1->

select name from Employee e

inner join (select managerId, count(\*)

from Employee where managerId is not null group by managerId having count(\*)>= 5)b on e.id = b.managerId

* import pandas as pd

# Assuming you have a DataFrame 'df' with Employee data

# Example DataFrame

# df = pd.DataFrame({'id': ..., 'name': ..., 'managerId': ...})

# Step 1: Create a groupby operation to count employees by managerId

manager\_count = df[df['managerId'].notnull()].groupby('managerId').size().reset\_index(name='employee\_count')

# Step 2: Filter out managers with fewer than 5 employees

managers\_with\_5\_or\_more = manager\_count[manager\_count['employee\_count'] >= 5]

# Step 3: Merge this filtered data back with the original Employee DataFrame

result = pd.merge(df, managers\_with\_5\_or\_more, how='inner', left\_on='id', right\_on='managerId')

# Step 4: Select only the 'name' column

result\_names = result['name']

# Show the result

print(result\_names)

2nd method   
  
import pandas as pd

def find\_managers(employee: pd.DataFrame) -> pd.DataFrame:

    manager\_count = employee[employee['managerId'].notnull()].groupby('managerId').size().reset\_index(name='manager\_count')

    managers\_with\_5\_or\_more = manager\_count[manager\_count['manager\_count'] >= 5]

    result = pd.merge(employee, managers\_with\_5\_or\_more, how='inner', left\_on='id', right\_on='managerId')

    return result[['name']]

2->

select s.user\_id,cast((coalesce((select count(\*) from Confirmations c1  where c1.user\_id = s.user\_id

and action = 'confirmed')/(select count(\*) from Confirmations c2  where c2.user\_id = s.user\_id ),0)) as decimal(10,2)) as confirmation\_rate

from Signups s

import pandas as pd

def calculate\_confirmation\_rate(signups: pd.DataFrame, confirmations: pd.DataFrame) -> pd.DataFrame:

# Step 1: Count confirmed actions for each user

confirmed\_count = confirmations[confirmations['action'] == 'confirmed'].groupby('user\_id').size().reset\_index(name='confirmed\_count')

# Step 2: Count total actions for each user

total\_count = confirmations.groupby('user\_id').size().reset\_index(name='total\_count')

# Step 3: Merge the two counts on 'user\_id'

user\_counts = pd.merge(confirmed\_count, total\_count, on='user\_id', how='right')

# Step 4: Calculate confirmation rate, handle division by zero (if total\_count is 0)

user\_counts['confirmation\_rate'] = (user\_counts['confirmed\_count'] / user\_counts['total\_count']).fillna(0)

# Step 5: Merge with signups DataFrame to get user\_id and confirmation\_rate

result = pd.merge(signups[['user\_id']], user\_counts[['user\_id', 'confirmation\_rate']], on='user\_id', how='left')

# Step 6: Format the confirmation rate to two decimal places

result['confirmation\_rate'] = result['confirmation\_rate'].apply(lambda x: round(x, 2))

return result

3->

select Department, Employee,salary

from

(select d.name as Department,e.name as Employee, e.salary,dense\_rank() over(partition by e.departmentId order by e.salary desc) as rank1

from Employee e

inner join Department d  on e.departmentId = d.id)s where rank1 <= 3



import pandas as pd

def get\_top\_employees\_by\_salary(employee: pd.DataFrame, department: pd.DataFrame) -> pd.DataFrame:

# Step 1: Merge the Employee and Department DataFrames on departmentId and id

merged\_df = pd.merge(employee, department, left\_on='departmentId', right\_on='id', how='inner')

# Step 2: Rank employees within each department based on salary in descending order

merged\_df['rank1'] = merged\_df.groupby('departmentId')['salary'].rank(method='dense', ascending=False)

# Step 3: Filter for employees with rank <= 3

top\_employees = merged\_df[merged\_df['rank1'] <= 3]

# Step 4: Select the desired columns (Department, Employee, salary)

result = top\_employees[['name\_x', 'name\_y', 'salary']] # 'name\_x' for Employee, 'name\_y' for Department

# Step 5: Rename columns to match the SQL output

result.columns = ['Department', 'Employee', 'salary']

return result

3->

select substring(trans\_date,1,7) as month,

country,

count(\*) as trans\_count,

sum(state = 'approved') as approved\_count,

coalesce(sum(amount),0) as trans\_total\_amount,

sum(if(state = 'approved',amount,0)) as approved\_total\_amount

from Transactions b

group by substring(trans\_date,1,7), country



import pandas as pd

# Assuming you have a DataFrame called 'df' with the columns: 'trans\_date', 'country', 'state', 'amount'

# Ensure the 'trans\_date' is in datetime format

df['trans\_date'] = pd.to\_datetime(df['trans\_date'])

# Extract the month from the 'trans\_date'

df['month'] = df['trans\_date'].dt.strftime('%Y-%m')

# Calculate the required aggregations

result = df.groupby(['month', 'country']).agg(

trans\_count=('trans\_date', 'size'),

approved\_count=('state', lambda x: (x == 'approved').sum()),

trans\_total\_amount=('amount', 'sum'),

approved\_total\_amount=('amount', lambda x: x[df['state'] == 'approved'].sum())

).reset\_index()

# If you want to ensure zero is used for missing amounts

result['trans\_total\_amount'] = result['trans\_total\_amount'].fillna(0)

result['approved\_total\_amount'] = result['approved\_total\_amount'].fillna(0)

print(result)