Introduction to SQL Queries

Pair Programming Exercise for DSE5002

HD Sheets, Feb. 5 2025

We will connect to the chinook database and work through some *queries*

The slang term for this is *pulling data from the database*

These are queries made to a single table, we will see later how to join tables together and create subqueries

The commands used in queries are:

SELECT --specify the variables or columns required

FROM--specify the table to obtain data from

LIMIT-- restrict the number of lines returned to a desired total N

WHERE-- this is a filtering function carried out on row elements, we can use AND, OR and NOT within the Where

ORDER BY-- This is a sorting function, it can sortascending or descending, and we can sort on multiple variables

GROUP BY--this is a grouping function

HAVING--Having is a filtering operation on group members

MAX(), MIN(), AVG(), SUM(), COUNT() are aggregating functions used with GROUP BY,

These are the available commands in SQL that we use in combination to create queries. We can do the same types of data slicing, sorting, filtering ect that we do in R or Python dataframes in SQL.

While the names are different, the actions appear in the TidyVerse library in R and in the Polars library in Python. Polars is an alternative to Pandas and has some advantages over Pandas with working with large data sets, and it's query system is almost identical to SQL, which is an advantage.

Source material

"Learning SQL", Beaulieu, O'Reilly 2005

https://www.sqlitetutorial.net/ - explains queries using the chinook database, albeit in the SQLite database system. The SELECT system used for queries is pretty standard for most SQL databases, the other aspects and commands seem to be a bit more variable from one server program to another.

That said, there are minor differences in variable names between the chinook database in postgres and the tutorial for SQlite, watch for underscores and pluralization (track vs tracks, etc). I have fixed all the examples shown here.

Connect to the Chinook Database

and figure out what we have in it

Set up the required libraries

In [1]:
```
#import psycopg2
import sqlalchemy

# we will want Pandas for the data frame structure

import pandas as pd
```

In [2]:
```
# Alter this to reflect your username and password,   this is for postgres c

# if you are using the default installation of Postgress and the databases 1
# command should work fine for the user bob whose password is pwd1

engine=sqlalchemy.create_engine('postgresql://bob:pwd1@localhost:5432/chinoo
```

In [3]:
```
pd.__version__
```

Out[3]:
```
'2.3.3'
```

what tables do we have

We can use a SELECT command to look for the table_name values in a built-in database called information_schema, in a table called tables.
This is a database and table that are built into postgres to hold information. It holds a lot of info, but our table names are all in the first 15 lines

In [4]:
```
pd.read_sql_query("SELECT table_name  FROM information_schema.tables LIMIT 1
```

Out[4]:

	table_name
0	album
1	artist
2	pg_type
3	customer
4	employee
5	genre
6	invoice
7	invoice_line
8	media_type
9	playlist
10	playlist_track
11	track
12	enames
13	pg_foreign_table
14	pg_roles

In [5]:

```
# Looking at the customer table, but only first 5 rows

pd.read_sql_query("SELECT * FROM customer LIMIT 5",engine)
```

Out[5]:

	customer_id	first_name	last_name	company	address	city	state	co
0	1	Luís	Gonçalves	Embraer - Empresa Brasileira de Aeronáutica S.A.	Av. Brigadeiro Faria Lima, 2170	São José dos Campos	SP	
1	2	Leonie	Köhler	None	Theodor-Heuss-Straße 34	Stuttgart	None	Ge
2	3	François	Tremblay	None	1498 rue Bélanger	Montréal	QC	C
3	4	Bjørn	Hansen	None	Ullevålsveien 14	Oslo	None	N
4	5	František	Wichterlová	JetBrains s.r.o.	Klanova 9/506	Prague	None	Re

In [6]:

```
#restrict this to only customer_id, first and last names


pd.read_sql_query("SELECT customer_id, first_name, last_name FROM customer L
```

Out[6]:

	customer_id	first_name	last_name
0	1	Luís	Gonçalves
1	2	Leonie	Köhler
2	3	François	Tremblay
3	4	Bjørn	Hansen
4	5	František	Wichterlová
5	6	Helena	Holý
6	7	Astrid	Gruber
7	8	Daan	Peeters

QUESTION/ACTION

Figure out what the table "invoices" looks like, display the first 5 lines of it so you can see the content

In [21]:

```
pd.read_sql_query("SELECT * FROM invoice LIMIT 5",engine)
```

Out[21]:

	invoice_id	customer_id	invoice_date	billing_address	billing_city	billing_state	bi
0	1	2	2021-01-01	Theodor-Heuss-Straße 34	Stuttgart	None	
1	2	4	2021-01-02	Ullevålsveien 14	Oslo	None	
2	3	8	2021-01-03	Grétrystraat 63	Brussels	None	
3	4	14	2021-01-06	8210 111 ST NW	Edmonton	AB	
4	5	23	2021-01-11	69 Salem Street	Boston	MA	

Question/Action

Show the variables customer_id, billing_country and total for the first 12 lines of invoice

In [22]:
```
pd.read_sql_query("SELECT customer_id, billing_country, total FROM invoice L
```

Out[22]:

	customer_id	billing_country	total
0	2	Germany	1.98
1	4	Norway	3.96
2	8	Belgium	5.94
3	14	Canada	8.91
4	23	USA	13.86
5	37	Germany	0.99
6	38	Germany	1.98
7	40	France	1.98
8	42	France	3.96
9	46	Ireland	5.94
10	52	United Kingdom	8.91
11	2	Germany	13.86

Ordering or Sorting Results

In [23]:
```
pd.read_sql_query("SELECT * FROM track ORDER BY Milliseconds LIMIT 12",engir
```

Out[23]:

	track_id	name	album_id	media_type_id	genre_id	composer	milliseconds
0	2461	É Uma Partida De Futebol	200	1	1	Samuel Rosa	1071
1	168	Now Sports	18	1	4	None	4884
2	170	A Statistic	18	1	4	None	6373
3	178	Oprah	18	1	4	None	6635
4	3304	Commercial 1	258	1	17	L. Muggerud	7941
5	172	The Real Problem	18	1	4	None	11650
6	3310	Commercial 2	258	1	17	L. Muggerud	21211
7	2241	Bossa	184	1	17	None	29048
8	1086	Casinha Feliz	85	1	10	Gilberto Gil	32287
9	246	Mateus Enter	24	1	7	Chico Science	33149
10	975	Deixa Entrar	78	1	7	None	33619
11	2797	Homem Primata (Vinheta)	224	1	4	Titãs	34168

In [24]:

```
# reversed order sort

# add DESC to sort descending, ASC to sort ascending

pd.read_sql_query("SELECT * FROM track ORDER BY Milliseconds DESC LIMIT 12",
```

Out[24]:

	track_id	name	album_id	media_type_id	genre_id	composer	milliseconds
0	2820	Occupation / Precipice	227	3	19	None	5286953
1	3224	Through a Looking Glass	229	3	21	None	5088838
2	3244	Greetings from Earth, Pt. 1	253	3	20	None	2960293
3	3242	The Man With Nine Lives	253	3	20	None	2956998
4	3227	Battlestar Galactica, Pt. 2	253	3	20	None	2956081
5	3226	Battlestar Galactica, Pt. 1	253	3	20	None	2952702
6	3243	Murder On the Rising Star	253	3	20	None	2935894
7	3228	Battlestar Galactica, Pt. 3	253	3	20	None	2927802
8	3248	Take the Celestra	253	3	20	None	2927677
9	3239	Fire In Space	253	3	20	None	2926593
10	3232	The Long Patrol	253	3	20	None	2925008
11	3235	The Magnificent Warriors	253	3	20	None	2924716

In [25]:

```
# sort by two variables

pd.read_sql_query("SELECT * FROM track ORDER BY composer ASC, milliseconds D
```

Out[25]:

	track_id	name	album_id	media_type_id	genre_id	co
0	2108	Children Of The Grave	174	1	3	A. F. Iommi, W. Ward, T. E O
1	2109	Paranoid	174	1	3	A. F. Iommi, W. Ward, T. E O
2	2107	Iron Man	174	1	3	A. F. Iommi, W. Ward, T. E O
3	1908	New Rhumba	157	1	2	/
4	415	Astronomy	35	1	3	A.Bouchard/J.Bouchard/S.P
5	2589	Hard To Handle	210	1	6	A.Isbell/A.Jones/O.
6	3427	Fanfare for the Common Man	296	2	24	Aaron (
7	3357	OAM's Blues	267	5	2	Aaron G
8	20	Overdose	4	1	1	
9	17	Let There Be Rock	4	1	1	
10	15	Go Down	4	1	1	
11	19	Problem Child	4	1	1	

Question/Action

Sort invoices by billing_city (ascending) and total purchase (descending), show the invoice_id, billing_city and total

In [26]:
```
pd.read_sql_query("SELECT invoice_id, billing_city, total FROM invoice ORDEF
```

Out[26]:

	invoice_id	billing_city	total
0	390	Amsterdam	13.86
1	206	Amsterdam	8.94
2	32	Amsterdam	8.91
3	184	Amsterdam	3.96
4	379	Amsterdam	1.98
...	...	...	...
407	388	Yellowknife	5.94
408	366	Yellowknife	3.96
409	343	Yellowknife	1.98
410	148	Yellowknife	1.98
411	27	Yellowknife	0.99

412 rows × 3 columns

Distinct

Selects only the unique values of a variable

In [27]:

```
# look at the Distinct cities in our customer list

pd.read_sql_query("""SELECT DISTINCT city
                    FROM customer
                    ORDER BY city
                    LIMIT 20;"""
                     ,engine)
```

Out[27]:

	city
0	Amsterdam
1	Bangalore
2	Berlin
3	Bordeaux
4	Boston
5	Brasília
6	Brussels
7	Budapest
8	Buenos Aires
9	Chicago
10	Copenhagen
11	Cupertino
12	Delhi
13	Dijon
14	Dublin
15	Edinburgh
16	Edmonton
17	Fort Worth
18	Frankfurt
19	Halifax

Question/Action

Find the list of distinct artists listed in Track, sort them

In [28]:
```
pd.read_sql_query("""SELECT DISTINCT name
                     FROM track
                     ORDER BY name;""", engine)
```

Out[28]:

	name
0	...And Found
1	...And Justice For All
2	...In Translation
3	.07%
4	'Round Midnight
...	...
3252	ZeroVinteUm
3253	Zither
3254	Zombie Eaters
3255	Zoo Station
3256	Zooropa

3257 rows × 1 columns

Where

Where is a filter that allows us to filter out only the rows that meet some desired condition.

Notice that the select command itself allows us to control the columns show, Where works on the rows

Comparison Operators

=, !=, <, >, >=, <= *Note equality is a single equal sign in postgres "="

Logical Operators

AND, NOT, OR

Other tests

ALL- 1 if all expressions are 1

ANY- 1 if any expressions is 1

BETWEEN- tests for a range of values

IN- comparison to a list of values

LIKE- used on strings, if they match a pattern

In [29]:
```
# we can select as specific album id for the tracks

pd.read_sql_query("""SELECT name, milliseconds,bytes,album_id
                     FROM track
                     WHERE album_id=6""", engine)
```

Out[29]:

	name	milliseconds	bytes	album_id
0	All I Really Want	284891	9375567	6
1	You Oughta Know	249234	8196916	6
2	Perfect	188133	6145404	6
3	Hand In My Pocket	221570	7224246	6
4	Right Through You	176117	5793082	6
5	Forgiven	300355	9753256	6
6	You Learn	239699	7824837	6
7	Head Over Feet	267493	8758008	6
8	Mary Jane	280607	9163588	6
9	Ironic	229825	7598866	6
10	Not The Doctor	227631	7604601	6
11	Wake Up	293485	9703359	6
12	You Oughta Know (Alternate)	491885	16008629	6

In [30]:
```
# we can select as specific album id for the tracks and restrict to relative

pd.read_sql_query("""SELECT name, milliseconds,bytes,album_id
                     FROM track
                     WHERE album_id=6 AND milliseconds<250000""", engine)
```

Out[30]:

	name	milliseconds	bytes	album_id
0	You Oughta Know	249234	8196916	6
1	Perfect	188133	6145404	6
2	Hand In My Pocket	221570	7224246	6
3	Right Through You	176117	5793082	6
4	You Learn	239699	7824837	6
5	Ironic	229825	7598866	6
6	Not The Doctor	227631	7604601	6

Question/Action

Find out how many invoices totals where over 25

In [31]:
```
pd.read_sql_query("""SELECT count(total)
                     FROM invoice
                     WHERE total > 25;""", engine)
```

Out[31]:

	count
0	1

LIKE

In [33]:
```
# The Like operator,  allows partial text matching

# note the use of the doubled percent signs %%
# also note that this is case sensitive

pd.read_sql_query("""SELECT name, album_id, composer
                     FROM track
                     WHERE composer LIKE '%%Smith%%'""",engine)
```

Out[33]:

	name	album_id	composer
0	Restless and Wild	3	F. Baltes, R.A. Smith-Diesel, S. Kaufman, U. D...
1	Princess of the Dawn	3	Deaffy & R.A. Smith-Diesel
2	Killing Floor	19	Adrian Smith
3	Machine Men	19	Adrian Smith
4	2 Minutes To Midnight	95	Adrian Smith/Bruce Dickinson
...	...	...	...
92	Savior	195	Anthony Kiedis/Chad Smith/Flea/John Frusciante
93	Dancing Barefoot	234	Ivan Kral/Patti Smith
94	Take the Box	322	Luke Smith
95	What Is It About Men	322	Delroy "Chris" Cooper, Donovan Jackson, Earl C...
96	Amy Amy Amy (Outro)	322	Astor Campbell, Delroy "Chris" Cooper, Donovan...

97 rows × 3 columns

Question/Action

Use the LIKE function to find all the invoice entries from Ireland

be sure to use LIKE, the = test would work here too, but practice using LIKE

In [36]:
```
pd.read_sql_query("""SELECT *
                     FROM invoice
                     WHERE billing_country LIKE '%%Ireland%%'""", engine)
```

Out [36]:

	invoice_id	customer_id	invoice_date	billing_address	billing_city	billing_state	bi
0	10	46	2021-02-03	3 Chatham Street	Dublin	Dublin	
1	62	46	2021-09-24	3 Chatham Street	Dublin	Dublin	
2	183	46	2023-03-18	3 Chatham Street	Dublin	Dublin	
3	194	46	2023-04-28	3 Chatham Street	Dublin	Dublin	
4	249	46	2023-12-27	3 Chatham Street	Dublin	Dublin	
5	378	46	2025-08-02	3 Chatham Street	Dublin	Dublin	
6	401	46	2025-11-04	3 Chatham Street	Dublin	Dublin	

IN

Tests for membership in a list

Also filtering out one AC/DC album using AND NOT combined with LIKE

In [37]:
```
pd.read_sql_query("""SELECT
                        name,
                        album_id,
                        media_type_id
                    FROM
                        track
                    WHERE
                        media_type_id IN (2, 3) AND NOT(name LIKE '%%Wall%%'
```

Out[37]:

	name	album_id	media_type_id
0	Fast As a Shark	3	2
1	Restless and Wild	3	2
2	Princess of the Dawn	3	2
3	Welcome to the Jungle	90	2
4	It's So Easy	90	2
...	...	...	...
444	There's No Place Like Home, Pt. 2	261	3
445	There's No Place Like Home, Pt. 3	261	3
446	Band Members Discuss Tracks from "Revelations"	271	3
447	Branch Closing	251	3
448	The Return	251	3

449 rows × 3 columns

AND

In [38]:
```
pd.read_sql_query("""SELECT
                      billing_address,
                      billing_city,
                      total
                    FROM
                      invoice
                    WHERE
                      billing_city= 'New York'
                    AND total > 5
                    ORDER BY
                      total;""",engine)
```

Out[38]:

	billing_address	billing_city	total
0	627 Broadway	New York	5.94
1	627 Broadway	New York	8.91
2	627 Broadway	New York	13.86

In [39]:
```
pd.read_sql_query("""SELECT * FROM invoice LIMIT 5""",engine)
```

Out[39]:

	invoice_id	customer_id	invoice_date	billing_address	billing_city	billing_state	bi
0	1	2	2021-01-01	Theodor-Heuss-Straße 34	Stuttgart	None	
1	2	4	2021-01-02	Ullevålsveien 14	Oslo	None	
2	3	8	2021-01-03	Grétrystraat 63	Brussels	None	
3	4	14	2021-01-06	8210 111 ST NW	Edmonton	AB	
4	5	23	2021-01-11	69 Salem Street	Boston	MA	

OR

Using AND and OR together

In [40]:

```
pd.read_sql_query("""SELECT
                      billing_address,
                      billing_city,
                      total
                    FROM
                      invoice
                    WHERE
                      (billing_city= 'New York' OR billing_city= 'Chicago')
                    AND total > 5
                    ORDER BY
                      total;""",engine)
```

Out[40]:

	billing_address	billing_city	total
0	162 E Superior Street	Chicago	5.94
1	627 Broadway	New York	5.94
2	162 E Superior Street	Chicago	7.96
3	162 E Superior Street	Chicago	8.91
4	627 Broadway	New York	8.91
5	627 Broadway	New York	13.86
6	162 E Superior Street	Chicago	15.86

BETWEEN

Looks for a range of values

In [41]:
```
pd.read_sql_query("""SELECT
                        invoice_id,
                        billing_address,
                        total
                    FROM
                        invoice
                    WHERE
                        total BETWEEN 14.91 and 18.86
                    ORDER BY
                        total; """,engine)
```

Out[41]:

	invoice_id	billing_address	total
0	193	Berger Straße 10	14.91
1	208	Ullevålsveien 14	15.86
2	103	162 E Superior Street	15.86
3	313	68, Rue Jouvence	16.86
4	306	Klanova 9/506	16.86
5	88	Calle Lira, 198	17.91
6	201	319 N. Frances Street	18.86
7	89	Rotenturmstraße 4, 1010 Innere Stadt	18.86

In [42]:
```
#NOT BETWEEN
#
# excluding a range


pd.read_sql_query("""SELECT
                        invoice_id,
                        billing_address,
                        total
                    FROM
                        invoice
                    WHERE
                        total NOT BETWEEN 1 and 20
                    ORDER BY
                        total; """,engine)
```

Out[42]:

	invoice_id	billing_address	total
0	405	541 Del Medio Avenue	0.99
1	13	1600 Amphitheatre Parkway	0.99
2	20	110 Raeburn Pl	0.99
3	27	5112 48 Street	0.99
4	34	Praça Pio X, 119	0.99
5	41	C/ San Bernardo 85	0.99
6	48	796 Dundas Street West	0.99
7	55	Grétrystraat 63	0.99
8	62	3 Chatham Street	0.99
9	69	319 N. Frances Street	0.99
10	76	Ullevålsveien 14	0.99
11	83	9, Place Louis Barthou	0.99
12	90	801 W 4th Street	0.99
13	384	162 E Superior Street	0.99
14	391	1498 rue Bélanger	0.99
15	398	11, Place Bellecour	0.99
16	6	Berger Straße 10	0.99
17	104	Barbarossastraße 19	0.99
18	111	1 Microsoft Way	0.99
19	118	421 Bourke Street	0.99
20	125	Rua da Assunção 53	0.99
21	132	Qe 7 Bloco G	0.99
22	139	Celsiusg. 9	0.99
23	146	230 Elgin Street	0.99
24	153	Sønder Boulevard 51	0.99
25	160	Via Degli Scipioni, 43	0.99
26	167	2211 W Berry Street	0.99
27	174	Klanova 9/506	0.99
28	181	68, Rue Jouvence	0.99
29	188	120 S Orange Ave	0.99
30	195	Av. Brigadeiro Faria Lima, 2170	0.99
31	209	627 Broadway	0.99

	invoice_id	billing_address	total
32	216	307 Macacha Güemes	0.99
33	223	Rua dos Campeões Europeus de Viena, 4350	0.99
34	230	8210 111 ST NW	0.99
35	237	202 Hoxton Street	0.99
36	244	194A Chain Lake Drive	0.99
37	251	Rua Dr. Falcão Filho, 155	0.99
38	258	Lijnbaansgracht 120bg	0.99
39	265	1033 N Park Ave	0.99
40	272	Rilská 3174/6	0.99
41	279	Porthaninkatu 9	0.99
42	286	69 Salem Street	0.99
43	293	Theodor-Heuss-Straße 34	0.99
44	300	8, Rue Hanovre	0.99
45	314	Calle Lira, 198	0.99
46	321	Tauentzienstraße 8	0.99
47	328	700 W Pender Street	0.99
48	335	113 Lupus St	0.99
49	342	696 Osborne Street	0.99
50	349	Av. Paulista, 2022	0.99
51	356	Ordynacka 10	0.99
52	363	302 S 700 E	0.99
53	370	Rotenturmstraße 4, 1010 Innere Stadt	0.99
54	377	Erzsébet krt. 58.	0.99
55	96	Erzsébet krt. 58.	21.86
56	194	3 Chatham Street	21.86
57	299	2211 W Berry Street	23.86
58	404	Rilská 3174/6	25.86

In [45]:
```
# shut down the engine to close the connection

engine.dispose()
```

In [ ]:
```
```