exercise-module-42

October 23, 2024

1 Module 42: Advance queries in SQL using Python

Author: Juliho David Castillo Colmenares

[49]:
```
import pandas as pd
import sqlite3

# Load the data from the CSV file
data_new = pd.read_csv("recursos_humanos.csv")
data_new.head()
```

[49]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	sales	salary
0	0.38	0.53	2	157	3	0	1	0	sales	low
1	0.80	0.86	5	262	6	0	1	0	sales	medium
2	0.11	0.88	7	272	4	0	1	0	sales	medium
3	0.72	0.87	5	223	5	0	1	0	sales	low
4	0.37	0.52	2	159	3	0	1	0	sales	low

[50]:
```
# Recreate the SQLite database and reinsert data into the 'Details' table
conn = sqlite3.connect("RH.db")
cursor = conn.cursor()
```

[51]:
```
# Create table and insert data again
cursor.execute(
```

```
    """
CREATE TABLE IF NOT EXISTS Details (
    satisfaction_level REAL,
    last_evaluation REAL,
    number_project INTEGER,
    average_monthly_hours INTEGER,
    time_spend_company INTEGER,
    work_accident INTEGER,
    left INTEGER,
    promotion_last_5years INTEGER,
    sales TEXT,
    salary TEXT
)
"""
)
```

[51]:
```
<sqlite3.Cursor at 0x193e3beef40>
```

[52]:
```
# Update column name to fix potential inconsistencies
data_new.columns = [
    col.replace("average_montly_hours", "average_monthly_hours").lower()
    for col in data_new.columns
]
data_new.to_sql("Details", conn, if_exists="replace", index=False)
```

[52]:
```
14999
```

[53]:
```
# 1. Display "sales", "salary", and "satisfaction_level" ordered by␣
 ↪satisfaction_level descending
query1 = """
SELECT sales, salary, satisfaction_level
FROM Details
ORDER BY satisfaction_level DESC
"""
result1 = pd.read_sql_query(query1, conn)
result1.head()
```

[53]:

	sales	salary	satisfaction_level
0	technical	low	1.0
1	technical	low	1.0
2	support	low	1.0
3	sales	low	1.0
4	technical	low	1.0

[54]:
```
# 2. Display "salary", "number_project", and "satisfaction_level" ordered by␣
 ↪number_project ascending and satisfaction_level descending
query2 = """
```

```
SELECT salary, number_project, satisfaction_level
FROM Details
ORDER BY number_project ASC, satisfaction_level DESC
"""
result2 = pd.read_sql_query(query2, conn)
result2.head()
```

[54]:

	salary	number_project	satisfaction_level
0	low	2	1.0
1	medium	2	1.0
2	medium	2	1.0
3	medium	2	1.0
4	medium	2	1.0

[55]:
```
# 3. Average of last_evaluation for each department (sales)
query3 = """
SELECT sales, AVG(last_evaluation) as avg_last_evaluation
FROM Details
GROUP BY sales
"""
result3 = pd.read_sql_query(query3, conn)
result3.head()
```

[55]:

	sales	avg_last_evaluation
0	IT	0.716830
1	RandD	0.712122
2	accounting	0.717718
3	hr	0.708850
4	management	0.724000

[56]:
```
# 4. Group by sales and salary, showing the average of last_evaluation
query4 = """
SELECT sales, salary, AVG(last_evaluation) as avg_last_evaluation
FROM Details
GROUP BY sales, salary
"""
result4 = pd.read_sql_query(query4, conn)
result4.head()
```

[56]:

	sales	salary	avg_last_evaluation
0	IT	high	0.716627
1	IT	low	0.715665
2	IT	medium	0.718187
3	RandD	high	0.700588
4	RandD	low	0.714176

[57]:
```
# 5. Extract departments where the average Work_accident is greater than 0.15
query5 = """
SELECT sales, AVG(work_accident) as avg_work_accident
FROM Details
GROUP BY sales
HAVING avg_work_accident > 0.15
"""
result5 = pd.read_sql_query(query5, conn)
result5.head()
```

[57]:

	sales	avg_work_accident
0	RandD	0.170267
1	management	0.163492
2	marketing	0.160839
3	support	0.154778

[58]:
```
# Alternative query: departments where total accidents are greater than 200
query5_alternative = """
SELECT sales, SUM(work_accident) as total_accidents
FROM Details
GROUP BY sales
HAVING total_accidents > 200
"""
result5_alt = pd.read_sql_query(query5_alternative, conn)
result5_alt.head()
```

[58]:

	sales	total_accidents
0	sales	587
1	support	345
2	technical	381

[59]:
```
# Close the connection
conn.close()
```