In [1]:
```
import pandas as pd
```

Question Set 1 - Easy

Q1: Who is the senior most employee based on job title?

In [4]:
```
df=pd.read_csv(r'C:\Users\rajga\OneDrive\Desktop\Business Analyst 2024\SQL\Projects
```

In [94]:
```
df.head(5)
```

Out[94]:

	employee_id	last_name	first_name	title	reports_to	levels	birthdate	hire_date	a
0	1	Adams	Andrew	General Manager	9.0	L6	18-02-1962 00:00	14-08-2016 00:00	A
1	2	Edwards	Nancy	Sales Manager	1.0	L4	08-12-1958 00:00	01-05-2016 00:00	
2	3	Peacock	Jane	Sales Support Agent	2.0	L1	29-08-1973 00:00	01-04-2017 00:00	
3	4	Park	Margaret	Sales Support Agent	2.0	L1	19-09-1947 00:00	03-05-2017 00:00	
4	5	Johnson	Steve	Sales Support Agent	2.0	L1	03-03-1965 00:00	17-10-2017 00:00	

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
employee_id    0
last_name      0
first_name     0
title          0
reports_to     1
levels         0
birthdate      0
hire_date      0
address        0
city           0
state          0
country        0
postal_code    0
phone          0
fax            0
email          0
dtype: int64
```

Explanation:

sort_values(by='levels', ascending=False): Sorts the DataFrame by the 'levels' column in descending order (similar to ORDER BY levels DESC).
.head(1): Retrieves the first row after sorting (similar to LIMIT 1).

In [9]:
```
# Sort the DataFrame by the 'levels' column in descending order and get the top row
top_employee = df.sort_values(by='levels', ascending=False).head(1)

# Display the result
print(top_employee)
```

```
   employee_id last_name first_name                  title  reports_to  \
8            9     Madan      Mohan  Senior General Manager         NaN

  levels         birthdate         hire_date             address      city  \
8     L7  26-01-1961 00:00  14-01-2016 00:00  1008 Vrinda Ave MT  Edmonton

  state country postal_code              phone                fax  \
8    AB  Canada     T5K 2N1  +1 (780) 428-9482  +1 (780) 428-3457

                        email
8  madan.mohan@chinookcorp.com
```

Using nlargest()

- nlargest() is a pandas function specifically designed to get the top N values in a column. In this case, we want the top 1 row based on the 'levels' column.

top_employee = df.nlargest(1, 'levels')
print(top_employee)

Using idxmax()

- idxmax() returns the index of the maximum value in a column. You can use this to locate the row with the highest 'levels'.

```
top_employee = df.loc[df['levels'].idxmax()]
print(top_employee)
```

Using Boolean Indexing

You can directly filter rows where the 'levels' column equals the maximum value.

```
top_employee = df[df['levels'] == df['levels'].max()]
print(top_employee)
```

Q2. Which countries have the most Invoices?

In [12]:
```
df_q2=pd.read_csv(r'C:\Users\rajga\OneDrive\Desktop\Business Analyst 2024\SQL\Proje
```

In [22]:
```
df_q2.head(2)
```

Out[22]:

	invoice_id	customer_id	invoice_date	billing_address	billing_city	billing_state	billing_c
0	1	18	2017-01-03 00:00:00	627 Broadway	New York	NY	
1	2	30	2017-01-03 00:00:00	230 Elgin Street	Ottawa	ON	

Use of reset index()

reset_index(name='c'): Now, reset_index() does the following:

- Before reset_index(), the billing_country is the index of the Series (it's not a normal column).
- reset_index(): This converts the index (which was billing_country) into a regular column, so the data becomes a\ DataFrame again, with a new integer index (0, 1, 2, etc.).\
- name='c': This renames the count column from its default name to 'c'

In [37]:
```
df_q2.groupby('customer_id').size().reset_index(name='count').head(5)
```

Out[37]:

	customer_id	count
0	1	13
1	2	11
2	3	9
3	4	9
4	5	18

In [38]:
```
df_q2.groupby('billing_country').size().reset_index(name='c').head(5)
```

Out[38]:

	billing_country	c
0	Argentina	5
1	Australia	10
2	Austria	9
3	Belgium	7
4	Brazil	61

In [45]:
```
# Group by 'billing_country' and count the number of rows for each country

result_q2 = df_q2.groupby('billing_country').size().reset_index(name='c')

# Sort by the count 'c' in descending order
result_q2 = result_q2.sort_values(by='c', ascending=False)

# Display the result
result_q2.head(5)
```

Out[45]:

	billing_country	c
22	USA	131
5	Canada	76
4	Brazil	61
10	France	50
11	Germany	41

To start the indexing from 0 we have to use the parameter drop = True

reset_index(drop=True)

In [46]:
```
# Display the result
result_q2.reset_index(drop=True).head(5)
```

Out[46]:

	billing_country	c
0	USA	131
1	Canada	76
2	Brazil	61
3	France	50
4	Germany	41

Q3. What are top 3 values of total invoice?

In [55]:
```
type(df_q2['total'])
```

Out[55]:
```
pandas.core.series.Series
```

For sorting the value we have to first convert the series into dataframe by using the double brackets [[ ]]

In [58]:
```
df_q2[['total']].sort_values(by='total',ascending=False).reset_index(drop=True).hea
```

Out[58]:

	total
0	23.76
1	19.80
2	19.80

Q.4 Which city has the best customers? We would like to throw a promotional Music Festival in the city we made the most money. Write a query that returns one city that has the highest sum of invoice totals. Return both the city name & sum of all invoice totals

In [68]:
```
result_q4=df_q2.groupby('billing_city')['total'].sum().reset_index(name='Total_Earn
```

In [72]:
```
result_q4.sort_values(by='Total_Earning_Per_City',ascending=False).reset_index(drop
```

Out[72]:

	billing_city	Total_Earning_Per_City
0	Prague	273.24
1	Mountain View	169.29
2	London	166.32
3	Berlin	158.40
4	Paris	151.47

Q.5 Who is the best customer? The customer who has spent the most money will be declared the best customer. Write a query that returns the person who has spent the most money

In [74]:
```
df_q5=pd.read_csv(r'C:\Users\rajga\OneDrive\Desktop\Business Analyst 2024\SQL\Proje
```

In [76]:
```
df_q5.head(3)
```

Out[76]:

	customer_id	first_name	last_name	company	address	city	state	country	p
0	1	Luís	Gonçalves	Embraer - Empresa Brasileira de Aeronáutica S.A.	Av. Brigadeiro Faria Lima, 2170	São José dos Campos	SP	Brazil	
1	2	Leonie	Köhler	NaN	Theodor-Heuss-Straße 34	Stuttgart	NaN	Germany	
2	3	François	Tremblay	NaN	1498 rue Bélanger	Montréal	QC	Canada	

In [73]:
```
df_q2.head(5)
```

Out[73]:

	invoice_id	customer_id	invoice_date	billing_address	billing_city	billing_state	billing_c
0	1	18	2017-01-03 00:00:00	627 Broadway	New York	NY	
1	2	30	2017-01-03 00:00:00	230 Elgin Street	Ottawa	ON	C
2	3	40	2017-01-05 00:00:00	8, Rue Hanovre	Paris	NaN	
3	4	18	2017-01-06 00:00:00	627 Broadway	New York	NY	
4	5	27	2017-01-07 00:00:00	1033 N Park Ave	Tucson	AZ	

In [79]:
```
# Combining the customer and invoice tables using a join
result_q5 = df_q5.merge(df_q2, on='customer_id')
result_q5.head(3)
```

Out[79]:

	customer_id	first_name	last_name	company	address	city	state	country	po
0	1	Luís	Gonçalves	Embraer - Empresa Brasileira de Aeronáutica S.A.	Av. Brigadeiro Faria Lima, 2170	São José dos Campos	SP	Brazil	1
1	1	Luís	Gonçalves	Embraer - Empresa Brasileira de Aeronáutica S.A.	Av. Brigadeiro Faria Lima, 2170	São José dos Campos	SP	Brazil	1
2	1	Luís	Gonçalves	Embraer - Empresa Brasileira de Aeronáutica S.A.	Av. Brigadeiro Faria Lima, 2170	São José dos Campos	SP	Brazil	1

3 rows × 21 columns

In [93]:
```
# Group by 'customer_id' and calculate the total sum for each customer
result = result_q5.groupby(['customer_id','first_name','last_name'])['total'].sum()


# Sort by 'total' in descending order and get the top result
top_customer = result.sort_values(by='total', ascending=False).head(1)

# Display the result
top_customer.reset_index(drop=True)
```

Out[93]:

	customer_id	first_name	last_name	total
0	5	František	Wichterlová	144.54

Second method without using join just by running two indivisual query

In [102…
```
# Group by 'customer_id' and calculate the sum for each customer
invoice_result = df_q2.groupby('customer_id')['total'].sum().reset_index(name='Max_

# Sort by 'Max_sum' in descending order and get the customer with the highest sum
max_sum_customer = invoice_result.sort_values(by='Max_sum', ascending=False).head(1

# Get the customer_id of the top customer
top_customer_id = max_sum_customer['customer_id'].values[0]

max_sum_customer  # This will display the customer_id and max sum
```

Out[102…

	customer_id	Max_sum
4	5	144.54

In [103…

```
# Select the first name and last name of the customer with the top customer_id
customer_info = df_q5[df_q5['customer_id'] == top_customer_id][['first_name', 'last

# Display the customer details
print(customer_info)
```

```
  first_name    last_name
4  František  Wichterlová
```

Question Set 2 – Moderate

Write query to return the email, first name, last name, & Genre of all Rock Music listeners. Return your list ordered alphabetically by email starting with A

In [106…

```
df_il=pd.read_csv(r'C:\Users\rajga\OneDrive\Desktop\Business Analyst 2024\SQL\Proje
```

In [107…

```
df_track=pd.read_csv(r'C:\Users\rajga\OneDrive\Desktop\Business Analyst 2024\SQL\Pr
```

In [108…

```
df_genre=pd.read_csv(r'C:\Users\rajga\OneDrive\Desktop\Business Analyst 2024\SQL\Pr
```

In [151…

```
# Step 1: Join the dataframes together as per the SQL query
result_s2_q1 = df_q5.merge(df_q2, on='customer_id') \
                    .merge(df_il, on='invoice_id') \
                    .merge(df_track, on='track_id') \
                    .merge(df_genre, on='genre_id')
result_s2_q1.head(2)
```

Out[151…

	customer_id	first_name	last_name	company	address	city	state	country	po
0	1	Luís	Gonçalves	Embraer - Empresa Brasileira de Aeronáutica S.A.	Av. Brigadeiro Faria Lima, 2170	São José dos Campos	SP	Brazil	1
1	1	Luís	Gonçalves	Embraer - Empresa Brasileira de Aeronáutica S.A.	Av. Brigadeiro Faria Lima, 2170	São José dos Campos	SP	Brazil	1

2 rows × 34 columns

In [152…
```
# Step 2: Filter where genre name is 'Rock'
result_s2_q1 = result_s2_q1[result_s2_q1['name_y'].str.contains('Rock', na=False)]
```

In [153…
```
# Step 3: Select distinct columns and rename them to match the SQL query
result_s2_q1 = result_s2_q1[['email', 'first_name', 'last_name', 'name_y']].drop_du

# Step 4: Sort by 'email' as in SQL
result_s2_q1 = result_s2_q1.sort_values(by='email',ascending=True)
```

In [157…
```
# Step 5: Rename columns as per SQL query output
result_s2_q1 = result_s2_q1.rename(columns={'email': 'Email', 'first_name': 'FirstN

# Display the result
result_s2_q1.reset_index(drop=True).head(5)
```

Out[157…

	Email	FirstName	LastName	Name
0	aaronmitchell@yahoo.ca	Aaron	Mitchell	Rock
1	alero@uol.com.br	Alexandre	Rocha	Rock
2	astrid.gruber@apple.at	Astrid	Gruber	Rock
3	bjorn.hansen@yahoo.no	Bjørn	Hansen	Rock
4	camille.bernard@yahoo.fr	Camille	Bernard	Rock

Q.2 Let's invite the artists who have written the most rock music in our dataset. Write a query that returns the Artist name and total track count of the top 10 rock bands

In [159…
```
df_album=pd.read_csv(r'C:\Users\rajga\OneDrive\Desktop\Business Analyst 2024\SQL\Pr
```

In [158…
```
df_artist=pd.read_csv(r'C:\Users\rajga\OneDrive\Desktop\Business Analyst 2024\SQL\P
```

In [171…
```
result_s2_q2=df_genre.merge(df_track,on='genre_id') \
        .merge(df_album,on='album_id') \
        .merge(df_artist,on='artist_id')

result_s2_q2 = result_s2_q2[result_s2_q2['name_x'].str.contains('Rock', na=False)]
result_s2_q2 = result_s2_q2.groupby(['artist_id', 'name']).size().reset_index(name=

result_s2_q2 = result_s2_q2.sort_values(by='number_of_songs', ascending=False).head

result_s2_q2.reset_index(drop=True)
```

Out[171…

	artist_id	name	number_of_songs
0	22	Led Zeppelin	114
1	150	U2	112
2	58	Deep Purple	92
3	90	Iron Maiden	81
4	118	Pearl Jam	54
5	152	Van Halen	52
6	51	Queen	45
7	142	The Rolling Stones	41
8	76	Creedence Clearwater Revival	40
9	52	Kiss	35

Q3. Return all the track names that have a song length longer than the average song length.Return the Name and Milliseconds for each track. Order by the song length with the longest songs listed first

In [186…

```
# Step 1: Calculate the average track length
avg_track_length = df_track['milliseconds'].mean()

# Step 2: Filter tracks where the track length is greater than the average
result_s2_q3 = df_track[df_track['milliseconds'] > avg_track_length]

# Step 3: Select 'name' and 'milliseconds' columns and sort by track length in desc
result_s2_q3_sorted = result_s2_q3[['name', 'milliseconds']].sort_values(by='millis

# Display the result
result_s2_q3_sorted.reset_index(drop=True).head(2)
```

Out[186…

	name	milliseconds
0	Occupation / Precipice	5286953
1	Through a Looking Glass	5088838

Question Set 3 – Advance

Q1. Find how much amount spent by each customer on artists? Write a query to return customer name, artist name and total spent

In [202…

```
    df_il.merge(df_track, on='track_id') \
    .merge(df_album, on='album_id') \
    .merge(df_artist, on='artist_id').head(4)
```

Out[202…

	invoice_line_id	invoice_id	track_id	unit_price_x	quantity	name_x	album_id	media_ty
0	1	1	1158	0.99	1	Right Next Door to Hell	91	
1	2	1	1159	0.99	1	Dust N' Bones	91	
2	3	1	1160	0.99	1	Live and Let Die	91	
3	4	1	1161	0.99	1	Don't Cry (Original)	91	

In [208…

```
# Step 1: Calculate the best-selling artist based on total sales
best_selling_artist = (
    df_il.merge(df_track, on='track_id') \
    .merge(df_album, on='album_id') \
    .merge(df_artist, on='artist_id') \
    .groupby(['artist_id','name_y'])['unit_price_y'].sum().reset_index(name='total_
    .sort_values(by='total_sales', ascending=False)
    .head(1)
)
```

In [209…

```
# Step 2: Join customer purchases data with the best-selling artist
result = (
    df_q2.merge(df_q5, on='customer_id')
    .merge(df_il, on='invoice_id')
    .merge(df_track, on='track_id')
    .merge(df_album, on='album_id')
    .merge(best_selling_artist, on='artist_id')
    .groupby(['customer_id', 'first_name', 'last_name', 'name_y'])['unit_price_y'].
    .sort_values(by='amount_spent', ascending=False)
)

# Display the result
print(result)
```

	customer_id	first_name	last_name	name_y	amount_spent
31	46	Hugh	O'Reilly	Queen	27.72
24	38	Niklas	Schröder	Queen	18.81
1	3	François	Tremblay	Queen	17.82
21	34	João	Fernandes	Queen	16.83
26	41	Marc	Dubois	Queen	11.88
37	53	Phil	Hughes	Queen	11.88
32	47	Lucas	Mancini	Queen	10.89
20	33	Ellie	Sullivan	Queen	10.89
11	20	Dan	Miller	Queen	3.96
2	5	František	Wichterlová	Queen	3.96
38	54	Steve	Murray	Queen	2.97
19	31	Martha	Silk	Queen	2.97
13	23	John	Gordon	Queen	2.97
36	52	Emma	Jones	Queen	1.98
9	17	Jack	Smith	Queen	1.98
7	15	Jennifer	Peterson	Queen	1.98
8	16	Frank	Harris	Queen	1.98
5	11	Alexandre	Rocha	Queen	1.98
4	8	Daan	Peeters	Queen	1.98
0	1	Luís	Gonçalves	Queen	1.98
40	57	Luis	Rojas	Queen	1.98
27	42	Wyatt	Girard	Queen	1.98
17	28	Julia	Barnett	Queen	1.98
18	30	Edward	Francis	Queen	1.98
22	35	Madalena	Sampaio	Queen	1.98
23	36	Hannah	Schneider	Queen	1.98
14	24	Frank	Ralston	Queen	1.98
33	48	Johannes	Van der Berg	Queen	1.98
34	49	Stanisław	Wójcik	Queen	1.98
29	44	Terhi	Hämäläinen	Queen	1.98
3	6	Helena	Holý	Queen	0.99
10	19	Tim	Goyer	Queen	0.99
6	13	Fernanda	Ramos	Queen	0.99
16	27	Patrick	Gray	Queen	0.99
15	26	Richard	Cunningham	Queen	0.99
25	39	Camille	Bernard	Queen	0.99
12	22	Heather	Leacock	Queen	0.99
28	43	Isabelle	Mercier	Queen	0.99
30	45	Ladislav	Kovács	Queen	0.99
35	50	Enrique	Muñoz	Queen	0.99
39	55	Mark	Taylor	Queen	0.99
41	58	Manoj	Pareek	Queen	0.99
42	59	Rishabh	Mishra	Queen	0.99

Q2. We want to find out the most popular music Genre for each country. We determine the most popular genre as the genre with the highest amount of purchases. Write a query that returns each country along with the top Genre. For countries where the maximum number of purchases is shared return all Genres

In [238…

```
# Step 1: Perform necessary joins
df_joined = df_il \
    .merge(df_q2, on='invoice_id') \
    .merge(df_q5, on='customer_id') \
    .merge(df_track, on='track_id') \
```

```
    .merge(df_genre, on='genre_id')

df_joined.head(2)
```

Out[238...

	invoice_line_id	invoice_id	track_id	unit_price_x	quantity	customer_id	invoice_date	bi
0	1	1	1158	0.99	1	18	2017-01-03 00:00:00	
1	2	1	1159	0.99	1	18	2017-01-03 00:00:00	

2 rows × 34 columns

In [239...

```
# Step 2: Group by country, genre, and genre_id, then count 'invoice_line.quantity'
df_grouped = df_joined.groupby(['country', 'name_y', 'genre_id']).size().reset_inde
df_grouped.head(5)
```

Out[239...

	country	name_y	genre_id	quantity
0	Argentina	Alternative	23	1
1	Argentina	Alternative & Punk	4	17
2	Argentina	Blues	6	2
3	Argentina	Easy Listening	12	1
4	Argentina	Heavy Metal	13	1

In [216...

```
Same as step 2 just used agg instead of size to calculate the count

# # Step 2: Group by country, genre, and genre_id, then count 'invoice_line.quantit
# df_grouped = df_joined.groupby(['country', 'name_y', 'genre_id']).agg({'quantity'
# df_grouped.head(5)
```

Out[216...

	country	name_y	genre_id	quantity
0	Argentina	Alternative	23	1
1	Argentina	Alternative & Punk	4	17
2	Argentina	Blues	6	2
3	Argentina	Easy Listening	12	1
4	Argentina	Heavy Metal	13	1

In [244...

```
# Step 3: Sort within each country by the count of purchases in descending order an
df_grouped['rowNo'] = df_grouped.groupby('country')['quantity'].rank(method='first'
df_grouped[['rowNo']].head(5)
```

Out[244…

	rowNo
0	7.0
1	1.0
2	3.0
3	8.0
4	9.0

In [236…

```
# Step 4: Filter rows where row number is less than or equal to 1
df_top_genre = df_grouped[df_grouped['rowNo'] <= 1]
df_top_genre.head(5)
```

Out[236…

	country	name_y	genre_id	quantity	rowNo
1	Argentina	Alternative & Punk	4	17	1.0
20	Australia	Rock	1	34	1.0
28	Austria	Rock	1	40	1.0
38	Belgium	Rock	1	26	1.0
51	Brazil	Rock	1	205	1.0

In [246…

```
# Step 5: Sort the result for better readability
df_top_genre = df_top_genre.sort_values(['country', 'quantity'], ascending=[True, F

# Display the result
df_top_genre.head(5)
```

Out[246…

	country	name_y	genre_id	quantity	rowNo
1	Argentina	Alternative & Punk	4	17	1.0
20	Australia	Rock	1	34	1.0
28	Austria	Rock	1	40	1.0
38	Belgium	Rock	1	26	1.0
51	Brazil	Rock	1	205	1.0

Q3. Write a query that determines the customer that has spent the most on music for each country. Write a query that returns the country along with the top customer and how much they spent. For countries where the top amount spent is shared, provide all customers who spent this amount

In [256…

```
# Step 1: Merge customer and invoice data
merged_df = df_q2.merge(df_q5, on='customer_id')
```

```
# Step 2: Group by customer details and billing country, summing total spending
grouped_df = merged_df.groupby(['customer_id', 'first_name', 'last_name', 'billing_
['total'].sum().rename(columns={'total': 'total_spending'})

# Step 3: Sort by billing_country and total_spending (descending)
grouped_df = grouped_df.sort_values(['billing_country', 'total_spending'], ascendin
```

In [253…

In [260…

```
# Step 4: Add a row number (rn) for each partition by billing_country
# grouped_df['rn'] = grouped_df.groupby('billing_country').cumcount() + 1
grouped_df['rn'] = grouped_df.groupby('billing_country')['total_spending'].rank(met


# Step 5: Filter to get the top spending customer per country
result = grouped_df[grouped_df['rn'] <= 1]

# Display the result
result.head(5)
```

Out[260…

	customer_id	first_name	last_name	billing_country	total_spending	rn
55	56	Diego	Gutiérrez	Argentina	39.60	1
54	55	Mark	Taylor	Australia	81.18	1
6	7	Astrid	Gruber	Austria	69.30	1
7	8	Daan	Peeters	Belgium	60.39	1
9	10	Eduardo	Martins	Brazil	60.39	1

THE END