Intro to RDBMS and SQL

ISTA 322 - Data Engineering

Working with a RDBMS

Connect to RDBMS

- To connect need credentials
  - Name
  - Host (location)
  - Username
  - Password
- DB Manger controls acces, etc.

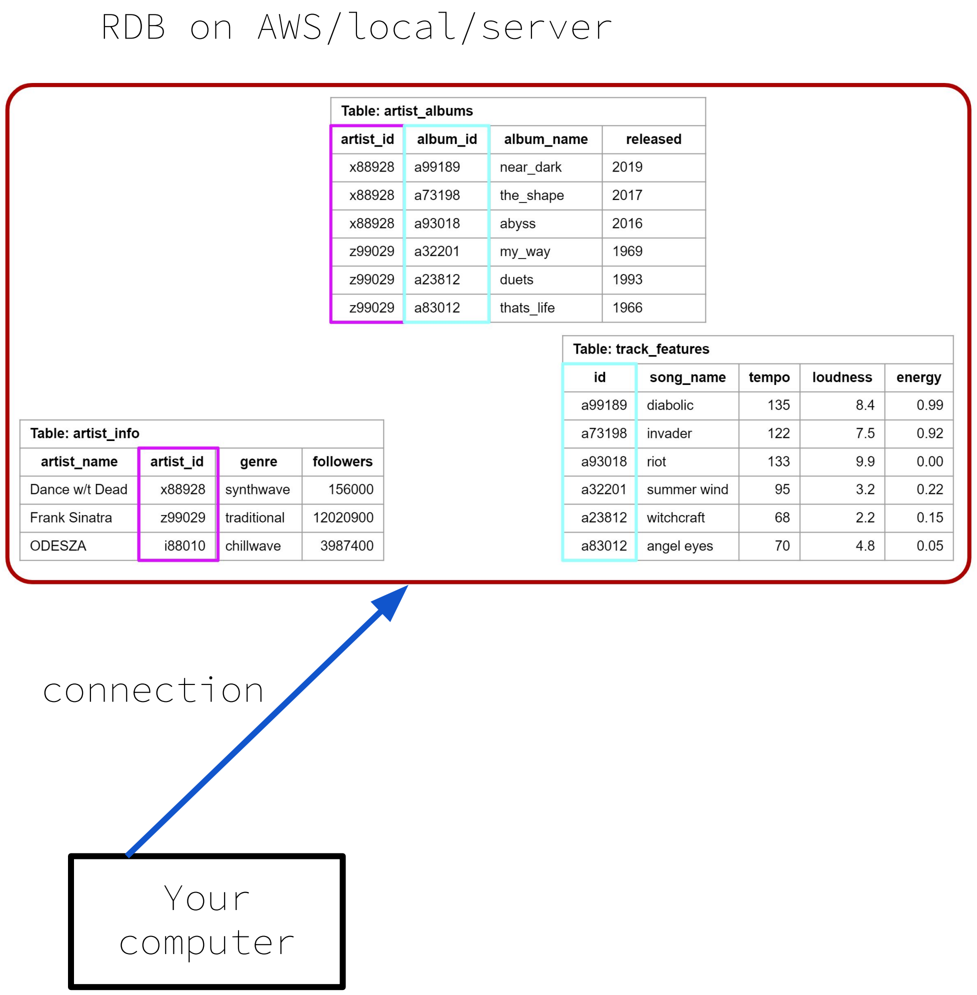

Table: artist_albums

artist_id	album_id	album_name	released
x88928	a99189	near_dark	2019
x88928	a73198	the_shape	2017
x88928	a93018	abyss	2016
z99029	a32201	my_way	1969
z99029	a23812	duets	1993
z99029	a83012	thats_life	1966

Table: artist_info

artist_name	artist_id	genre	followers
Dance w/t Dead	x88928	synthwave	156000
Frank Sinatra	z99029	traditional	12020900
ODESZA	i88010	chillwave	3987400

Table: track_features

id	song_name	tempo	loudness	energy
a99189	diabolic	135	8.4	0.99
a73198	invader	122	7.5	0.92
a93018	riot	133	9.9	0.00
a32201	summer wind	95	3.2	0.22
a23812	witchcraft	68	2.2	0.15
a83012	angel eyes	70	4.8	0.05

Working with a RDBMS

Create cursor

- Cursor allocates memory at the RDB

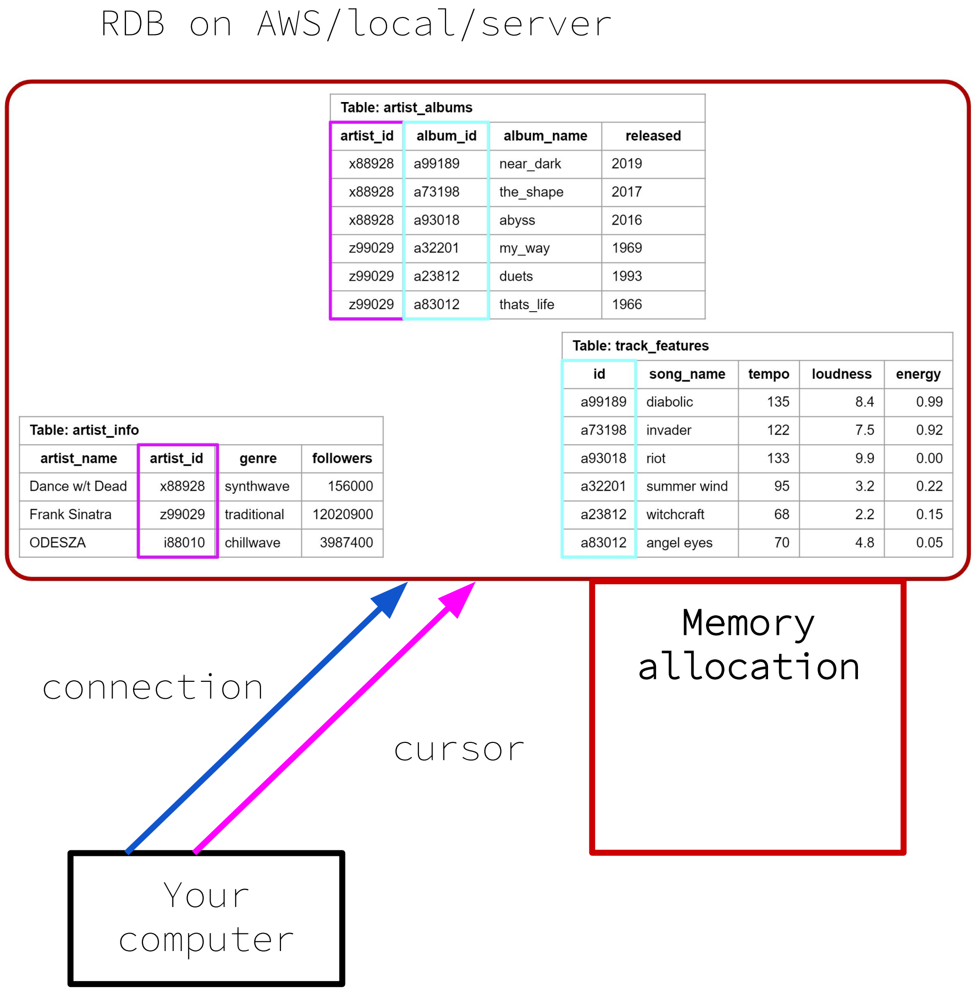

Table: artist_albums

artist_id	album_id	album_name	released
x88928	a99189	near_dark	2019
x88928	a73198	the_shape	2017
x88928	a93018	abyss	2016
z99029	a32201	my_way	1969
z99029	a23812	duets	1993
z99029	a83012	thats_life	1966

Table: artist_info

artist_name	artist_id	genre	followers
Dance w/t Dead	x88928	synthwave	156000
Frank Sinatra	z99029	traditional	12020900
ODESZA	i88010	chillwave	3987400

Table: track_features

id	song_name	tempo	loudness	energy
a99189	diabolic	135	8.4	0.99
a73198	invader	122	7.5	0.92
a93018	riot	133	9.9	0.00
a32201	summer wind	95	3.2	0.22
a23812	witchcraft	68	2.2	0.15
a83012	angel eyes	70	4.8	0.05

Working with a RDBMS

Fetch data using cursor

- Cursor allocates memory at the RDB
- Cursor object executes query

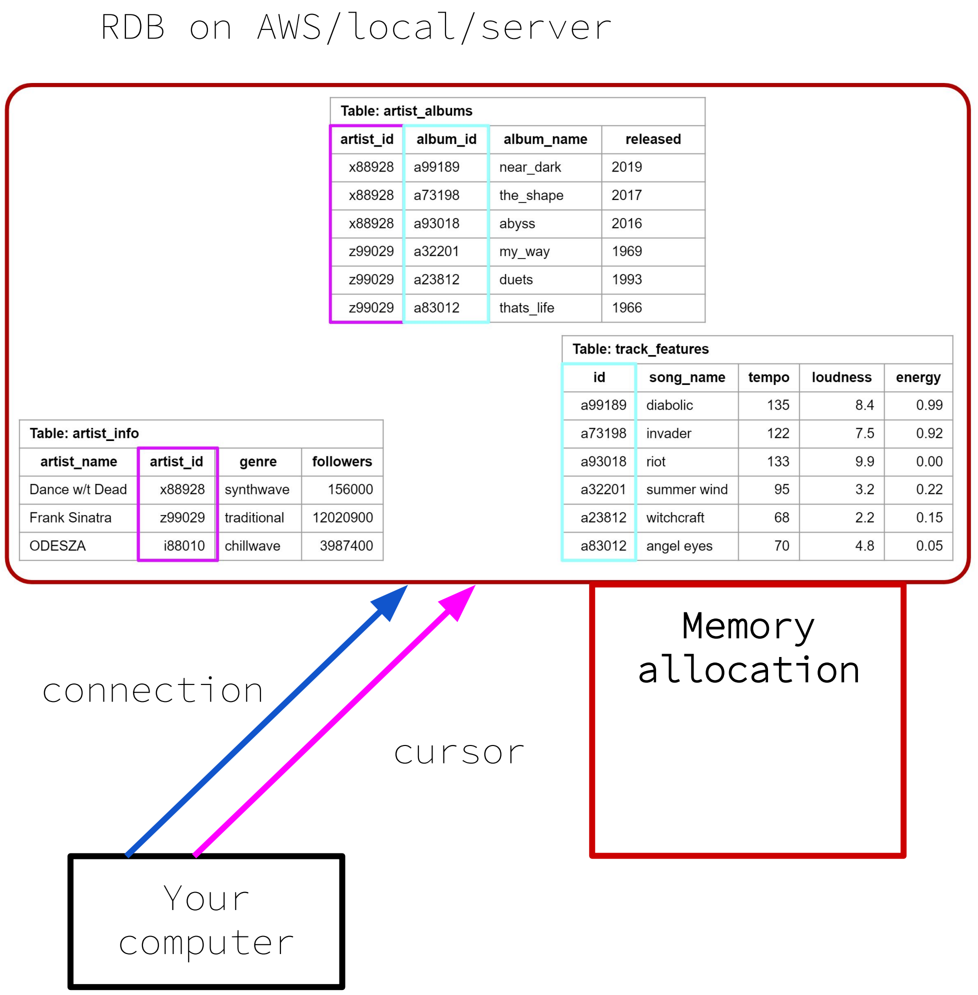

Table: artist_albums

artist_id	album_id	album_name	released
x88928	a99189	near_dark	2019
x88928	a73198	the_shape	2017
x88928	a93018	abyss	2016
z99029	a32201	my_way	1969
z99029	a23812	duets	1993
z99029	a83012	thats_life	1966

Table: track_features

id	song_name	tempo	loudness	energy
a99189	diabolic	135	8.4	0.99
a73198	invader	122	7.5	0.92
a93018	riot	133	9.9	0.00
a32201	summer wind	95	3.2	0.22
a23812	witchcraft	68	2.2	0.15
a83012	angel eyes	70	4.8	0.05

Table: artist_info

artist_name	artist_id	genre	followers
Dance w/t Dead	x88928	synthwave	156000
Frank Sinatra	z99029	traditional	12020900
ODESZA	i88010	chillwave	3987400

Working with a RDBMS

Fetch data using cursor

- Cursor allocates memory at the RDB
- Cursor object executes query
- Cursor used to fetch results which are then displayed or stored locally

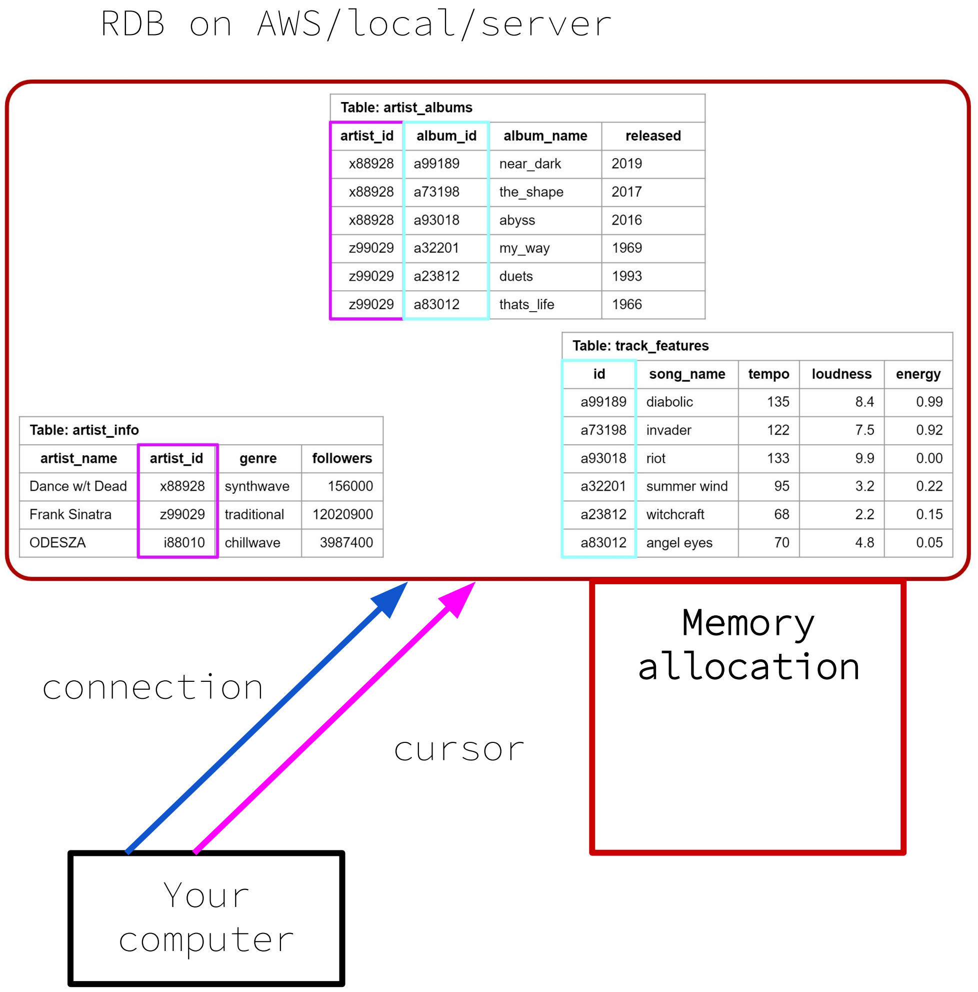

Table: artist_albums

artist_id	album_id	album_name	released
x88928	a99189	near_dark	2019
x88928	a73198	the_shape	2017
x88928	a93018	abyss	2016
z99029	a32201	my_way	1969
z99029	a23812	duets	1993
z99029	a83012	thats_life	1966

Table: track_features

id	song_name	tempo	loudness	energy
a99189	diabolic	135	8.4	0.99
a73198	invader	122	7.5	0.92
a93018	riot	133	9.9	0.00
a32201	summer wind	95	3.2	0.22
a23812	witchcraft	68	2.2	0.15
a83012	angel eyes	70	4.8	0.05

Table: artist_info

artist_name	artist_id	genre	followers
Dance w/t Dead	x88928	synthwave	156000
Frank Sinatra	z99029	traditional	12020900
ODESZA	i88010	chillwave	3987400

Working with a RDBMS

- It's important to not leave connections and cursors open
- Your functions should close them at the end!

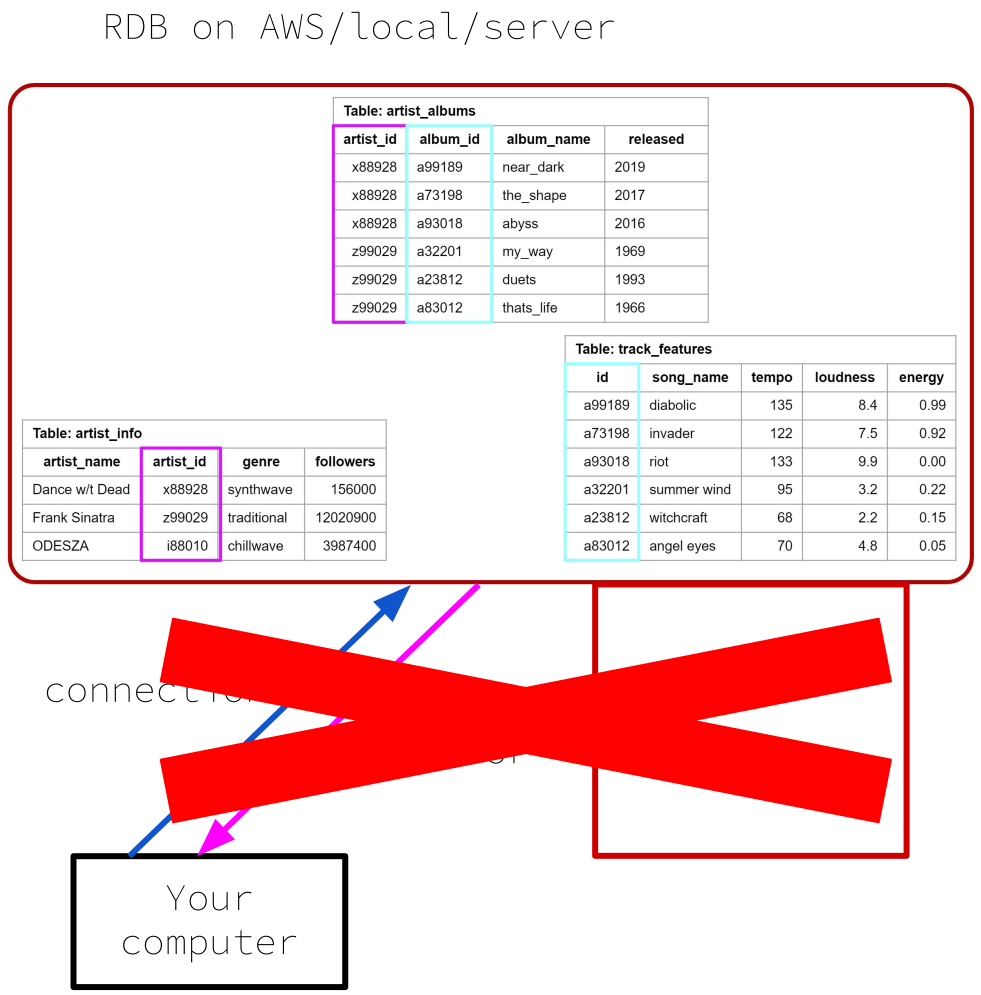

Table: artist_albums

artist_id	album_id	album_name	released
x88928	a99189	near_dark	2019
x88928	a73198	the_shape	2017
x88928	a93018	abyss	2016
z99029	a32201	my_way	1969
z99029	a23812	duets	1993
z99029	a83012	thats_life	1966

Table: track_features

id	song_name	tempo	loudness	energy
a99189	diabolic	135	8.4	0.99
a73198	invader	122	7.5	0.92
a93018	riot	133	9.9	0.00
a32201	summer wind	95	3.2	0.22
a23812	witchcraft	68	2.2	0.15
a83012	angel eyes	70	4.8	0.05

Table: artist_info

artist_name	artist_id	genre	followers
Dance w/t Dead	x88928	synthwave	156000
Frank Sinatra	z99029	traditional	12020900
ODESZA	i88010	chillwave	3987400

SQL Syntax

- After getting your connection and cursor you can write a query
- Query is the statement of actions that you want to perform
- SELECT, FROM, WHERE, GROUPBY...

Syntactical Order of Operations

1. SELECT
2. DISTINCT
3. AGGREGATIONS
4. FROM
5. JOIN
6. WHERE
7. GROUP BY
8. HAVING
9. ORDER BY

Starting with
the basics

Syntactical Order of Operations

1. SELECT
2. DISTINCT
3. AGGREGATIONS
4. FROM
5. JOIN
6. WHERE
7. GROUP BY
8. HAVING
9. ORDER BY

SELECT

- SELECT - Select the columns you want from a table
- Call columns by names
- Or call all using *

```
SELECT song_name, artist_id
```

SELECT

Table: top_track_features				
artist_id	song_name	tempo	loudness	energy
x88928	diabolic	135	8.4	0.99
x88928	invader	122	7.5	0.92
x88928	riot	133	9.9	0.00
z99029	summer wind	95	3.2	0.22
z99029	witchcraft	68	2.2	0.15
z99029	angel eyes	70	4.8	0.05

- SELECT - Select the columns you want from a table
- Call columns by names
- Or call all using *

```
SELECT song_name, artist_id
```

Syntactical Order of Operations

1. SELECT
2. DISTINCT
3. AGGREGATIONS
4. FROM
5. JOIN
6. WHERE
7. GROUP BY
8. HAVING
9. ORDER BY

FROM

- FROM - What table do you want the columns from?
- Obviously weird to call this second… more on this later.

```
SELECT song_name, artist_id
    FROM top_track_features
```

FROM

Table: top_track_features				
artist_id	**song_name**	**tempo**	**loudness**	**energy**
x88928	diabolic	135	8.4	0.99
x88928	invader	122	7.5	0.92
x88928	riot	133	9.9	0.00
z99029	summer wind	95	3.2	0.22
z99029	witchcraft	68	2.2	0.15
z99029	angel eyes	70	4.8	0.05

- FROM - What table do you want the columns from?
- Obviously weird to call this second… more on this later.

```
SELECT song_name, artist_id
    FROM top_track_features
```

Syntactical Order of Operations

1. SELECT
2. DISTINCT
3. AGGREGATIONS
4. FROM
5. JOIN
6. WHERE
7. GROUP BY
8. HAVING
9. ORDER BY

WHERE

- WHERE - allows you to filter rows by condition
- Comparison
  - = > < <>(not)

```
SELECT song_name, artist_id
    FROM top_track_features
    WHERE artist_id = ‘x88928’
```

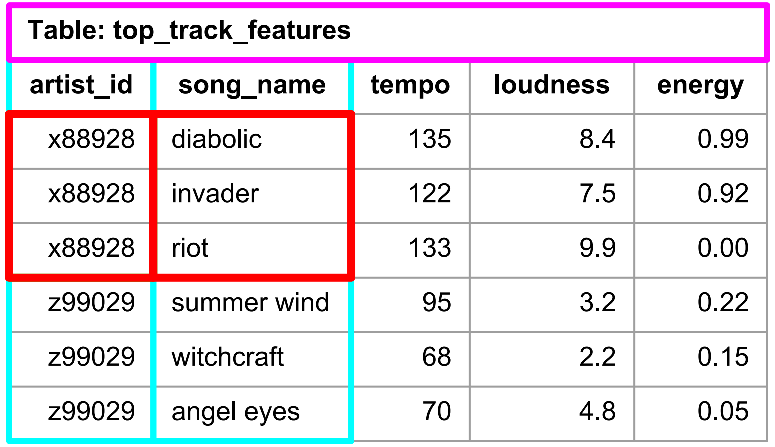

Table: top_track_features				
artist_id	song_name	tempo	loudness	energy
x88928	diabolic	135	8.4	0.99
x88928	invader	122	7.5	0.92
x88928	riot	133	9.9	0.00
z99029	summer wind	95	3.2	0.22
z99029	witchcraft	68	2.2	0.15
z99029	angel eyes	70	4.8	0.05

WHERE

- WHERE - allows you to filter rows by condition
- Comparison
  - = > < <>(not)

```
SELECT song_name, artist_id
    FROM top_track_features
    WHERE artist_id = ‘x88928’
```

Syntactical Order of Operations

1. SELECT
2. DISTINCT
3. AGGREGATIONS
4. FROM
5. JOIN
6. WHERE
7. GROUP BY
8. HAVING
9. ORDER BY

Slightly more
advanced functions

Syntactical Order of Operations

1. SELECT
2. DISTINCT
3. AGGREGATIONS
4. FROM
5. JOIN
6. WHERE
7. GROUP BY
8. HAVING
9. ORDER BY

GROUP BY

- GROUP BY - Allows you to apply aggregation functions to columns for each grouping level

```
SELECT AVG(tempo)
    FROM top_track_features
    GROUP BY artist_id
```

Syntactical Order of Operations

1. SELECT
2. DISTINCT
3. AGGREGATIONS
4. FROM
5. JOIN
6. WHERE
7. GROUP BY
8. HAVING
9. ORDER BY

GROUP BY

- GROUP BY
- Aggregation functions applied to columns
- MIN, MAX, AVG, COUNT, SUM

```
SELECT AVG(tempo)
    FROM top_track_features
    GROUP BY artist_id
```

Table: top_track_features				
artist_id	**song_name**	**tempo**	**loudness**	**energy**
x88928	diabolic	135	8.4	0.99
x88928	invader	122	7.5	0.92
x88928	riot	133	9.9	0.00
z99029	summer wind	95	3.2	0.22
z99029	witchcraft	68	2.2	0.15
z99029	angel eyes	70	4.8	0.05

RETURN	
artist_id	**AVG(tempo)**
x88928	(135+122+133)/3 = **130**
z99029	(95+68+70)/3 = **78.6**

```
SELECT AVG(tempo)
    FROM top_track_features
    GROUP BY artist_id
```

Table: top_track_features				
artist_id	song_name	tempo	loudness	energy
x88928	diabolic	135	8.4	0.99
x88928	invader	122	7.5	0.92
x88928	riot	133	9.9	0.00
z99029	summer wind	95	3.2	0.22
z99029	witchcraft	68	2.2	0.15
z99029	angel eyes	70	4.8	0.05

RETURN	
artist_id	AVG(tempo)
x88928	(135+122+133)/3 = **130**
z99029	(95+68+70)/3 = **78.6**

- A good time to talk about aliasing!
- Allows you to rename column
- Put ‘as new_name’ right after
- Later operations will use aliased name

```
SELECT AVG(tempo)
    FROM top_track_features
    GROUP BY artist_id
```

Table: top_track_features				
artist_id	song_name	tempo	loudness	energy
x88928	diabolic	135	8.4	0.99
x88928	invader	122	7.5	0.92
x88928	riot	133	9.9	0.00
z99029	summer wind	95	3.2	0.22
z99029	witchcraft	68	2.2	0.15
z99029	angel eyes	70	4.8	0.05

RETURN	
artist_id	avg_tempo
x88928	(135+122+133)/3 = **130**
z99029	(95+68+70)/3 = **78.6**

- A good time to talk about aliasing!
- Allows you to rename column
- Put 'as new_name' right after
- Later operations will use aliased name

```
SELECT AVG(tempo) as avg_tempo
    FROM top_track_features
    GROUP BY artist_id
```

Syntactical Order of Operations

1. SELECT
2. DISTINCT
3. AGGREGATIONS
4. FROM
5. JOIN
6. WHERE
7. GROUP BY
8. HAVING
9. ORDER BY

HAVING

- HAVING allows you to filter your aggregated data
- HAVING AVG(tempo)>100

```
SELECT AVG(tempo)
    FROM top_track_features
    GROUP BY artist_id
    HAVING AVG(tempo) > 100
```

Table: top_track_features

artist_id	song_name	tempo	loudness	energy
x88928	diabolic	135	8.4	0.99
x88928	invader	122	7.5	0.92
x88928	riot	133	9.9	0.00
z99029	summer wind	95	3.2	0.22
z99029	witchcraft	68	2.2	0.15
z99029	angel eyes	70	4.8	0.05

RETURN

artist_id	AVG(tempo)
x88928	(135+122+133)/3 = **130**
~~z99029~~	~~(95+68+70)/3 = **78.6**~~

```
SELECT AVG(tempo) as avg_temp
    FROM top_track_features
    GROUP BY artist_id
    HAVING AVG(tempo) > 100
```

Must use the aggregating function, not alias!

Syntactical Order of Operations

1. SELECT
2. DISTINCT
3. AGGREGATIONS
4. FROM
5. JOIN
6. WHERE
7. GROUP BY
8. HAVING
9. ORDER BY

ORDER BY

- ORDER BY values in a column
- Can be ascending or descending
- ASC | DESC

```
SELECT tempo
    FROM top_track_features
    ORDER BY tempo DESC
```

Table: top_track_features				
artist_id	**song_name**	**tempo**	**loudness**	**energy**
x88928	diabolic	135	8.4	0.99
x88928	invader	122	7.5	0.92
x88928	riot	133	9.9	0.00
z99029	summer wind	95	3.2	0.22
z99029	witchcraft	68	2.2	0.15
z99029	angel eyes	70	4.8	0.05

RETURN	
song_name	**tempo**
diabolic	135
riot	133
invader	122
summer wind	95
angel eyes	70
witchcraft	68

```
SELECT tempo, song_name
    FROM top_track_features
    ORDER BY tempo DESC
```

Table: top_track_features				
artist_id	**song_name**	**tempo**	**loudness**	**energy**
x88928	diabolic	135	8.4	0.99
x88928	invader	122	7.5	0.92
x88928	riot	133	9.9	0.00
z99029	summer wind	95	3.2	0.22
z99029	witchcraft	68	2.2	0.15
z99029	angel eyes	70	4.8	0.05

RETURN	
song_name	**tempo**
witchcraft	68
angel eyes	70
summer wind	95
invader	122
riot	133
diabolic	135

```
SELECT tempo, song_name
    FROM top_track_features
    ORDER BY tempo ASC
```

Syntactical Order of Operations

1. SELECT
2. DISTINCT
3. AGGREGATIONS
4. FROM
5. JOIN
6. WHERE
7. GROUP BY
8. HAVING
9. ORDER BY

JOIN

- JOIN lets you merge multiple tables

```
SELECT *
    FROM top_track_features
    LEFT JOIN artist_info ON
    top_track_features.artist_id =
    artist_info.artist_id
```

Table: top_track_features

artist_id	song_name	tempo	loudness	energy
x88928	diabolic	135	8.4	0.99
x88928	invader	122	7.5	0.92
x88928	riot	133	9.9	0.00
z99029	summer wind	95	3.2	0.22
z99029	witchcraft	68	2.2	0.15
z99029	angel eyes	70	4.8	0.05

RETURN

artist_id	song_name	tempo	loudness	energy	artist_name	...
x88928	diabolic	135	8.4	0.99	Dance w/t Dead	...
x88928	invader	122	7.5	0.92	Dance w/t Dead	...
x88928	riot	133	9.9	0.00	Dance w/t Dead	...
z99029	summer wind	95	3.2	0.22	Frank Sinatra	...
z99029	witchcraft	68	2.2	0.15	Frank Sinatra	...
z99029	angel eyes	70	4.8	0.05	Frank Sinatra	...

Table: artist_info

artist_name	artist_id	genre	followers
Dance w/t Dead	x88928	synthwave	156000
Frank Sinatra	z99029	traditional	12020900
ODESZA	i88010	chillwave	3987400

```
SELECT *
     FROM top_track_features
     LEFT JOIN artist_info ON
          top_track_features.artist_id =
          artist_info.artist_id
```

Table: top_track_features				
artist_id	song_name	tempo	loudness	energy
x88928	diabolic	135	8.4	0.99
x88928	invader	122	7.5	0.92
x88928	riot	133	9.9	0.00
z99029	summer wind	95	3.2	0.22
z99029	witchcraft	68	2.2	0.15
z99029	angel eyes	70	4.8	0.05

RETURN			
artist_id	song_name	tempo	artist_name
x88928	diabolic	135	Dance w/t Dead
x88928	invader	122	Dance w/t Dead
x88928	riot	133	Dance w/t Dead
z99029	summer wind	95	Frank Sinatra
z99029	witchcraft	68	Frank Sinatra
z99029	angel eyes	70	Frank Sinatra

Table: artist_info			
artist_name	artist_id	genre	followers
Dance w/t Dead	x88928	synthwave	156000
Frank Sinatra	z99029	traditional	12020900
~~ODESZA~~	~~i88010~~	~~chillwave~~	~~3987400~~

```
SELECT artist_info.artist_id, song_name,
tempo, artist_name
     FROM top_track_features
     LEFT JOIN artist_info ON
          top_track_features.artist_id =
          artist_info.artist_id
```

Syntactical Order of Operations

1. SELECT
2. DISTINCT
3. AGGREGATIONS
4. FROM
5. WHERE
6. GROUP BY
7. HAVING
8. ORDER BY

Logical Order of Operations

1. FROM
2. WHERE
3. GROUP BY
4. AGGREGATION
5. HAVING
6. SELECT
7. DISTINCT
8. ORDER BY