

# **[301] Database 3**

Tyler Caraza-Harter

# Learning Objectives Today

## GROUP BY

- how to break data into buckets
- combination of GROUP BY with ORDER BY
- WHERE vs. HAVING

## Aggregates

- SUM, COUNT, MAX, MIN, SUM
- Aliases with AS

# Outline

## Aggregation Queries

Grouping with Python

Grouping with SQL

Combining with LIMIT and ORDER BY

WHERE vs. HAVING

Practice

# Example: Movies Database

```
In [32]: c = sqlite3.connect('movies.db')
df = pd.read_sql("select * from movies", c)
c.close()
df
```

Out[32]:

	Title	Director	Year	Runtime	Rating	Revenue
0	Guardians of the Galaxy	James Gunn	2014	121	8.1	333.13
1	Prometheus	Ridley Scott	2012	124	7.0	126.46
2	Split	M. Night Shyamalan	2016	117	7.3	138.12
3	Sing	Christophe Lourdelet	2016	108	7.2	270.32
4	Suicide Squad	David Ayer	2016	123	6.2	325.02
5	The Great Wall	Yimou Zhang	2016	103	6.1	45.13
6	La La Land	Damien Chazelle	2016	128	8.3	151.06

Title	Director	Year	Runtime	Rating	Revenue
Guardians of the Galaxy	James Gunn	2014	121	8.1	333.13
Prometheus	Ridley Scott	2012	124	7.0	126.46
Split	M. Night Shyamalan	2016	117	7.3	138.12
Sing	Christophe Lourdelet	2016	108	7.2	270.32
Suicide Squad	David Ayer	2016	123	6.2	325.02
The Great Wall	Yimou Zhang	2016	103	6.1	45.13
La La Land	Damien Chazelle	2016	128	8.3	151.06

**Question:** which **movie** has the **highest rating**?

**SQL Query:**

Title	Director	Year	Runtime	Rating	Revenue
Guardians of the Galaxy	James Gunn	2014	121	8.1	333.13
Prometheus	Ridley Scott	2012	124	7.0	126.46
Split	M. Night Shyamalan	2016	117	7.3	138.12
Sing	Christophe Lourdelet	2016	108	7.2	270.32
Suicide Squad	David Ayer	2016	123	6.2	325.02
The Great Wall	Yimou Zhang	2016	103	6.1	45.13
La La Land	Damien Chazelle	2016	128	8.3	151.06

**Question:** which **movie** has the **highest rating**?

**SQL Query:** `SELECT Title FROM Movies  
ORDER BY Rating DESC  
LIMIT 1`

Title	Director	Year	Runtime	Rating	Revenue
Guardians of the Galaxy	James Gunn	2014	121	8.1	333.13
Prometheus	Ridley Scott	2012	124	7.0	126.46
Split	M. Night Shyamalan	2016	117	7.3	138.12
Sing	Christophe Lourdelet	2016	108	7.2	270.32
Suicide Squad	David Ayer	2016	123	6.2	325.02
The Great Wall	Yimou Zhang	2016	103	6.1	45.13
La La Land	Damien Chazelle	2016	128	8.3	151.06

**Question:** which **director** made the **shortest movie**?

**SQL Query:**

Title	Director	Year	Runtime	Rating	Revenue
Guardians of the Galaxy	James Gunn	2014	121	8.1	333.13
Prometheus	Ridley Scott	2012	124	7.0	126.46
Split	M. Night Shyamalan	2016	117	7.3	138.12
Sing	Christophe Lourdelet	2016	108	7.2	270.32
Suicide Squad	David Ayer	2016	123	6.2	325.02
The Great Wall	Yimou Zhang	2016	103	6.1	45.13
La La Land	Damien Chazelle	2016	128	8.3	151.06

**Question:** which **director** made the **shortest movie**?

**SQL Query:** `SELECT Director FROM Movies  
ORDER BY Runtime ASC  
LIMIT 1`



Title	Director	Year	Runtime	Rating	Revenue
Guardians of the Galaxy	James Gunn	2014	121	8.1	333.13
Prometheus	Ridley Scott	2012	124	7.0	126.46
Split	M. Night Shyamalan	2016	117	7.3	138.12
Sing	Christophe Lourdelet	2016	108	7.2	270.32
Suicide Squad	David Ayer	2016	123	6.2	325.02
The Great Wall	Yimou Zhang	2016	103	6.1	45.13
La La Land	Damien Chazelle	2016	128	8.3	151.06

**Question:** which **director** made the **highest-revenue movie**?

**SQL Query:**

Title	Director	Year	Runtime	Rating	Revenue
Guardians of the Galaxy	James Gunn	2014	121	8.1	333.13
Prometheus	Ridley Scott	2012	124	7.0	126.46
Split	M. Night Shyamalan	2016	117	7.3	138.12
Sing	Christophe Lourdelet	2016	108	7.2	270.32
Suicide Squad	David Ayer	2016	123	6.2	325.02
The Great Wall	Yimou Zhang	2016	103	6.1	45.13
La La Land	Damien Chazelle	2016	128	8.3	151.06

**Question:** which **director** made the **highest-revenue movie**?

**SQL Query:** `SELECT Director FROM Movies  
ORDER BY Revenue DESC  
LIMIT 1`

Title	Director	Year	Runtime	Rating	Revenue
Guardians of the Galaxy	James Gunn	2014	121	8.1	333.13
Prometheus	Ridley Scott	2012	124	7.0	126.46
Split	M. Night Shyamalan	2016	117	7.3	138.12
Sing	Christophe Lourdelet	2016	108	7.2	270.32
Suicide Squad	David Ayer	2016	123	6.2	325.02
The Great Wall	Yimou Zhang	2016	103	6.1	45.13
La La Land	Damien Chazelle	2016	128	8.3	151.06

**Question:** which **movie** had the **highest revenue** in **2016**?

**SQL Query:**

Title	Director	Year	Runtime	Rating	Revenue
Guardians of the Galaxy	James Gunn	2014	121	8.1	333.13
Prometheus	Ridley Scott	2012	124	7.0	126.46
Split	M. Night Shyamalan	2016	117	7.3	138.12
Sing	Christophe Lourdelet	2016	108	7.2	270.32
Suicide Squad	David Ayer	2016	123	6.2	325.02
The Great Wall	Yimou Zhang	2016	103	6.1	45.13
La La Land	Damien Chazelle	2016	128	8.3	151.06

**Question:** which **movie** had the **highest revenue** in **2016**?

**SQL Query:**  
SELECT **Title** FROM Movies  
WHERE **Year** = 2016  
ORDER BY **Revenue** DESC  
**LIMIT 1**

Title	Director	Year	Runtime	Rating	Revenue
Guardians of the Galaxy	James Gunn	2014	121	8.1	333.13
Prometheus	Ridley Scott	2012	124	7.0	126.46
Split	M. Night Shyamalan	2016	117	7.3	138.12
Sing	Christophe Lourdelet	2016	108	7.2	270.32
Suicide Squad	David Ayer	2016	123	6.2	325.02
The Great Wall	Yimou Zhang	2016	103	6.1	45.13
La La Land	Damien Chazelle	2016	128	8.3	151.06

**Question:** which 3 movies had the highest revenues in 2016?

**SQL Query:**

Title	Director	Year	Runtime	Rating	Revenue
Guardians of the Galaxy	James Gunn	2014	121	8.1	333.13
Prometheus	Ridley Scott	2012	124	7.0	126.46
Split	M. Night Shyamalan	2016	117	7.3	138.12
Sing	Christophe Lourdelet	2016	108	7.2	270.32
Suicide Squad	David Ayer	2016	123	6.2	325.02
The Great Wall	Yimou Zhang	2016	103	6.1	45.13
La La Land	Damien Chazelle	2016	128	8.3	151.06

**Question:** which 3 movies had the highest revenues in 2016?

**SQL Query:**

```
SELECT Title FROM Movies
WHERE Year = 2016
ORDER BY Revenue DESC
LIMIT 3
```

# Data Questions

which **movie** has the **highest rating**?

which **director** made the **shortest movie**?

which **director** made the **highest-revenue movie**?

which **movie** had the **highest revenue** in **2016**?

which **3 movies** had the **highest revenues** in **2016**?

**These questions all have something in common:**

# Data Questions

which **movie** has the **highest rating**?

which **director** made the **shortest movie**?

which **director** made the **highest-revenue movie**?

which **movie** had the **highest revenue** in **2016**?

which **3 movies** had the **highest revenues** in **2016**?

**These questions all have something in common:**

identify certain rows, then **just extract**  
specific columns from those rows to answer



# Data Questions

which **movie** has the **highest rating**?

which **director** made the **shortest movie**?

which **director** made the **highest-revenue movie**?


which **movie** had the **highest revenue** in **2016**?

which **3 movies** had the **highest revenues** in **2016**?

**These questions all have something in common:**

identify certain rows, then **just extract**  
specific columns from those rows to answer

**Sometimes we want a **summary** over multiple rows**

 called an **“aggregate”**

## Extract data:

which **movie** has the **highest rating**?

which **director** made the **shortest movie**?

which **director** made the **highest-revenue movie**?

which **movie** had the **highest revenue** in **2016**?

which **3 movies** had the **highest revenues** in **2016**?

## Summarize data:

what is the **average** rating across **all movies**?

what is the **average** runtime for a **James Gunn** movie?

what is the **average** revenue for a **Ridley Scott** movie?

**how many** movies were there in **2016**?

what was the **total** revenue of all movies in **2016**?

## Extract data:

which **movie** has the **highest rating**?

which **director** made the **shortest movie**?

which **director** made the **highest-revenue movie**?

which **movie** had the **highest revenue** in **2016**?

which **3 movies** had the **highest revenues** in **2016**?

## Summarize data:

what is the **average** rating across **all movies**?

what is the **average** runtime for a **James Gunn** movie?

what is the **average** revenue for a **Ridley Scott** movie?

**how many** movies were there in **2016**?

what was the **total** revenue of all movies in **2016**?

today

Title	Director	Year	Runtime	Rating	Revenue
Guardians of the Galaxy	James Gunn	2014	121	8.1	333.13
Prometheus	Ridley Scott	2012	124	7.0	126.46
Split	M. Night Shyamalan	2016	117	7.3	138.12
Sing	Christophe Lourdelet	2016	108	7.2	270.32
Suicide Squad	David Ayer	2016	123	6.2	325.02
The Great Wall	Yimou Zhang	2016	103	6.1	45.13
La La Land	Damien Chazelle	2016	128	8.3	151.06

**Question:** what is the **total revenue** of **all the movies**?

**SQL Query:** `SELECT * FROM Movies`

	Revenue
0	333.13
1	126.46
2	138.12
3	270.32
4	325.02
5	45.13
6	151.06
7	0.00
8	8.01

**Question:** what is the **total revenue** of **all the movies**?

**SQL Query:** `SELECT Revenue FROM Movies`

SUM(Revenue)	
0	72215.45

**Question:** what is the **total revenue** of **all the movies**?

**SQL Query:** `SELECT SUM(Revenue) FROM Movies`

SUM(Revenue)	
0	72215.45

**Question:** what is the **total revenue** of **all the movies**?


**SQL Query:** `SELECT SUM(Revenue) FROM Movies`

 SUM is an aggregate function

SUM(Revenue)	
0	72215.45

**Question:** what is the **total revenue** of **all the movies**?

**SQL Query:** `SELECT SUM(Revenue) FROM Movies`

 SUM is an aggregate function


**aggregates functions:** SUM, AVG, COUNT, MIN, MAX



	SUM(Revenue)	COUNT()
0	72215.45	998

**Question:** what is the **total revenue** of **all the movies**?  
and how **many movies** are there?

**SQL Query:** `SELECT SUM(Revenue), COUNT() FROM Movies`

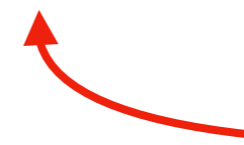
 COUNT doesn't need  
an argument

**aggregates functions:** SUM, AVG, COUNT, MIN, MAX

SUM(Revenue) / COUNT()	
0	72.36017

**Question:** what is the **average revenue** of **all the movies**?

**SQL Query:** `SELECT SUM(Revenue) / COUNT() FROM Movies`

 you can combine aggregates

**aggregates functions:** SUM, AVG, COUNT, MIN, MAX

AVG(Revenue)	
0	72.36017

**Question:** what is the **average revenue** of **all the movies**?

**SQL Query:** `SELECT AVG(Revenue) FROM Movies`  
 SUM divided by COUNT

**aggregates functions:** SUM, AVG, COUNT, MIN, MAX

	<b>AVG(Revenue)</b>	<b>AVG(Runtime)</b>
<b>0</b>	72.36017	113.170341

**Question:** what is the **average revenue** of **all the movies**?  
what is the **average runtime** of **all the movies**?

**SQL Query:** `SELECT AVG(Revenue), AVG(Runtime) FROM Movies`

**aggregates functions:** SUM, AVG, COUNT, MIN, MAX

**Question:** what **percentage** of the **total revenue** came from the **highest-revenue movie**?

**SQL Query:** ???

**aggregates functions:** SUM, AVG, COUNT, MIN, MAX

**MAX(revenue) / SUM(revenue)**

0

0.01297

**Question:** what **percentage** of the **total revenue** came from the **highest-revenue movie**?

**SQL Query:** `SELECT MAX ( Revenue ) / SUM ( Revenue ) FROM Movies`

**aggregates functions:** SUM, AVG, COUNT, MIN, MAX

$\text{MAX}(\text{revenue}) * 100.0 / \text{SUM}(\text{revenue})$

0	1.296994
---	----------

**Question:** what **percentage** of the **total revenue** came from the **highest-revenue movie**?

**SQL Query:** `SELECT 100 * MAX(Revenue) / SUM(Revenue) FROM Movies`

**aggregates functions:** SUM, AVG, COUNT, MIN, MAX

clunky column name for Pandas DataFrame

	<code>MAX(revenue) * 100.0 / SUM(revenue)</code>
0	1.296994

**Question:** what **percentage** of the **total revenue** came from the **highest-revenue movie**?

**SQL Query:** `SELECT 100 * MAX(Revenue) / SUM(Revenue) FROM Movies`

**aggregates functions:** SUM, AVG, COUNT, MIN, MAX



	percent
0	1.296994

**Question:** what **percentage** of the **total revenue** came from the **highest-revenue movie**?

**SQL Query:** `SELECT 100 * MAX(Revenue) / SUM(Revenue) AS percent  
FROM Movies`

**aggregates functions:** SUM, AVG, COUNT, MIN, MAX

	percent
0	1.296994

**Question:** what **percentage** of the **total revenue** came from the **highest-revenue movie**?

**SQL Query:** `SELECT 100 * MAX(Revenue) / SUM(Revenue) AS percent`  
`FROM Movies`

you can use "AS" to  
call columns whatever  
you like...

**aggregates functions:** SUM, AVG, COUNT, MIN, MAX

	percent
0	1.296994

**Question:** what **percentage** of the **total revenue** came from the **highest-revenue movie**?

**SQL Query:** `SELECT 100 * MAX(Revenue) / SUM(Revenue) AS percent  
FROM Movies`

**aggregates functions:** SUM, AVG, COUNT, MIN, MAX

	percent
0	1.296994

what if we want to  
answer this question just  
for movies in 2016?

**Question:** what **percentage** of the **total revenue** came from the **highest-revenue movie**?

**SQL Query:** `SELECT 100 * MAX(Revenue) / SUM(Revenue) AS percent  
FROM Movies`

**aggregates functions:** SUM, AVG, COUNT, MIN, MAX

what if we want to  
answer this question just  
for movies in 2016?

**Question:** what **percentage** of the **total revenue in 2016** came from the **highest-revenue movie**?

**SQL Query:** `SELECT 100 * MAX(Revenue) / SUM(Revenue) AS percent  
FROM Movies`

**aggregates functions:** SUM, AVG, COUNT, MIN, MAX

**Question:** what **percentage** of the **total revenue in 2016** came from the **highest-revenue movie**?

**SQL Query:** `SELECT 100 * MAX(Revenue) / SUM(Revenue) AS percent  
FROM Movies  
WHERE year = 2016`

you can combine WHERE with aggregates  
(filtering is done before aggregation)

**aggregates functions:** SUM, AVG, COUNT, MIN, MAX

Year	percent
2014	333.13
2012	126.46
2016	138.12
2016	270.32
2016	325.02
2016	45.13
2016	151.06

in progress...

**Question:** what **percentage** of the **total revenue in 2016** came from the **highest-revenue movie**?

**SQL Query:** `SELECT 100 * MAX(Revenue) / SUM(Revenue) AS percent  
FROM Movies  
WHERE year = 2016`

you can combine WHERE with aggregates  
(filtering is done before aggregation)

**aggregates functions:** SUM, AVG, COUNT, MIN, MAX

Year	percent
2016	138.12
2016	270.32
2016	138.12
2016	270.32
2016	325.02
2016	45.13
2016	151.06

in progress...

**Question:** what **percentage** of the **total revenue in 2016** came from the **highest-revenue movie**?

**SQL Query:** `SELECT 100 * MAX(Revenue) / SUM(Revenue) AS percent  
FROM Movies  
WHERE year = 2016`

you can combine WHERE with aggregates  
(filtering is done before aggregation)

**aggregates functions:** SUM, AVG, COUNT, MIN, MAX



max(revenue)	sum(revenue)
532.17	11211.65

in progress...

**Question:** what **percentage** of the **total revenue in 2016** came from the **highest-revenue movie**?

**SQL Query:** `SELECT 100 * MAX(Revenue) / SUM(Revenue) AS percent  
FROM Movies  
WHERE year = 2016`

you can combine WHERE with aggregates  
(filtering is done before aggregation)

**aggregates functions:** SUM, AVG, COUNT, MIN, MAX

	percent
0	4.746581

**Question:** what **percentage** of the **total revenue in 2016** came from the **highest-revenue movie**?

**SQL Query:** `SELECT 100 * MAX(Revenue) / SUM(Revenue) AS percent  
FROM Movies  
WHERE year = 2016`

you can combine WHERE with aggregates  
(filtering is done before aggregation)

**aggregates functions:** SUM, AVG, COUNT, MIN, MAX

# Outline

Aggregation Queries

Grouping with Python

Grouping with SQL

Combining with LIMIT and ORDER BY

WHERE vs. HAVING

Practice

previously

**Extract data:**

- which **movie** has the **highest rating**?
- which **director** made the **shortest movie**?
- which **director** made the **highest-revenue movie**?
- which **movie** had the **highest revenue** in **2016**?
- which **3 movies** had the **highest revenues** in **2016**?

just now

**Summarize data:**

- what is the **average rating** across **all movies**?
- what is the **average runtime** for a **James Gunn** movie?
- what is the **average revenue** for a **Ridley Scott** movie?
- how many movies** were there in **2016**?
- what was the **total revenue** of all movies in **2016**?

now...

**Summarize data across groups:**

- what is the **average rating** for **each** director?
- what is the **average runtime** for **each** director?
- what is the **average revenue** for **each** year?
- how many movies** were there in **each** year?
- what was the **total revenue** for **each** year?

## Extract data:

previously

which **movie** has the **highest rating**?

which **director** made the **shortest movie**?

which **director** made the **highest-revenue movie**?

which **movie** had the **highest revenue** in **2016**?

which **3 movies** had the **highest revenues** in **2016**?

## Summarize data:

just now

what is the **average rating** across **all movies**?

what is the **average runtime** for a **James Gunn** movie?

what is the **average revenue** for a **Ridley Scott** movie?

**how many movies** were there in **2016**?

what was the **total revenue** of all movies in **2016**?

## Summarize data across groups:

now...

what is the **average rating** for **each** director?

what is the **average runtime** for **each** director?

what is the **average revenue** for **each** year?

**how many movies** were there in **each** year?

what was the **total revenue** for **each** year?

# Demo 1: Average Rating per Director

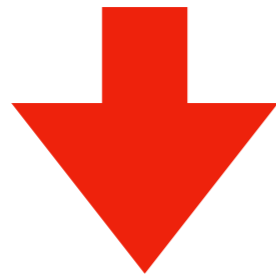
Database  
Table

many rows for each director

# Demo 1: Average Rating per Director

Database  
Table

many rows for each director



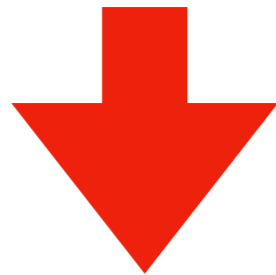
break data into groups (per director)

```
groups = {  
  "James Gunn": [ ROW ROW ROW ],  
  "Ridley Scott": [ ROW ROW ],  
  "M. Night Shyamalan": [ ROW ROW ROW ROW ],  
  ... more actors and lists of rows ...  
}
```

# Demo 1: Average Rating per Director

Database  
Table

many rows for each director



break data into groups (per director)

```
groups = {  
  "James Gunn": [ ROW ROW ROW ],  
  "Ridley Scott": [ ROW ROW ],  
  "M. Night Shyamalan": [ ROW ROW ROW ROW ],  
  ... more actors and lists of rows ...  
}
```



do stats on each group

```
James Gunn 7.13  
Ridley Scott 6.85  
M. Night Shyamalan 5.53  
...
```



# Demo 2: Average Y per X



many rows for each X



break data into groups (per X)

```
groups = {  
  "X": [ ROW ROW ROW ],  
  "X": [ ROW ROW ],  
  "X": [ ROW ROW ROW ROW ],  
  ... more X's and lists of rows ...  
}
```



do stats on each group

```
X1 Y1avg  
X2 Y2avg  
X3 Y3avg  
...
```

# Outline

Aggregation Queries

Grouping with Python

Grouping with SQL

Combining with LIMIT and ORDER BY

WHERE vs. HAVING

Practice

## Group By, by example...

	SUM(Revenue)
0	72215.45

**Question:** what is the **total revenue** of **all the movies**?

**SQL Query:** `SELECT SUM(Revenue) FROM Movies`

## Group By, by example...

**Question:** what is the **total revenue** of **per year**?

**SQL Query:** `SELECT SUM(Revenue) FROM Movies`



## Group By, by example...

**Question:** what is the **total revenue** of **per year**?

**SQL Query:** `SELECT SUM(Revenue) FROM Movies  
GROUP BY year`

## Group By, by example...

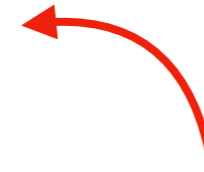
	SUM(Revenue)
0	3624.46
1	4306.23
2	5053.22
3	5292.26
4	5989.65
5	5431.96

**Question:** what is the **total revenue** of **per year**?

**SQL Query:** `SELECT SUM(Revenue) FROM Movies  
GROUP BY year`

## Group By, by example...

	SUM(Revenue)
0	3624.46
1	4306.23
2	5053.22
3	5292.26
4	5989.65
5	5431.96



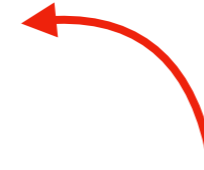
each row was the total revenue for one year.

**Question:** what is the **total revenue** of **per year**?

**SQL Query:** `SELECT SUM(Revenue) FROM Movies  
GROUP BY year`

## Group By, by example...

	SUM(Revenue)
0	3624.46
1	4306.23
2	5053.22
3	5292.26
4	5989.65
5	5431.96



each row was the total revenue for one year.  
*But which year?*

**Question:** what is the total revenue of per year?

**SQL Query:** `SELECT SUM(Revenue) FROM Movies  
GROUP BY year`



## Group By, by example...

	Year	SUM(Revenue)
0	2006	3624.46
1	2007	4306.23
2	2008	5053.22
3	2009	5292.26
4	2010	5989.65
5	2011	5431.96

**Question:** what is the **total revenue** of **per year**?

**SQL Query:** `SELECT year, SUM(Revenue) FROM Movies  
GROUP BY year`

## Group By, by example...

	Year	SUM(Revenue)
0	2006	3624.46
1	2007	4306.23
2	2008	5053.22
3	2009	5292.26
4	2010	5989.65
5	2011	5431.96

**Question:** what is the **total revenue** of **per year**?

it's very common that in the SELECT,  
the group-by variable is plain, and  
the others are aggregated

**SQL Query:** `SELECT year, SUM(Revenue) FROM Movies  
GROUP BY year`

## Group By, by example...

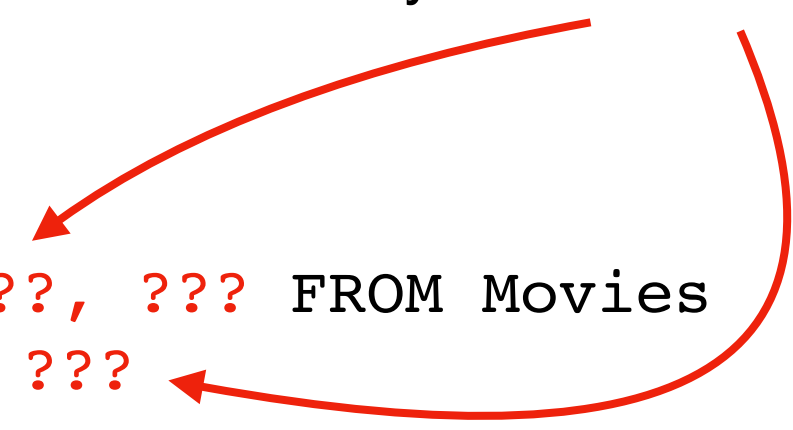
**Question:**      how many movies were by each director?

**SQL Query:**      SELECT ???, ??? FROM Movies  
GROUP BY ???

## Group By, by example...

**Question:** how many movies were by each director?

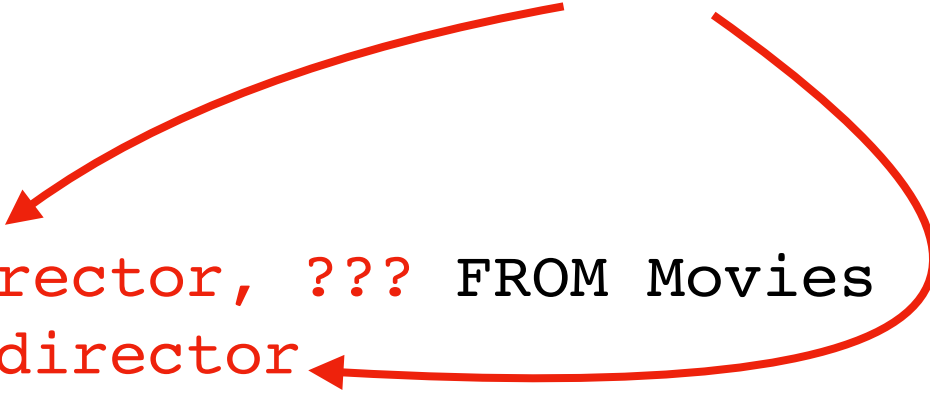
**SQL Query:** `SELECT ???, ??? FROM Movies  
GROUP BY ???`



## Group By, by example...

**Question:** how many movies were by each director?

**SQL Query:** `SELECT director, ??? FROM Movies  
GROUP BY director`



## Group By, by example...

**Question:** how many movies were by each director?

**SQL Query:** `SELECT director, ??? FROM Movies  
GROUP BY director`



## Group By, by example...

**Question:** how many movies were by each director?

**SQL Query:** `SELECT director, COUNT() FROM Movies  
GROUP BY director`



## Group By, by example...

	Director	COUNT()
0	Aamir Khan	1
1	Abdellatif Kechiche	1
2	Adam Leon	1
3	Adam McKay	4
4	Adam Shankman	2
5	Adam Wingard	2
6	Afonso Poyart	1

### Question:

how many movies were by each director?

### SQL Query:

```
SELECT director, COUNT() FROM Movies  
GROUP BY director
```





# Outline

Aggregation Queries

Grouping with Python

Grouping with SQL

Combining with LIMIT and ORDER BY

WHERE vs. HAVING

Practice

# Review from previous lecture

SELECT



FROM



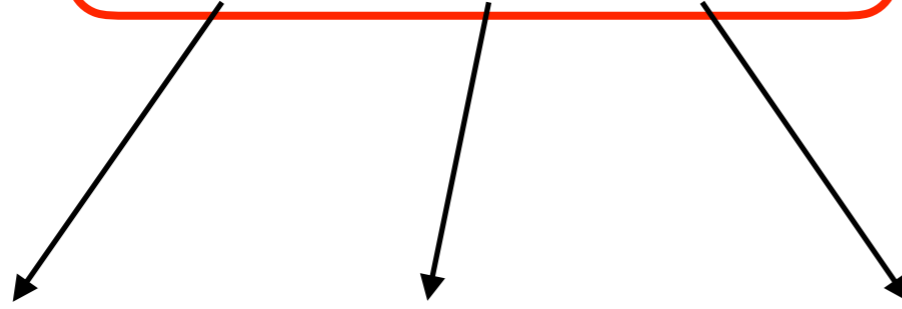
*optional stuff*

;

**where**

**order by**

**limit**



# Review from previous lecture

SELECT

FROM

*WHERE (optional)*

*ORDER BY (optional)*

*LIMIT (optional)*

;

# Mixing in GROUP BY

SELECT

FROM

*WHERE (optional)*

*ORDER BY (optional)*

*LIMIT (optional)*



**GROUP BY**

;

# Mixing in GROUP BY

SELECT

FROM

*WHERE (optional)*

***GROUP BY (optional)***

*ORDER BY (optional)*

*LIMIT (optional)*

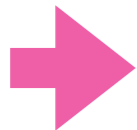
;

# Mixing in GROUP BY

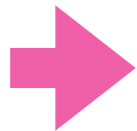
SELECT

FROM

*WHERE (optional)*



***GROUP BY (optional)***



*ORDER BY (optional)*

*LIMIT (optional)*

;

	Year	SUM(Revenue)
0	2006	3624.46
1	2007	4306.23
2	2008	5053.22
3	2009	5292.26
4	2010	5989.65
5	2011	5431.96
6	2012	6910.29

**Question:** what is the **total revenue** of **per year**?

**SQL Query:** `SELECT year, SUM(Revenue) FROM Movies  
GROUP BY year`

**Question:** what is the **total revenue** of **per year**?  
show most-recent years first...

**SQL Query:** `SELECT year, SUM(Revenue) FROM Movies  
GROUP BY year`



**Question:** what is the **total revenue** of **per year**?  
show most-recent years first...

**SQL Query:** SELECT **year**, SUM(**Revenue**) FROM **Movies**  
GROUP BY **year**  
ORDER BY **year** DESC


	Year	SUM(Revenue)
0	2016	11211.65
1	2015	8854.12
2	2014	7997.40
3	2013	7544.21
4	2012	6910.29
5	2011	5431.96
6	2010	5989.65

**Question:** what is the **total revenue** of **per year**?  
show most-recent years first...

**SQL Query:** SELECT **year**, **SUM(Revenue)** FROM Movies  
GROUP BY **year**  
ORDER BY **year** DESC

**Question:** what is the **total revenue** of **per year**?  
show highest-revenue years first...

**SQL Query:** SELECT **year**, **SUM(Revenue)** FROM Movies  
GROUP BY **year**  
ORDER BY **year** DESC



order by this

**Question:** what is the **total revenue** of **per year**?  
show highest-revenue years first...

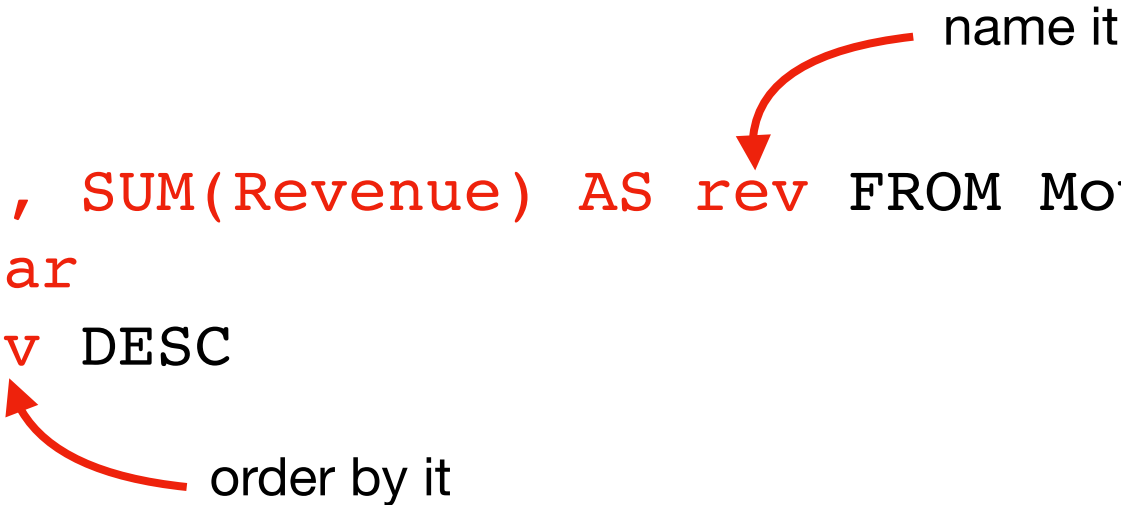
**SQL Query:** SELECT **year**, SUM(Revenue) AS **rev** FROM Movies  
GROUP BY **year**  
ORDER BY **year** DESC



name it

**Question:** what is the **total revenue** of **per year**?  
show highest-revenue years first...

**SQL Query:** SELECT **year**, SUM(Revenue) AS **rev** FROM Movies  
GROUP BY **year**  
ORDER BY **rev** DESC



name it

order by it

	Year	rev
0	2016	11211.65
1	2015	8854.12
2	2014	7997.40
3	2013	7544.21
4	2012	6910.29
5	2010	5989.65
6	2011	5431.96

**Question:** what is the **total revenue** of **per year**?  
show highest-revenue years first...

**SQL Query:** `SELECT year, SUM(Revenue) AS rev FROM Movies  
GROUP BY year  
ORDER BY rev DESC`

name it

order by it

# Mixing in GROUP BY

SELECT

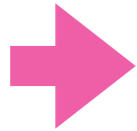
FROM



*WHERE (optional)*



***GROUP BY (optional)***



*ORDER BY (optional)*

*LIMIT (optional)*

;

**Question:** which **directors** have had the **largest number of movies** earning **over \$100M**?

**SQL Query:** SELECT ??? FROM Movies  
WHERE ???  
GROUP BY ???  
ORDER BY ???




**Question:**

which **directors** have had the **largest number of movies** earning **over \$100M**?

**SQL Query:**

```
SELECT ??? FROM Movies
WHERE revenue > 100
GROUP BY ???
ORDER BY ???
```




**Question:**

which **directors** have had the **largest number of movies** earning over \$100M?

**SQL Query:**

```
SELECT director FROM Movies
WHERE revenue > 100
GROUP BY director
ORDER BY ???
```

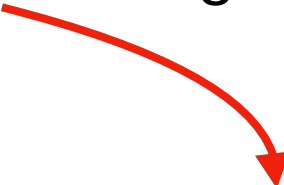


**Question:**

which **directors** have had the **largest number of movies** earning **over \$100M**?

**SQL Query:**

```
SELECT director, COUNT() FROM Movies  
WHERE revenue > 100  
GROUP BY director  
ORDER BY ???
```

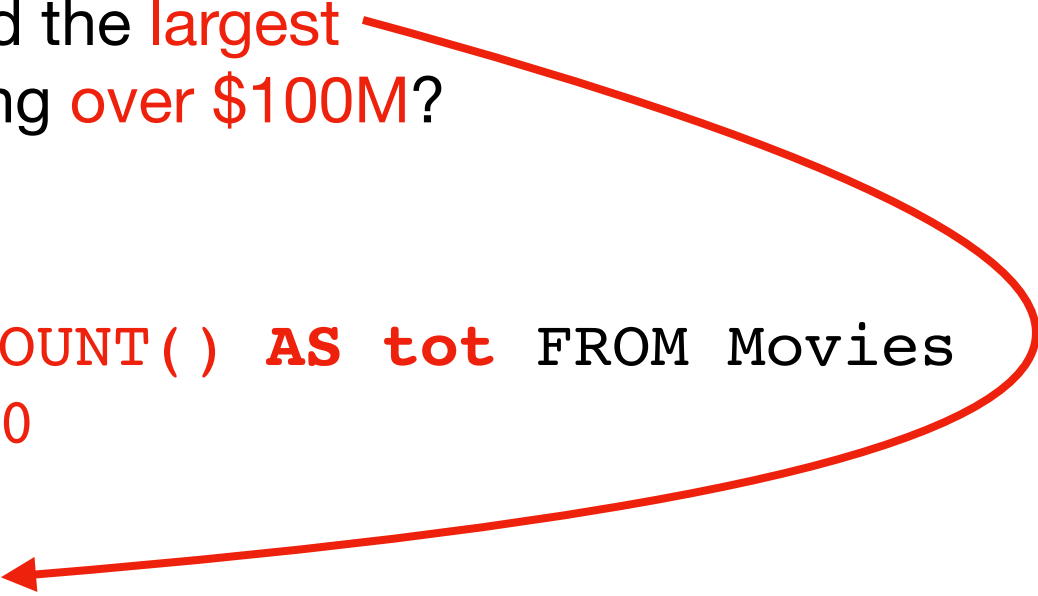


**Question:**

which **directors** have had the **largest number of movies** earning **over \$100M**?

**SQL Query:**

```
SELECT director, COUNT() AS tot FROM Movies  
WHERE revenue > 100  
GROUP BY director  
ORDER BY tot DESC
```



	Director	tot
0	David Yates	6
1	J.J. Abrams	5
2	Christopher Nolan	4
3	Dennis Dugan	4
4	Francis Lawrence	4
5	Justin Lin	4
6	Michael Bay	4

**Question:** which **directors** have had the **largest number of movies** earning **over \$100M**?

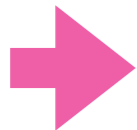
**SQL Query:** `SELECT director, COUNT() AS tot FROM Movies  
WHERE revenue > 100  
GROUP BY director  
ORDER BY tot DESC`

# Mixing in GROUP BY

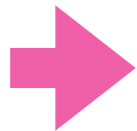
SELECT

FROM

*WHERE (optional)*



***GROUP BY (optional)***



*ORDER BY (optional)*



*LIMIT (optional)*

;

**sound familiar?**

**Question:** which **3** of the **directors** have the **greatest average rating**?

**SQL Query:**

```
SELECT ??? FROM Movies
GROUP BY ???
ORDER BY ???
LIMIT ???
```

**Question:**

which 3 of the **directors** have the **greatest average rating**?

**SQL Query:**

```
SELECT director FROM Movies  
GROUP BY director  
ORDER BY ???  
LIMIT ???
```



**Question:**

which 3 of the directors have the greatest average rating?

**SQL Query:**

```
SELECT director, AVG(Rating) FROM Movies
GROUP BY director
ORDER BY ???
LIMIT ???
```

**Question:**

which 3 of the directors have the **greatest** average rating?

**SQL Query:**

```
SELECT director, AVG(Rating) AS r FROM Movies
GROUP BY director
ORDER BY r DESC
LIMIT ???
```

**Question:** which **3** of the **directors** have the greatest average rating?

**SQL Query:** `SELECT director, AVG(Rating) AS r FROM Movies  
GROUP BY director  
ORDER BY r DESC  
LIMIT 3`

	Director	r
0	Nitesh Tiwari	8.80
1	Christopher Nolan	8.68
2	Makoto Shinkai	8.60

**Question:** which 3 of the directors have the greatest average rating?

**SQL Query:**

```
SELECT director, AVG(Rating) AS r FROM Movies
GROUP BY director
ORDER BY r DESC
LIMIT 3
```

*avg over how many?*

	Director	r
0	Nitesh Tiwari	8.80
1	Christopher Nolan	8.68
2	Makoto Shinkai	8.60

**Question:** which 3 of the directors have the greatest average rating?

**SQL Query:**

```
SELECT director, AVG(Rating) AS r FROM Movies
GROUP BY director
ORDER BY r DESC
LIMIT 3
```

	Director	r	COUNT()
0	Nitesh Tiwari	8.80	1
1	Christopher Nolan	8.68	5
2	Makoto Shinkai	8.60	1

**Question:** which 3 of the directors have the greatest average rating?

**SQL Query:**

```
SELECT director, AVG(Rating) AS r, COUNT()  
FROM Movies  
GROUP BY director  
ORDER BY r DESC  
LIMIT 3
```

	Director	r	COUNT()
0	Nitesh Tiwari	8.80	1
1	Christopher Nolan	8.68	5
2	Makoto Shinkai	8.60	1

what if we only want  
to consider actors in  
5+ movies?

can't use WHERE.  
WHERE filters rows  
**before** groups are made

**Question:** which 3 of the directors have the greatest average rating?

**SQL Query:**

```
SELECT director, AVG(Rating) AS r, COUNT()  
FROM Movies  
GROUP BY director  
ORDER BY r DESC  
LIMIT 3
```

# Outline

Aggregation Queries

Grouping with Python

Grouping with SQL

Combining with LIMIT and ORDER BY

WHERE vs. HAVING

Practice



# Filtering, before and after GROUP BY

Sometimes we want to filter data **before we form groups**.

Sometimes we want to **filter groups themselves**.

# Filtering, before and after GROUP BY

Sometimes we want to filter data **before we form groups**.

Sometimes we want to **filter groups themselves**.

Which directors have had more than 5 movies since 2014?

# Filtering, before and after GROUP BY

Sometimes we want to filter data **before we form groups**.

Sometimes we want to **filter groups themselves**.

Which directors have had **more than 5 movies** **since 2014**?

**filter groups** **filter input rows**

# Filtering, before and after GROUP BY

Sometimes we want to filter data **before we form groups**.

Sometimes we want to **filter groups themselves**.

Which directors have had **more than 5 movies** **since 2014**?

**filter groups** **filter input rows**

**HAVING** **WHERE**

“**HAVING**” is basically a “**WHERE**” for groups...

# Filtering

SELECT

FROM

*WHERE (optional)*

*GROUP BY (optional)*

*ORDER BY (optional)*

*LIMIT (optional)*

;

# Filtering

SELECT   
FROM   
*WHERE (optional)*  
*GROUP BY (optional)* ← **HAVING**  
*ORDER BY (optional)*  
*LIMIT (optional)* ;

# Filtering

SELECT

FROM

*WHERE (optional)*

*filter rows **before** bucketing into groups*

*GROUP BY (optional)*

*HAVING (optional)*

*filter groups **after** computing aggregates*

*ORDER BY (optional)*

*LIMIT (optional)*

;

# Filtering

SELECT

FROM

*WHERE (optional)*

*filter rows **before** bucketing into groups*

*GROUP BY (optional)*

*HAVING (optional)*

*filter groups **after** computing aggregates*

*ORDER BY (optional)*

*LIMIT (optional)*

;



**Note: this is very similar to “top directors” in P7**

**Question:** which **3** of the **directors** with **at least 5 movies** have **the greatest average rating**?

**SQL Query:**

```
SELECT ???  
FROM Movies  
GROUP BY ???  
HAVING ???  
ORDER BY ???  
LIMIT ???
```

**Note: this is very similar to “top directors” in P7**

**Question:** which 3 of the **directors** with at least 5 movies have the greatest average rating?

**SQL Query:**

```
SELECT director
FROM Movies
GROUP BY director
HAVING ???
ORDER BY ???
LIMIT ???
```

**Note: this is very similar to “top directors” in P7**

**Question:**

which 3 of the directors with **at least 5 movies** have the greatest average rating?

**SQL Query:**

```
SELECT director, COUNT() as c
FROM Movies
GROUP BY director
HAVING c >= 5
ORDER BY ???
LIMIT ???
```

**Note: this is very similar to “top directors” in P7**

**Question:** which 3 of the directors with at least 5 movies have the greatest average rating?

**SQL Query:**

```
SELECT director, COUNT() as c, AVG(rating) as r
FROM Movies
GROUP BY director
HAVING c >= 5
ORDER BY ???
LIMIT ???
```

**Note: this is very similar to “top directors” in P7**

**Question:** which 3 of the directors with at least 5 movies have the greatest average rating?

**SQL Query:**

```
SELECT director, COUNT() as c, AVG(rating) as r
FROM Movies
GROUP BY director
HAVING c >= 5
ORDER BY r DESC
LIMIT ???
```

**Note: this is very similar to “top directors” in P7**

**Question:** which **3** of the **directors** with **at least 5 movies** have **the greatest average rating**?

**SQL Query:**

```
SELECT director, COUNT() as c, AVG(rating) as r
FROM Movies
GROUP BY director
HAVING c >= 5
ORDER BY r DESC
LIMIT 3
```

	Director	c	r
0	Christopher Nolan	5	8.68
1	Martin Scorsese	5	7.92
2	David Fincher	5	7.82

**Question:** which 3 of the directors with at least 5 movies have the greatest average rating?

**SQL Query:**

```
SELECT director, COUNT() as c, AVG(rating) as r
FROM Movies
GROUP BY director
HAVING c >= 5
ORDER BY r DESC
LIMIT 3
```

# Outline

Aggregation Queries

Grouping with Python

Grouping with SQL

Combining with LIMIT and ORDER BY

WHERE vs. HAVING

Practice



**Fill in this notebook:**

<https://github.com/tylerharter/caraza-harter-com/blob/master/tyler/cs301/fall18/materials/code/lec-33/bus.ipynb>