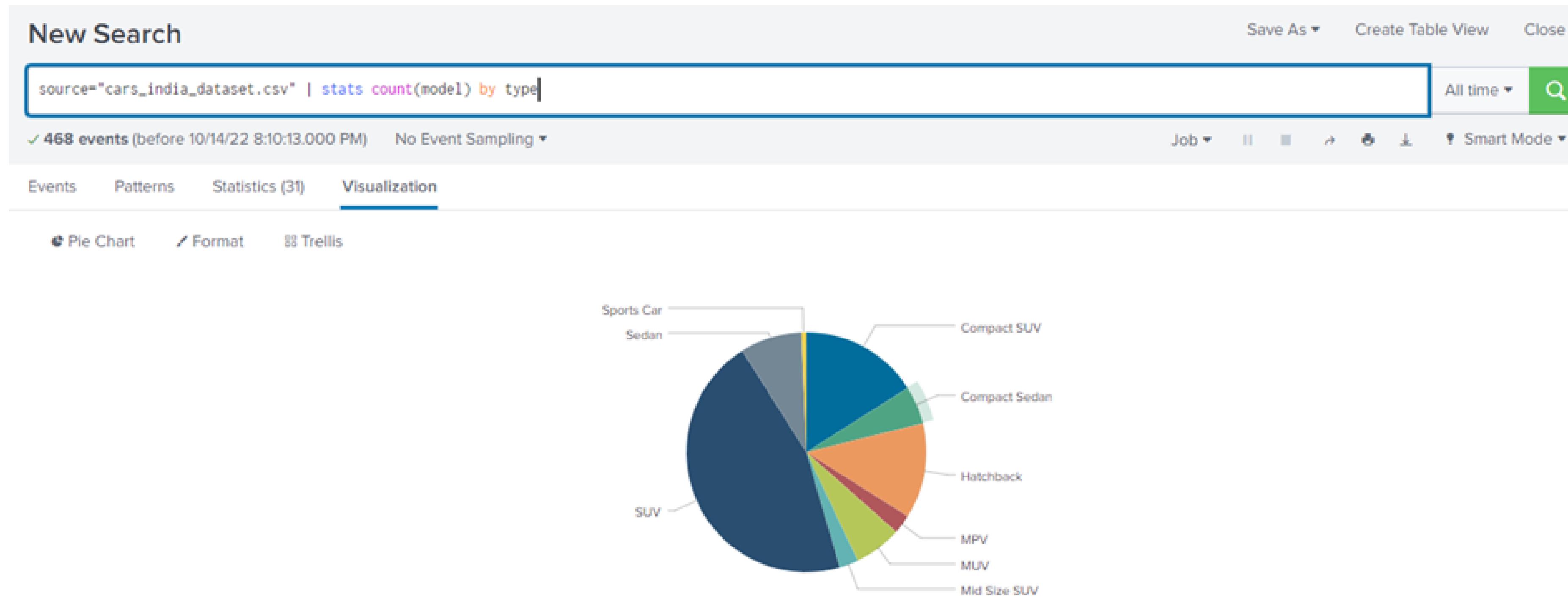
20 Per Page ▾ ✓ Format Preview ▾

values(model) ▾

- Alcazar
- Carens
- Carnival
- Fortuner
- Fortuner Legender
- Marazzo
- Safari
- Scorpio Classic
- Scorpio-N
- XUV 700

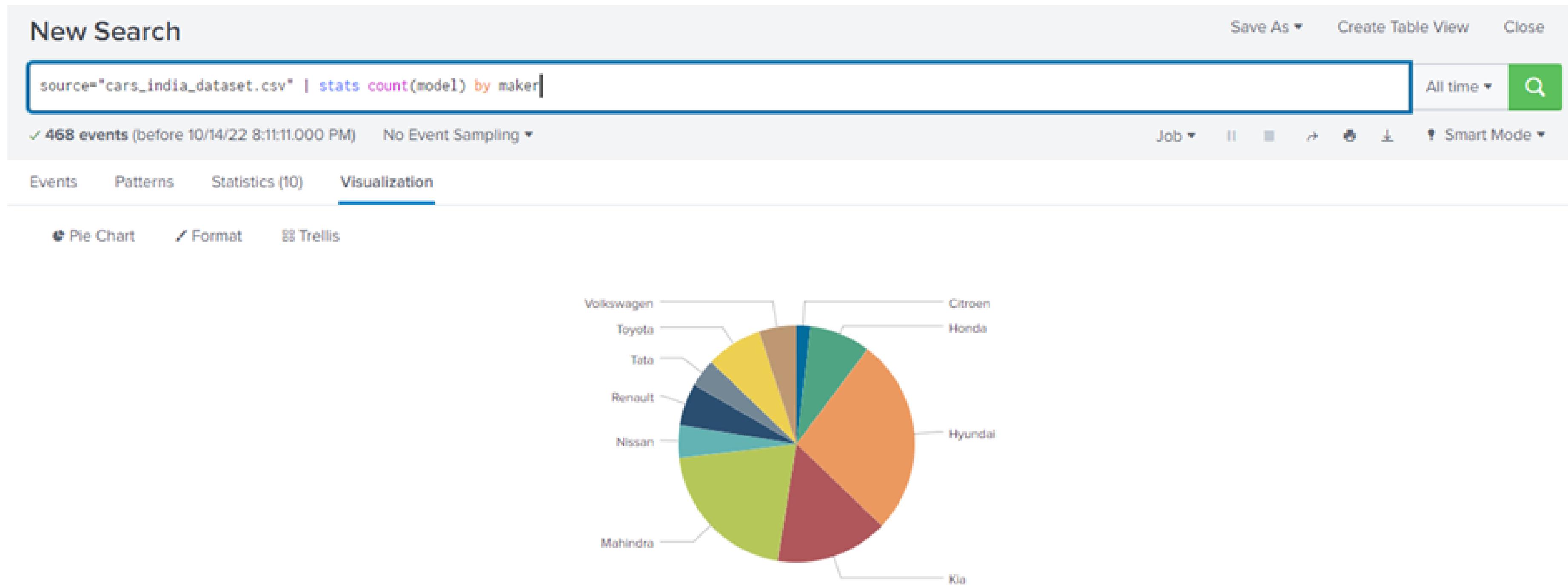
Splunk

Visualizing types of car models via a pie chart



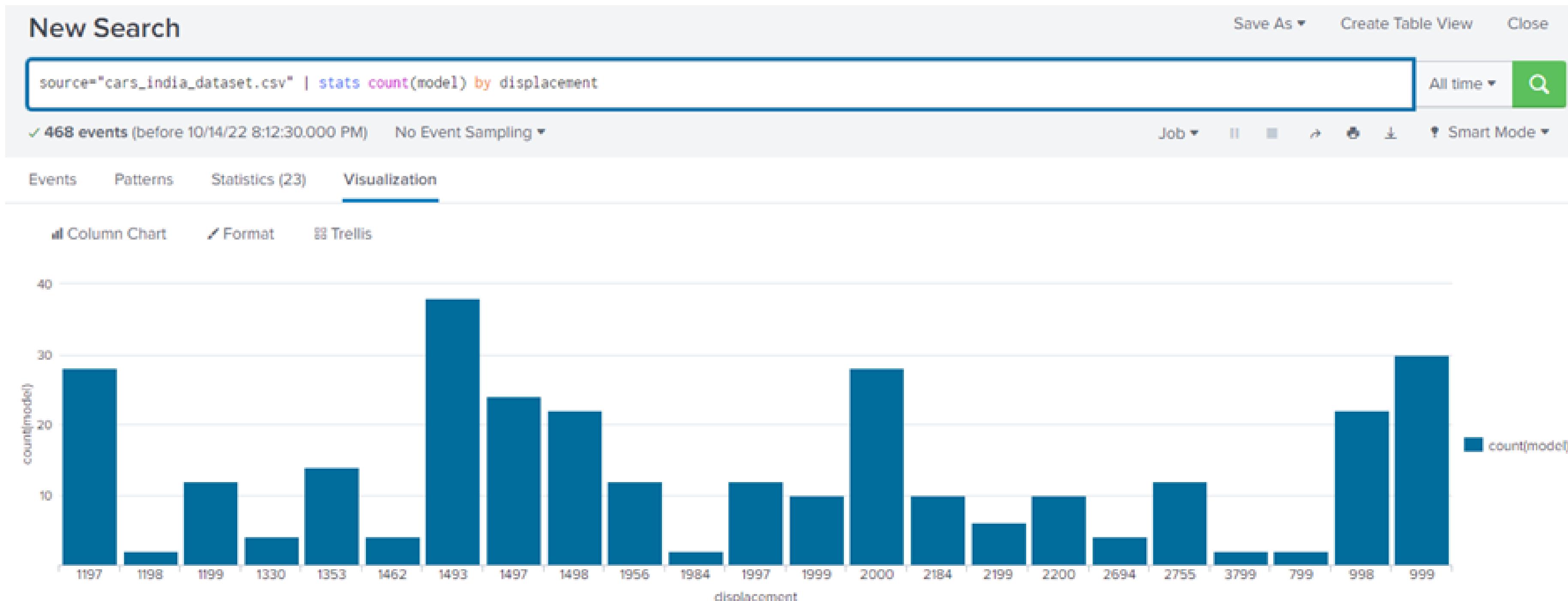
Splunk

Visualizing different car makers via a pie chart



Splunk

Visualizing no.of cars under each displacement via a bar chart



MongoDB

Updating the categorical values such as “Not Rated”, “Not Tested” as 0 in NCAP Rating column

```
> db.car.updateMany({"NCAP Rating": {$in:["Not Tested","Not Rated"]}},{$set:{ "NCAP Rating":0}})  
{ "acknowledged" : true, "matchedCount" : 33, "modifiedCount" : 33 }
```

MongoDB

Display all car models whose transmission is 6AT or 6MT

```
> db.car.distinct("Model", {"Transmission":{$in:["6 AT","6 MT"]}})  
[  
    "Alcazar",  
    "C3",  
    "Carens",  
    "City 5th Gen",  
    "Creta",  
    "Fortuner",  
    "Fortuner Legender",  
    "Harrier",  
    "Kicks",  
    "Marazzo",  
    "Safari",  
    "Scorpio Classic",  
    "Scorpio-N",  
    "Seltos",  
    "Sonet",  
    "Thar AX OPT",  
    "Thar LX",  
    "Tucson",  
    "Venue",  
    "Verna",  
    "WR-V",  
    "XUV 300",  
    "XUV 700",  
    "i20"  
]
```

MongoDB

Find all car makers who produce cars with NCAP Rating between 3 and 4. And fuel efficiency is less than 10

```
> db.car.find({"Fuel Efficiency":{$lte:10}, "NCAP Rating":{$lte:5,$gte:3}}).pretty()
{
    "_id" : ObjectId("634ae70ae6b4582cfee10143"),
    "Model" : "Thar LX",
    "Maker" : "Mahindra",
    "Type" : "SUV",
    "Seats" : 4,
    "Displacement" : 2184,
    "Length" : 3985,
    "Width" : 1855,
    "Height" : 1844,
    "Wheelbase" : 2450,
    "No_of_Cylinders" : 4,
    "Fuel" : "Diesel",
    "Engine Type" : "mHawk 130",
    "Transmission" : "6 TC Automatic",
    "Front Brake" : "Disc",
    "Rear Brake" : "Drum",
    "Drive" : "4WD",
    "Turning Radius" : 5.25,
    "Fuel Tank Capacity" : 57,
    "Boot Space" : "",
    "Fuel Efficiency" : 9,
    "Emission Type" : "BS VI",
    "Tyre Size" : "255/65 R18",
    "Variants" : 1,
    "NCAP Rating" : 4
}
```

MongoDB

No.of different types of Petrol cars manufactured by Mahindra car maker

```
> db.car.aggregate([{$match:{ "Fuel":{$ne:"Diesel"}, "Maker":"Mahindra"}}, {$count:"carTypes"}])  
{ "carTypes" : 13 }
```

MongoDB

Average fuel efficiency Model wise

```
> db.car.aggregate([{$group:{_id:"$Model",avgEfficiency:{$avg:"$Fuel Efficiency"}}}])  
{ "_id" : "Kona Electric", "avgEfficiency" : null }  
{ "_id" : "Aura", "avgEfficiency" : 22.5 }  
{ "_id" : "Amaze", "avgEfficiency" : 20.65 }  
{ "_id" : "Triber", "avgEfficiency" : 18.6 }  
{ "_id" : "Creta", "avgEfficiency" : 18.018 }  
{ "_id" : "i20 N-Line", "avgEfficiency" : 20.25 }  
{ "_id" : "Thar LX", "avgEfficiency" : 13.649999999999999 }  
{ "_id" : "C5 Aircross", "avgEfficiency" : 18.6 }  
{ "_id" : "Fortuner Legender", "avgEfficiency" : 14.3 }  
{ "_id" : "Scorpio-N", "avgEfficiency" : null }  
{ "_id" : "Scorpio Classic", "avgEfficiency" : null }  
{ "_id" : "Jazz", "avgEfficiency" : 16.85 }  
{ "_id" : "Kicks", "avgEfficiency" : 15.299999999999999 }  
{ "_id" : "Marazzo", "avgEfficiency" : 17 }  
{ "_id" : "Venue N-Line", "avgEfficiency" : 18.1 }  
{ "_id" : "Urban Cruiser", "avgEfficiency" : 17.895000000000003 }  
{ "_id" : "GT-R", "avgEfficiency" : 10.16 }  
{ "_id" : "Kwid", "avgEfficiency" : 21.466666666666667 }  
{ "_id" : "Verna", "avgEfficiency" : 20.32 }  
{ "_id" : "i20", "avgEfficiency" : 21.14 }  
Type "it" for more
```

MongoDB

Model and type of car wise avg mileage - group by model and type of car

```
> db.car.aggregate([{$group:{_id:{'model':'$Model','type':'$Type'},avgMileage:{$avg:'$Displacement'}}}])  
{ "_id" : { "model" : "XUV 300", "type" : "Compact SUV" }, "avgMileage" : 1347 }  
{ "_id" : { "model" : "City 5th Gen", "type" : "Sedan" }, "avgMileage" : 1498 }  
{ "_id" : { "model" : "Urban Cruiser", "type" : "Compact SUV" }, "avgMileage" : 1462 }  
{ "_id" : { "model" : "Amaze", "type" : "Compact Sedan" }, "avgMileage" : 1348.5 }  
{ "_id" : { "model" : "XUV 700", "type" : "SUV" }, "avgMileage" : 2111.111111111113 }  
{ "_id" : { "model" : "Tiguan", "type" : "SUV" }, "avgMileage" : 1984 }  
{ "_id" : { "model" : "Safari", "type" : "SUV" }, "avgMileage" : 1956 }  
{ "_id" : { "model" : "Tucson", "type" : "SUV" }, "avgMileage" : 1997.6666666666667 }  
{ "_id" : { "model" : "Kiger", "type" : "Compact SUV" }, "avgMileage" : 999 }  
{ "_id" : { "model" : "Kwid", "type" : "Hatchback" }, "avgMileage" : 932.333333333334 }  
{ "_id" : { "model" : "C5 Aircross", "type" : "SUV" }, "avgMileage" : 1997 }  
{ "_id" : { "model" : "Taigun", "type" : "Mid Size SUV" }, "avgMileage" : 1248.5 }  
{ "_id" : { "model" : "GT-R", "type" : "Sports Car" }, "avgMileage" : 3799 }  
{ "_id" : { "model" : "Kicks", "type" : "SUV" }, "avgMileage" : 1386 }  
{ "_id" : { "model" : "Scorpio-N", "type" : "SUV" }, "avgMileage" : 2000 }  
{ "_id" : { "model" : "Thar LX", "type" : "SUV" }, "avgMileage" : 2090.5 }  
{ "_id" : { "model" : "Thar AX OPT", "type" : "SUV" }, "avgMileage" : 2090.5 }  
{ "_id" : { "model" : "Marazzo", "type" : "MPV" }, "avgMileage" : 1497 }  
{ "_id" : { "model" : "Scorpio Classic", "type" : "SUV" }, "avgMileage" : 2184 }  
{ "_id" : { "model" : "Glanza", "type" : "Hatchback" }, "avgMileage" : 1197 }  
Type "it" for more
```

MongoDB

Max NCAP Rating according to car maker wise, among all those 5 seater Petrol cars

```
> db.car.aggregate([{$match:{ "Seats":5, "Fuel":"Petrol"}}, {$group:{_id:"$Maker",maxRating:{$max:"$NCAP Rating"} }}, {$sort:{maxRating:-1}}])  
{ "_id" : "Volkswagen", "maxRating" : 5 }  
{ "_id" : "Honda", "maxRating" : 5 }  
{ "_id" : "Nissan", "maxRating" : 5 }  
{ "_id" : "Mahindra", "maxRating" : 5 }  
{ "_id" : "Toyota", "maxRating" : 4 }  
{ "_id" : "Hyundai", "maxRating" : 4 }  
{ "_id" : "Renault", "maxRating" : 4 }  
{ "_id" : "Kia", "maxRating" : 3 }  
{ "_id" : "Citroen", "maxRating" : 0 }  
>
```

Thank You !